JAVA FOR THE STUDY OF EVOLUTION

VOLUME VII

TWISTING RANDOMNESS

José del Carmen Rodríguez Santamaría
The EvolJava Community

Contents

Preface

The series *Java for the Study of Evolution* is directed to scientists that want to manage a serious but not excessively expensive tool to study evolution by direct experimentation under perfectly controlled conditions. These requirements cannot be met in nature but only in simulations and mathematical models. In consequence, the series has three main purposes:

1. To endow the community of **researchers in biology and evolution** with *high level programming* enabling an accurate study of models and simulations of the most diverse nature.

2. To clearly show how this tool is used to study the fundamental questions of evolution.

3. To suggest that the study of Java could be *very fruitful* for **undergraduates** in biological sciences even more than calculus alone.

This is the seventh volume: The purpose of this volume is to discuss Darwinism (evolution is a natural phenomenon fueled by randomness) by contrast to Lamarckism (natural evolution might be helped by twisting randomness). We discover that all successful applications of evolution somehow twist randomness and that the Evolutionary Theory of the Scientific International Literature is neither one or the other but a misconception. This volume can be worked out after vol I or vol V.

Bogotá, Colombia, *José Rodríguez*
2/XII/ 2011

Introduction

Evolution is a process in which objects might change, reproduce with variability and/or be selected for. The idea of evolution is inborn to every human being. So, one can find *evolution everywhere*, such as it is discussed in the first chapter. An *evolutionary theory* aims at explaining facts by means of evolution. In ordinary life one is used to formulate, study, accept or reject evolutionary theories including those in which evolution is dominated by *selection as a driving force*, chapter two.

When human beings are engaged in an evolutionary process, they usually try to make changes specially targeted to improve good qualities. This is the very idea of ***Lamarck***. Contrary to this, **Darwinism** shows that mutations at random, which are blindly executed and without any intention whatever, are capable of producing marvels if only the fruits of mutation are selected for the desired qualities.We contrast them in chapter three *Lamarck and Darwin*. Lamarckism is the way we use to experience evolution in ordinary life while Darwinism is a theorem of science. Darwinian evolution is very difficult to teach because every one tends to believe that evolution is by its very nature Lamarckian and that that mutations might be guided to swiftly achieve results.

In biology, **evolutionism** teaches that all living beings on Earth are linked by descend to a common ancestor that appeared by randomness and self organization. The **Darwinian Evolutionary Theory** claims that *random mutation + selection* compose the underlying mechanism of evolution. To discuss the power of this mechanism, we need to be endowed with a fair insight given by an authorized referee. Our election is to choose **complexity as the judge**, whose insight is that *the genome is software and evolution is a software developer*, chapter four. We conclude that everything is possible for evolution if only it has enough time and resources to commit as many bugs as desired.

In regard with *the origin of species*, chapter five, Modern International Literature shows how well the fossil record and taxonomic data are fit by a genetic tree dominated by a trend towards complexity and hominization and then teaches that life is obviously explained by evolution. Nevertheless, the envisaged type of evolution cannot be Darwinian since it provides no cue in regard with the mandatory

price that Darwinian evolution must pay: bugs followed by corrections that create more bugs. The evolution of the Modern International Literature cannot be Lamarckian either because there is no mention to external agents or forces that could help evolution to defeat complexity. So, the Evolutionary Theory of the Modern Scientific Literature is a contradiction: it is evolution (of form and function) without evolution (from imperfection towards perfection along every sort of malfunctions and malformations).

The inertia created by the International Scientific Literature is so great that scientists need special formation to remove their blindness. We provide in chapter six a motivation to begin with: *evolution without evolvability* is the common form of implementing evolution but the evolution that we find in nature makes of it a great project that marvels every expert. In general, every scientist must seriously examine his or her **heart and mind** to see whether or not his or her beliefs fill in the vacuum for eternity that is in his or her heart. This might be a very nice but extremely difficult work, as we show in the last chapter, where we examine the history of evolution paying attention to the authors that left a track over it.

Chapter 1

Evolution everywhere

`Evolution is part of our ordinary life`

1 ***Purpose.*** *To study over toy models the way as we concoct and test scientific evolutionary theories in our ordinary life.*

1.1 Elementary definitions

Evolution is a process in which objects might change, reproduce with variability and/or be selected for.

An **evolutionary environment** is a world, real, artificial or gedanken (imagined), that enables evolution.

A **proposition** is a given affirmation that can be either false else true in regard with the **universal set** or set of reference. A **theory** is a set of propositions about a given universal set. **An evolutionary theory** is one that proposes an explanation to given facts using an evolutionary environment as the main basis. We will prefer **scientific theories** that are those that can be studied statistically to determine whether they are to be accepted or rejected. This first chapter shows how concepts are intertwined by resorting to **toy theories**, which are those that can be computed at once, maybe by inspection or with the help of very simple programs in Java. In the second chapter we deal with more complex theories that are more difficult to simulate.

The genetic material of living beings is composed of DNA, which can be described as a string of the form ATTCGGCTA.... Thus, we define an **alphabet** as a set of symbols or **letters**. For DNA, the alphabet is $\{A,T,C,G\}$. A **string** is a linear array without gaps of letters of the given alphabet, say, ATTCGGCTA. Strings can be ordered by a partial order relation $s_1 < s_2$ if and only if s_1 is a substring

of s_2, i.e., if coincides with a part of s_2. A maximal element in this relation is a **chromosome**, i.e., a string that is not substring of another string.

A **genome** is a set of chromosomes that encodes for the information to construct an **organism** given that living organisms already exists. Organisms form a **population**.

The members of a population can reproduce, live and die differentially in response to genetic composition, to the interaction with other members (lions chase in group and this enhance their probability of surviving and getting offspring) and to the interaction with the environment that might include effects over long periods of time. All this is generically called **natural selection** or selection. Our evolutionary theories deal with abstractions of these features.

1.2 Basic examples

Evolution is an everyday phenomenon and biological evolution is just one particular case. So, let us extract from ordinary life some examples to see what are they compound of.

2 ***The grand father with cellular phone.***

Old people might be reluctant to accept and use new technologies. Say, it is a strike when some grandfather that is used to wired telephones buys and uses a cellular one.

We rewrite this story that describes a usual behavior that suddenly changes to a new one as follows:

$T \to T \to T \ldots T \to C$

We see that our alphabet is composed of just two elements that are behaviors, to use telephone (T) and to use use a cellular phone (C). We register the chosen behavior every day and we noticed that during a long period of time, day after day, T was the rule. But some day in the recent past we observed a **mutation**, a change, from T to C. This change caused a great surprise among the family and friends of the grandfather. Why?

From our point of view, everyone believed that the behavior of the grandfather can be explained by the theory:

$T \to T \to T \ldots T \to T \ldots$

which expressed the conviction of the grandfather of being happy with the old technology. This theory was **deterministic** because the present creates the future in a single way according to its state.

But the experimental finding of C brought the theory to a collapse. We say that this theory was **falsified** by C, so it declared inadequate to explain observed facts.

The proposed **falsification** suggests that the new behavior cannot be explained by the state of mind that explained the old behavior of getting stuck to the old technology but that rather a new vision enlightened the decision of the grandfather of adopting the new tech, say, he must not move from his place to answer a call.

This evolutionary theory relies on an **evolutionary environment** that consists in:

- An alphabet $\{T\}$. There is only one chromosome with just one letter, T. This chromosome is an organism, the only one member of its population.
- A set of **rules for evolution**, for the construction of the future from the present, that consist in just one element: the **identity**. This rule builds the future as an exact image of the present. This function generates a deterministic evolution.
- A **scoring function** measures the degree of fitting of each organism by assigning a real number (say, 0.23) to each organism of the population. In this case, the theory says that behavior T is optimal, with score, say, 100. The scoring function represents an explanation of the observed behavior and is a synonymous of **selection**.
- An **observable** is a function that measures something, so it assigns numbers to the elements of some set. We enable in that way statistical studies. In this case, we have the observable that reports the proportion of T's. According to the theory, it is always one with full certainty. Therefore, any other value will falsify the theory. In fact, the theory was falsified once the value 0 was registered when C appeared.

3 ***Reproduction and recombination***

By looking at the sequence of events about telephone calls given by
$T \to T \to T \ldots T \to T \ldots$
one might wonder and exclaim: where are recombination, mutation, reproduction and selection that are so important in biology?

Our answer is as follows:

- We can interpret the given sequence as the evolution of a population of individuals in which T is at the same time a chromosome, a genome, an individual and a population.
- **Recombination** takes two chromosomes and produces a third one taking some substrings from the first and some others from the second. When T recombines with T, taken as chromosomes, T is produced.

- Mutation is turn off until the last event in which T is changed into C.
- The individual T begets T.
- Individual T is adapted to its milieu during all that time previous to the mutation of T into C. But due to a change of mind of the grandfather, from that moment on, T and C working in ensemble do it better than T alone. This is so because the grandfather is supposed to be now happier than before when he was bound to the wired telephone alone.

We are proposing that the characteristics of biological evolution that are important for accepting or rejecting the Evolutionary Theory of the origin of species can be found all around, everywhere in ordinary life. Therefore, *ordinary life is a great and extremely rich lab of evolution*.

4 ***Individuals vs groups***

Let us consider the proposition found at the end of the last list: T and C working in ensemble do it better than T alone. We are clearly declaring that we consider group effects as elementary descriptions of nature and that therefore are very important, fundamental for our general theories. Nevertheless, we will be able to prove in some few cases that a group description can be reduced to a sum of individual events. Let us refer to this reduction as the **individual vs group conundrum**.

5 ***A non-deterministic variation***

With a slight modification of the previous example, we can include non-determinism in the evolutionary process, as it follows:

Once the grandfather learned to use his cellular phone, he had two options, the old technology, T, and the new one C. It happens that after achieving equilibrium, the corresponding proportions were 60 and 40 respectively estimated weekly. But one of his granddaughters recently became engaged and the family of her husband is composed of a great majority of young women, all used to C, with the addendum of having no grand father with them. So, all those ladies like to call the grandfather and that is why he is very happy now. His happiness corresponds to the **falsification** of the Evolutionary Theory that relies on the next evolutionary environment that describes the life of the grandfather before the marriage:

- An alphabet = $\{T,C\}$. The strings describing the actual behavior of the grandfather over a week look something like this: $TTCTCTTCCT$. These

are our chromosomes. The population consists in one chromosome of length 10 and with a proportion of 6 T's against 4 C's. So the grandfather is called 10 times per week because he receives calls twice a week from every one of the five persons that like to call him. The order of calls for different persons can vary but not the numbers for each person. We do not know in general what will come next to a T or to a C, thus we have a non deterministic evolution, but we do know the proportion of T's per week: it is 4. Say, if the present week is described by $TTCTCTTCCT$, the next one might be $TTTCCTTCCT$, $TTCTCTTCTC$ or one among many others of its kindred.

- A set of rules for evolution that builds the future from the present: the only chromosome of the future is just a permutation of that describing the present. The permutation is made at random, so the evolution is not deterministic.

- A scoring function that measures the degree of fitting of the weekly behavior. In this case, the theory says that the weekly behavior of 6 T's against 4 C's is optimal, with score, say, 100. Thus, the behavior observed at equilibrium defines an optimum. The fine structure of the scoring function is not known and might be very complex as it was the evolutionary force that moved frequencies to its equilibrium values.

- An observable that reports the proportion of T's. According to the theory, it is 0.6 without uncertainty.

The proposal of **falsification** of this theory arises from the fact that the observed frequencies after the marriage and over the past 7 weeks gave a stable estimate of 4 T's against 6 C's over strings of length 20. This new state of affairs reflects the fact that the grand father receives now more calls from cellular phones because of the various young women of the family of the husband of his granddaughter. They are five and each one calls him religiously 2 times a week.

We see that the value of the observable that reports the proportion of T's changed from 0.6 to 0.4 and that difference includes no uncertainty. It is precisely the lack of uncertainty that allows us to claim that the new findings falsify the theory. Therefore, we have devised an evolutionary framework that models the life of the grandfather and explains why he is so happy now even more than before.

6 *A probabilistic variation*

Let us venture now on a probabilistic version of the story about the grandfather. We deal with behaviors of real people and will try to capture their way of living

into a theory. This is in general a very difficult task and we must be ready to make some few attempts before a satisfactory theory might be achieved. Our first attempt follows:

Since the marriage of his granddaughter three years have passed by. People still calls him but, it is painful to say it, not so much as it was before. The grandfather is fading into oblivion even so he is still alive. From the 20 calls that he received every week three years ago, he received in the last time 4, 5 or 6 with equal probability. And people behave in a week trying to fit probabilities of T and C that happened in the previous week. Initial frequency of T is 0.4. To worsen things, the family has recently discovered that they can have fun in a near lake and so they have been very busy. In consequence, the total numbers of calls during the last three weeks has been 13. So, the grandfather proclaimed the new situation as one of national disaster. His feeling was corroborated later because in the next two weeks he received just 8 calls, so that in the last five week he received 21 calls at all.

The frustration of the grandfather seems to corresponds to the **falsification** of the Evolutionary Theory that relies on the next evolutionary environment that describes the life of the grandfather before the discovering of the lake:

- An alphabet = $\{T,C\}$. The strings describing the actual behavior of the grandfather over a week look something like this: $TTCTC$. These are our chromosomes. The population consists in one chromosome of length 4,5,6 with equal probabilities. The length represents the total number of calls that he receives per week. People calls rather disorderly but some how trying to resemble the behavior of the previous week.

- A set of rules for evolution to build the future from the present: the only chromosome of the future is composed of a random sample of letters that describes the present.

- A scoring function that measures the degree of fitting of the weekly behavior. Since the grandfather worries about the number of calls, a good measure of the fitting of each chromosome is just its length divided by 20, the maximal numbers of calls that he has received. Let us remark that the scoring function is intended to be the driving force of evolution. Nevertheless, the crude reality is that apart from the continuous quarrels of the grandfather to achieve more calls there are other forces with contrary sign that also exert their influence over the dynamics, say, the lack of time, and that is why the observed behavior is not a maximum of the scoring function.

- A suitable observable is just the length of each chromosome.

The proposal of **falsification** of this theory arises from the fact that the number of calls during the last three weeks should have been greater than 12 and less than 18 but it was just 13 while over 5 weeks together the number of calls was 21 when it should have been at around 25.

Can we agree with the grandfather and accept that the theory envisaged by the previous evolutionary environment has been falsified? It happens that in scientific studies and under the presence of noise and random outcomes, we have a protocol for deciding when a null hypothesis or theory must be accepted or rejected. Let us recall it (after vol V of this series).

7 ***The scientific method in a nut shell.***

Science *is basically a contrast between what one sees and what one beliefs. For simple models, we can measure the discrepancy between facts and ideas. If this discrepancy is in within certain bounds that define what is normally expected, one usually prefers to remain quietly assuming that what one believes is the truth. But if the discrepancy is too large, one is invited to invent and test new explanations for observed facts. More operationally:*

1. *We characterize the output of a system as a random variable with a given distribution. We formulate a proposition about its mean or variance or shape. This is the null hypothesis: it is about the global behavior of the system.*

2. *Over the background of the formulated distribution, our aim is to contrast the null hypothesis with a specific distinguished event or set of events that possibly came from observation or experiment.*

3. *To that aim, we built an artificial, virtual world, with the same characteristics as those proposed by the null hypotheses.*

4. *One runs many times the virtual world under the same conditions as those declared by the null hypothesis. Mathematicians runs the system infinitely many times but in simulations, we run the system 1000, 10000 or one million times.*

5. *For each run in the virtual world one measures, always using the same protocol, the discrepancy between each event and that value predicted by the null hypothesis. One registers the distribution of those discrepancies and decides what events are normal and which are outliers. To that aim, one previously defines which is the fraction of events that one must consider as outliers, which is the level of significance. This allows to calculate barriers*

that divide the normal events or region of acceptance of the null hypothesis from the region of outliers or region of rejection of the null hypothesis.

6. *One employs the very same protocol used in the last point to measure the discrepancy between the distinguished event, that possibly came from an experiment of from an observation, and that value predicted by the null hypothesis.*

7. *If the discrepancy originated by the distinguished event is an outlier with respect to the distribution of the virtual world, one declares that the experiment does not support the null hypotheses. Otherwise, one declares that the null hypothesis explains the distinguished event or, more rigorously, one should say: our data or distinguished event was unable to reject the null hypothesis.*

8 *The code for calculating the theory*

Let us apply now this protocol to decide the probabilistic theory about the grandfather. The first thing to do is to simulate the evolutionary environment described by the theory to see how it looks.

The theory that describes the world of the grandfather before the discovering of the lake by his family is simple, and might be mentally calculated but only if one have training in mathematics. Nevertheless, we have nothing to worry because we can resort to a program in Java.

The theory represents the usual behavior before the discovery of the lake. It has two main elements. First: received number of calls per week in the last time was 4, 5 or 6 and with equal probability. Second: people try to repeat the behavior of the last week, i.e., the observed frequencies of T and C that happened in the previous week served as probabilities to design the future in the following week. Initial frequency of T is 0.4. Observed facts that must be contrasted with the theory and that presumably falsify it: the total number of calls during the last three weeks has been 13 while in the last five he received just 21 calls.

The code follows:

```
//Program G8 grandFather
//Calculation of an evolutionary theory.
/*  The  theory represents the usual behavior
 *  of calls from wired telephones (T)  or cell phones (C)
 *  to the grandfather by his family and relatives
 *  before the discovery of the lake. It has
 *  two main elements. First: received number
 *  of calls per week in the last time was  4, 5 or 6
```

```
 *  and with equal probability.  Second: people
 *  try to repeat the behavior of the last week,
 *  i.e.,  the  observed frequencies  of T and
 *  C that happened in the previous week served
 *  as probabilities to design the future in
 *  the following week. Initial frequency of T
 *  is 0.4.
 */

import java.util.Random;

public class grandFather
{
 private static int averageNCalls = 5;
 private static int randomShift;
 private static int numberCalls;
 private static int nWeeks = 30;
 private static double initialFreqT = 0.4;
 //Turn on of the random generator
 private static Random r  = new Random();


//Basic Test of the random generator for random generation
//of 4's, 5's, 6's with equal probabilities
public static void testRandGen1()
{
 System.out.println("Tests the random generator" +
" \nfor random generation of 4's, 5's, 6's " +
"\nwith equal probabilties");
 int counter4 = 0;
 int counter5 = 0;
 int counter6 = 0;
 int nExp = 4000;
 for(int i= 0; i < 3*nExp; i++)
  {
randomShift = r.nextInt(3);
numberCalls = averageNCalls + randomShift - 1;
System.out.println(numberCalls);
//counters are incremented in one
if(numberCalls == 4) counter4++;
```

```
if(numberCalls == 5) counter5++;
if(numberCalls == 6) counter6++;
  }
 System.out.println("Number of 4's = " + counter4);
 System.out.println("Number of 5's = " + counter5);
 System.out.println("Number of 6's = " + counter6);
 System.out.println("Expected number = " + nExp);
 System.out.println("Optional: carry out a z-test. = ");
}

//Basic test of the random generator for random generation
//of T's with prob probT;
public static void testRandGen2(double probT)
{
 System.out.println("Tests the random generator" +
" \nfor random generation of T'2" +
"\nwith probability " + probT);
 int counterT = 0;
 int counterC = 0;
 //Total number of chars
 int n = 4000;
 double t;
 for(int i= 0; i < n; i++)
  {
   t = r.nextDouble();
   if (t < probT )
   {
    System.out.println("T");
    counterT++;
   }
  else
   {
System.out.println("C");
counterC++;
   }
 }
 System.out.println("Number of T's = " + counterT);
 System.out.println("Number of C's = " + counterC);
 System.out.println("Total number of chars = "  + n);
 double expT= n*probT;
```

```
 System.out.println("Expected number of T's = "  + expT);
}

public static double generation(double probT)
{
 //generates the number of calls for this week
 randomShift = r.nextInt(3);
 numberCalls = averageNCalls + randomShift - 1;
 System.out.print("Number of calls = "
          + numberCalls + ": ");
 int counterT = 0;
 double t;
 //Generates actual type of calls with prob of T = probT;
 for(int i = 0; i < numberCalls; i++)
  {
   t = r.nextDouble();
   if (t < probT )
{
 System.out.print("T");
 counterT++;
}
   else  System.out.print("C");
  }
 double cT = counterT;
 double freqT = cT/numberCalls;
 System.out.print("    FreqT  = " + freqT);
 System.out.println();
 return freqT;
}

public static void evolution(double probT)
{
 for(int i = 0; i < nWeeks; i++)
  {
   System.out.print("Week = " +  i + " ");
   //the system try to reproduce the frequency of T calls
   probT = generation(probT);
  }
}
```

```
public static void main(String[] args)
{
 boolean testRandGen1 = false;
 if (testRandGen1) testRandGen1();
 double probT = initialFreqT;
 System.out.println("Simulation of calls week by week.");
 System.out.println("T = call from a wired telephone.");
 System.out.println("C = call from a cell phone.");
 System.out.println("Initial frequency of T's = " + probT);
 evolution(probT);
}//End of main
}//End of class
```

9 ***Exercise.*** *Run the previous program. Activate the enclosed tests to confirm that the program generate the expected frequencies. Verify else reject the next conclusions drawn by the Author:*

1. No mater which initial proportion of T's one chooses, the system tends to evolve towards fixation: everything is T else everything is C.

2. This implies that the theory predicts that shortly after any initial setup, those that have cell phones ceases to call else those that have wired lines.

3. This situation contradicts the feeling of the grandfather: he declares the situation as one of national interest but not as the end of the world. So, there remain some persons from both groups that still call him.

4. This is precluded by our simulation that predicts that he will forgotten by one group taken at random but not by both.

5. We conclude that our theory does not fairly represent the situation previous to the discovery of the lake.

6. We must do another try.

10 ***Sampling effects, drift and fixation.*** *Previous simulation describes a very important aspect of evolution:*

1. When one uses probabilities to construct the future from the present, one must make a sample of the present population. Because natural populations

are finite, there is always a difference between expected probabilities and generated relative frequencies. This causes the relative frequencies of a given trait in a population to **drift**, to move but by mere chance, by **sampling effects**.

2. We conclude that there is always in every evolutionary environment with finite populations a driven **agent** that causes change: drift. This agent is immanent in evolution, cannot be silenced and is the default explanation of every change. This agent is not deterministic as a **force** or as a systematic, selective effect. Drift is **dispersive**: if various populations are set up in the same initial condition, a bit later and due to drift, the relative frequencies of tracing traits will be completely different. By contrast, a force is deterministic: it produces upon the same inputs the same results over and over again. So, it would be better to say that drift is a non-deterministic agent instead of saying that it is force o deterministic agent.

3. The final result of drift is **fixation**: some traits will get fixed at the expense of others which disappear from the population not because there are worst but because of randomness.

4. It is well known that there are many **loci** (encoded genetic functions) in genetics that are **monomorphic**, with only one **allele** or version. This can be explained in two ways: by fixation due to sampling else by functional demands that select the optimal variant. Which option to take is the matter of study of **population genetics**, the study of changes in genetic observables under evolution.

5. Since there are many loci that are **polymorphic**, with many variants, one must infer either that variants are the result of mutation else that there is some selective force that tend to maintain the variants in the population, say in heterozygous form. Which option to choose is another problem in population genetics.

11 ***Exercise.*** *Invent a very simple mathematical reasoning showing that fixation is unavoidable when evolution rely on sampling. So, drift and fixation are robust concepts in evolution.*

12 ***Exercise.*** *Mathematical models in population genetics predict that the proportion of fixation of a given trait is just the initial frequency of that trait in the population. Modify the previous program to test this asseveration.*

13 ***Exercise.*** *The previous program builds the future by measuring the relative proportions of the present generation and abstracting them as probabilities that guide the generation of new individuals for the next generation. Therefore, we see here a group directive: probabilities are computed over the whole sample space and not by looking at isolate individuals. Prove that this group directive can be reduced to treatment of events that are directly related to individuals.*

14 ***The grandfather: a modified version***

Humans are able to superpose random or pseudo-random behavior over a deterministic way of life. Determinism might be dictated by necessity, as in the need to take lunch but randomness determines the hour of taking it. The family of the grandfather is not an exception: their habits and moral law say that they must call him but who, when and how many times might be a matter of randomness.

Let us try to combine deterministic and random effects to produce a theory, a model, to capture the calling behavior of the family of the grandfather whose main two points are: First: received number of calls per week in the last time was 4, 5 or 6 and with equal probability. Second: people try to repeat the behavior of the last week, i.e., the observed frequencies of T and C that happened in the previous week served as probabilities to design the future in the following week. Old people use wired telephone and call him at least once because the force of the custom but the young ladies with cell phones call him at least one time in a week by moral law and compassion. Initial frequency of T is 0.4. Observed facts that must be contrasted with the theory and that presumably falsify it: the total number of calls during the last three weeks has been 13 while in the last five he received just 21 calls.

15 ***The code to simulate this new situation follows:***

```
//Program G15 grandFather2
//Calculation of an evolutionary theory.
/*  The  theory represents the usual behavior
 *  of calls from wired telephones (T)  or cell phones (C)
 *  to the grandfather by his family and relatives
 *  before the discovery of the lake. It has
 *  two main elements. First: received number
 *  of calls per week in the last time was  4, 5 or 6
 *  and with equal probability.  Second: people
 *  try to repeat the behavior of the last week,
 *  i.e.,  the  observed frequencies  of T and
```

```
 *  C that happened in the previous week served
 *  as probabilities to design the future in
 *  the following week. Moral law and the force
 *  of habits dictate that there must be at least
 *  one T and one C every week.
 *  The initial frequency of T  is 0.4.
 *  Observed facts that must be contrasted with
 *  the theory and that presumably falsify it:
 *  the total number of calls during the last three
 *  weeks has been 13 while in the  last five he
 *  received just 21 calls.
 */

import java.util.Random;

public class grandFather2
{
 private static int averageNCalls = 5;
 private static int randomShift;
 private static double initialFreqT = 0.4;
 private static int nEvents = 30000;
 private static int RecordCalls[ ] = new int[nEvents];
 private static int Dist[ ] = new int[nEvents];
 //Number of groups of weeks
 private static int nGroups = 1000;
 //Number of weeks per group
 private static int nWeeks;
 private static int counterCalls;
 //Observed number of calls
 private static int observed;
 private static int min, max;
 private static boolean print;
 //Turn on of the random generator
 private static Random r  = new Random();


//Basic Test of the random generator for random generation
//of 4's, 5's, 6's with equal probabilities
public static void testRandGen1()
{
```

```
 System.out.println("Tests the random generator" +
" \nfor random generation of 4’s, 5’s, 6’s " +
"\nwith equal probabilties");
 int counter4 = 0;
 int counter5 = 0;
 int counter6 = 0;
 int nExp = 4000;
 for(int i= 0; i < 3*nExp; i++)
  {
randomShift = r.nextInt(3);
int numberCalls = averageNCalls + randomShift - 1;
System.out.println(numberCalls);
//counters are incremented in one
if(numberCalls == 4) counter4++;
if(numberCalls == 5) counter5++;
if(numberCalls == 6) counter6++;
  }
 System.out.println("Number of 4’s = " + counter4);
 System.out.println("Number of 5’s = " + counter5);
 System.out.println("Number of 6’s = " + counter6);
 System.out.println("Expected number = " + nExp);
}

//Basic test of the random generator for random generation
//of T’s with prob probT;
public static void testRandGen2(double probT)
{
 System.out.println("Tests the random generator" +
" \nfor random generation of T’2" +
"\nwith probability " + probT);
 int counterT = 0;
 int counterC = 0;
 //Total number of chars
 int n = 4000;
 double t;
 for(int i= 0; i < n; i++)
  {
   t = r.nextDouble();
   if (t < probT )
   {
```

```
    System.out.println("T");
    counterT++;
   }
  else
   {
System.out.println("C");
counterC++;
   }
 }
 System.out.println("Number of T's = " + counterT);
 System.out.println("Number of C's = " + counterC);
 System.out.println("Total number of chars = "  + n);
 double expT= n*probT;
 System.out.println("Expected number of T's = "  + expT);
}

public static double generation(double probT)
{
 //generates the number of calls for this week
 randomShift = r.nextInt(3);
 int callsThisWeek = averageNCalls + randomShift - 1;
 if (print)
 System.out.print("Number of calls = "
          + callsThisWeek + ": ");
 counterCalls = counterCalls + callsThisWeek;
 int counterT = 0;
 double t;
 //Generates actual type of calls:
 //Two of them are TC,
 if (print) System.out.print("T");
 counterT++;
 if (print) System.out.print("C");
 //Others calls are
 //generated with  prob of T = probT;
 for(int i = 2; i < callsThisWeek; i++)
  {
   t = r.nextDouble();
   if (t < probT )
{
   if (print) System.out.print("T");
```

```
 counterT++;
}
   else
   if (print) System.out.print("C");
  }
 double cT = counterT;
 double freqT = cT/callsThisWeek;
 if (print) System.out.print("    FreqT  = " + freqT);
 if (print) System.out.println();
 return freqT;
}

public static void evolution(double probT)
{
 int counterRecord = 0;
 for(int i = 0; i < nGroups; i++)
  {
   counterCalls = 0;
   for(int j = 0; j < nWeeks; j++)
    {
     if (print) System.out.print(  i + " " + j);
     probT = generation(probT);
    }
   if (print)
   System.out.println("Number of calls in " + nWeeks
   + " weeks = " + counterCalls);
   RecordCalls[counterRecord] = counterCalls;
   //counter is incremented by one.
   counterRecord++;
  }
}

//Find the distribution of numbers of total
//calls recorded in RecordCalls[]
public static void findDistribution()
{
 min = nEvents;
 max = 0;
 for(int i = 0; i < nEvents; i++)
  {
```

```
Dist[RecordCalls[i]] ++;
//! means boolean negation
if (!(RecordCalls[i]==0))
 {
  if (RecordCalls[i] > max ) max = RecordCalls[i];
  if (RecordCalls[i] < min ) min = RecordCalls[i];
 }
  }
 System.out.println("Distribution of calls");
 System.out.println("Number of calls and frequency:");
 for(int i = min; i <= max; i++)
 System.out.println( i + " " + Dist[i]);
}

//The pValue of observed value is found, i.e.
//the probability of getting an event as extreme or more
//than the observed value
public static void findSignificance(int observed)
{
  double totalSum = 0;
  double partialSum = 0;
  for(int i = min; i <= max; i++)
   totalSum = totalSum + Dist[i];
  for(int i = min; i <= observed; i++)
  partialSum = partialSum + Dist[i];
  double pValue = partialSum/totalSum;
  System.out.println("Partial sum until " + observed +
  " =  " + partialSum);
  System.out.println("Total sum = " + totalSum);
  System.out.println("pValue = " + pValue);
}
//The proposed theory is studied:
//First: we find the distribution of calls during
//a group of nWeeks.
//Second: we find the significance of the
//given barriers. First barrier = 13 for groups of 3 weeks
//and the second is 21 for groups of 5 weeks.
public static void studyTheory()
{
 findDistribution();
```

```
 findSignificance(observed);
}

public static void main(String[] args)
{
 print = false;
 boolean testRandGen1 = false;
 if (testRandGen1) testRandGen1();
 double probT = initialFreqT;
 System.out.println("Simulation of calls week by week.");
 System.out.println("T = call from a wired telephone.");
 System.out.println("C = call from a cell phone.");
 System.out.println("Initial frequency of T's = " + probT);

 boolean testRandGen2 = false;
 if (testRandGen2) testRandGen2(probT);
 //Initialization
 for(int i = 0; i < nEvents; i++)
 RecordCalls[i] = 0;
 nWeeks = 3;
 observed = 13;
 System.out.println("\nWeeks are grouped by  " + nWeeks);
 System.out.println("Observed number of calls  "
 + observed);
 evolution(probT);
 studyTheory();
//Initialization
 for(int i = 0; i < nEvents; i++)
 RecordCalls[i] = 0;
 nWeeks = 5;
 observed = 21;
 System.out.println("Observed number of calls  "
 + observed);
 System.out.println("\nWeeks are grouped by  " + nWeeks);
 evolution(probT);
 studyTheory();
}//End of main method
}//End of class
```

16 ***Exercise.*** *Run and play with the code. Accept else reject the next conclusions drawn by the Author:*

1. The feeling of desolation of the grandfather began by the third week but nevertheless there were not enough observations to claim with a significance less than 0.05 that the family forgot him because the p-value of 13 was greater than 0.1, a fact that says that this can eventually happens by mere randomness in within the old established behavior of the family. Recall that the **p-value** of an event is the probability of being at least as extreme as the event.

2. When one suspects something but cannot scientifically prove it, one must pick up more observations, more data. In this case, 3 weeks were not enough to prove that the grandfather was right but 5 were sufficient to achieve this: the p-value of 21 is as a rule less than 0.02, a value that invites everyone to consider that something bad has happened with the relation of the family with the grandfather.

3. We have illustrated a very general fact: behind ordinary decisions there is almost always a theory that can be recast in evolutionary form, so we can say that evolution pervades our lives in a rich variety of forms.

4. We conclude that rejection of evolutionary theories in usual life of every person is as ordinary as breathing.

17 ***Exercise.*** *We have interpreted the results of the simulation in the previous program by assuming that the studied theory indeed produces a probability density function. This means that the tables of relative frequencies tend to converge to a limit. In other words, two tables taken at an interval of, say, thousand generations must differ very little one from another. Adapt the previous program to test this. Challenge: use a formal statistical test to guide your study.*

1.3 Conclusion

Life is plenty of evolution, evolutionary processes and evolutionary theories, whose rejection or acceptance is as frequent and ordinary as breathing. By the way, the complexity of that natural evolutionary processes easily can become very high. We know this because the simulation of very simple instances already present more or less serious challenges.

Chapter 2

Selection as a driving force

`Artificial selection in action`

Measures of fitting of given objects were given in our previous toy theories but it happened that they appeared more as witnesses than as actors of evolution. This was so because the evolutionary environments were observed at equilibrium. Let us see now a case of ordinary life in which selection is precisely the motor of change, of evolution. We emphasize that we choose folds of ordinary life because we want the Reader to make sure that evolution can be seen everywhere in nature and in our lives, where it is directly or indirectly implemented, and that acceptance and rejection of evolutionary theories is something inescapable that every human does at every moment.

2.1 Organismic selection

Let us a case of ordinary life that illustrates that selection is made over whole organisms.

18 ***Learning the order of words***

Texts are formed from sentences which in their turn are formed of words. In consequence, **grammar** is the study of the formation of words, say, in declination of verbs, while **syntax** studies the rules for forming admissible sentences and in particular the right ordering of words.

Let be **Syntax** the problem that receives a disordered phrase and must produce a syntactically perfect text. We inbuilt an automatic mechanism to give feedback, the selective agent which simulates a teacher that knows the answer, and with its help, we will look for the perfect text.

To implement evolution we proceed in various steps. The first one is to discuss the form as the problem must be encoded in an evolutionary environment.

Our original phrase is *syntax deals with the correct formation of sentences*. We separate this phrase into words, permute words and end up with a disordered string, say: *the deals syntax correct with formation sentences of*. Our task is to restore the original correct phrase. Words are kept in a vector and are addressed by their corresponding index, say, the word *correct* is word number 4 because Java begins counting with 0. Therefore, a given phrase, ordered or not, can be represented by the string corresponding to the orders in which words of the vector shall appear. Say, the string 01234567 is just the original string but 47135062 encodes the disordered phrase : *correct sentences deals the formation syntax of with*.

To each disordered phrase a score is given: it is the very same ordering string but taken as a number. String a is better than string b if the first is less than the second taken both as numbers. Say, the perfect phrase is 01234567 and it is better than 47135062, which is very disordered.

We will test the following theory: evolution can indeed solve Syntax. Thus, we must define an observable that will say whether or not we have succeeded.

We can specify now the evolutionary environment:

- The alphabet = $\{0,1,2,3,4,5,6,7\}$. A chromosome is a permutation of the string 01234567. A chromosome is at the same time a genome and an individual.

- A set of rules for evolution that builds the future from the present: the future generation is formed from the present one by a process of mutation, recombination and selection. We choose a smart set of mutations, which is based on transpositions. A **transposition** is a change of place between two letters in a string. A succession of transpositions produces a **permutation**, a change of order, and any one permutation can be decomposed as a succession of transpositions. Recombination takes two chromosomes and produces a third one by interchanging parts of them.

- Selection is represented by a scoring function that measures the degree of fitness of each chromosome. The fitness of a chromosome is precisely the number that it represents. Individuals are ordered by fitness in decreasing order: the less they are, the more ordered is the phrase they represent.

- To decide that we indeed can solve the given problem by evolution, we have the observable that associates to each chromosome the difference between

it and 01234567. If this difference is zero, we can claim that we have succeeded and we will tell the world that evolution functions. This is really very serious.

19 ***The code that simulates evolution as a tool to solve Syntax follows. An algorithm of this type, that simulates evolution to solve a specific problem is a*** ***genetic algorithm.***

```
//Program G19 syntax
//We mimic a game that a teacher
//proposes to her pupils:
//A disordered phrase is given to pupils.
//A disordered phrase is just a permutation of
//the original one, a variation of order of its words.
//They must restore from this
//string the original sentence
//with correct meaning and syntax.
//The problem is solved by means of EVOLUTION,
//in which selection is the driving force of the
//ensuing evolutionary process.
//In this case, selection is given by the teacher,
//who already knows the answer and can estimate
//how good are the attempts of the pupils.


//To disorder the original correct phrase,
//a series of swaps or transpositions is done.
//The best way to correctly reorder the
//given disordered phrase is to use also swaps.


import java.util.Random;

public class syntax
{
 private static String phrase =
  "syntax deals with the formation of correct sentences";
 private static int nLetters = phrase.length();
 private static int nWords;
```

```
//The phrase is decomposed in words
private static String[] Word = new String[10];
//Words are numbered. A disordered phrase is encoded
//in a vector of digits, Permutation[], such that
//Permutation[i] indicates what word goes
//at place i of the possibly disordered phrase.
private static int[] Permutation = new int[10];
//Turn on the random generator
private static Random r  = new Random();

//====Genetic part of declaration of variables===
// Individuals are kept in the array
// Individual[]. It is an array of strings.
//Each individual encodes
//the permutation that it represents.
//The individual is a string that encodes for
//a number.

// The number of individuals must be
// less than limit.
static double zMax; static double N;
static int limit = 50000;
static double Fitness[];
static String Individual[ ], Individualc [];
//Actual number of individuals
static int nIndiv ;
static int Order[];
static String b;
static int generation;
static  int  nGen;
static int ReportMin[], ReportMax[];
static double oldError, newError;
static double deltaError;
static boolean print,testEncoding;
//Our observable that declares success.
static boolean done = false;
static boolean recombination;
static double ERROR, FITNESS;
static String SOLUTION;
static int GENERATION;
```

```
//Words are recognized: they are separated by a space.
//They are kept in a vector.
private static void numberWords(String s)
 {
  int counter = 0;
  String f = "";
  for(int i = 0; i < nLetters; i++)
   {
char d = s.charAt(i);
//System.out.println(d);
if (!(d == ' '))  f = f+d;
else
 {
  Word[counter] = f;
  f = "";
  counter++;
 }
//last word
if (i == nLetters-1) Word[counter] = f;
   }
  //System.out.println( counter);
  System.out.println("\nThe words of the phrase " +
                       "with their indexes: ");
  nWords = counter +1;
  for(int i = 0; i < nWords; i++)
   {
System.out.println(i + "  " +  Word[i]);
Permutation[i] =  i;
   }
 }

 //A transposition  or swap is the  interchange of
 //place between two items.
private static void transposition()
 {
  int m = r.nextInt(nWords);
  boolean flag = false;
  //Sites must be different
```

```
  int n=0;
  while(!(flag))
   { n = r.nextInt(nWords);
   if (!(n == m)) flag = true;
   }
  if (print) System.out.println("Permuation of "
                               +  m + " and " + n);
  int k = Permutation[m];
  Permutation[m] = Permutation[n];
  Permutation[n] = k;
 }

//Mixer:  a sequence of transpositions creates variability.
//The output is encoded in a string, a chromosome.
private static String mixer(int numberOfTransp)
 {
  for(int i= 0; i < numberOfTransp; i++ )
 transposition();
  String v = "";
  for(int i = 0; i < nWords; i++)
    {
  Integer e = Permutation[i];
  v = v + e.toString();
    }
  return v;
 }



//Words in the sentence encoded by s are permuted

private static String permutate (String s)
 {
  numberWords(s);
  //Number of transpositions
  int n =  nWords;
  return  mixer(n);
 }

//String number  is decoded into a phrase
```

```
private static String stringToPhrase(String number)
  {
   String phrase= "";
   for(int j=0;j  < number.length();j++ )
    {
     char s3  =   number.charAt(j);
    int  l = Character.getNumericValue(s3);
    phrase = phrase + Word[l] + " ";

    }
   return phrase;
  }

//Declarations and default initializations
// of other arrays.
private static void otherInit( )
{
 Order= new int[limit+1];
 ReportMin= new int[limit+1];
 ReportMax= new int[limit+1];
 Fitness= new double[limit +1];
 for(int i = 0; i< nIndiv; i++)
  {
   Order[i]=0;
   ReportMin[i]=0;
   ReportMax[i]=0;
   Fitness= new double[limit +1];
 }
}

//An individual is a string that is a permutation
//of another one.
private static String generateIndividual( )
{
 //Number of transpositions
 int numberOfTransp = r.nextInt(nWords);
 String ind = mixer(numberOfTransp);
 return ind;
}
```

```
/* We generate nIndiv individuals (strings)
 encoding for permutations of 01234567 */
private static void Initialization( )
{
 //Formal declaration of our array.
 Individual= new String[limit];
 if (print)  System.out.println("ORIGINAL POPULATION");
 for(int i = 0; i< nIndiv; i++)
 {
  if (print) System.out.println( "\ni = " + i);
     Individual[i]="";
 // Individual[i] is a string,
 // it is a genotype that encodes a number,
 //its phenotype.
     Individual[i] = generateIndividual();
     if (print)
      {
       System.out.println(" = " + Individual[i] );
      }
    }
 otherInit();
  }



//String s is decoded into a number
private static long recoverNumber(String s)
 {
  long number = 0;
  for(int j=0;j  < nWords;j++ )
   {
char c  =   s.charAt(j);

  int  l = Character.getNumericValue(c);
  int q = j+1;
  long number1 = (long) (l * Math.pow(10,nWords-q));
  number = number + number1;

   }
  return number;
```

```
 }

// This method transforms a string in a number
//of type long. Some  outputs to console are displayed.
private static long stringToNumber(String s)
  {
if (print)
  {
   System.out.println( "\nIndividual =  " + s);
   System.out.println( "length of s  = " + s.length());
  }

if (print) System.out.println( "Recovering number");
long k = recoverNumber(s);

if (print)  System.out.println("String "
    + s +  " as number = "       + k + "\n");
 return k;
 }

//This method decodes the string into the entries
//of permutation[]
private static void stringToVector(String s)
{
 for(int j = 0; j  < nWords;j++ )
  {
char c  =   s.charAt(j);
    int  l = Character.getNumericValue(c);
Permutation[j] = l;
  }
}


private static double fitness(double error)
{
  return  1/(1+error*error);
}


//This fitness of each individual is found
```

```
private static void fitness(int gen)
{
 //We measure the error of individual[i]
 //with respect to the perfect one 01234567.
 for(int i = 0; i< nIndiv; i++)
  {
   double  error =
   Math.abs(1234567-stringToNumber(Individual[i]));

   //The error defines the fitness: less error, more fitness.
   //Maximal fitness = 1; minimal = 0.
   Fitness[i] = fitness(error);
   if (print)
  System.out.println("i = " + i
  + " Individual = " + Individual[i]
  + " Error  = "+ error
  + " Fitness"  + Fitness[i]    );
  if (error < 0.000000000001)
   {
 GENERATION = gen;
 ERROR = error;
 SOLUTION = Individual[i];
 FITNESS =  Fitness[i];
 done = true;
}
  if (print)  System.out.println();
  }
 }

//Individuals are sorted by fitness
private static void Sorting(int gen)
  {
   fitness(gen);
   if (print) System.out.println("\nSORTING");
   int   Champ;
  //Sorting by fitness.
  for(int i = 0; i< nIndiv;i++)
{
 Champ = 0;
 for(int j = 0; j< nIndiv;j++)
```

```
    if (Fitness[j] >= Fitness[Champ])   Champ = j;
     //The array Order classifies individuals by  fitness
  // by equal or decreasing order.
  Order[i] = Champ;
     if (print)
      {
       System.out.println(  i + "th ind. is  No "
    + Champ + " " + Individual[Order[i] ]
           + " Fitness = "  + Fitness[Champ]);
      }
     Fitness[Champ] = 0;
     }

String t = Individual[Order[0] ];
System.out.println(" Best solution "  + t  +
" -> " + stringToPhrase(t));
  }



//Each individual of the top ten
// produces 10  copies.
private static void Reproduction()
{
 if (print)
  System.out.println( "Reproduction");
 if (print)
  System.out.println( "The best = " + Order[0]);
 Individualc= new String[limit];
 int counter = 0;
 for (int top = 0; top < 10; top++)
 {
 for(int j = 0; j< 10; j++)
  {
    Individualc[counter] = Individual[Order[top]];
counter = counter +1;
  }
 }
 for(int j = 0; j< counter; j++)
Individual[j] = Individualc[j];
```

```
}

//A a sequence of transpositions creates variability.
private static String mixWords(String s,
                                int numberOfTransp)
 {
  //Chromosome s is transformed into a vector
  stringToVector(s);
  //Entries in vector are transposed
  for(int i= 0; i < numberOfTransp; i++ )
 transposition();
  //Vector is encoded into a chromosome, a string.
  String v = "";
  for(int i = 0; i < nWords; i++)
  {
  Integer e = Permutation[i];
  v = v + e.toString();
  }
  return v;
 }


//A mutation results from a series of transpositions
private static void Mutation()
{
  for(int j = 1; j<  nIndiv; j++)
{
 //Number of transpositions
 int n = r.nextInt(nWords);
 Individual[j] = mixWords( Individual[j],n);
}
}

//Two strings recombine and produce two offspring.
private static void Recombination()
{
 for(int j = 50; j<  nIndiv; j++)
  {
//Define place of recombination
```

```
int m =  r.nextInt(nIndiv);
int n = r.nextInt(nIndiv);
String a = Individual[m];
String b = Individual[n];
int placeRec = r.nextInt(nWords);
Individual[m] = a.substring(0, placeRec)
               + b.substring( placeRec);
Individual[n] = b.substring(0, placeRec)
       + a.substring( placeRec);
 }
}

//Master for a generation
private static void dynamics(int gen)
 {
  //Individuals are sorted by fitness
  Sorting(gen);
  //The top ten are preferentially reproduced
  Reproduction();
  //The new population is subjected to mutation
  Mutation();
  if (recombination) Recombination();
 }

//The original ordered phrase is rebuilt
//by a genetic algorithm.
//This program simulates a teacher that knows
//the answer and guides their students
//for a better solution.
private static void restorePhrase()
 {
  System.out.println("\n\nRunning ");
  nIndiv = 150;
  Initialization( ) ;
  nGen = 200000;
  oldError = 1000;
  for(int gen = 1; ( gen <= nGen) & (!(done)); gen++)
 {
   System.out.print("\nGENERATION = " + gen);
   dynamics(gen);
```

```
 }
  if (done)
   {
 System.out.println("\nGENERATION = " + GENERATION +
       " SOLUTION FOUND");
     System.out.println("ERROR = " + ERROR);
     System.out.println("SOLUTION = " + SOLUTION
      + " -> " +  stringToPhrase(SOLUTION) );
     System.out.println("FITNESS = " + FITNESS);
     System.out.println("Number of generations = "
      + GENERATION);
     System.out.println("Number of individuals " +
      "per generation = " + nIndiv);
     int totalCost = GENERATION * nIndiv;
     System.out.println("NUmber of Generations x " +
      "number of individuals = " + totalCost);
   }
 }

public static void main(String[] args)
 {
  print = false;
  recombination = false;
  System.out.println("This program disorders a sentence"
                +  " \nand reorders it by evolution "
             + "\nusing random transpositions.\n");
  System.out.println("ORIGINAL SENTENCE: ");
  System.out.println(phrase);
  String p = permutate(phrase);
  System.out.println( "\nCHALLENGING DISORDERED SENTENCE" +
   " WITH ITS INDEXATION : \n" );
  System.out.println( p + " -> " + stringToPhrase(p) );
   restorePhrase();
 }//End of main method
}
```

20 ***Exercise.*** *Run the program and play with the code. Decide whether or not evolution solves Syntax.*

21 ***What is and what is not science.***

We have solved a concrete instance of a puzzle for children with age under 7. We have obtained a great victory. This very feeling of glory is shared all around the world at every moment: evolution is an ordinary tool to solve problems. This feeling of joy propels many people to claim that biological evolution is the explanation of our existence. Dear Reader: this is not science. What is science?

Science begins with very clear, simple and sharp concepts. The very first one is that to falsify a theory all one needs is a statistical **falsification** along anyone observable. But to prove a theory, we need to show that it is robust along all possible observables which must be tested over any possible initial setting and diversity of parameters that are allowed by that theory. One uses to think that it is by far easier to falsify a theory than to prove it to such a degree that to prove a theory may be considered unrealistic and out of the scope of science.

Let us elaborate this point a bit better.

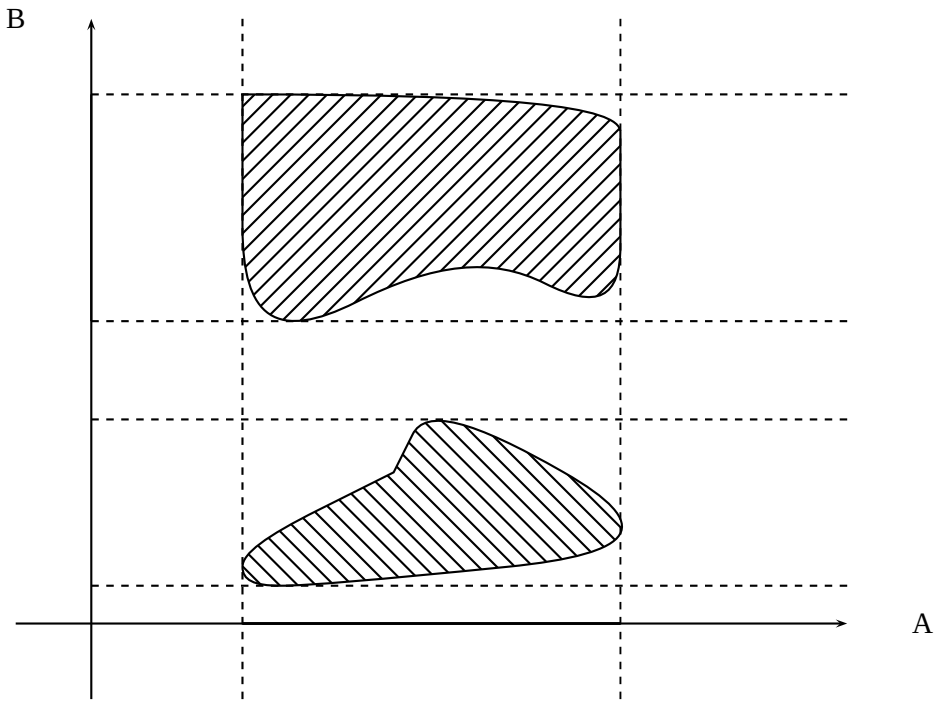

Figure 2.0. Two objects are different if and only if they are discriminated by a given observable (axis B) no mater that they look the same according to every other one (bundle of axes, A).

Let us look at Marina and Alla. Bot have two legs and two eyes, both are woman, both are Russian, both like Igor and both have the very same 10000 genetic loci that distinguishes a human being. But they are different: in spite of being identical twin sisters, Alla has, unlike Marina, a small mole in the left hand. In more formal terms, two objects are different if and only if they are discriminated by a given observable no mater that they look the same according to every other one.

The same happens with theories. These are objects composed of axioms, suppositions, allowed initial and border conditions, predictions. A theory with partial validity matches facts along several observables or across certain regions of observations. A true theory matches facts along every observable.

By mathematical reasons, only very simple universes have true theories. All others have only theories with partial validity else with inner contradictions. Therefore, the concept of all-proof truth is alien to science. Nevertheless, each person and each scientist ranks observations (objective or subjective) according to its importance, a classification that he or a she makes on personal or cultural grounds, and them applies a simple principle: an important set of facts cannot be contradicted or ignored by a theory. It is in this way that personal and social truths arise. The Evolutionary Theory in the light of all religions of the most varied kind is a good example to think about.

22 ***Exercise.*** *In the program G19 turn on recombination (at the very end of the program, in the main method). Proof statistically that the number of generations to achieve the desired solution is dramatically augmented by recombination. Explain and interpret this result.*

2.2 Selection over genes

Sometimes selection, which is organismic, can be traced down to genes.

23 ***Selection over genes***

Selection is an abstract term that refers to the differential reproduction and surviving of individuals in response to the challenges of the environment. Thus, the object of selection are the organisms. When an organism dies, its genome also dies and with it the genes that it carried. So, at its face value and in regard with selection, genes are enslaved to organisms. Nevertheless, there are some mathematical difficult problems that enlighten that reading. To fix ideas, let us consider the previous program about ordering some words to form a sentence with meaning:

We encoded each phrase as a chromosome which was a permutation of 01234567. This chromosome was at the same time a genotype, carrying genetic information, and a phenotype, suffering the impact of selection. The genes of this chromosome were the alphabet $\{0,1,2,3,4,5,6,7\}$. Selection was based on the absolute difference of the number representing a chromosome and the perfect one, 1234567. Thus selection operated at the organism or chromosome level, looking at the genetic information as a whole.

Let us pose now the next question: Does the organismic selection induces a selection of genes so that we could say that selection operates over genes? We might understand better what is it all about if we pay attention to the following observation: the organismic selection in our syntax problem induces a selection among genes by positional effects. In effect, each gene strives to get a place at the foremost left but the force is higher the less it is. The ensuing conflict is resolved by a perfect ordering. For instance, the chromosome 01234576 is strained because in 76, the last pair of genes, 6 is stronger than 7 and so 6 forces 7 to a transposition that results in 01234567.

Let us assign a fitness to each chromosome based on positional effects of genes:

- The fitness of 01234576 is -1 because there is a pair of contiguous genes, 76, that violate a correct ordering.
- The fitness of 01324657 is -2 because there are two pairs of contiguous genes, 32 and 65, that violate a correct ordering.
- The fitness of 10324657 is -3 because there are three pairs of contiguous genes, 10, 32 and 65, that violate a correct ordering.

The fitter reproduces more.

24 ***Exercise.*** *Adapt the previous program G19, page 25, to check the theory that organismic selection induces a type of gene selection that for all purposes replaces the former.*

25 *The concept of function*

We must be aware of the fact that according to modern biology the purpose of life is to fuel evolution. Let us see in which way evolution gives rise to the concept of function.

To fix ideas, let us consider the enzyme exonuclease. Studies in vitro have allowed us to conclude that the function of this enzyme is to cleave foreign DNA. But we currently think that this conclusion can be extrapolated to the living cell. Let us

notice that nobody disagrees on this. But, what, if any, is the role of evolution in that definition of the function of the exonuclease? The answer is that **evolutionism** is a culture that one must fabricate by mere pleasure:

Evolution entails selection, which is in first term organismic because organisms are the units that live, die and reproduce. More rigorously, the basic evolutionary item must be the family, which is the smallest unity with the capability of reproducing. A family reduces to a couple, when the offspring is abandoned to their luck, or to a single individual, if it can reproduce alone, as in the case of bacteria or some plants. So, when one says in modern biology that for instance a given enzyme has a function, one is contending that the overall selection that exerts pressure over families and whole organisms can somehow induce pressure over genes from which they acquire a function, an evolutionary function.

Thus, our evolutionary understanding of life is built on system analysis that formulates an invitation to divide a complex system into tractable subsystems to next merge their behavior to compute the overall response.

A prerequisite for a successful division is the stability of function against some variations of the milieu so that it becomes possible to reproduce the cellular milieu sufficiently well in such a way that the function of subsystems is not distorted.

So, what is the evolutionary function of that enzyme that in vitro is given the exonuclease function? The two concepts, in vitro and evolutionary, coincide: if that enzyme fails, foreign DNA may get inserted into own genome and most surely it will kill life because of incompatibility problems. The final result is that evolution is terminated together with a less probably event of diversification.

It is astonishing how transparent the induction from organismic selection down to function of genes works. That is why some people think and teach that a gene centered approach to biology and evolution is the right scientific stand (Dawkins, 1976). Nevertheless, we are not ready to accept that. Instead, we gladly invite everyone to consider group and sexual selection, among others, to enrich the personal vision over evolution (Mayr, 1997).

2.3 Conclusion

To solve a problem means to find a solution. Most problems can be solved by evolution because to find a solution can be simply understood as to find the best approximated solution. Evolution becomes the iteration of superposition of a selection procedure over a source of variability. Automatic iteration is implemented by means of reproduction. The overall procedure is called a genetic algorithm. In some cases, the selection that works at a high organismic level can be reduced to selection to genes and it is this way that genes acquieres an evolutionary function.

This assumes the validity of the system paradigm: divide a complex system into tractable subsystems to next merge their behavior to compute the overall response.

Chapter 3

Lamarck and Darwin

`Two flavors of evolution`

26 ***Objective.*** *Here we claim that the merit of Lamarck was twofold: to invent the evolutionary idea, that species evolve, and to propose a solution to the problem of making evolution into a reliable, believable theory: all one needs to do is to twist randomness to accelerate evolution. That it is really possible is a fact that every successful simulation of evolution to solve complex problems shows. Nevertheless, believable and realizable theories are not necessarily the correct explanation of nature. Thus, the proposed by him mechanism to accelerate evolution by means of inheritance of acquired characters was shown to be false. By contrast, Darwin proposes that natural selection of variants are all nature needs to explain itself. His theory is simple, direct and plainly scientific and that is why we study it.*

3.1 The transformism of Lamarck

I was taught to consider Lamarck as an historical error that was corrected by Darwin. I blame the culture that educated me for this wrong formation. To begin with, the modern idea of evolution, of the transformation of species by natural and mechanistic phenomena, comes from Lamarck:

Aussi l' on peut assurer que, parmi ses productions, la nature n' a réellement formé ni classes, ni ordres, ni familles, ni genres, ni espèces constantes, mais seulement des individus qui se succèdent les uns aux autres, et qui ressemblent à ceux qui les ont produits. (Philosophie zoologique, Pag 46 in the pdf numeration, 1809)

Next, he made his best effort to sustain this ideal from the natural point of view. Nevertheless, his ideas was covered by oblivion and disdain mainly because

he relied on alchemy precisely when modern chemistry was impressing everyone with its power.

27 ***Helping evolution***

According to Lamarck, reproduction plus the local effect of circumstances are the key process that must explain evolution. Nevertheless, his academic formation and continuous study of nature, imposes on him the heavy load of explaining the anatomic and behavioral complexity of living beings. His solution was to help reproduction with a mechanism to rapidly gathering complexity: the inheritance of acquired characters.

28 ***Darwin***

Adopting the transformism of Lamarck and the belief in the inheritance of acquired characters, Darwin added the crucial role of natural selection as the main mechanism of adaptation that was also useful to gather complexity and to create diversity.

Further scientific development has shown that a very simple and robust evolutionary theory can be formulated by resorting to *reproduction with mutation + recombination and with natural selection of the fittest*.

29 ***Darwinian vs Lamarckian evolutions***

We voice that Darwin and Lamarck proposed two flavors of evolution that can be implemented in simulations and differentiated in a clear cut fashion:

Mutation is in Darwinian evolution at random over the alphabet that the evolutionary environment is built upon. In Lamarckian evolution, mutation is guided from above in such a way that mutation is not random over that alphabet. The next program implements a Lamarckian version of the program *syntax*. In it, we use the random generator to offer mutation at random, but we pose a filter: we accept and execute only those mutations that are good to solve the problem at hand. Specifically, since our purpose is to produce an ordered version of the number from 0 to 7, we accept those mutations that favor the demanded ordering but not others.

30 ***A Lamarckian version of the program*** **`syntax`.** *Mutations is initially proposed at random, but they are filtered: only those mutations that produce progress are allowed to be part of an individual. In a well designed Lamarckian evolutionary environment, progress is unavoidable but it is a nice surprise to see the form as that actually happens.*

The code follows:

```
//Program G30 syntax3
//We mimic a game that a teacher
//proposes to her pupils:
//A disordered phrase is given to pupils.
//A disordered phrase is just a permutation of
//the original one, a variation of order of its words.
//They must restore from this
//string the original sentence
//with correct meaning and syntax.
//The problem is solved by means of LAMARCKIAN EVOLUTION,
//in which  mutation over genes is guided to produce
//the desired output.
//In this case, fitness is given by the teacher,
//who already knows the answer and can estimate
//how good are the attempts of the pupils.

//To disorder the original correct phrase,
//a series of swaps or transpositions is done.
//The best way to correctly reorder the
//given disordered phrase is to use also swaps.


import java.util.Random;

public class syntax3
{
 private static String phrase =
  "syntax deals with the formation of correct sentences";
 private static int nLetters = phrase.length();
 private static int nWords;
 //The phrase is decomposed in words
 private static String[] Word = new String[10];
 //Words are numbered. A disordered phrase is encoded
 //in a vector of digits, Permutation[], such that
 //Permutation[i] indicates what word goes
 //at place i of the possibly disordered phrase.
 private static int[] Permutation = new int[10];
 //Turn on of the random generator
```

```
 private static Random r  = new Random();

 //======Genetic part of declaration of variables=====
 // Individuals are kept in the array
 // Individual[]. It is an array of strings.
 //Each individual encodes
 //the permutation that it represents.
 //The individual is a string that encodes for
 //a number.


 // The number of individuals must be
 // less than limit.
 static double zMax; static double N;
 static int limit = 50000;
 static double Fitness[];
 static String Individual[ ], Individualc [];
 //Actual number of individuals
 static int nIndiv ;
 static int Order[];
 static String b;
 static int generation;
 static  int  nGen;
 static int ReportMin[], ReportMax[];
 static double oldError, newError;
 static double deltaError;
 static boolean print,testEncoding;
 //Our observable that declares success.
 static boolean done = false;
 static boolean recombination;
 static double ERROR, FITNESS;
 static String SOLUTION;
 static int GENERATION;

//Words are recognized: they are separated by a space.
//They are kept in a vector.
private static void numberWords(String s)
 {
  int counter = 0;
  String f = "";
```

```
  for(int i = 0; i < nLetters; i++)
   {
char d = s.charAt(i);
//System.out.println(d);
if (!(d == ' '))  f = f+d;
else
 {
  Word[counter] = f;
  f = "";
  counter++;
 }
//last word
if (i == nLetters-1) Word[counter] = f;
   }
  //System.out.println( counter);
  System.out.println("\nThe words of the phrase " +
                       "with their indexes: ");
  nWords = counter +1;
  for(int i = 0; i < nWords; i++)
   {
System.out.println(i + "  " +  Word[i]);
Permutation[i] =  i;
   }
 }

 //A transposition  is the  interchange of
 //place between two items.
private static void transposition()
 {
  int m = r.nextInt(nWords);
  boolean flag = false;
  //Sites must be different
  int n=0;
  while(!(flag))
   { n = r.nextInt(nWords);
   if (!(n == m)) flag = true;
   }
  if (print) System.out.println("Permuation of "
                               +  m + " and " + n);
  int k = Permutation[m];
```

```
  Permutation[m] = Permutation[n];
  Permutation[n] = k;
 }

//Mixer:  a sequence of transpositions creates variability.
//The output is encoded in a string, a chromosome.
private static String mixer(int numberOfTransp)
 {
  for(int i= 0; i < numberOfTransp; i++ )
 transposition();
  String v = "";
  for(int i = 0; i < nWords; i++)
    {
  Integer e = Permutation[i];
  v = v + e.toString();
    }
  return v;
 }



//Words in the sentence encoded by s are permuted

private static String permutate (String s)
 {
  numberWords(s);
  //Number of transpositions
  int n =  nWords;
  return  mixer(n);
 }

//String number  is decoded into a phrase
private static String stringToPhrase(String number)
  {
   String phrase= "";
   for(int j=0;j  < number.length();j++ )
    {
     char s3  =   number.charAt(j);
    int  l = Character.getNumericValue(s3);
    phrase = phrase + Word[l] + " ";
```

```
    }
   return phrase;
  }

//Declarations and default initializations
// of other arrays.
private static void otherInit( )
{
 Order= new int[limit+1];
 ReportMin= new int[limit+1];
 ReportMax= new int[limit+1];
 Fitness= new double[limit +1];
 for(int i = 0; i< nIndiv; i++)
  {
   Order[i]=0;
   ReportMin[i]=0;
   ReportMax[i]=0;
   Fitness= new double[limit +1];
 }
}

//An individual is a string that is a permutation
//of another one.
private static String generateIndividual( )
{
 //Number of transpositions
 int numberOfTransp = r.nextInt(nWords);
 String ind = mixer(numberOfTransp);
 return ind;
}

/* We generate nIndiv individuals (strings)
 encoding for permutations of 01234567 */
private static void Initialization( )
{
 //Formal declaration of our array.
 Individual= new String[limit];
 if (print)  System.out.println("ORIGINAL POPULATION");
 for(int i = 0; i< nIndiv; i++)
```

```
 {
  if (print) System.out.println( "\ni = " + i);
     Individual[i]="";
 // Individual[i] is a string,
 // it is a genotype that encodes a number,
 //its phenotype.
     Individual[i] = generateIndividual();
     if (print)
      {
       System.out.println(" = " + Individual[i] );
      }
    }
 otherInit();
  }



//This method decodes the string into the entries
//of permutation[]
private static void stringToVector(String s)
{
 for(int j = 0; j  < nWords;j++ )
  {
char c  =   s.charAt(j);
    int  l = Character.getNumericValue(c);
Permutation[j] = l;
  }
}

//The error of   chromosome s is minus
//the number of wrong pairwise orderings.
//Say, 54 is wrong and obtains a penalty of -1.
private static double error(String s)
{
 stringToVector(s);
 int counter = 0;
 int a,b ;

 for(int j = 0; j  < nWords-1;j++ )
```

```
 {
char c  =   s.charAt(j);
    a = Character.getNumericValue(c);
     c  =   s.charAt(j+1);
    b = Character.getNumericValue(c);
    if (a > b ) counter--;
    //System.out.println( a + " " + b + " " + counter);
 }
 return counter;
}


//This fitness of each individual is found
private static void fitness(int gen)
{
 //We measure the error of individual[i]
 //with respect to the perfect one 01234567.
 //We adopt a gene insight.
 for(int i = 0; i< nIndiv; i++)
  {
   double  error = error(Individual[i]);

   Fitness[i] = error;
   if (print)
  System.out.println("i = " + i
  + " Individual = " + Individual[i]
  + " Error  = "+ error
  + " Fitness"  + Fitness[i]    );
  if (error > - 0.000000000001)
   {
 GENERATION = gen;
 ERROR = error;
 SOLUTION = Individual[i];
 FITNESS =  Fitness[i];
 done = true;
}
  if (print)  System.out.println();
  }
 }
```

```
//Individuals are sorted by fitness
private static void Sorting(int gen)
  {
   fitness(gen);
   if (print) System.out.println("\nSORTING");
   int   Champ;
  //Sorting by fitness.
   //Minimal fitness = - 6. Maximal 0.
  for(int i = 0; i< nIndiv;i++)
{
 Champ = 0;
 for(int j = 0; j< nIndiv;j++)
    if (Fitness[j] >= Fitness[Champ])   Champ = j;
     //The array Order classifies individuals by  fitness
  // by equal or decreasing order.
  Order[i] = Champ;
     if (print)
      {
       System.out.println(  i + "th ind. is  No "
    + Champ + " " + Individual[Order[i] ]
           + " Fitness = "  + Fitness[Champ]);
      }
     Fitness[Champ] = -10;
     }

String t = Individual[Order[0] ];
System.out.println(" Best solution "  + t  +
" -> " + stringToPhrase(t));
  }



//Each individual of the top ten
// produces 10  copies.
private static void Reproduction()
{
 if (print)
  System.out.println( "Reproduction");
 if (print)
  System.out.println( "The best = " + Order[0]);
```

```
 Individualc= new String[limit];
 int counter = 0;
 for (int top = 0; top < 10; top++)
 {
 for(int j = 0; j< 10; j++)
  {
    Individualc[counter] = Individual[Order[top]];
counter = counter +1;
  }
 }
 for(int j = 0; j< counter; j++)
Individual[j] = Individualc[j];

}

//Lamarckian mutations:
//originally random mutations are at random
//but they are filtered and admitted only if
//they accelerate evolution.
//Our mutations implement swaps, transpositions.
//A transposition  is the  interchange of
//place between two items.
private static void transpositionL()
{
 boolean again = true;
 if (done) again = false;
int m = r.nextInt(nWords-1);
 int n = m+1;
 //Let us keep in mind that
 //the perfectly ordered phrase is encoded as 01234567.
 //Suppose we have the phrase 01235467 and we are checking
 //whether nor not the digit 5 is worth be exchanged
 //with digit 4. The answer is affirmative because
 //the present situation generates a strain that is relaxed
 //by a swap. But not with other digits.
 if (Permutation[m] > Permutation[m+1])
 {
  if (print) System.out.println("Permuation of "
                               +  m + " and " + n);
  int k = Permutation[m];
```

```
  Permutation[m] = Permutation[n];
  Permutation[n] = k;
  again = false;
 }
 if (again) transpositionL();
}

//A a sequence of transpositions creates variability.
private static String mixWords(String s,
                                int numberOfTransp)
 {
  //Chromosome s is transformed into a vector
  stringToVector(s);
  //Entries in vector are transposed
  for(int i= 0; i < numberOfTransp; i++ )
 transpositionL();
  //Vector is encoded into a chromosome, a string.
  String v = "";
  for(int i = 0; i < nWords; i++)
  {
  Integer e = Permutation[i];
  v = v + e.toString();
  }
  return v;
 }


//A mutation results from a series of transpositions
private static void Mutation()
{
if (print) System.out.println("MUTATION");
for(int j = 0; j<  nIndiv; j++)
{
 if (print) System.out.println("Ind = " + j
+ " " + Individual[j]);
 int n = 1;
 Individual[j] = mixWords( Individual[j],n);
}
}
```

```
//Two strings recombine and produce two offspring.
private static void Recombination()
{
 for(int j = 50; j<  nIndiv; j++)
  {
//Define place of recombination
int m =  r.nextInt(nIndiv);
int n = r.nextInt(nIndiv);
String a = Individual[m];
String b = Individual[n];
int placeRec = r.nextInt(nWords);
Individual[m] = a.substring(0, placeRec)
               + b.substring( placeRec);
Individual[n] = b.substring(0, placeRec)
       + a.substring( placeRec);
 }
}

//Master for a generation
private static void dynamics(int gen)
 {
  //Individuals are sorted by fitness
  Sorting(gen);
  //The top ten are preferentially reproduced
  Reproduction();
  //The new population is subjected to mutation
  Mutation();
  if (recombination) Recombination();
 }

//The original ordered phrase is rebuilt
//by a genetic algorithm.
//This program simulates a teacher that knows
//the answer and guides their students
//for a better solution.
private static void restorePhrase()
 {
  System.out.println("\n\nRunning ");
  nIndiv = 1;
  Initialization( ) ;
```

```
  nGen = 200000;
  oldError = 1000;
  for(int gen = 1; ( gen <= nGen) & (!(done)); gen++)
 {
   System.out.print("\nGENERATION = " + gen);
   dynamics(gen);
 }
  if (done)
   {
 System.out.println("\nGENERATION = " + GENERATION +
       " SOLUTION FOUND");
     System.out.println("ERROR = " + ERROR);
     System.out.println("SOLUTION = " + SOLUTION
      + " -> " +  stringToPhrase(SOLUTION) );
     System.out.println("FITNESS = " + FITNESS);
     System.out.println("Number of generations = "
      + GENERATION);
     System.out.println("Number of individuals " +
      "per generation = " + nIndiv);
     int totalCost = GENERATION * nIndiv;
     System.out.println("Number of Generations x " +
      "number of individuals = " + totalCost);
   }
 }

public static void main(String[] args)
 {
  print = false;
  recombination = false;
  System.out.println("This program disorders a sentence"
         + " \nand reorders it by evolution "
         + "\nusing random transpositions."
         + "\nMutation is guided to swiftly produce"
         + "\ndesired result. Evolution is Lamarckian.\n");
  System.out.println("ORIGINAL SENTENCE: ");
  System.out.println(phrase);
  String p = permutate(phrase);
  System.out.println( "\nCHALLENGING DISORDERED SENTENCE" +
   " WITH ITS INDEXATION : \n" );
  System.out.println( p + " -> " + stringToPhrase(p) );
```

```
   restorePhrase();
 }//End of main method
}
```

31 ***Exercise.*** *Run the program and play with the code.*

3.2 Darwinian evolution

Evolution may be imagined as a walk along a landscape. If you have a guiding instrument, you will give a step only in the right direction towards the summit. That is evolution according to Lamarck. But, is the instrument necessary? If it is optional, what is the election of nature to build complexity and perfection? The answer to these questions given by Darwin is astonishing:

32 ***A simple, clear and powerful mechanism***

Various formulations of the evolutionary theory are possible. The next one could be the most transparent:

1. Dead is the nearest reality to life. So, dead must be at the very core of any realistic theorization about life.

2. The remedy against dead is reproduction. That is why sex is so important for every living being. Like begets like but not exactly, rather organisms are born different in many ways and so they respond differently to the challenges given by the environment. Thus, life is a machine that creates variability, which is intrinsically random, i.e., variability is not directed to fit any particular requirement.

3. All living beings die but not at the same time neither for the same reason. The existence of small differences in dead rates that correlate with qualities of involved organisms is called selection.

4. The recursive effect of selection over generations gathers small change (Dawkins, 1986) and produces adaptation to the milieu, perfection and beauty.

The emphasis on small change and small differences point to a multivariate universe. Nevertheless, one usually works with univariate models in which simple functions are considered. Take, for instance, the function exonuclease, which is implemented by the corresponding enzyme, whose task is to cleave foreign DNA. With a high probability, the lack of that function leads to dead and extinction and

as the function improves, surviving gets more probably. This gradation implies that an evolution from no function to function to the full is not excluded. But on the other side, this univariate model allows two forms of implementing selection:

- The first is negative, demographic with dead at its center: those individuals at the lower tail of the axis of exonuclease performance are chastised more severely than those at higher positions.
- The second is positive, functional and is designed to reward with more opportunities of reproduction those individuals that are in the upper tail of performance. Until now, we have chosen this second form of simulating selection: our simulations rely on reward.

The next exercise shows that it is indeed a difficult task to decide when evolution is Lamarckian else Darwinian.

33 ***Exercise.*** *Show that the first version of* `syntax`*, G19, which used transpositions, is also Lamarkian in spite of the fact that seemingly it lacks guiding instrumentation and that mutations were at random but over the set of transpositions.*

34 ***Challenge.*** *Imagine what could be of evolution without recombination. Prove else refute that recombination is a way to create variability that violates randomness. So, recombination is an evolutionary machinery that is aligned in the direction of Lamarck much better that along that of Darwin. Hint: help yourself by studying the molecular machinery that executes recombination and decide how probably is that it could had appeared by Darwinian evolution.*

Why do we use transpositions to solve our problem *syntax*? Because random mutations produce from 12536470 mutants as 12233432 which clearly are not in the direction of the solution to our puzzle which is 01234567. If we are correctly thinking, Darwinian evolution shall arrive to the solution using a greater number of generations than the Lamarckian version. Let us test this idea.

35 ***The code that solves* `syntax` *according to Darwinian evolution.***

```
//Program G35 syntax4

//We mimic a game that a teacher
```

```
//proposes to her pupils:
//A disordered phrase is given to pupils.
//A disordered phrase is just a permutation of
//the original one, a variation of order of its words.
//They must restore from this
//string the original sentence
//with correct meaning and syntax.
//The problem is solved by means of DARWINIAN EVOLUTION,
//in which selection is the driving force of the
//ensuing evolutionary process and
//mutation is random over the alphabet
//that is dictated by reproduction.
//Every phrase is encoded by a number
//
//In this case, fitness is given by the teacher,
//who already knows the answer and can estimate
//how good are the attempts of the pupils.
//Specifically, the fitness of a phrase is defined by
//number - 01234567
//where number is the number that encodes for the
//phrase at trial.


import java.util.Random;

public class syntax4
{
 private static String phrase =
  "syntax deals with the formation of correct sentences";
 private static int nLetters = phrase.length();
 private static int nWords;
 //The phrase is decomposed in words
 private static String[] Word = new String[10];
 //Words are numbered. A disordered phrase is encoded
 //in a vector of digits, Permutation[], such that
 //Permutation[i] indicates what word goes
 //at place i of the possibly disordered phrase.
 private static int[] Permutation = new int[10];
 //Turn on of the random generator
 private static Random r  = new Random();
```

```
 //Mutation rate per symbol per generation
 private static double mutRate;

 //======Genetic part of declaration of variables=====
 // Individuals are kept in the array
 // Individual[]. It is an array of strings.
 //Each individual encodes
 //the permutation that it represents.
 //The individual is a string that encodes for
 //a number.


 // The number of individuals must be
 // less than limit.
 static double zMax; static double N;
 static int limit = 50000;
 static double Fitness[];
 static String Individual[ ], Individualc [];
 //Actual number of individuals
 static int nIndiv ;
 static int Order[];
 static String b;
 static int generation;
 static  int  nGen;
 static int ReportMin[], ReportMax[];
 static double oldError, newError;
 static double deltaError;
 static boolean print,testEncoding;
 //Our observable that declares success.
 static boolean done = false;
 static boolean recombination;
 static double ERROR, FITNESS;
 static String SOLUTION;
 static int GENERATION;

//Words are recognized: they are separated by a space.
//They are kept in a vector.
private static void numberWords(String s)
 {
  int counter = 0;
```

```
  String f = "";
  for(int i = 0; i < nLetters; i++)
   {
char d = s.charAt(i);
//System.out.println(d);
if (!(d == ' '))  f = f+d;
else
 {
  Word[counter] = f;
  f = "";
  counter++;
 }
//last word
if (i == nLetters-1) Word[counter] = f;
   }
  //System.out.println( counter);
  System.out.println("\nThe words of the phrase " +
                       "with their indexes: ");
  nWords = counter +1;
  for(int i = 0; i < nWords; i++)
   {
System.out.println(i + "  " +  Word[i]);
Permutation[i] =  i;
   }
 }

 //A transposition  is the  interchange of
 //place between two items.
private static void transposition()
 {
  int m = r.nextInt(nWords);
  boolean flag = false;
  //Sites must be different
  int n=0;
  while(!(flag))
   { n = r.nextInt(nWords);
   if (!(n == m)) flag = true;
   }
  if (print) System.out.println("Permuation of "
                              +  m + " and " + n);
```

```
  int k = Permutation[m];
  Permutation[m] = Permutation[n];
  Permutation[n] = k;
 }

//Mixer:  a sequence of transpositions creates variability.
//The output is encoded in a string, a chromosome.
private static String mixer(int numberOfTransp)
 {
  for(int i= 0; i < numberOfTransp; i++ )
 transposition();
  String v = "";
  for(int i = 0; i < nWords; i++)
    {
  Integer e = Permutation[i];
  v = v + e.toString();
    }
  return v;
 }



//Words in the sentence encoded by s are permuted

private static String permutate (String s)
 {
  numberWords(s);
  //Number of transpositions
  int n =  nWords;
  return  mixer(n);
 }

//String number  is decoded into a phrase
private static String stringToPhrase(String number)
  {
   String phrase= "";
   for(int j=0;j  < number.length();j++ )
    {
     char s3  =   number.charAt(j);
    int  l = Character.getNumericValue(s3);
```

```
    phrase = phrase + Word[l] + " ";

    }
   return phrase;
  }

//Declarations and default initializations
// of other arrays.
private static void otherInit( )
{
 Order= new int[limit+1];
 ReportMin= new int[limit+1];
 ReportMax= new int[limit+1];
 Fitness= new double[limit +1];
 for(int i = 0; i< nIndiv; i++)
  {
   Order[i]=0;
   ReportMin[i]=0;
   ReportMax[i]=0;
   Fitness= new double[limit +1];
 }
}

//An individual is a string that is a permutation
//of another one.
private static String generateIndividual( )
{
 //Number of transpositions
 int numberOfTransp = r.nextInt(nWords);
 String ind = mixer(numberOfTransp);
 return ind;
}

/* We generate nIndiv individuals (strings)
 encoding for permutations of 01234567 */
private static void Initialization( )
{
 //Formal declaration of our array.
 Individual= new String[limit];
 if (print)  System.out.println("ORIGINAL POPULATION");
```

```
 for(int i = 0; i< nIndiv; i++)
 {
  if (print) System.out.println( "\ni = " + i);
     Individual[i]="";
 // Individual[i] is a string,
 // it is a genotype that encodes a number,
 //its phenotype.
     Individual[i] = generateIndividual();
     if (print)
      {
       System.out.println(" = " + Individual[i] );
      }
    }
 otherInit();
  }



//String s is decoded into a number
private static long recoverNumber(String s)
 {
  long number = 0;
  for(int j=0;j  < nWords;j++ )
   {
char c  =   s.charAt(j);

  int  l = Character.getNumericValue(c);
  int q = j+1;
  long number1 = (long) (l * Math.pow(10,nWords-q));
  number = number + number1;

   }
  return number;
 }

// This method transforms a string in a number
//of type long. Some  outputs to console are displayed.
private static long stringToNumber(String s)
  {
if (print)
```

```
  {
   System.out.println( "\nIndividual =  " + s);
   System.out.println( "length of s  = " + s.length());
  }

if (print) System.out.println( "Recovering number");
long k = recoverNumber(s);

if (print)  System.out.println("String "
    + s +  " as number = "       + k + "\n");
 return k;
 }

//This method decodes the string into the entries
//of permutation[]
private static void stringToVector(String s)
{
 for(int j = 0; j  < nWords;j++ )
  {
char c  =   s.charAt(j);
    int  l = Character.getNumericValue(c);
Permutation[j] = l;
  }
}


private static double fitness(double error)
{
  return  1/(1+error*error);
}


//This fitness of each individual is found
private static void fitness(int gen)
{
 //We measure the error of individual[i]
 //with respect to the perfect one 01234567.
 for(int i = 0; i< nIndiv; i++)
  {
   double  error =
```

```
   Math.abs(1234567-stringToNumber(Individual[i]));

   //The error defines the fitness: less error, more fitness.
   //Maximal fitness = 1; minimal = 0.
   Fitness[i] = fitness(error);
   if (print)
  System.out.println("i = " + i
  + " Individual = " + Individual[i]
  + " Error  = "+ error
  + " Fitness"  + Fitness[i]    );
  if (error < 0.000000000001)
   {
 GENERATION = gen;
 ERROR = error;
 SOLUTION = Individual[i];
 FITNESS =  Fitness[i];
 done = true;
}
  if (print)  System.out.println();
  }
 }

//Individuals are sorted by fitness
private static void Sorting(int gen)
  {
   fitness(gen);
   if (print) System.out.println("\nSORTING");
   int   Champ;
  //Sorting by fitness.
  for(int i = 0; i< nIndiv;i++)
{
 Champ = 0;
 for(int j = 0; j< nIndiv;j++)
    if (Fitness[j] >= Fitness[Champ])   Champ = j;
     //The array Order classifies individuals by  fitness
  // by equal or decreasing order.
  Order[i] = Champ;
     if (print)
      {
       System.out.println(  i + "th ind. is  No "
```

```
    + Champ + " " + Individual[Order[i] ]
            + " Fitness = "  + Fitness[Champ]);
      }
     Fitness[Champ] = 0;
     }

String t = Individual[Order[0] ];
System.out.println(" Best solution "  + t  +
" -> " + stringToPhrase(t));
  }



//Each individual of the top ten
// produces 10  copies.
private static void Reproduction()
{
 if (print)
  System.out.println( "Reproduction");
 if (print)
  System.out.println( "The best = " + Order[0]);
 Individualc= new String[limit];
 int counter = 0;
 for (int top = 0; top < 10; top++)
 {
 for(int j = 0; j< 10; j++)
  {
    Individualc[counter] = Individual[Order[top]];
counter = counter +1;
  }
 }
 for(int j = 0; j< counter; j++)
Individual[j] = Individualc[j];

}

//A a sequence of transpositions creates variability.
private static String mutate(String s)
 {
  //Chromosome s is transformed into vector Permutation[]
```

```
  stringToVector(s);
  //Mutate or not mutate
  double p = r.nextDouble();
  //Entries in vector are subject to mutation
  if ( p < mutRate )
   {
//Site of mutation is chosen at random
int site = r.nextInt(nWords);
//Effect of mutation is chosen at random
int change = r.nextInt(nWords);
Permutation[site] = change;
   }
  //Vector is encoded into a chromosome, a string.
  String v = "";
  for(int i = 0; i < nWords; i++)
  {
  Integer e = Permutation[i];
  v = v + e.toString();
  }
  return v;
 }

//A mutation is Darwinian:
//both the site of mutation and the change
//are defined by randomness.
private static void Mutation()
{
  for(int j = 1; j<  nIndiv; j++)
{
 Individual[j] = mutate( Individual[j]);
}
}



//Two strings recombine and produce two offspring.
private static void Recombination()
{
 for(int j = 50; j<  nIndiv; j++)
  {
```

```
//Define place of recombination
int m =  r.nextInt(nIndiv);
int n = r.nextInt(nIndiv);
String a = Individual[m];
String b = Individual[n];
int placeRec = r.nextInt(nWords);
Individual[m] = a.substring(0, placeRec)
               + b.substring( placeRec);
Individual[n] = b.substring(0, placeRec)
       + a.substring( placeRec);
 }
}

//Master for a generation
private static void dynamics(int gen)
 {
  //Individuals are sorted by fitness
  Sorting(gen);
  //The top ten are preferentially reproduced
  Reproduction();
  //The new population is subjected to mutation
  Mutation();
  if (recombination) Recombination();
 }

//The original ordered phrase is rebuilt
//by a genetic algorithm.
//This program simulates a teacher that knows
//the answer and guides their students
//for a better solution.
private static void restorePhrase()
 {
  System.out.println("\n\nRunning ");
  nIndiv = 150;
  Initialization( ) ;
  nGen = 500000;
  oldError = 1000;
  for(int gen = 1; ( gen <= nGen) & (!(done)); gen++)
 {
   System.out.print("\nGENERATION = " + gen);
```

```
   dynamics(gen);
 }
  if (done)
   {
 System.out.println("\nGENERATION = " + GENERATION +
       " SOLUTION FOUND");
     System.out.println("ERROR = " + ERROR);
     System.out.println("SOLUTION = " + SOLUTION
      + " -> " +  stringToPhrase(SOLUTION) );
     System.out.println("FITNESS = " + FITNESS);
     System.out.println("Number of generations = "
      + GENERATION);
     System.out.println("Number of individuals " +
      "per generation = " + nIndiv);
     long totalCost = GENERATION * nIndiv;
     System.out.println("NUmber of Generations x " +
      "number of individuals = " + totalCost);
   }
 }

public static void main(String[] args)
 {
  print = false;
  recombination = false;
  mutRate = 0.5;
  System.out.println("This program disorders a sentence"
         + " \nand reorders it by evolution "
         + "\nusing random mutations over the"
         + "\nalphabet 0,1,2,3,4,5,6,7."
         + "\nEvolution is Darwinian.\n");
  System.out.println("ORIGINAL SENTENCE: ");
  System.out.println(phrase);
  String p = permutate(phrase);
  System.out.println( "\nCHALLENGING DISORDERED SENTENCE" +
   " WITH ITS INDEXATION : \n" );
  System.out.println( p + " -> " + stringToPhrase(p) );
   restorePhrase();
 }//End of main method
}
```

36 ***Exercise.*** *Run the program and play with the code. Turn on and off recombination.*

37 ***Exercise.*** *Verify else refute the following conclusions of the Author in regard with Darwinian evolution:*

1. *Evolution that relies on random mutation but without recombination arrives to the solution but in the generality of cases very late with respect to our best Lamarckian version (program G30).*

2. *Evolution that relies on random mutation plus recombination arrives to the solution and can compete with some Lamarckian versions but not with the best (G30).*

38 ***Challenge.*** *Add to your solution to the previous exercise a statistical justification that must be calculated by the program itself.*

39 ***Saving our honor***

If one is able to program evolution to solve a problem, the immediate duty is to verify that the personal honor is in high. To that aim, one must test whether or not chosen implementation is better than chance. While one usually does this by some mathematical reasoning, let us illustrate how one programs chance to solve the given problem. The code follows:

```
//Program G39 syntax5


//We mimic a game that a teacher
//proposes to her pupils:
//A disordered phrase is given to pupils.
//A disordered phrase is just a permutation of
//the original one, a variation of order of its words.
//They must restore from this
//string the original sentence
//with correct meaning and syntax.
//The problem is solved by means of CHANCE alone.
//In this case, fitness is given by the teacher,
//who already knows the answer and can estimate
//how good are the attempts of the pupils.
```

```
import java.util.Random;

public class syntax5
{
 private static String phrase =
  "syntax deals with the formation of correct sentences";
 private static int nLetters = phrase.length();
 private static int nWords;
 //The phrase is decomposed in words
 private static String[] Word = new String[10];
 //Words are numbered. A disordered phrase is encoded
 //in a vector of digits, Permutation[], such that
 //Permutation[i] indicates what word goes
 //at place i of the possibly disordered phrase.
 private static int[] Permutation = new int[10];
 //Turn on of the random generator
 private static Random r  = new Random();
 private static boolean print;



//Words are recognized: they are separated by a space.
//They are kept in a vector.
private static void numberWords(String s)
 {
  int counter = 0;
  String f = "";
  for(int i = 0; i < nLetters; i++)
   {
char d = s.charAt(i);
//System.out.println(d);
if (!(d == ' '))  f = f+d;
else
 {
  Word[counter] = f;
  f = "";
  counter++;
 }
```

```
//last word
if (i == nLetters-1) Word[counter] = f;
   }
  //System.out.println( counter);
  System.out.println("\nThe words of the phrase " +
                      "with their indexes: ");
  nWords = counter +1;
  for(int i = 0; i < nWords; i++)
   {
System.out.println(i + "  " +  Word[i]);
Permutation[i] =  i;
   }
 }

 //A transposition  is the  interchange of
 //place between two items.
private static void transposition()
 {
  int m = r.nextInt(nWords);
  boolean flag = false;
  //Sites must be different
  int n=0;
  while(!(flag))
   { n = r.nextInt(nWords);
   if (!(n == m)) flag = true;
   }
  if (print) System.out.println("Permuation of "
                              +  m + " and " + n);
  int k = Permutation[m];
  Permutation[m] = Permutation[n];
  Permutation[n] = k;
 }

//Mixer:  a sequence of transpositions creates variability.
//The output is encoded in a string, a chromosome.
private static String mixer(int numberOfTransp)
 {
  for(int i= 0; i < numberOfTransp; i++ )
 transposition();
  String v = "";
```

```
  for(int i = 0; i < nWords; i++)
    {
  Integer e = Permutation[i];
  v = v + e.toString();
    }
  return v;
 }



//Words in the sentence encoded by s are permuted

private static String permutate (String s)
 {
  numberWords(s);
  //Number of transpositions
  int n =  nWords;
  return  mixer(n);
 }

//String number  is decoded into a phrase
private static String stringToPhrase(String number)
  {
   String phrase= "";
   for(int j=0;j  < number.length();j++ )
    {
     char s3  =   number.charAt(j);
    int  l = Character.getNumericValue(s3);
    phrase = phrase + Word[l] + " ";

    }
   return phrase;
  }

//Numbers that encode phrases
//are generated at random
//and compared with the target.
//Phrases are decoded and publicized.
private static void matchPhrase()
 {
```

```
  System.out.println("\n\nRunning ");
  boolean go = true;
  int i = 0;
  Integer n = 1234567;
  String target = n.toString();
  target = "0" + target;
  String trial = "";
  Integer k;
   while (go)
 {
   i++;
   System.out.print("\n i  = " + i);
   //Generate at random a possible solution
   for(int j = 0; j < nWords; j++)
   {
 k = r.nextInt(nWords);
    trial =  trial + k.toString();
   }
   System.out.print("\n i  = " + i);

   System.out.print(" trial  = " + trial
 +  " -> " + stringToPhrase(trial) );
   //If the solution is found, halt
   if (trial == target) go = false;
   trial = "";

 }

 System.out.println("\nAttempt = " + i +
       " SOLUTION FOUND");

     System.out.println("Number of attempts = "
      + i);
     System.out.println("Number of individuals " +
      "per attempt = 1 " );
 }

public static void main(String[] args)
 {
  print = false;
```

```
  System.out.println("This program disorders a sentence"
        + " \nand reorders it by chance alone. ");
  System.out.println("ORIGINAL SENTENCE: ");
  System.out.println(phrase);
  String p = permutate(phrase);
  System.out.println( "\nCHALLENGING DISORDERED SENTENCE" +
   " WITH ITS INDEXATION : \n" );
  System.out.println( p + " -> " + stringToPhrase(p) );
   matchPhrase();
 }//End of main method
}
```

40 ***Exercise.*** *Run the program and play with the code. Agree with else refute the conclusion of the Author: for this problem, chance alone does not overcome Darwinian evolution. If that is the case, our honor has been saved.*

3.3 Conclusion

Behind the erroneous mechanism of Lamarck proclaiming the inheritance of acquired characters, we find a simple and powerful idea that is at the very core of applied evolution, such as in genetic engineering: randomness can be twisted to accelerate the race of evolution towards perfection. The use of filters and suitable super-alphabets is common to twist randomness (use bricks rather than atoms or molecules to build a house). The modern version of the Darwinian Evolutionary Theory proposes a mechanism that also enables perfection but that needs no twisting because it relies on variation created at randomness. A positive formulation of the mechanism reads: those variants that are nearer to the functional optimum are given more opportunities of reproduction. Or if one prefers a negative, demographic insight, one could also say: the existence of small differences in dead rates that correlate with qualities of involved organisms is called selection and produces a trend towards adaptation, perfection and beauty.

Chapter 4

Complexity as judge

`A severe, authorized referee.`

41 ***Objective.*** *The idea of evolution is so natural and embedded in our lives that our simulations of evolution could look futile or at least redundant. Anyway, we have made sure that evolution functions. And given that its power is so obvious, a direct question arises: which are the limits, if any, of evolution? To answer this question we need a referee that must be endowed with a fair insight. Our choice is as follows: evolution solve problems and problems are difficult. So, the natural referee of evolution is complexity. It is the referee. And where is its insight? It arises naturally from the molecular biology of the gen: the genome is software and evolution is a software developer. We claim that this insight is both necessary and enough to deal with all important scientific questions related to evolution.*

4.1 Landscapes

Evolution can be understood graphically thanks to landscapes.

42 ***Driving through a landscape***

In *evolution*, real or simulated, we always hang over selection, which gives to every individual a score that measures how good it does in regard with solving the posed problem. In nature, the posed problem is to survive and to leave offspring. The fitness function always can be recast in such a way that the overall problem can be understood as one of climbing a mountain to attain its highest peak. The floor over which the mountain rises is the space of genomes and the height of the mountain over that point or genome is exactly the fitness function evaluated at it. The higher is the elevation at a point, the higher is the fitness of the corresponding

genome. It helps to imagine that the mountain is dressed in mist son that one cannot see beyond own nose although one can determine when one rises or when one goes down. This means that according to Darwin, evolution is not guided by an overall plan but it is instead a local phenomenon in which mutations are blindly produced but their fruits are assessed and selected for the desired qualities.

The difference between applied and natural evolution is that the landscape in the applied case is given by the designer while in natural evolution the landscape arises from the interaction of the genome with its environment.

Landscapes are perfect to depict optimization problems, those in which an optimum is searched for. The optimum can be a maximal or a minimal value. So, we imagine problems in which a peak must be reached and others in which the maximal depression must be located. Problems are difficult so one does not expect to find the exact optimum but only a good approximation. The interesting point is that many, many problems can be stated as optimization problems. Consider, for instance, the problem of solving an equation, or for that matter, a system of equations, say, in partial derivatives. All those problem read: *find x such that* $f(x) = y$. This problem can be reformulated as follows: *find x such that* $f(x) - y = 0$, which in turn can be restated as *find x such that* $f(x) - y$ *attains an object with the minimum length or norm.* This last problem is an optimization one, in which has not mentioned the constraints associated with initial and border conditions.

With constructions like this evolution is shown to be an ordinary part of our lives and also of the finest engineering. In short, evolution is an ordinary tool to solve every kind of problems. But problems are difficult. That is why we can earn a life in this world. It is now good to think of mountain climbers and of excursionists that get lost in a desert... So, complexity is the natural referee of evolution to assess its limits, if any. But that referee must have a correct insight to take a decision. The appropriate vision has arrived to us through a very lengthy and complex process. It has two main points. The first is that the genome is software and the second is that evolution is a software developer (Rodríguez, 2010).

43 *The genome is software*

We can define **software** as the generality of verbal instructions that convey information to solve problems and that is decoded by means of a compilation table that assigns specific actions to verbal tokens or subunits. It is immediate to catch up why the genome is software:

The genome of our cells is composed of DNA which is arranged in chromosomes, which are very long strings composed with letters of the alphabet given by $\{A,T,C,G\}$. So, the genetic information looks something like:

$...TTACTCGATCGTTCAGCT...$

But, why do we call such a thing genetic information? Where is its information? In modern terms, the answer is succinct and clear: the genome is software. Expanding, we have:

1. The genome conveys instructions that are verbal, i.e., than can be depicted with letters of the alphabet $\{A,T,C,G\}$.

2. A string of the form $...TTACTCGATCGTTCAGCT...$ means nothing in itself but acquires a meaning if it is considered together with a code to decipher all strings that can be formed with the alphabet. That code is the genetic code that associates amino acids to codons, which are triplets of letters, say *AAA* is associated with *lysine*. Now, this code is verbal and can be redefined at pleasure if only one pays attention to the underlying mechanism. Actually, there are some natural variants of the genetic code that seem to be softly explained by evolution.

3. So, the genome contains information to assemble proteins, agglomerates of linear chains of amino acids. Some proteins fill in structural needs, such as shelter in the walls of a cell. Others fulfill enzymatic functions, say, form sophisticated lines of work that digest sugar to produce energy in the form of ATP.

4. Looking from above, the genome contains very specific, verbal instructions to solve problems. Those instructions are compiled, deciphered and executed, automatically by the ribosome with the help of t-RNAs. That is why **the genome is software**.

44 ***Evolution is a software developer***

Let us consider our program G35, pag 58, in which we solved *syntax* thanks to Darwinian evolution. In hindsight, the program composes the appropriate verbal information, the software, to order a set of words to form a syntactically correct phrase that makes meaning in English. The program begins its work with strings assembled at random and uses the information given by the environment, which simulates a teacher that knows the answer, to arrive to the solution.

It would be interesting to make explicit the relation of aforementioned program with the cellular computing machine: Our strings are of the form 1324567. This strings make sense when a code is provided. In this case, the code associates a word to each letter: letter 0 is associated with the word *syntax*. So, to convert a string with 7 letters into phrases, we convert the string (the genotype that suffer mutations) into a chain of seven digits which functions like a number (the phenotype that is subject to selection) and replace the digit with a word according to the

appropriate table which contains our genetic code or compilation table. All decoding procedures that convert strings into a chain of digits simulate the ribosome. And the ontogenic process is simulated by the overall conversion of strings into phrases.

Thus, we have proven that evolution can synthesize software albeit of very low complexity. What can we expect when complexity is increased?

4.2 Bugs of evolution

The most direct experience of every software developer is that software design is an extremely complex affair. What does this mean?

45 ***Bugs of evolution***

In operational terms, the complexity of software design is reflected in two inviolable laws:

1. No one can compose a piece of software, even a tiny one, without committing errors, **bugs**.
2. There is a long path that every developer must follow with sweat and tears from his or her original situation to perfection. This is valid when one begins from scratch as when one reuses old software.

The evolutionary version of these laws can be elaborated as follows (after Vol 5 of this series):

Consider the case of artificial selection of race horses in which an evolutionary trend towards extreme and specialized perfection is attempted. Let us clearly depict where are the bugs and where are the tracks of an evolutionary process toward complexity and perfection.

When specialized perfection is looked for, the evolutionary pattern is always the same: the evolutionary trend is fast and cheap at the beginning, but then it slows down and finally stops. The character of being a bug is determined here by a fault against the evolutionary trend dictated by the selective criterion, which in the case of race horses is to be faster than everybody. So, we have a bug each time that appears an offspring that is slower than their parents and this is the rule when the evolutionary trend approaches stabilization.

We have therefore abundant bugs in the evolutionary process that is awaken by artificial selection, but where are the left tracks corresponding to an evolution

towards perfection? In general they do not appear in the fossil record: the differences between perfection, good quality and mediocrity are many times a matter of coordination of small details and not of great changes. Is this a falsification of our law of abundant bugs and tracks of an evolution towards perfection? No. Let us explain this with some detail.

Evolution comprises two levels (that might be intertwined). The first is the design of the evolutionary environment with the combinatorial basis and the rules for reproduction, mutation, recombination and selection. And the second, is evolution properly, the evolutionary process, in which the elements of the combinatorial basis are blindly recombined to fit better and better solutions. Experience shows that

- The design of the evolutionary environment is an extremely difficult task because one can easily end with too poor solutions to targeted problems.
- Once one has an evolutionary environment, the ensued evolution has no magic, rather it obeys the ordinary laws of computation, i.e., if a task is feasible, it takes time. So there are easy problems that can be resolved in some few blind computations. But there are complex problems whose solutions demand huge amount of computations. In this last case, one can see that local optima are persistently visited and so they are sure candidates to appear in any process of random sampling. These are the tracks we speak about, thanks to which we can trace an evolution towards high complexity and perfection. Local optima is what defines horses that are valued in 500, 1000, 10000, 1000000 or million of dollars. Their fossils will be almost indistinguishable for most people.

Let us now see how the bookkeeping of bugs is implemented. To that aim, we add to a previous program a piece of code to show the behavior of some observables related with bugs.

46 ***Code with bookkeeping of bugs.***

```
//Program G46 syntax6
//This program shows one very simple and
//inexorable law: if evolution can indeed find the answer,
//a price must be paid: a lot of bugs must be made,
//and therefore,
//a clear path towards perfection must exist.
//We show here
```

```
//how long one is frozen in a local minimum and
//how many bugs evolution makes.
//We have a BUG each time that appears
//an offspring that is less fitted than their parents.
//
//Problem to solve:
//We mimic a game that a teacher
//proposes to her pupils:
//A disordered phrase is given to pupils.
//A disordered phrase is just a permutation of
//the original one, a variation of order of its words.
//They must restore from this
//string the original sentence
//with correct meaning and syntax.
//The problem is solved by means of DARWINIAN EVOLUTION,
//in which selection is the driving force of the
//ensuing evolutionary process and
//mutation is random over the alphabet
//that is dictated by reproduction.
//In this case, selection is given by the teacher,
//who already knows the answer and can estimate
//how good are the attempts of the pupils.
//Specifically, the fitness of a phrase is defined by
//number - 01234567
//where number is the number that encodes for the
//phrase at trial.


import java.util.Random;

public class syntax6
{
 private static String phrase =
  "syntax deals with the formation of correct sentences";
 private static int nLetters = phrase.length();
 private static int nWords;
 //The phrase is decomposed in words
 private static String[] Word = new String[10];
 //Words are numbered. A disordered phrase is encoded
 //in a vector of digits, Permutation[], such that
```

```
//Permutation[i] indicates what word goes
//at place i of the possibly disordered phrase.
private static int[] Permutation = new int[10];
//Turn on of the random generator
private static Random r  = new Random();
//Mutation rate per symbol per generation
private static double mutRate;

//=====Genetic part of declaration of variables==========
// Individuals are kept in the array
// Individual[]. It is an array of strings.
//Each individual encodes
//the permutation that it represents.
//The individual is a string that encodes for
//a number.


// The number of individuals must be
// less than limit.
static double zMax; static double N;
static int limit = 50000;
static int Fitness[], FitnessCopy[];
static String Individual[ ], Individualc [];
//Actual number of individuals
static int nIndiv ;
static int Order[];
static String b;
static int generation;
static  int  nGen;
static int ReportMin[], ReportMax[];
static double oldError, newError;
static double deltaError;
static boolean print,testEncoding;
//Our observable that declares success.
static boolean done = false;
static boolean recombination;
static double ERROR, FITNESS;
static String SOLUTION;
static int GENERATION;
//Number of generation with no progress
```

```
 //because  of a local minimum.
 static int nEcstasy = 1;
 static int oldBestFitness = -1000000000;
 static int newBestFitness = 0;
 //Gained fitness
 static int gainedFitness;
 //Number of bugs during a generation
 //(individuals whose performance is worst
 //than that of their parents).
 static int nBugs = 0;
 //Number of bugs while expecting a jump
 static int nTotalBugsEcstasy = 0;
 static int bigTotalBugs = 0;

//Words are recognized: they are separated by a space.
//They are kept in a vector.
private static void numberWords(String s)
 {
  int counter = 0;
  String f = "";
  for(int i = 0; i < nLetters; i++)
   {
char d = s.charAt(i);
//System.out.println(d);
if (!(d == ' '))  f = f+d;
else
 {
  Word[counter] = f;
  f = "";
  counter++;
 }
//last word
if (i == nLetters-1) Word[counter] = f;
   }
  //System.out.println( counter);
  System.out.println("\nThe words of the phrase " +
                       "with their indexes: ");
  nWords = counter +1;
  for(int i = 0; i < nWords; i++)
   {
```

```
System.out.println(i + "  " +  Word[i]);
Permutation[i] =  i;
   }
 }

 //A transposition  is the  interchange of
 //place between two items.
private static void transposition()
 {
  int m = r.nextInt(nWords);
  boolean flag = false;
  //Sites must be different
  int n=0;
  while(!(flag))
   { n = r.nextInt(nWords);
   if (!(n == m)) flag = true;
   }
  if (print) System.out.println("Permuation of "
                              +  m + " and " + n);
  int k = Permutation[m];
  Permutation[m] = Permutation[n];
  Permutation[n] = k;
 }

//Mixer:  a sequence of transpositions creates variability.
//The output is encoded in a string, a chromosome.
private static String mixer(int numberOfTransp)
 {
  for(int i= 0; i < numberOfTransp; i++ )
 transposition();
  String v = "";
  for(int i = 0; i < nWords; i++)
    {
  Integer e = Permutation[i];
  v = v + e.toString();
    }
  return v;
 }
```

```
//Words in the sentence encoded by s are permuted

private static String permutate (String s)
 {
  numberWords(s);
  //Number of transpositions
  int n =  nWords;
  return  mixer(n);
 }

//String number  is decoded into a phrase
private static String stringToPhrase(String number)
  {
   String phrase= "";
   for(int j=0;j  < number.length();j++ )
    {
     char s3  =   number.charAt(j);
    int  l = Character.getNumericValue(s3);
    phrase = phrase + Word[l] + " ";

    }
   return phrase;
  }

//Declarations and default initializations
// of other arrays.
private static void otherInit( )
{
 Order = new int[limit+1];
 ReportMin = new int[limit+1];
 ReportMax = new int[limit+1];
 Fitness = new int[limit +1];
 FitnessCopy = new int[limit +1];
 for(int i = 0; i< nIndiv; i++)
  {
   Order[i] =0;
   ReportMin[i] =0;
   ReportMax[i] =0;
   Fitness[i] = 0;
```

```
   FitnessCopy[i] = 0;
 }
}

//An individual is a string that is a permutation
//of another one.
private static String generateIndividual( )
{
 //Number of transpositions
 int numberOfTransp = r.nextInt(nWords);
 String ind = mixer(numberOfTransp);
 return ind;
}

/* We generate nIndiv individuals (strings)
 encoding for permutations of 01234567 */
private static void Initialization( )
{
 //Formal declaration of our array.
 Individual= new String[limit];
 if (print)  System.out.println("ORIGINAL POPULATION");
 for(int i = 0; i< nIndiv; i++)
  {
   if (print) System.out.println( "\ni = " + i);
   Individual[i]="";
   // Individual[i] is a string,
   // it is a genotype that encodes a number,
   //its phenotype.
   Individual[i] = generateIndividual();
   if (print)
      System.out.println(" = " + Individual[i] );
  }
  otherInit();
}



//String s is decoded into a number
private static long recoverNumber(String s)
 {
```

```
  long number = 0;
  for(int j=0;j  < nWords;j++ )
   {
char c  =   s.charAt(j);

  int  l = Character.getNumericValue(c);
  int q = j+1;
  long number1 = (long) (l * Math.pow(10,nWords-q));
  number = number + number1;

   }
  return number;
 }

// This method transforms a string in a number
//of type long. Some  outputs to console are displayed.
private static long stringToNumber(String s)
  {
if (print)
  {
   System.out.println( "\nIndividual =  " + s);
   System.out.println( "length of s  = " + s.length());
  }

if (print) System.out.println( "Recovering number");
long k = recoverNumber(s);

if (print)  System.out.println("String "
    + s +  " as number = "       + k + "\n");
 return k;
 }

//This method decodes the string into the entries
//of permutation[]
private static void stringToVector(String s)
{
 for(int j = 0; j  < nWords;j++ )
  {
char c  =   s.charAt(j);
    int  l = Character.getNumericValue(c);
```

```
Permutation[j] = l;
  }
}


//The fitness of each individual is found
private static void fitness(int gen)
{
 //We measure the error of individual[i]
 //with respect to the perfect one 01234567.
 for(int i = 0; i< nIndiv; i++)
  {
   int  error =
(int) Math.abs(1234567-stringToNumber(Individual[i]));

   //The error defines the fitness: less error,
   //more fitness.
   //Maximal fitness = 0; minimal = -1E10.
   Fitness[i] = - error;
   FitnessCopy[i] = - error;
   if (print)
  System.out.println("i = " + i
  + " Individual = " + Individual[i]
  + " Error  = "+ error
  + " Fitness"  + Fitness[i]    );
  if (error  == 0)
   {
 GENERATION = gen;
 ERROR = error;
 SOLUTION = Individual[i];
 FITNESS =  Fitness[i];
 done = true;
}
  if (print)  System.out.println();
  }
 }



//Individuals are sorted by fitness
```

```
private static void sorting(int gen)
{
 fitness(gen);
 nBugs = 0;
 if (print) System.out.println("\nSORTING");
 int   Champ = 0;
 //Sorting by fitness.
 for(int i = 0; i< nIndiv;i++)
  {
   //The i-th individual is picked out
   for(int j = 0; j< nIndiv;j++)
    if (Fitness[j] >  Fitness[Champ])  Champ = j;
   //The array Order classifies individuals by  fitness
   Order[i] = Champ;
   if (print)
    {
     System.out.println(  i + "th ind. is  No "
    + Champ + " " + Individual[Order[i] ]
            + " Fitness = "  + Fitness[Champ]);
    }
   if (i == 0) newBestFitness = Fitness[Champ];
   Fitness[Champ] = -100000000;
   Champ = 0;
  } //end of for i
  //Bug's section
  for(int j = 0; j< nIndiv;j++)
  if (FitnessCopy[j] <  FitnessCopy[Champ])  nBugs++;
  if (print) System.out.println( "Number of bugs  = "
                       + nBugs);
  nTotalBugsEcstasy  = nTotalBugsEcstasy  + nBugs;
  bigTotalBugs = bigTotalBugs + nBugs;
  String t = Individual[Order[0] ];
  gainedFitness = newBestFitness - oldBestFitness;
  //If an evolutionary  jump, an improvement in fitness,
  //is detected...
  if (gainedFitness > 0)
   {
System.out.println("\nGENERATION = " + gen);
System.out.println(" Best solution "  + t  +
" -> " + stringToPhrase(t));
```

```
System.out.print("Generations needed for" +
" a new jump = " +  nEcstasy);
System.out.println(", with " + nTotalBugsEcstasy
                     + " bugs");
System.out.println("Old best fitness = "
+ oldBestFitness);
System.out.println("New best fitness = "
+ newBestFitness);
System.out.println("Gained fitness = " +
 gainedFitness);
oldBestFitness = newBestFitness;
nEcstasy = 1;
nTotalBugsEcstasy  = 0;
   }
 else
  {
   nEcstasy++;
  }
} //End of sorting



//Each individual of the top ten
// produces 10  copies.
private static void Reproduction()
{
 if (print)
 System.out.println( "Reproduction");
 if (print)
 System.out.println( "The best = " + Order[0]);
 Individualc= new String[limit];
 int counter = 0;
 for (int top = 0; top < 10; top++)
  {
   for(int j = 0; j< 10; j++)
 {
   Individualc[counter] = Individual[Order[top]];
   counter = counter +1;
  }
  }
```

```
  for(int j = 0; j< counter; j++)
Individual[j] = Individualc[j];
}

//A a sequence of transpositions creates variability.
private static String mutate(String s)
{
 //Chromosome s is transformed into vector Permutation[]
 stringToVector(s);
 //Mutate or not mutate
 double p = r.nextDouble();
 //Entries in vector are subject to mutation
 if ( p < mutRate )
   {
//Site of mutation is chosen at random
int site = r.nextInt(nWords);
//Effect of mutation is chosen at random
int change = r.nextInt(nWords);
Permutation[site] = change;
   }
 //Vector is encoded into a chromosome, a string.
 String v = "";
 for(int i = 0; i < nWords; i++)
  {
  Integer e = Permutation[i];
  v = v + e.toString();
  }
 return v;
}

//A mutation is Darwinian:
//both the site of mutation and the change
//are defined by randomness.
private static void Mutation()
{
  for(int j = 1; j<  nIndiv; j++)
{
 Individual[j] = mutate( Individual[j]);
}
}
```

```
//Two strings recombine and produce two offspring.
private static void Recombination()
{
 for(int j = 50; j<  nIndiv; j++)
  {
//Define place of recombination
int m =  r.nextInt(nIndiv);
int n = r.nextInt(nIndiv);
String a = Individual[m];
String b = Individual[n];
int placeRec = r.nextInt(nWords);
Individual[m] = a.substring(0, placeRec)
               + b.substring( placeRec);
Individual[n] = b.substring(0, placeRec)
       + a.substring( placeRec);
 }
}

//Master for a generation
private static void dynamics(int gen)
 {
  //Individuals are sorted by fitness
  sorting(gen);
  //The top ten are preferentially reproduced
  Reproduction();
  //The new population is subjected to mutation
  Mutation();
  if (recombination) Recombination();
 }

//The original ordered phrase is rebuilt
//by a genetic algorithm.
//This program simulates a teacher that knows
//the answer and guides their students
//for a better solution.
private static void restorePhrase()
 {
```

```
  System.out.println("\n\nRunning ");
  nIndiv = 70;
  Initialization( ) ;
  nGen = 5000000;
  oldError = 1000;
  for(int gen = 1; ( gen <= nGen) & (!(done)); gen++)
 {

   dynamics(gen);
 }
  if (done)
   {
 System.out.println("\nGENERATION = " + GENERATION +
       " SOLUTION FOUND");
     System.out.println("ERROR = " + ERROR);
     System.out.println("SOLUTION = " + SOLUTION
      + " -> " +  stringToPhrase(SOLUTION) );
     System.out.println("FITNESS = " + FITNESS);
     System.out.println("Number of generations = "
      + GENERATION);
     System.out.println("Number of individuals " +
      "per generation = " + nIndiv);
     long totalCost = GENERATION * nIndiv;
     System.out.println("Total cost = " +
      "Number of Generations x " +
      "number of individuals = " + totalCost);
     System.out.println("Big Total  of bugs "
                              + bigTotalBugs);
     double tC = totalCost;
     double tB = bigTotalBugs;
     System.out.println("Normalized cost = " +
      "Big Total of bugs/ Total cost = "
             + tB/tC);
   }
 }

public static void main(String[] args)
 {
  print = false;
  recombination = true;
```

```
  mutRate = 0.5;
  System.out.println("This program disorders a sentence"
     + " (with the help of transpositions,)"
         + " \nand reorders it by evolution "
         + "\nusing random mutations over the"
         + "\nalphabet 0,1,2,3,4,5,6,7."
         + "\nEvolution is Darwinian.\n");
  System.out.println("ORIGINAL SENTENCE: ");
  System.out.println(phrase);
  String p = permutate(phrase);
  System.out.println( "\nCHALLENGING DISORDERED SENTENCE" +
   " WITH ITS INDEXATION : \n" );
  System.out.println( p + " -> " + stringToPhrase(p) );
   restorePhrase();
 }//End of main method
}
```

47 ***Exercise.*** *Run the program and play with the code. Make sure that you can see the path towards perfection and the cost in bugs that evolution must pay for finding the answer.*

4.3 Battling complexity

One can solve a problem using evolution and it can be implemented in diverse forms. So, a question comes to the mind: Are all forms of implementing evolution equally efficient in the waste of needed time and other resources? To solve this question, we need to test diversity to see what it gives of itself.

48 ***Another measure of fitness***

Most problems (maybe all) can be solved by evolution and many are the ways as this can be done. Let us see, for instance, another form as we can implement selection to solve our problem *syntax*. This is done as follows: we compare letter by letter the given chromosome with that representing the solution, which is the string 01234567, and count the number of mismatches. Example: we take chromosome 10324567 and want to find its fitness. Firstly, we align the objective string with the given chromosome:

10324567
01234567

and next count the number of mismatches: they are 4. Let us refer to this measure as the *mismatch metric*. This number represents an error, so we can define fitness as -4 and declare that the higher is the fitness the fitter is the chromosome. The corresponding code follows:

```
//Program G48 syntax7
//This program shows one very simple and
//inexorable law: if evolution can indeed find the answer,
//a price must be paid: a lot of bugs must be made,
//and therefore, a clear path from imperfection
//towards perfection must exist.
//We show here
//how long one is frozen in a local minimum and
//how many bugs evolution makes.
//We have a BUG each time that appears
//an offspring that is less fitted than their parents.
//
//We define fitness as minus the number of mismatches
//of  given chromosome with respect to the optimal one.
//
//Problem to solve:
//We mimic a game that a teacher
//proposes to her pupils:
//A phrase is subject to a series of
//permutations of its words to end with a string
//with no meaning. Pupils must produce from this
//string the original sentence
//with correct meaning and syntax.
//The problem is solved by means of DARWINIAN EVOLUTION,
//in which selection is the driving force of the
//ensuing evolutionary process and
//mutation is random over the alphabet.
//In this case, selection is given by the teacher,
//who already knows the answer and can estimate
//how good are the attempts of the pupils.
//Style= global variables.


import java.util.Random;
```

```
public class syntax7
{
 private static String phrase =
  "syntax deals with the formation of correct sentences";
 private static int nLetters = phrase.length();
 private static int nWords;
 //The phrase is decomposed in words
 private static String[] Word = new String[10];
 //Words are numbered. A disordered phrase is encoded
 //in a vector of digits, Permutation[], such that
 //Permutation[i] indicates what word goes
 //at place i of the possibly disordered phrase.
 private static int[] Permutation = new int[10];
 //Turn on of the random generator
 private static Random r  = new Random();
 //Mutation rate per symbol per generation
 private static double mutRate;

 //====Genetic part of declaration of variables==========
 // Individuals are kept in the array
 // Individual[]. It is an array of strings.
 //Each individual encodes
 //the permutation that it represents.
 //The individual is a string that encodes for
 //a number.


 // The number of individuals must be
 // less than limit.
 static double zMax; static double N;
 static int limit = 50000;
 static int Fitness[], FitnessCopy[];
 static String Individual[ ], Individualc [];
 //Actual number of individuals
 static int nIndiv ;
 static int Order[];
 static String b;
 static int generation;
 static  int  nGen;
```

```
 static int ReportMin[], ReportMax[];
 static double oldError, newError;
 static double deltaError;
 static boolean print,testEncoding;
 //Our observable that declares success.
 static boolean done = false;
 static boolean recombination;
 static double ERROR, FITNESS;
 static String SOLUTION;
 static int GENERATION;
 //Number of generation with no progress
 //because  of a local minimum.
 static int nEcstasy = 1;
 static int oldBestFitness = -10;
 static int newBestFitness = 0;
 //Gained fitness
 static int gainedFitness;
 //Number of bugs during a generation
 //(individuals whose performance is worst
 //than that of their parents).
 static int nBugs = 0;
 //Number of bugs while expecting a jump
 static int nTotalBugsEcstasy = 0;
 static int bigTotalBugs = 0;

//Words are recognized: they are separated by a space.
//They are kept in a vector.
private static void numberWords(String s)
 {
  int counter = 0;
  String f = "";
  for(int i = 0; i < nLetters; i++)
   {
char d = s.charAt(i);
//System.out.println(d);
if (!(d == ' '))  f = f+d;
else
 {
  Word[counter] = f;
  f = "";
```

```
  counter++;
 }
//last word
if (i == nLetters-1) Word[counter] = f;
   }
  //System.out.println( counter);
  System.out.println("\nThe words of the phrase " +
                       "with their indexes: ");
  nWords = counter +1;
  for(int i = 0; i < nWords; i++)
   {
System.out.println(i + "  " +  Word[i]);
Permutation[i] =  i;
   }
 }

 //A transposition  is the  interchange of
 //place between two items.
private static void transposition()
 {
  int m = r.nextInt(nWords);
  boolean flag = false;
  //Sites must be different
  int n=0;
  while(!(flag))
   { n = r.nextInt(nWords);
   if (!(n == m)) flag = true;
   }
  if (print) System.out.println("Permuation of "
                              +  m + " and " + n);
  int k = Permutation[m];
  Permutation[m] = Permutation[n];
  Permutation[n] = k;
 }

//Mixer:  a sequence of transpositions creates variability.
//The output is encoded in a string, a chromosome.
private static String mixer(int numberOfTransp)
 {
  for(int i= 0; i < numberOfTransp; i++ )
```

```
 transposition();
  String v = "";
  for(int i = 0; i < nWords; i++)
    {
  Integer e = Permutation[i];
  v = v + e.toString();
    }
  return v;
 }



//Words in the sentence encoded by s are permuted

private static String permutate (String s)
 {
  numberWords(s);
  //Number of transpositions
  int n =  nWords;
  return  mixer(n);
 }

//String number  is decoded into a phrase
private static String stringToPhrase(String number)
  {
   String phrase= "";
   for(int j=0;j  < number.length();j++ )
    {
     char s3  =   number.charAt(j);
    int  l = Character.getNumericValue(s3);
    phrase = phrase + Word[l] + " ";

    }
   return phrase;
  }

//Declarations and default initializations
// of other arrays.
private static void otherInit( )
{
```

```
 Order = new int[limit+1];
 ReportMin = new int[limit+1];
 ReportMax = new int[limit+1];
 Fitness = new int[limit +1];
 FitnessCopy = new int[limit +1];
 for(int i = 0; i< nIndiv; i++)
  {
   Order[i] =0;
   ReportMin[i] =0;
   ReportMax[i] =0;
   Fitness[i] = 0;
   FitnessCopy[i] = 0;
 }
}

//An individual is a string that is a permutation
//of another one.
private static String generateIndividual( )
{
 //Number of transpositions
 int numberOfTransp = r.nextInt(nWords);
 String ind = mixer(numberOfTransp);
 return ind;
}

/* We generate nIndiv individuals (strings)
 encoding for permutations of 01234567 */
private static void Initialization( )
{
 //Formal declaration of our array.
 Individual= new String[limit];
 if (print)  System.out.println("ORIGINAL POPULATION");
 for(int i = 0; i< nIndiv; i++)
  {
   if (print) System.out.println( "\ni = " + i);
   Individual[i]="";
   // Individual[i] is a string,
   // it is a genotype that encodes a number,
   //its phenotype.
   Individual[i] = generateIndividual();
```

```
   if (print)
      System.out.println(" = " + Individual[i] );
  }
  otherInit();
}


//This method decodes the string into the entries
//of permutation[]
private static void stringToVector(String s)
{
 for(int j = 0; j  < nWords;j++ )
  {
char c  =   s.charAt(j);
    int  l = Character.getNumericValue(c);
Permutation[j] = l;
  }
}


//The fitness of each individual is found
private static void fitness(int gen)
{
 //We measure the error of individual[i]
 //with respect to the perfect one 01234567.
 for(int i = 0; i< nIndiv; i++)
  {
   int  error = 0;
   for(int j = 0; j  < nWords;j++ )
   {
  char c  =   Individual[i].charAt(j);
  Integer e = j;
  char d = (e.toString().charAt(0));
    if (!(c == d)) error++;
   }

   //The error defines the fitness: less error,
   //more fitness.
   //Maximal fitness = 0; minimal = -1E10.
   Fitness[i] = - error;
```

```
   FitnessCopy[i] = - error;
   if (print)
  System.out.println("i = " + i
  + " Individual = " + Individual[i]
  + " Error  = "+ error
  + " Fitness"  + Fitness[i]    );
  if (error  == 0)
   {
 GENERATION = gen;
 ERROR = error;
 SOLUTION = Individual[i];
 FITNESS =  Fitness[i];
 done = true;
}
  if (print)  System.out.println();
  }
 }



//Individuals are sorted by fitness
private static void sorting(int gen)
{
 fitness(gen);
 nBugs = 0;
 if (print) System.out.println("\nSORTING");
 int   Champ = 0;
 //Sorting by fitness.
 for(int i = 0; i< nIndiv;i++)
  {
   //The i-th individual is picked out
   for(int j = 0; j< nIndiv;j++)
    if (Fitness[j] >  Fitness[Champ])  Champ = j;
   //The array Order classifies individuals by  fitness
   Order[i] = Champ;
   if (print)
    {
     System.out.println(  i + "th ind. is  No "
    + Champ + " " + Individual[Order[i] ]
            + " Fitness = "  + Fitness[Champ]);
```

```
    }
   if (i == 0) newBestFitness = Fitness[Champ];
   Fitness[Champ] = -10;
   Champ = 0;
  } //end of for i
  //Bug's section
  for(int j = 0; j< nIndiv;j++)
  if (FitnessCopy[j] <  FitnessCopy[Champ])  nBugs++;
  if (print) System.out.println( "Number of bugs  = "
                        + nBugs);
  nTotalBugsEcstasy  = nTotalBugsEcstasy  + nBugs;
  bigTotalBugs = bigTotalBugs + nBugs;
  String t = Individual[Order[0] ];
  gainedFitness = newBestFitness - oldBestFitness;
  //If an evolutionary  jump, an improvement in fitness,
  //is detected...
  if (gainedFitness > 0)
   {
System.out.println("\nGENERATION = " + gen);
System.out.println(" Best solution "  + t  +
" -> " + stringToPhrase(t));
System.out.print("Generations needed for" +
" a new jump = " +  nEcstasy);
System.out.println(", with " + nTotalBugsEcstasy
                     + " bugs");
System.out.println("Old best fitness = "
+ oldBestFitness);
System.out.println("New best fitness = "
+ newBestFitness);
System.out.println("Gained fitness = " +
 gainedFitness);
oldBestFitness = newBestFitness;
nEcstasy = 1;
nTotalBugsEcstasy  = 0;
   }
 else
  {
   nEcstasy++;
  }
} //End of sorting
```

```
//Each individual of the top ten
// produces 10  copies.
private static void Reproduction()
{
 if (print)
 System.out.println( "Reproduction");
 if (print)
 System.out.println( "The best = " + Order[0]);
 Individualc= new String[limit];
 int counter = 0;
 for (int top = 0; top < 10; top++)
  {
   for(int j = 0; j< 10; j++)
 {
   Individualc[counter] = Individual[Order[top]];
   counter = counter +1;
  }
  }
  for(int j = 0; j< counter; j++)
Individual[j] = Individualc[j];
}

//A a sequence of transpositions creates variability.
private static String mutate(String s)
{
 //Chromosome s is transformed into vector Permutation[]
 stringToVector(s);
 //Mutate or not mutate
 double p = r.nextDouble();
 //Entries in vector are subject to mutation
 if ( p < mutRate )
   {
//Site of mutation is chosen at random
int site = r.nextInt(nWords);
//Effect of mutation is chosen at random
int change = r.nextInt(nWords);
Permutation[site] = change;
```

```
   }
 //Vector is encoded into a chromosome, a string.
 String v = "";
 for(int i = 0; i < nWords; i++)
  {
  Integer e = Permutation[i];
  v = v + e.toString();
  }
 return v;
}

//A mutation is Darwinian:
//both the site of mutation and the change
//are defined by randomness.
private static void Mutation()
{
  for(int j = 1; j<  nIndiv; j++)
{
 Individual[j] = mutate( Individual[j]);
}
}



//Two strings recombine and produce two offspring.
private static void Recombination()
{
 for(int j = 50; j<  nIndiv; j++)
  {
//Define place of recombination
int m =  r.nextInt(nIndiv);
int n = r.nextInt(nIndiv);
String a = Individual[m];
String b = Individual[n];
int placeRec = r.nextInt(nWords);
Individual[m] = a.substring(0, placeRec)
               + b.substring( placeRec);
Individual[n] = b.substring(0, placeRec)
       + a.substring( placeRec);
 }
```

```
}

//Master for a generation
private static void dynamics(int gen)
 {
  //Individuals are sorted by fitness
  sorting(gen);
  //The top ten are preferentially reproduced
  Reproduction();
  //The new population is subjected to mutation
  Mutation();
  if (recombination) Recombination();
 }

//The original ordered phrase is rebuilt
//by a genetic algorithm.
//This program simulates a teacher that knows
//the answer and guides their students
//for a better solution.
private static void restorePhrase()
 {
  System.out.println("\n\nRunning ");
  nIndiv = 70;
  Initialization( ) ;
  nGen = 5000000;
  oldError = 1000;
  for(int gen = 1; ( gen <= nGen) & (!(done)); gen++)
 {

   dynamics(gen);
 }
  if (done)
   {
 System.out.println("\nGENERATION = " + GENERATION +
       " SOLUTION FOUND");
     System.out.println("ERROR = " + ERROR);
     System.out.println("SOLUTION = " + SOLUTION
      + " -> " +  stringToPhrase(SOLUTION) );
     System.out.println("FITNESS = " + FITNESS);
     System.out.println("Number of generations = "
```

```
      + GENERATION);
     System.out.println("Number of individuals " +
      "per generation = " + nIndiv);
     long totalCost = GENERATION * nIndiv;
     System.out.println("Total cost = " +
      "Number of Generations x " +
      "number of individuals = " + totalCost);
     System.out.println("Big Total  of bugs "
                           + bigTotalBugs);
     double tC = totalCost;
     double tB = bigTotalBugs;
     System.out.println("Normalized cost = " +
      "Big Total of bugs/ Total cost = "
             + tB/tC);
   }
 }

public static void main(String[] args)
 {
  print = false;
  recombination = true;
  mutRate = 0.5;
  System.out.println("This program disorders a sentence"
     + " (with the help of transpositions,)"
         + " \nand reorders it by evolution "
         + "\nusing random mutations over the"
         + "\nalphabet 0,1,2,3,4,5,6,7."
         + "\nEvolution is Darwinian.\n");
  System.out.println("ORIGINAL SENTENCE: ");
  System.out.println(phrase);
  String p = permutate(phrase);
  System.out.println( "\nCHALLENGING DISORDERED SENTENCE" +
   " WITH ITS INDEXATION : \n" );
  System.out.println( p + " -> " + stringToPhrase(p) );
   restorePhrase();
 }//End of main method
}
```

49 ***Exercise.*** *Run the program and play with the code.*

50 ***Exercise.*** *Verify else reject the conclusions drawn by the Author in regard with*

previous programs: the mismatch metric produces an evolutionary environment that solves the posed problem in less generations than the old one, in which string were transformed to numbers. Additionally, the total cost together with the relation Big Total-of-bugs/Total-cost were lower for the mismatch metric.

51 ***Selection according to creationism***

Up to this very moment we have modeled selection as an entity that rewards the best performance. Concretely, our scheme of reproduction assigned more possibilities to fitter individuals. This is the **positive flavor** of selection. Such a vision is akin to the insight according to which evolution goes upwards, heading for progress. Nevertheless, there is a contrary vision that might be welcome by some *creationists*. The core of the protest could be formulated as follows:

Living beings were created perfect and therefore mutation can create variants that behave as well as their parents else that perish or that suffer somehow.

This vision is best modeled by a **negative flavor of selection**, which might be implemented in its simplest form as a probability of death, which shall be higher the worst is the fitness. A question immediately begins to spur: can this type of selection be used in applied evolution to solve problems and to guide an evolutionary process from scratch to perfection? To study this question, let us consider the next code that implements a negative type of selection that is also proposed by the demographic stand of understanding selection:

```
//Program G51 syntax8
//This program shows one very simple and
//inexorable law: if evolution can indeed find the answer,
//a price must be paid: a lot of bugs must be made,
//and therefore,
//a clear path towards perfection must exist.
//We show here
//how long one is frozen in a local minimum and
//how many bugs evolution makes.
//We have a BUG each time that appears
//an offspring that is less fitted than their parents.
//
//We define fitness as minus the number of mismatches
//of  given chromosome with respect to the optimal one.
//
//SELECTION IS NEGATIVE: the probability of death
```

```
//before reproduction is the higher the
//less is the fitness of the individual.
//
//Problem to solve:
//We mimic a game that a teacher
//proposes to her pupils:
//A phrase is subject to a series of
//permutations of its words to end with a string
//with no meaning. Pupils must produce from this
//string the original sentence
//with correct meaning and syntax.
//The problem is solved by means of DARWINIAN EVOLUTION,
//in which selection is the driving force of the
//ensuing evolutionary process and
//mutation is random over the alphabet.
//In this case, selection is given by the teacher,
//who already knows the answer and can estimate
//how good are the attempts of the pupils.
//Style= global variables.


import java.util.Random;

public class syntax8
{
 private static String phrase =
  "the problem is to have evolution to be easily evolvable";
 private static int nLetters = phrase.length();
 private static int nWords;
 //The phrase is decomposed in words
 private static String[] Word = new String[100];
 //Words are numbered. A disordered phrase is encoded
 //in a vector of digits, Permutation[], such that
 //Permutation[i] indicates what word goes
 //at place i of the possibly disordered phrase.
 private static int[] Permutation = new int[100];
 //Turn on of the random generator
 private static Random r  = new Random();
 //Mutation rate per symbol per generation
 private static double mutRate;
```

```
//====Genetic part of declaration of variables====
// Individuals are kept in the array
// Individual[]. It is an array of strings.
//Each individual encodes
//the permutation that it represents.
//The individual is a string that encodes for
//a number.


// The number of individuals must be
// less than limit.
static double zMax; static double N;
static int limit = 50000;
static int Fitness[];
static double  FitnessCopy[];
static String Individual[ ], Individualc [];
//Actual number of individuals
static int nIndiv ;
static int Order[];
static String b;
static int generation;
static  int  nGen;
static int ReportMin[], ReportMax[];
static double oldError, newError;
static double deltaError;
static boolean print,testEncoding;
//Our observable that declares success.
static boolean done = false;
static boolean recombination;
static double ERROR, FITNESS;
static String SOLUTION;
static int GENERATION;
//Number of generation with no progress
//because  of a local minimum.
static int nEcstasy = 1;
static int oldBestFitness = -10;
static int newBestFitness = 0;
//Gained fitness
static int gainedFitness;
```

```
 //Number of bugs during a generation
 //(individuals whose performance is worst
 //than that of their parents).
 static int nBugs = 0;
 //Number of bugs while expecting a jump
 static int nTotalBugsEcstasy = 0;
 static int bigTotalBugs = 0;

//Words are recognized: they are separated by a space.
//They are kept in a vector.
private static void numberWords(String s)
 {
  int counter = 0;
  String f = "";
  for(int i = 0; i < nLetters; i++)
   {
char d = s.charAt(i);
//System.out.println(d);
if (!(d == ' '))  f = f+d;
else
 {
  Word[counter] = f;
  f = "";
  counter++;
 }
//last word
if (i == nLetters-1) Word[counter] = f;
   }
  //System.out.println( counter);
  System.out.println("\nThe words of the phrase " +
                       "with their indexes: ");
  nWords = counter +1;
  for(int i = 0; i < nWords; i++)
   {
System.out.println(i + "  " +  Word[i]);
Permutation[i] =  i;
   }
 }

 //A transposition  is the  interchange of
```

```
 //place between two items.
private static void transposition()
 {
  int m = r.nextInt(nWords);
  boolean flag = false;
  //Sites must be different
  int n=0;
  while(!(flag))
   { n = r.nextInt(nWords);
   if (!(n == m)) flag = true;
   }
  if (print) System.out.println("Permuation of "
                              +  m + " and " + n);
  int k = Permutation[m];
  Permutation[m] = Permutation[n];
  Permutation[n] = k;
 }

//Mixer:  a sequence of transpositions creates variability.
//The output is encoded in a string, a chromosome.
private static String mixer(int numberOfTransp)
 {
  for(int i= 0; i < numberOfTransp; i++ )
 transposition();
  String v = "";
  for(int i = 0; i < nWords; i++)
    {
  Integer e = Permutation[i];
  v = v + e.toString();
    }
  return v;
 }



//Words in the sentence encoded by s are permuted

private static String permutate (String s)
 {
  numberWords(s);
```

```
  //Number of transpositions
  int n =  nWords;
  return  mixer(n);
 }

//String number  is decoded into a phrase
private static String stringToPhrase(String number)
  {
   String phrase= "";
   for(int j=0;j  < number.length();j++ )
    {
     char s3  =   number.charAt(j);
    int  l = Character.getNumericValue(s3);
    phrase = phrase + Word[l] + " ";

    }
   return phrase;
  }

//Declarations and default initializations
// of other arrays.
private static void otherInit( )
{
 Order = new int[limit+1];
 ReportMin = new int[limit+1];
 ReportMax = new int[limit+1];
 Fitness = new int[limit +1];
 FitnessCopy = new double[limit +1];
 for(int i = 0; i< nIndiv; i++)
  {
   Order[i] =0;
   ReportMin[i] =0;
   ReportMax[i] =0;
   Fitness[i] = 0;
   FitnessCopy[i] = 0;
 }
}

//An individual is a string that is a permutation
//of another one.
```

```
private static String generateIndividual( )
{
 //Number of transpositions
 int numberOfTransp = r.nextInt(nWords);
 String ind = mixer(numberOfTransp);
 return ind;
}

/* We generate nIndiv individuals (strings)
 encoding for permutations of 01234567 */
private static void Initialization( )
{
 //Formal declaration of our array.
 Individual = new String[limit];
 Individualc = new String[limit];
 if (print)  System.out.println("ORIGINAL POPULATION");
 for(int i = 0; i< nIndiv; i++)
  {
   if (print) System.out.println( "\ni = " + i);
   Individual[i]="";
   Individualc[i]="";
   // Individual[i] is a string,
   // it is a genotype that encodes a number,
   //its phenotype.
   Individual[i] = generateIndividual();
   if (print)
      System.out.println(" = " + Individual[i] );
  }
  otherInit();
}


//This method decodes the string into the entries
//of permutation[]
private static void stringToVector(String s)
{
 for(int j = 0; j  < nWords;j++ )
  {
char c  =   s.charAt(j);
    int  l = Character.getNumericValue(c);
```

```
Permutation[j] = l;
  }
}


//The fitness of each individual is found
private static void fitness(int gen)
{
 //We measure the error of individual[i]
 //with respect to the perfect one 01234567.
 for(int i = 0; i< nIndiv; i++)
  {
   int  error = 0;
   for(int j = 0; j  < nWords;j++ )
   {
  char c  =   Individual[i].charAt(j);
  Integer e = j;
  char d = (e.toString().charAt(0));
    if (!(c == d)) error++;
   }

   //The error defines the fitness: less error,
   //more fitness.
   //Maximal fitness = 0; minimal = -10.
   Fitness[i] = - error;
   FitnessCopy[i] = - error;
   if (print)
  System.out.println("i = " + i
  + " Individual = " + Individual[i]
  + " Error  = "+ error
  + " Fitness"  + Fitness[i]    );
  if (error  == 0)
   {
 GENERATION = gen;
 ERROR = error;
 SOLUTION = Individual[i];
 FITNESS =  Fitness[i];
 done = true;
}
  if (print)  System.out.println();
```

```
  }
 }



//Individuals are sorted by fitness
private static void sorting(int gen)
{
 fitness(gen);
 nBugs = 0;
 if (print) System.out.println("\nSORTING");
 int   Champ = 0;
 //Sorting by fitness.
 for(int i = 0; i< nIndiv;i++)
  {
   //The i-th individual is picked out
   for(int j = 0; j< nIndiv;j++)
    if (Fitness[j] >  Fitness[Champ])  Champ = j;
   //The array Order classifies individuals by  fitness
   Order[i] = Champ;
   if (print)
    {
     System.out.println(  i + "th ind. is  No "
    + Champ + " " + Individual[Order[i] ]
            + " Fitness = "  + Fitness[Champ]);
    }
   if (i == 0) newBestFitness = Fitness[Champ];
   Fitness[Champ] = -10;
   Champ = 0;
  } //end of for i
  //Bug's section
  for(int j = 0; j< nIndiv;j++)
  if (FitnessCopy[j] <  FitnessCopy[Champ])  nBugs++;
  if (print) System.out.println( "Number of bugs  = "
                       + nBugs);
  nTotalBugsEcstasy  = nTotalBugsEcstasy  + nBugs;
  bigTotalBugs = bigTotalBugs + nBugs;
  String t = Individual[Order[0] ];
  gainedFitness = newBestFitness - oldBestFitness;
  //If an evolutionary  jump, an improvement in fitness,
```

```
  //is detected...
  if (gainedFitness > 0)
   {
System.out.println("\nGENERATION = " + gen);
System.out.println(" Best solution "  + t  +
" -> " + stringToPhrase(t));
System.out.print("Generations needed for" +
" a new jump = " +  nEcstasy);
System.out.println(", with " + nTotalBugsEcstasy
                     + " bugs");
System.out.println("Old best fitness = "
+ oldBestFitness);
System.out.println("New best fitness = "
+ newBestFitness);
System.out.println("Gained fitness = " +
 gainedFitness);
oldBestFitness = newBestFitness;
nEcstasy = 1;
nTotalBugsEcstasy  = 0;
   }
 else
  {
   nEcstasy++;
  }
} //End of sorting


//Negative selection:
//The probability of dead is higher,
//the lower is the fitness
private static void death()
{
 //Fitness is rescaled
 double min = 100;
 double max = -100;
 for(int j = 0; j< nIndiv;j++)
  {
   if (FitnessCopy[j] > max ) max = FitnessCopy[j];
   if (FitnessCopy[j] < min ) min = FitnessCopy[j];
  }
```

```
 //
 for(int j = 0; j< nIndiv;j++)
  {
   double probDeath = -FitnessCopy[j]/(max-min) + max/(max-min);
   //System.out.println("ProbDeath = " + probDeath);
   double p = r.nextDouble();
   if (p <=probDeath) Individual[j] = "";
  }
}

//A random not nil individual is chosen.
private static int  randomIndiv()
{
 boolean cloned = false;
 int k = 0;
 while (cloned == false)
  {
k = r.nextInt(nIndiv);
if (!(Individual[k] == ""))
cloned = true;
  }
  if (print) System.out.println("Cloned from = " + k);
 return k;
}

//The pre-population of the new generation
//is built with clones
//of living individuals  sampled at random
//with replacement.
private static void Reproduction()
{
 if (print)
 System.out.println( "Reproduction");
 if (print)
 System.out.println( "The best = " + Order[0]);
 for(int j = 0; j<  nIndiv; j++)
 Individualc[j] = Individual[randomIndiv()];
  for(int j = 0; j< nIndiv; j++)
Individual[j] = Individualc[j];
```

```
}

//A a sequence of transpositions creates variability.
private static String mutate(String s)
{
 //Chromosome s is transformed into vector Permutation[]
 stringToVector(s);
 //Mutate or not mutate
 double p = r.nextDouble();
 //Entries in vector are subject to mutation
 if ( p < mutRate )
   {
//Site of mutation is chosen at random
int site = r.nextInt(nWords);
//Effect of mutation is chosen at random
int change = r.nextInt(nWords);
Permutation[site] = change;
   }
 //Vector is encoded into a chromosome, a string.
 String v = "";
 for(int i = 0; i < nWords; i++)
  {
  Integer e = Permutation[i];
  v = v + e.toString();
  }
 return v;
}

//A mutation is Darwinian:
//both the site of mutation and the change
//are defined by randomness.
private static void Mutation()
{
  for(int j = 1; j<  nIndiv; j++)
{
 Individual[j] = mutate( Individual[j]);
}
}
```

```
//Two strings recombine and produce two offspring.
private static void Recombination()
{
 for(int j = 50; j<  nIndiv; j++)
  {
//Define place of recombination
int m =  r.nextInt(nIndiv);
int n = r.nextInt(nIndiv);
String a = Individual[m];
String b = Individual[n];
int placeRec = r.nextInt(nWords);
Individual[m] = a.substring(0, placeRec)
               + b.substring( placeRec);
Individual[n] = b.substring(0, placeRec)
       + a.substring( placeRec);
 }
}

//Master for a generation
private static void dynamics(int gen)
 {
  //Individuals are sorted by fitness
  sorting(gen);
  //Differential death
  death();
  //The top ten are preferentially reproduced
  Reproduction();
  //The new population is subjected to mutation
  Mutation();
  if (recombination) Recombination();
 }

//The original ordered phrase is rebuilt
//by a genetic algorithm.
//This program simulates a teacher that knows
//the answer and guides their students
//for a better solution.
private static void restorePhrase()
 {
```

```
  System.out.println("\n\nRunning ");
  nIndiv = 70;
  Initialization( ) ;
  nGen = 5000000;
  oldError = 1000;
  for(int gen = 1; ( gen <= nGen) & (!(done)); gen++)
 {

   dynamics(gen);
 }
  if (done)
   {
 System.out.println("\nGENERATION = " + GENERATION +
       " SOLUTION FOUND");
     System.out.println("ERROR = " + ERROR);
     System.out.println("SOLUTION = " + SOLUTION
      + " -> " +  stringToPhrase(SOLUTION) );
     System.out.println("FITNESS = " + FITNESS);
     System.out.println("Number of generations = "
      + GENERATION);
     System.out.println("Number of individuals " +
      "per generation = " + nIndiv);
     long totalCost = GENERATION * nIndiv;
     System.out.println("Total cost = " +
      "Number of Generations x " +
      "number of individuals = " + totalCost);
     System.out.println("Big Total  of bugs "
                              + bigTotalBugs);
     double tC = totalCost;
     double tB = bigTotalBugs;
     System.out.println("Normalized cost = " +
      "Big Total of bugs/ Total cost = "
             + tB/tC);
   }
 }

public static void main(String[] args)
 {
  print = false;
  recombination = true;
```

```
  mutRate = 0.5;
  System.out.println("This program disorders a sentence"
     + " (with the help of transpositions,)"
         + " \nand reorders it by evolution "
         + "\nusing random mutations over the"
         + "\nalphabet 0,1,2,3,4,5,6,7."
         + "\nEvolution is Darwinian.\n");
  System.out.println("ORIGINAL SENTENCE: ");
  System.out.println(phrase);
  String p = permutate(phrase);
  System.out.println( "\nCHALLENGING DISORDERED SENTENCE" +
   " WITH ITS INDEXATION : \n" );
  System.out.println( p + " -> " + stringToPhrase(p) );
   restorePhrase();
 }//End of main method
}
```

52 ***Exercise.*** *Run the program and play with the code. Decide once and for all times whether or not negative selection can guide evolution towards perfection beginning from scratch. If that is the case, Which type of selection is better for applied purposes, positive or negative? What is your opinion about the relevance of the aforementioned creationist complaint?*

53 ***Exercise.*** *Randomness alone demands too much resources for solving complex problem. Selection alone without variability is also hopeless. But randomness that creates a great span of variability working together with selection form a marvelous duo that is capable of solving very complex problems. And, what about twisted randomness? Make an attempt to twist randomness in solo by modifying program syntax5, G39, pag 71. Hint: implement the directive of thinking well to next make things in perfect form. This is possible because we already know where we want to arrive at.*

4.4 Conclusion

To decide over the limits of evolution, we have chosen complexity as the suitable referee which is endowed with a powerful insight: the genome is software and evolution is a software developer. Our simulations show results that are not against the belief that evolution can solve all sort of optimization problems if only enough time and resources are given. Nevertheless, the way as we have experienced the heavy load of complexity is that a price must be inescapably paid: a lot of bugs

shall be committed and a clear path from imperfection to perfection shall be easily traced. So, one automatically tries to diminish the demanded cost. In particular, we have seen that one can play with different forms of measuring fitness and that one can also choose between the positive vs negative flavors of selection. They are almost equally good for applied purposes but the negative one is simpler to implement. We have also witnessed that twisted randomness seems to be almighty to solve problems of retro-engineering in which we already know the answer.

Chapter 5

The origin of species

Let us imagine a person that considers that he or she has enough reasons to belief that species of living beings on Earth were a result of evolution. He also knows that there are two flavors of evolution, the first is Darwinian, the second is Lamarckian. The difference is that if evolution is Darwinian nature unfolds itself naturally, without external help, while in Lamarckian evolution, nature is somehow guided to complexity and perfection by an outside force or intelligence that somehow interferes with nature (Lamarck was alchemist).

Now, she or he asks her or himself: which model fits better as field data as the fossil record?

54 ***Exercise.*** *Let us imagine that we indeed appeared as a fruit of evolution on Earth:*

1. *Propose a simple and clear cut test to decide between Darwinian or Lamarckian evolution to explain our existence.*

2. *Lamarckism to be scientific needs a material mechanism to twist randomness. Can you see some hope in that respect?*

3. *Prove beyond doubts and criticisms that the Modern International Literature relies neither on Lamarckian or Darwinian evolution but on a lack of clear concepts. Which ones?*

4. *Does this discussion converts religions and creationism into science?*

Chapter 6

Evolution without evolvability

`Yes but not`

55 ***Purpose*** *As a hard protest against those that consider that evolution is the obvious, tautological explanation of life in Earth, we present here a clear illustration of the fact that evolution is an idea that can be implemented as bad as desired. Concretely, one can use evolution to solve a problem but without a trace of evolvability, i.e., without the less possibility of easy adaptation to a slightly modified problem. A challenge is now clear: the evolvability of the genome, which is real, is a marvel that deserves more than a cheap explanation.*

6.1 Evolvability

The development of software is among the most difficult enterprises known to man. A very natural and smart idea to diminish that difficulty is **reuse**, to use extant software to develop further, possibly more complex projects. The property of being reusable is also known as **evolvability**.

56 ***A difficult aim***

Experience shows at every moment all around the world that evolvability is a property of software that must be actively pursued and worked out with wisdom, knowledge, sweat and tears. So, it is at the very front of the most crucial and actual problems of computing science (Martuffi, 2007). To understand what it is all about, please, solve the next very easy task:

57 ***Exercise.*** *Use the previous program, G46 syntax6, pag 81, that use Darwinian evolution to disorder and reorder each one of the following sentences:*

1. *evolution can be implemented as bad as desired*
2. *the problem is to have evolution to be easily evolvable*
3. *one can use evolution to solve a problem but without a trace of evolvability*

We have witnessed that program G46 is broken to pieces by a sentence with more than 10 words. We see here that one can solve a problem by evolution but that the achieved solution could be so poor that it is unsuitable to solve even slightly modified problems. That is why we say that evolution can be programmed as bad as desired. This is a crude reality that most works are inescapably plagued with.

58 ***Evolving robustness***

Let us make the necessary amendments in program G46 to enable the acceptance of input phrases with a number of words less than 100.

The problem we face is the following: a number in base 10 is something like 34589012. This number is immediately encoded as the string 34589012. But to encode numbers in base, say, 13, we must use another alphabet because $\{0,1,2,3,4,5,6,7,8,9\}$ does not suffice. For base 13, the next alphabet might be good: $\{0,1,2,3,4,5,6,7,8,9,a,b,c\}$. If we use this alphabet, we can solve problems in base 13, and this idea can be easily adapted to problems in base 30. And what about base 10000? We consider that the next idea not only solves the problem with base 13 but additionally opens the way to program a solution with whatever base. Thus, we are not only interesting in modifying, evolving, a program to achieve a new function but we aim at doing that in such a way that coming challenges could be more easily solved. Evolution alone, adaptation to new tasks, is not sufficient for us: evolvability is what we aim at. The idea is the following:

We need a new type of **codon** (substring that encodes for an elementary object), not with one digit but with two. Example for numbers in base 10: the number 5 is encoded by the string 05. The number 23 is encoded as the string 23 and so on until 99. The relation of this encoding with syntax, our target problem, is as follows: 11052327 encodes for a phrase that contains in order those words of the original phrase whose numbers are 11, 5, 23 and 27. The corresponding code follows.

59 ***The code.***

```
//Program G59 syntaxb1
//This program shows one very simple and
//inexorable law: if evolution can indeed find the answer,
```

```
//a price must be paid: a lot of bugs must be made,
//and therefore,
//a clear path towards perfection must exist.
//We show here
//how long one is frozen in a local minimum and
//how many bugs evolution makes.
//We have a BUG each time that appears
//an offspring that is less fitted than their parents.
//
//We define fitness as minus the number of mismatches
//of  given chromosome with respect to the optimal one.
//
//Selection is negative: the probability of dead
//before reproduction is the higher the
//less is the fitness of the individual.
//
//Problem to solve:
//We mimic a game that a teacher
//proposes to her pupils:
//A phrase is subject to a series of
//permutations of its words to end with a string
//with no meaning. Pupils must produce from this
//string the original sentence
//with correct meaning and syntax.
//The problem is solved by means of
//LAMARCKIAN EVOLUTION,
//in which evolution is helped with suitable cues.
//In this case, selection is given by the teacher,
//who already knows the answer and can estimate
//how good are the attempts of the pupils.
//Style= global variables.
//
//Robustness of the code against perturbations
//of the input  is enhanced to deal
//with sentences with 100  words or less.
//This is done trying to enable EVOLVABILITY,
//i.e., facilitating the future adaptation of
//the code to digest phrases with more than 100 words.

import java.util.Random;
```

```
public class syntaxb1
{
 private static String phrase =
  "one can use evolution to solve a problem " +
  "but without a trace of evolvability";
 private static String target ="";
 private static int nLetters = phrase.length();
 private static int nWords;
 //The phrase is decomposed in words
 private static String[] Word = new String[100];
 //A disordered sentence is just a permutation of
 //the original one:
 private static String[] Permutation = new String[100];
 //Turn on of the random generator
 private static Random r  = new Random();
 //Mutation rate per symbol per generation
 private static double mutRate;

 //=====Genetic part of declaration of variables=====
 // Individuals are kept in the array
 // Individual[]. It is an array of strings.
 //Each individual encodes
 //the permutation that it represents.
 //The individual is a string that encodes for
 //a number.


 // The number of individuals must be
 // less than limit.
 static double zMax; static double N;
 static int limit = 50000;
 static int[] Fitness;
 static double[]  FitnessCopy;
 static String[] Individual, Individualc ;
 //Actual number of individuals
 static int nIndiv ;
 static int[] Order;
 static String b;
 static int generation;
```

```
 static  int  nGen;
 static int[] ReportMin, ReportMax;
 static double oldError, newError;
 static double deltaError;
 static boolean print,testEncoding;
 //Our observable that declares success.
 static boolean done = false;
 static boolean recombination;
 static double ERROR, FITNESS;
 static String SOLUTION;
 static int GENERATION;
 //Number of generation with no progress
 //because  of a local minimum.
 static int nEcstasy = 1;
 static int oldBestFitness = -100;
 static int newBestFitness = 0;
 //Gained fitness
 static int gainedFitness;
 //Number of bugs during a generation
 //(individuals whose performance is worst
 //than that of their parents).
 static int nBugs = 0;
 //Number of bugs while expecting a jump
 static int nTotalBugsEcstasy = 0;
 static int bigTotalBugs = 0;

//Words are recognized: they are separated by a space.
//They are kept in a vector.
private static void numberWords(String s)
 {
  int counter = 0;
  String f = "";
  for(int i = 0; i < nLetters; i++)
   {
char d = s.charAt(i);
//System.out.println(d);
if (!(d == ' '))  f = f+d;
else
 {
  Word[counter] = f;
```

```
  f = "";
  counter++;
 }
//last word
if (i == nLetters-1) Word[counter] = f;
   }
  //System.out.println( counter);
  System.out.println("\nThe words of the phrase " +
                      "with their indexes: ");
  nWords = counter +1;
  for(int i = 0; i < nWords; i++)
   {
System.out.println(i + "  " +  Word[i]);
Integer e = i;
String h = e.toString();
  if (h.length() == 1) h = '0'+h;
Permutation[i] =  h;
target =target + h;
   }
  System.out.println("Target = phrase is " +
            "encoded in a chromosome = " + target);
 }

 //A transposition  is the  interchange of
 //place between two items.
private static void transposition()
 {
  int m = r.nextInt(nWords);
  boolean flag = false;
  //Sites must be different
  int n=0;
  while(!(flag))
   { n = r.nextInt(nWords);
   if (!(n == m)) flag = true;
   }
  if (print) System.out.println("Permuation of "
                              +  m + " and " + n);
  String k = Permutation[m];
  Permutation[m] = Permutation[n];
  Permutation[n] = k;
```

```
 }

//Mixer:  a sequence of transpositions creates variability.
//The output is encoded in a string, a chromosome.
private static String mixer(int numberOfTransp)
 {
  for(int i= 0; i < numberOfTransp; i++ )
 transposition();
  String v = "";
  for(int i = 0; i < nWords; i++)
    {
  String e = Permutation[i];
  v = v + e.toString();
    }
  return v;
 }



//Words in the sentence encoded by s are permuted

private static String permutate (String s)
 {
  numberWords(s);
  //Number of transpositions
  int n =  nWords;
  return  mixer(n);
 }



//String number  is decoded into a phrase
private static String stringToPhrase(String s)
 {
  String phrase= "";
  for(int j=0;j  < nWords;j++ )
   {
    char s1 =   s.charAt(2*j);
    int  p1 = Character.getNumericValue(s1);
  s1  =   s.charAt(2*j+1);
```

```
  int  p2 = Character.getNumericValue(s1);
  int number = p1*10 + p2;
  if (print) System.out.println( "number = " + number);
  phrase = phrase + Word[number] + " ";
   }
  return phrase;
 }


//Declarations and default initializations
// of other arrays.
private static void otherInit( )
{
 Order = new int[limit+1];
 ReportMin = new int[limit+1];
 ReportMax = new int[limit+1];
 Fitness = new int[limit +1];
 FitnessCopy = new double[limit +1];
 for(int i = 0; i< nIndiv; i++)
  {
   Order[i] =0;
   ReportMin[i] =0;
   ReportMax[i] =0;
   Fitness[i] = 0;
   FitnessCopy[i] = 0;
 }
}

//An individual is a string that is a permutation
//of another one.
private static String generateIndividual( )
{
 //Number of transpositions
 int numberOfTransp = r.nextInt(nWords);
 String ind = mixer(numberOfTransp);
 return ind;
}

/* We generate nIndiv individuals (strings)
 encoding for permutations of 01234567 */
```

```
private static void Initialization( )
{
 //Formal declaration of our array.
 Individual = new String[limit];
 Individualc = new String[limit];
 if (print)  System.out.println("ORIGINAL POPULATION");
 for(int i = 0; i< nIndiv; i++)
  {
   if (print) System.out.println( "\ni = " + i);
   Individual[i]="";
   Individualc[i]="";
   // Individual[i] is a string,
   // it is a genotype that encodes a number,
   //its phenotype.
   Individual[i] = generateIndividual();
   if (print)
      System.out.println(" = " + Individual[i] );
  }
  otherInit();
}


//This method decodes the string s into the entries
//of permutation[]. Letters are taken by pairs.
private static void stringToVector(String s)
{
 if (print) System.out.println("s = " + s);
 for(int j = 0; j  < nWords;j++ )
  {
String t = s.substring(2*j, 2*j + 2);
Permutation[j] = t;
if (print)  System.out.println("t = " + t);
  }
}


//The fitness of each individual is found
private static void fitness(int gen)
{
 //We measure the error of individual[i]
```

```
 //with respect to the perfect one 01234567.
 for(int i = 0; i< nIndiv; i++)
  {
   int  error = 0;
   String s = Individual[i];
   if (print) System.out.println( "s = " + s);
   for(int j = 0; j  < nWords;j++ )
   {
String a = s.substring(2*j, 2*j + 2);
String b = target.substring(2*j, 2*j + 2);
if (print) System.out.println( "a = " + a + " b = " + b);
    if (!(a.equals(b))) error++;
   }

   //The error defines the fitness: less error,
   //more fitness.
   //Maximal fitness = 0; minimal = -10.
   Fitness[i] = - error;
   FitnessCopy[i] = - error;
   if (print)
  System.out.println("i = " + i
  + " Individual = " + Individual[i]
  + " Error  = "+ error
  + " Fitness"  + Fitness[i]    );
  if (error  == 0)
   {
 GENERATION = gen;
 ERROR = error;
 SOLUTION = Individual[i];
 FITNESS =  Fitness[i];
 done = true;
}
  if (print)  System.out.println();
  }
 }



//Individuals are sorted by fitness
private static void sorting(int gen)
```

```
{
 fitness(gen);
 nBugs = 0;
 if (print) System.out.println("\nSORTING");
 int   Champ = 0;
 //Sorting by fitness.
 for(int i = 0; i< nIndiv;i++)
  {
   //The i-th individual is picked out
   for(int j = 0; j< nIndiv;j++)
    if (Fitness[j] >  Fitness[Champ])  Champ = j;
   //The array Order classifies individuals by  fitness
   Order[i] = Champ;
   if (print)
    {
     System.out.println(  i + "th ind. is  No "
    + Champ + " " + Individual[Order[i] ]
            + " Fitness = "  + Fitness[Champ]);
    }
   if (i == 0) newBestFitness = Fitness[Champ];
   Fitness[Champ] = -10;
   Champ = 0;
  } //end of for i
  //Bug's section
  for(int j = 0; j< nIndiv;j++)
  if (FitnessCopy[j] <  FitnessCopy[Champ])  nBugs++;
  if (print) System.out.println( "Number of bugs  = "
                        + nBugs);
  nTotalBugsEcstasy  = nTotalBugsEcstasy  + nBugs;
  bigTotalBugs = bigTotalBugs + nBugs;
  String t = Individual[Order[0] ];
  gainedFitness = newBestFitness - oldBestFitness;
  //If an evolutionary  jump, an improvement in fitness,
  //is detected...
  if (gainedFitness > 0)
   {
System.out.println("\nGENERATION = " + gen);
System.out.println(" Best solution "  + t  +
" -> " + stringToPhrase(t));
System.out.print("Generations needed for" +
```

```
" a new jump = " +  nEcstasy);
System.out.println(", with " + nTotalBugsEcstasy
                     + " bugs");
System.out.println("Old best fitness = "
+ oldBestFitness);
System.out.println("New best fitness = "
+ newBestFitness);
System.out.println("Gained fitness = " +
 gainedFitness);
oldBestFitness = newBestFitness;
nEcstasy = 1;
nTotalBugsEcstasy  = 0;
   }
 else
  {
   nEcstasy++;
  }
} //End of sorting


//Negative selection:
//The probability of dead is higher,
//the lower is the fitness
private static void death()
{
 //Fitness is rescaled
 double min = 100;
 double max = -100;
 for(int j = 0; j< nIndiv;j++)
  {
   if (FitnessCopy[j] > max ) max = FitnessCopy[j];
   if (FitnessCopy[j] < min ) min = FitnessCopy[j];
  }

 //
 for(int j = 0; j< nIndiv;j++)
  {
   double probDeath = -FitnessCopy[j]/(max-min) + max/(max-min);
   //System.out.println("ProbDeath = " + probDeath);
   double p = r.nextDouble();
```

```
   if (p <=probDeath) Individual[j] = "";
  }
}

//A random not nil individual is chosen.
private static int  randomIndiv()
{
 boolean cloned = false;
 int k = 0;
 while (cloned == false)
  {
k = r.nextInt(nIndiv);
if (!(Individual[k] == ""))
cloned = true;
  }
  if (print) System.out.println("Cloned from = " + k);
 return k;
}

//The pre-population of the new generation
//is built with clones
//of living individuals  sampled at random
//with replacement.
private static void Reproduction()
{
 if (print)
 System.out.println( "Reproduction");
 if (print)
 System.out.println( "The best = " + Order[0]);
 for(int j = 0; j<  nIndiv; j++)
 Individualc[j] = Individual[randomIndiv()];
  for(int j = 0; j< nIndiv; j++)
Individual[j] = Individualc[j];
}

//A a sequence of transpositions creates variability.
private static String mutate(String s)
{
 //Chromosome s is transformed into vector Permutation[]
 stringToVector(s);
```

```
 //Mutate or not mutate
 double p = r.nextDouble();
 //Entries in vector are subject to mutation
 if ( p < mutRate )
   {
//Site of mutation is chosen at random
int site = r.nextInt(nWords);
//Effect of mutation is chosen at random
Integer change = r.nextInt(nWords);
String h = change.toString();
  if (h.length() == 1) h = '0'+h;
Permutation[site] = h;
   }
 //Vector is encoded into a chromosome, a string.
 String v = "";
 for(int i = 0; i < nWords; i++)
  {
  String h = Permutation[i];
  v = v + h;
  }
 return v;
}

//A mutation is Darwinian:
//both the site of mutation and the change
//are defined by randomness.
private static void Mutation()
{
  for(int j = 1; j<  nIndiv; j++)
{
 Individual[j] = mutate( Individual[j]);
}
}



//Two strings recombine and produce two offspring.
private static void Recombination()
{
 for(int j = 50; j<  nIndiv; j++)
```

```
  {
//Define place of recombination
int m =  r.nextInt(nIndiv);
int n = r.nextInt(nIndiv);
String a = Individual[m];
String b = Individual[n];
int placeRec = r.nextInt(2*nWords);
Individual[m] = a.substring(0, placeRec)
               + b.substring( placeRec);
Individual[n] = b.substring(0, placeRec)
       + a.substring( placeRec);
 }
}

//Master for a generation
private static void dynamics(int gen)
 {
  //Individuals are sorted by fitness
  sorting(gen);
  //Differential death
  death();
  //The top ten are preferentially reproduced
  Reproduction();
  //The new population is subjected to mutation
  Mutation();
  if (recombination) Recombination();
 }

//The original ordered phrase is rebuilt
//by a genetic algorithm.
//This program simulates a teacher that knows
//the answer and guides their students
//for a better solution.
private static void restorePhrase()
 {
  System.out.println("\n\nRunning ");
  nIndiv = 70;
  Initialization( ) ;
  nGen = 5000000;
  oldError = 1000;
```

```
  for(int gen = 1; ( gen <= nGen) & (!(done)); gen++)
 {

   dynamics(gen);
 }
  if (done)
   {
 System.out.println("\nGENERATION = " + GENERATION +
       " SOLUTION FOUND");
     System.out.println("ERROR = " + ERROR);
     System.out.println("SOLUTION = " + SOLUTION
      + " -> " +  stringToPhrase(SOLUTION) );
     System.out.println("FITNESS = " + FITNESS);
     System.out.println("Number of generations = "
      + GENERATION);
     System.out.println("Number of individuals " +
      "per generation = " + nIndiv);
     long totalCost = GENERATION * nIndiv;
     System.out.println("Total cost = " +
      "Number of Generations x " +
      "number of individuals = " + totalCost);
     System.out.println("Big Total  of bugs "
                           + bigTotalBugs);
     double tC = totalCost;
     double tB = bigTotalBugs;
     System.out.println("Normalized cost = " +
      "Big Total of bugs/ Total cost = "
             + tB/tC);
   }
 }

public static void main(String[] args)
 {
  print = false;
  recombination = true;
  mutRate = 0.5;
  System.out.println("This program disorders a sentence"
     + " (with the help of transpositions), \n"
         + "and reorders it by evolution \n"
         + "using random mutations over the \n"
```

```
        + "alphabet 0,1,2,3,...,n, \n"
        + "where n is the number of words \n"
        + "of the sentence. \n"
        + "\nEvolution is Darwinian.\n");
  System.out.println("ORIGINAL SENTENCE: ");
  System.out.println(phrase);
  String p = permutate(phrase);
  System.out.println( "\nCHALLENGING DISORDERED SENTENCE" +
   " WITH ITS INDEXATION : \n" );
  System.out.println( p + " -> " + stringToPhrase(p) );
   restorePhrase();
 }//End of main method
}
```

60 ***Exercise.*** *Run the program and play with the code.*

61 ***Exercise.*** *The previous code solves a problem with the help of Lamarckian evolution. Clearly explains why it is not Darwinian. Recall that evolution is Darwinian when mutation is random over the alphabet dictated by reproduction.*

62 ***Exercise.*** *Program G51 syntax8, pag 109, solves syntax by Darwinian evolution. Make the necessary modifications to extend robustness against variations of the input to accept phrases with 100 words or less. Verify that Darwinian evolution is by far slower than the Lamarckian. For applied purposes, one uses to say that evolution can be implemented as bad as desired. As we see, Darwinian evolution might be a candidate for the worst way of implementing evolution. It is astonishing that the International Literature teaches that species arouse by Darwinian evolution. In general, all simulations of evolution, in genetic algorithms and in genetic programming (the design of software by means of evolution) are Lamarckian and are oriented by refined mathematical ideas or by powerful heuristics that may be hidden just in the way as one chooses the alphabet of the evolutionary environment. Thus, applied evolution is a very difficult art that is worth a career for a life.*

63 ***Exercise.*** *We have passed, with a good doses of difficulty, from a program that accepts sentences with less than 10 words to another that accepts sentences with a maximum of 100. And this kind of robustness was promised to had been devised with evolvability (the quality of being easily modifiable to solve related but more complex problems). How to achieve evolvability is one of the crucial problems of computing science. This is due to the fact that developing software is just going over corrections after corrections that generate more corrections. In ordinary life, evolvability might be a case of good luck. To look at a concrete case,*

modify program G59 syntaxb1, page 128, to accept phrases, maybe paragraphs, with more than 100 words. Decide whether or not our program G59 awakes an evolvable branch.

64 ***Challenge.*** *Generalize the program that solves syntax to one that accepts any sentence with whatever number of words.*

65 *Ruling out Lamarckian evolution*

We have contended that life on Earth cannot be explained by Darwinian evolution because there is no trace of evolution towards complexity with the expected long path paved with every sort of malformations and malfunctions. Now, Does somebody has something to say about the proposal that life appeared as a fruit of Lamarckian evolution? Yes, we do. We have a lot of experience with Lamarckian evolution:

Software development by human beings, with or without computing assistance, always involves an evolutionary process in which complexity is built following a very hard path paved in bugs. One can say that the closest reality to a human software designer is a permanent head to head collision with bugs. The ensuing evolutionary process classifies as Lamarckian since every developer does his or her best to improve productivity. So, our experience with Lamarckian evolution that deals with complex design is that it is plagued with bugs. Life is among the most complex marvels that exist or may exist. Thus, if one supposes that life on Earth was the fruit of a helped evolutionary process, one must expect a very rich record of bugs whose fixing generates more bugs. These are not found neither in the fossil record nor in extant populations.

We conclude that the argument of bugs falsifies as the Darwinian as the Lamarckian theories of evolution relying on an evolutionary process run on the Earth. Our arguments can be extended to falsify most sorts of creationism because humans are very smart creators but nevertheless they cannot design software without committing mountains of bugs whose correction generate more bugs.

66 *The mystery of life*

Life is the epitome of evolvable software. Its evolvability is so immense that it is considered in many circles that the obvious explanation of life is that it arouse by evolution and that to challenge this premise is insane.

Anymore.

There is more pleasure for the mind in thinking of life as a mystery than in accepting cheap explanations based on Darwinian or Lamarckian evolution. Now, what is a mystery? A mystery is a mystery than can only be killed by a greater mystery.

6.2 Conclusion

Evolution is a tool to solve every kind of problems. But according to modern standards this is anymore sufficient: we need evolvable solutions that are those that are easily adapted to solve more complex problems. Experience shows that this is usually not the case: one one side, evolution can be programmed as bad as desired and on the other, most solutions are not evolvable. Now, life is the epitome of a bunch of evolvable solutions, a marvel that wonders every expert without a single exception: Where does it come from? We can say with certainty that it did not appear by Darwinian or Lamarckian (helped) evolution because there is no track of malfunctions or malformations neither along the fossil record nor in extant populations. So, life is a mystery. In recognizing this, the mind attains peace because a mystery can only be killed by a greater mystery.

Chapter 7

Hearts and minds

`Lamarck, Darwin, Mendel...`

7.1 Introduction

Science is a project to which one contributes with the best although some contributions are more appreciated than others. In regard with the formulation of the Evolutionary Theory, Lamarck will share with Darwin a preferential site (Rens, 2009). The work of Mendel sparked the research of the mechanism of inheritance, thanks to which the theory has been reformulated in a complete materialist basis. We contend that the vision of Lamarck was enlightened by Alchemy as a spiritualist credo, that Mendel inaugurated creationism and that the Evolutionary Theory of the modern international scientific literature is a flaw. At the heart of our discussion we posit the concept of **twisted randomness**.

7.2 The transformism of Lamarck

The first person that tried to formulate a scientific evolutionary theory was Lamarck.

67 ***Reinventing Lamarck***

Lamarck (1744-1829) is difficult to understand (compare Gould, 2002, pags 170-192). The problem is that his writings were addressed to people in the nascent science with their demands on objectiveness and logical reasoning while his ideas were illuminated with alchemy. This connection is clearly stated in his *Recherches sur les causes de principaux faits physiques, Tome premier (1794)* , where he organizes his perception of reality around the Earth, the Water, the Air and the Fire.

We can witness how alchemy enters into his mechanistically oriented thought if we pay attention to the next remarks of his *Philosophie zoologique (1809), pág 398-400*:

1) que toute l' opération de la nature pour former ses créations directes, consiste à organiser en tissu cellulaire les petites masses de matière gélatineuse ou mucilagineuse qu' elle trouve à sa disposition et dans des circonstances favorables; à remplir ces petites masses celluleuses de fluides contenables ; et à les vivifier en mettant ces fluides contenables en mouvement, à l' aide des fluides subtils excitateurs qui y affluent sans cesse des milieux environnans;

2) que le tissu cellulaire est la gangue dans laquelle toute organisation a été formée, et au milieu de laquelle les différens organes se sont successivement développés, par la voie du mouvement des fluides contenables qui ont graduellement modifié ce tissu cellulaire;

3) qu' effectivement, le propre du mouvement des fluides dans les parties souples des corps vivans qui les contiennent, est de s' y frayer des routes, des lieux de dépôt et des issues; d' y créer des canaux, et par suite des organes divers; d' y varier ces canaux et ces organes à raison de la diversité, soit des mouvemens, soit de la nature des fluides qui y donnent lieu et qui s' y modifient ; enfin, d' agrandir, d' allonger, de diviser et de solidifier graduellement ces canaux et ces organes par les matières qui se forment et se séparent sans cesse des fluides essentiels qui y sont en mouvement; matières dont une partie s' assimile et s' unit aux organes, tandis que l' autre est rejetée au dehors;

4) qu' enfin, le propre du mouvement organique est, non-seulement de développer l' organisation, d' étendre les parties et de donner lieu à l' accroissement, mais encore de multiplier les organes et les fonctions à remplir. Après avoir exposé ces grandes considérations qui me semblent présenter des vérités incontestables, et cependant jusqu' à ce jour inaperçues, j'examinerai quelles sont les facultés communes à tous les corps vivans, et conséquemment à tous les animaux ; ensuite je passerai en revue les principales de celles qui sont nécessairement particulières à certains animaux, les autres ne pouvant nullement en être doués. J' ose le dire, c' est un abus très-nuisible à l' avancement de nos connoissances physiologiques, que de supposer inconsidérément que tous les animaux, sans exception, possèdent les mêmes organes et jouissent des mêmes facultés; comme si la nature étoit forcée d' employer partout les mêmes moyens pour arriver à son but. Dès que, sans s'arrêter à la considération des faits, il n' en coûte que quelques actes de l' imagination pour créer des principes, que ne suppose-t-on de suite que tous les corps vivans possèdent généralement les mêmes organes, et jouissent en conséquence des mêmes facultés? Un objet que je n' ai pas dû négliger dans cette seconde partie de mon ouvrage, est la considération des résultats immédiats de la vie dans un corps. Or, je puis faire voir que ces résultats donnent lieu à des combinaisons entre des

principes qui, sans cette circonstance, ne se fussent jamais unis ensemble. Ces combinaisons se surchargent même de plus en plus, à mesure que l' énergie vitale augmente; en sorte que, dans les animaux les plus parfaits, elles offrent une grande complication et une surcharge considérable dans leurs principes combinés. Ainsi les corps vivans constituent, par le pouvoir de la vie qu' ils possèdent, le principal moyen que la nature emploie pour faire exister une multitude de composés différens qui n' eussent jamais eu lieu sans cette cause remarquable.

Lamarck command somewhere to interpret this and related passages following a clean materialistic frame (pag 453):

Que la vie en elle-même n'est qu'un phénomène physique, qui donne graduellement lieu à beaucoup d'autres, et qui résulte uniquement des relations qui existent entre les parties contenantes et appropriées d'un corps, les fluides contenus qui y sont en mouvement, et la cause excitatrice des mouvemens et des changemens qui s'y opèrent.

but the problem is that he nowhere explains exactly the nature of the factor that excites corporal fluids. Why? Given his alchemist orientation, we must conclude that that happens because he considers that fluids are the field of action of a Water as a cosmic power: please, read again these two quotations from the stand of Alchemy and look there for Water, Fire and Terra. Next add Air (in the respiration of every living being) to close the sacred square of the 4 elements.

So, Lamarck should had filtered his true beliefs to project them down to science. The result was not a highly clear one. Let us invite a reconstruction of his thoughts, just to understand his heart.

The history of alchemy boils down to Aristotle (300 BC), who adhered to the by that time current belief that there are only four fundamental elements: fire, air, water and earth (Lienhard, 1997).

Contrary to modern chemistry, alchemy was not a materialistic vision but an spiritualist one. To learn about this, we have two sources: The Emerald Tablet of Hermes (The Alchemy Web Site, 2011) and the Kybalion (The Three Initiates, 1908). To reconstruct Lamarck, we find it suitable to rely on our own interpretation of the second, which is a book that presumingly contains the teachings of spiritual masters from Egypt and Greece, and that guided through ages some schools of hidden spiritualism. Our considerations follow:

Te first truth is that the universe is not material but rather it is mental: *all the phenomenal world or universe is simply a Mental Creation of THE ALL, subject to the Laws of Created Things, and that the universe, as a whole, and in its parts or units, has its existence in the Mind of THE ALL, in which Mind we "live and move and have our being.*" (The Kybalion, pag 9).

Given that the ALL is mind and that the Universe is mental, the ensuing preoccupation is to decide how we will judge about spiritual things, in which the mind governs, being that we are so material. The solution is given by the principle of Correspondence:

As above, so below; as below, so above.

This principle enables us to reason about the ALL, nature included, on a secure background: all we need is to look attentively. This principle also allows us to study the material plane of existence as if it were isolated of the other planes. So, the purest materialism does not contradict ancient wisdom, it just explores one plane of the All and so Lamarck made his best to impulse materialism and to combat vitalism. Nevertheless, one must not deceive oneself: if water and fluids are the cause of life on the material plane, it is just because Water + Fire are the source of life in the spiritual one.

Now, the material plane and the spiritual one go shoulder to shoulder, although in a tempered interaction such as it is prescribed by the principle of Vibration that tells us that

Nothing stands still, everything moves, evolves, everything vibrates.

It might be argued that the Principle of Vibration predicted quantum mechanics as early as 2000 years ago. The problem is that the principle of vibration is qualitative but quantum mechanics is quantitative: there are infinitely many different forms of quantum mechanics while only some ten to twenty equations make sense in the quantum mechanics obeyed by nature but none of these has been predicted by ancient wisdom.

On the other hand, if everything moves and evolves, robust truths are best built of processes rather than on stable structures. So, a doctrine about the origin of life must have at its very heart the feeling of evolution and of change. In fact, Lamarck proposed his transformism to explain the complexity and diversity of life: since life begets life and complex life is the only source of complex life, and given that there are varying degrees of complexity among living beings, one arrives to transformism or evolution as a logical necessity.

Evolution is the product of two main contrary forces, such as it is written in the principle of Polarity:

Everything is the result of the struggle between two forces that opposite one another.

So is life, which can be understood as the result of the struggle between Simplicity and Complexity. But, beware, the material version of Simplicity amounts to the most simple organisms that arise spontaneously. Now, life exists because *the very essence of the All is to beget things like it itself,* such as it is said by the principle of Generation, which by the principle of Polarity uses to appear in two forms that are universally called masculine and feminine. The struggle between

simplicity and complexity has various solutions which are the diverse species and that unfold in time: so we have evolution along time from simplicity to complexity. This happens not by chance, because *there is nothing left to chance* as the principle of Cause and Effect reads, but with Rhythm, *with measured motion between possibly many pairs of extremes* that try to enrich the process of the generation of life.

We see that a rather interesting ideology was transmitted by secret loggias. So, we must ask ourselves: why did they need to remain secret? The reason is that if one believes this credo to full terms, one must conclude, for instance, that God and Satan are the two dominant poles that govern the unfolding of the All and that the first has no sense without the second because the All cannot distinguish them apart. All this is contrary and insulting to dominant religions, Christianity, Islam and Judaism, according to which God is good and holy by its very essence. Moreover, Alchemy just makes Satan equal to God, an equation that unveils the longing of Satan to be worshiped just as God is.

On the other hand, the principle of correspondence invites one to investigate (pag 386-388):

Ce sont, sans doute, des objets bien importans, que ceux de rechercher en quoi consiste ce qu' on nomme la vie dans un corps ; quelles sont les conditions essentielles de l' organisation pour que la vie puisse exister ; quelle est la source de cette force singulière qui donne lieu aux mouvemens vitaux tant que l' état de l' organisation le permet ; enfin, comment les différens phénomènes qui résultent de la présence et de la durée de la vie dans un corps peuvent s' opérer, et donner à ce corps les facultés qu' on y observe ; mais aussi, de tous les problèmes que l' on puisse se proposer, ce sont, sans contredit, ceux qui sont les plus difficiles à résoudre. Il étoit, ce me semble, beaucoup plus aisé de déterminer le cours des astres observés dans l' espace, et de reconnoître les distances, les grosseurs, les masses et les mouvemens des planètes qui appartiennent au système de notre soleil, que de résoudre le problème relatif à la source de la vie dans les corps qui en sont doués, et, conséquemment, à l' origine ainsi qu' à la production des différens corps vivans qui existent.

Quelque difficile que soit ce grand sujet de recherches, les difficultés qu' il nous présente ne sont point insurmontables ; car il n' est question, dans tout ceci, que de phénomènes purement physiques. or, il est évident que les phénomènes dont il s' agit ne sont, d' une part, que les résultats directs des relations de différens corps entre eux, et que les suites d' un ordre et d' un état de choses qui, dans certains d' entre eux, donnent lieu à ces relations; et de l' autre part, qu' ils résultent de mouvemens excités dans les parties de ces corps, par une force dont il est possible

d' apercevoir la source.

In that line, and keeping in mind that the evolutionary process is by its essence an spiritual process intertwining simplicity and complexity, we ask: how would the evolutionary process be perceived from our material plane? Lamarck saw, as every human beings sees, that making corrections leads to majestic deeds and so he proposed that nature has a mechanism to make corrections to improve life. Now, we correct precisely what we need and we trow away what we do not need. So is Nature. His introduction of this theme is both simple and smart:

à la vérité, depuis assez long-temps on a remarqué l' influence des différens états de notre organisation sur notre caractère, nos penchans, nos actions, et même nos idées ; mais il me semble que personne encore n' a fait connoître celle de nos actions et de nos habitudes sur notre organisation même. (Lamarck, 1809, pag 243).

Next, he invites us to recognize that the organization is plugged to the milieu and is modified by habits (l.c., 250).:

Ceux qui ont beaucoup observé, et qui ont consulté les grandes collections, ont pu se convaincre qu' à mesure que les circonstances d' habitation, d' exposition, de climat, de nourriture, d' habitude de vivre, etc., viennent à changer ; les caractères de taille, de forme, de proportion entre les parties, de couleur, de consistance, d' agilité et d' industrie pour les animaux, changent proportionnellement.

Ce que la nature fait avec beaucoup de temps, nous le faisons tous les jours, en changeant nous-mêmes subitement, par rapport à un végétal vivant, les circonstances dans lesquelles lui et tous les individus de son espèce se rencontroient. Tous les botanistes savent que les végétaux qu' ils transportent de leur lieu natal dans les jardins pour les y cultiver, y subissent peu à peu des changemens qui les rendent à la fin méconnoissables. Beaucoup de plantes très-velues naturellement, y deviennent glabres, ou à peu près; quantité de celles qui étoient couchées et traînantes, y voient redresser leur tige ; d' autres y perdent leurs épines ou leurs aspérités ; d' autres encore, de l' état ligneux et vivace que leur tige possédoit dans les climats chauds qu' elles habitoient, passent, dans nos climats, à l'état herbacé, et parmi elles, plusieurs ne sont plus que des plantes annuelles ; enfin, les dimensions de leurs parties y subissent elles-mêmes des changemens très-considérables. Ces effets des changemens de circonstances sont tellement reconnus, que les botanistes n'aiment point à décrire les plantes de jardins, à moins qu' elles n' y soient nouvellement cultivées.

The passage from habits of individuals to inheritance is declared as follows (l.c., pag 261):

Première Loi.

Dans tout animal qui n' a point dépassé le terme de ses développemens, l' emploi plus fréquent et soutenu d' un organe quelconque, fortifie peu à peu cet organe, le développe, l' agrandit, et lui donne une puissance proportionnée à la durée de cet emploi ; tandis que le défaut constant d' usage de tel organe, l' affoiblit insensiblement, le détériore, diminue progressivement ses facultés, et finit par le faire disparoître.

Deuxième Loi.

Tout ce que la nature a fait acquérir ou perdre aux individus par l' influence des circonstances où leur race se trouve depuis long-temps exposée, et, par conséquent, par l' influence de l' emploi prédominant de tel organe, ou par celle d' un défaut constant d' usage de telle partie ; elle le conserve par la génération aux nouveaux individus qui en proviennent, pourvu que les changemens acquis soient communs aux deux sexes, ou à ceux qui ont produit ces nouveaux individus.

And, what can we say about the certainty of these laws? The answer is determinant: structure follows habits (l.c., pag 261)

Ce sont là deux vérités constantes qui ne peuvent être méconnues que de ceux qui n' ont jamais observé ni suivi la nature dans ses opérations, ou que de ceux qui se sont laissés entraîner à l' erreur que je vais combattre.

Les naturalistes ayant remarqué que les formes des parties des animaux, comparées aux usages de ces parties, sont toujours parfaitement en rapport, ont pensé que les formes et l' état des parties en avoient amené l' emploi : or, c' est là l' erreur; car il est facile de démontrer, par l' observation, que ce sont, au contraire, les besoins et les usages des parties qui ont développé ces mêmes parties, qui les ont même fait naître lorsqu' elles n'existoient pas, et qui, conséquemment, ont donné lieu à l' état où nous les observons dans chaque animal.

7.3 Lavoisier

One cannot avoid to get struck at the fact that Lamarck published in 1809 his Alchemy based theory of transformism given the previous publication by Lavoisier (1789) of his *Traité élémentaire de chimie,* in which he empowers a purely materialist vision of transformation.

68 ***The glorious birth of modern chemistry:***

Let us see how Lavoisier builds his ideas upon the hardest science, that one that we are used to love so much (Lavoisier, l.c., Pag 1-2):

nous ne pensons qu'avec le secours des mots ; que les langues sont de véritables méthodes analytiques ; que l'algèbre la plus simple, la plus exacte et la mieux adaptée à son objet de toutes les manières de s'énoncer, est à la fois une langue

et une méthode analytique ; enfin, que l'art de raisonner se réduit en une langue bien faite. Et en effet, tandis que je croyais ne m'occuper que de nomenclature, tandis que je n'avais pour objet que de perfectionner le langage de la chimie, mon ouvrage s'est transformé insensiblement entre mes mains, sans qu'il m'ait été possible de m'en défendre, en un traité élémentaire de chimie. L'impossibilité d'isoler la nomenclature de la science et la science de la nomenclature tient à ce que toute science physique est nécessairement formée de trois choses : la série des faits qui constituent. la science ; les idées qui les rappellent ; les mots qui les expriment. Le mot doit faire naître l'idée ; l'idée doit peindre le fait : ce sont trois empreintes d'un même cachet ; et, comme ce sont les mots qui conservent les idées et qui les transmettent, il en résulte qu'on ne peut perfectionner le langage sans perfectionner la science, ni la science sans le langage, et que, quelque certains que fussent les faits, quelque justes que fussent les idées qu'ils auraient fait naître, ils ne transmettraient encore que des impressions fausses, si nous n'avions pas des expressions exactes pour les rendre.

69 *Alchemy is broken down*

It is very nice to see the way as the sacred square of Alchemy is ruled out by observation and experiment and instead a science as a very complex project is born to light (pags 6-9):

On ne manquera pas d'être surpris de ne point trouver dans un traité élémentaire de chimie un chapitre sur les parties constituantes et élémentaires des corps ; mais je ferai remarquer ici que cette tendance que nous avons à vouloir que tous les corps de la nature ne soient composés que de trois ou quatre éléments tient à un préjugé qui nous vient originairement des philosophes grecs. L'admission de quatre éléments, qui, par la variété de leurs proportions, composent tous les corps que nous connaissons, est une pure hypothèse, imaginée longtemps avant qu'on eût les premières notions de la physique expérimentale et de la chimie. On n'avait point encore de faits, et l'on formait des systèmes ; et aujourd'hui que nous avons rassemblé des faits, il semble que nous nous efforcions de les repousser, quand ils ne cadrent pas avec nos préjugés ; tant il est vrai que le poids de l'autorité de ces pères de la philosophie humaine se fait encore sentir, et qu'elle pèsera sans doute encore sur les générations à venir. Une chose très-remarquable, c'est que, tout en enseignant la doctrine des quatre éléments, il n'est aucun chimiste qui, par la force des faits, n'ait été conduit à en admettre un plus grand nombre. Les premiers chimistes qui ont écrit depuis le renouvellement des lettres regardaient le soufre et le sel comme des substances, élémentaires qui entraient dans la combinaison d'un grand nombre de corps : ils reconnaissaient donc l'existence de six éléments au lieu de quatre. Becher admettait trois terres, et cétait de leur combinaison et

de la différence des proportions que résultait suivant lui, la différence qui existe entre les substances métalliques. Stahl a modifié ce système : tous les chimistes qui lui ont succédé se sont permis d'y faire des changements, même d'en imaginer d'autres, mais tous se sont laissé entraîner à l'esprit de leur siècle, qui se contentait d'assertions sans preuves, ou du moins qui regardait souvent comme telles de très-légères probabilités. Tout ce qu'on peut dire sur le nombre et sur la nature des éléments se borne, suivant moi, à des discussions purement métaphysiques : ce sont des problèmes indéterminés qu'on se propose de résoudre, qui sont susceptibles dune infinité de solutions, mais dont il est très-probable qu'aucune en particulier n'est d'accord avec la nature. Je me contenterai donc de dire que, si par le nom d'éléments nous entendons désigner les molécules simples et indivisibles qui composent les corps, il est probable que nous ne les connaissons pas : que, si, au contraire, nous attachons au nom d'éléments ou de principes des corps l'idée du dernier terme auquel parvient l'analyse, toutes les substances que nous n'avons encore pu décomposer par aucun moyen sont pour nous des éléments ; non pas que nous puissions assurer que ces corps, que nous regardons comme simples, ne soient pas eux-mêmes composés de deux ou même d'un plus grand nombre de principes ; mais, puisque ces principes ne se séparent jamais, ou plutôt puisque nous n'avons aucun moyen de les séparer, ils agissent à notre égard à la manière des corps simples, et nous ne devons les supposer composés qu'au moment où l'expérience et l'observation nous en auront fourni la preuve.

Now, we must ask ourselves: why Lamarck did not gave up alchemy in favor of chemistry as instructed people did?

The following explanation might help: alchemy is footed on the belief that there is a whole universe beyond ordinary matter and continues with the terrifying belief that that world can be understood and that moreover is rather simple. Thus, alchemy is very attractive for people with a spiritualist tendency that enjoy intellectual discussions.

7.4 The hallmark of Darwin

While the nice materialism of Lavoisier is incontestable, one must not decide that Lamarck is completely wrong. Rather, one must scrutinize his ideas to pick promissory ones. In particular, one must separate an idea from its mechanistic implementation. Darwin chose to belief that the transformism of Lamarck was good while he proposes his own mechanistic implementation to build complexity and perfection, variability, adaptation and coadaptation. His proposal is simple, reliable and

perfectly materialistic, although with some errors that are better understood as vacuums or open problems to be investigated.

70 *The theory of Darwin*

The Evolutionary theory consists of three points (Darwin, 1859):

1. Populations always have variability.
2. Populations are always under pressure be from the milieu or from siblings. So, individuals struggle for existence.
3. Some individuals do it better than others in the struggle for life and eventually leave an offspring whose survivors are better than themselves. In this way, populations get adapted to their milieu, acquire complexity and beauty.

Let us see from first hand how Darwin supports these three points:

71 *Variability in nature*

In spite of similitudes, the members of a human family are all different, so one readily accepts that nature is endowed with huge levels of variability at any level of inquiry. Is there some order, some science, behind this quench for diversity? Yes, an instance follows (chapter II):

From looking at species as only strongly-marked and well-defined varieties, I was led to anticipate that the species of the larger genera in each country would oftener present varieties, than the species of the smaller genera; for wherever many closely related species (i.e. species of the same genus) have been formed, many varieties or incipient species ought, as a general rule, to be now forming. Where many large trees grow, we expect to find saplings. Where many species of a genus have been formed through variation, circumstances have been favourable for variation; and hence we might expect that the circumstances would generally be still favourable to variation. On the other hand, if we look at each species as a special act of creation, there is no apparent reason why more varieties should occur in a group having many species, than in one having few.

To test the truth of this anticipation I have arranged the plants of twelve countries, and the coleopterous insects of two districts, into two nearly equal masses, the species of the larger genera on one side, and those of the smaller genera on the other side, and it has invariably proved to be the case that a larger proportion of the species on the side of the larger genera present varieties, than on the side of the smaller genera. Moreover, the species of the large genera which present

any varieties, invariably present a larger average number of varieties than do the species of the small genera. Both these results follow when another division is made, and when all the smallest genera, with from only one to four species, are absolutely excluded from the tables. These facts are of plain signification on the view that species are only strongly marked and permanent varieties; for whenever many species of the same genus have been formed, or where, if we may use the expression, the manufactory of species has been active, we ought generally to find the manufactory still in action, more especially as we have every reason to believe the process of manufacturing new species to be a slow one. And this certainly is the case, if varieties be looked at as incipient species; for my tables clearly show as a general rule that, wherever many species of a genus have been formed, the species of that genus present a number of varieties, that is of incipient species, beyond the average. It is not that all large genera are now varying much, and are thus increasing in the number of their species, or that no small genera are now varying and increasing; for if this had been so, it would have been fatal to my theory; inasmuch as geology plainly tells us that small genera have in the lapse of time often increased greatly in size; and that large genera have often come to their maxima, declined, and disappeared. All that we want to show is, that where many species of a genus have been formed, on an average many are still forming; and this holds good.

72 ***Struggle for existence***

Life is not easy, nor for me neither for you. And this happens to every living being (chapter III):

I should premise that I use the term Struggle for Existence in a large and metaphorical sense, including dependence of one being on another, and including (which is more important) not only the life of the individual, but success in leaving progeny. Two canine animals in a time of dearth, may be truly said to struggle with each other which shall get food and live. But a plant on the edge of a desert is said to struggle for life against the drought, though more properly it should be said to be dependent on the moisture. A plant which annually produces a thousand seeds, of which on an average only one comes to maturity, may be more truly said to struggle with the plants of the same and other kinds which already clothe the ground. The missletoe is dependent on the apple and a few other trees, but can only in a far-fetched sense be said to struggle with these trees, for if too many of these parasites grow on the same tree, it will languish and die. But several seedling missletoes, growing close together on the same branch, may more truly be said to struggle with each other. As the missletoe is disseminated by birds, its existence depends

on birds; and it may metaphorically be said to struggle with other fruit-bearing plants, in order to tempt birds to devour and thus disseminate its seeds rather than those of other plants. In these several senses, which pass into each other, I use for convenience sake the general term of struggle for existence. A struggle for existence inevitably follows from the high rate at which all organic beings tend to increase. Every being, which during its natural lifetime produces several eggs or seeds, must suffer destruction during some period of its life, and during some season or occasional year, otherwise, on the principle of geometrical increase, its numbers would quickly become so inordinately great that no country could support the product. Hence, as more individuals are produced than can possibly survive, there must in every case be a struggle for existence, either one individual with another of the same species, or with the individuals of distinct species, or with the physical conditions of life. It is the doctrine of Malthus applied with manifold force to the whole animal and vegetable kingdoms; for in this case there can be no artificial increase of food, and no prudential restraint from marriage. Although some species may be now increasing, more or less rapidly, in numbers, all cannot do so, for the world would not hold them.

73 ***Natural selection***

The difficulty in the struggle for life can translate itself in differential rates of surviving and reproduction (chapter IV):

How will the struggle for existence, discussed too briefly in the last chapter, act in regard to variation? Can the principle of selection, which we have seen is so potent in the hands of man, apply in nature? I think we shall see that it can act most effectually. Let it be borne in mind in what an endless number of strange peculiarities our domestic productions, and, in a lesser degree, those under nature, vary; and how strong the hereditary tendency is. Under domestication, it may be truly said that the, whole organisation becomes in some degree plastic. Let it be borne in mind how infinitely complex and close-fitting are the mutual relations of all organic beings to each other and to their physical conditions of life. Can it, then, be thought improbable, seeing that variations useful to man have undoubtedly occurred, that other variations useful in some way to each being in the great and complex battle of life, should sometimes occur in the course of thousands of generations? If such do occur, can we doubt (remembering that many more individuals are born than can possibly survive) that individuals having any advantage, however slight, over others, would have the best chance of surviving and of procreating their kind? On the other hand, we may feel sure that any variation in the least degree injurious would be rigidly destroyed. This preservation of favourable variations and the rejection of injurious variations, I call Natural Selection. Variations

neither useful nor injurious would not be affected by natural selection, and would be left a fluctuating element, as perhaps we see in the species called polymorphic.

74 ***Assessing the theory***

One must at last assess how well the theory fits data. In fact, it is so good that the idea of a direct creation of the diverse species seems now as making fun of God (chapter XII) :

The most striking and important fact for us in regard to the inhabitants of islands, is their affinity to those of the nearest mainland, without being actually the same species. Numerous instances could be given of this fact. I will give only one, that of the Galapagos Archipelago, situated under the equator, between 500 and 600 miles from the shores of South America. Here almost every product of the land and water bears the unmistakeable stamp of the American continent. There are twenty-six land birds, and twenty-five of those are ranked by Mr Gould as distinct species, supposed to have been created here; yet the close affinity of most of these birds to American species in every character, in their habits, gestures, and tones of voice, was manifest. So it is with the other animals, and with nearly all the plants, as shown by Dr. Hooker in his admirable memoir on the Flora of this archipelago. The naturalist, looking at the inhabitants of these volcanic islands in the Pacific, distant several hundred miles from the continent, yet feels that he is standing on American land. Why should this be so? why should the species which are supposed to have been created in the Galapagos Archipelago, and nowhere else, bear so plain a stamp of affinity to those created in America? There is nothing in the conditions of life, in the geological nature of the islands, in their height or climate, or in the proportions in which the several classes are associated together, which resembles closely the conditions of the South American coast: in fact there is a considerable dissimilarity in all these respects. On the other hand, there is a considerable degree of resemblance in the volcanic nature of the soil, in climate, height, and size of the islands, between the Galapagos and Cape de Verde Archipelagos: but what an entire and absolute difference in their inhabitants! The inhabitants of the Cape de Verde Islands are related to those of Africa, like those of the Galapagos to America. I believe this grand fact can receive no sort of explanation on the ordinary view of independent creation; whereas on the view here maintained, it is obvious that the Galapagos Islands would be likely to receive colonists, whether by occasional means of transport or by formerly continuous land, from America; and the Cape de Verde Islands from Africa; and that such colonists would be liable to modifications; the principle of inheritance still betraying their original birthplace. Many analogous facts could be given: indeed it is an almost universal rule that the endemic productions of islands are related

to those of the nearest continent, or of other near islands. The exceptions are few, and most of them can be explained. Thus the plants of Kerguelen Land, though standing nearer to Africa than to America, are related, and that very closely, as we know from Dr. Hooker's account, to those of America: but on the view that this island has been mainly stocked by seeds brought with earth and stones on icebergs, drifted by the prevailing currents, this anomaly disappears. New Zealand in its endemic plants is much more closely related to Australia, the nearest mainland, than to any other region: and this is what might have been expected; but it is also plainly related to South America, which, although the next nearest continent, is so enormously remote, that the fact becomes an anomaly. But this difficulty almost disappears on the view that both New Zealand, South America, and other southern lands were long ago partially stocked from a nearly intermediate though distant point, namely from the antarctic islands, when they were clothed with vegetation, before the commencement of the Glacial period. The affinity, which, though feeble, I am assured by Dr. Hooker is real, between the flora of the south-western corner of Australia and of the Cape of Good Hope, is a far more remarkable case, and is at present inexplicable: but this affinity is confined to the plants, and will, I do not doubt, be some day explained. The law which causes the inhabitants of an archipelago, though specifically distinct, to be closely allied to those of the nearest continent, we sometimes see displayed on a small scale, yet in a most interesting manner, within the limits of the same archipelago. Thus the several islands of the Galapagos Archipelago are tenanted, as I have elsewhere shown, in a quite marvellous manner, by very closely related species; so that the inhabitants of each separate island, though mostly distinct, are related in an incomparably closer degree to each other than to the inhabitants of any other part of the world. And this is just what might have been expected on my view, for the islands are situated so near each other that they would almost certainly receive immigrants from the same original source, or from each other. But this dissimilarity between the endemic inhabitants of the islands may be used as an argument against my views; for it may be asked, how has it happened in the several islands situated within sight of each other, having the same geological nature, the same height, climate, &c., that many of the immigrants should have been differently modified, though only in a small degree. This long appeared to me a great difficulty: but it arises in chief part from the deeply-seated error of considering the physical conditions of a country as the most important for its inhabitants; whereas it cannot, I think, be disputed that the nature of the other inhabitants, with which each has to compete, is at least as important, and generally a far more important element of success. Now if we look to those inhabitants of the Galapagos Archipelago which are found in other parts of the world (having on one side for the moment the endemic species, which cannot be here fairly included, as we are considering how they have come to be

modified since their arrival), we find a considerable amount of difference in the several islands. This difference might indeed have been expected on the view of the islands having been stocked by occasional means of transport a seed, for instance, of one plant having been brought to one island, and that of another plant to another island. Hence when in former times an immigrant settled on any one or more of the islands, or when it subsequently spread from one island to another, it would undoubtedly be exposed to different conditions of life in the different islands, for it would have to compete with different sets of organisms: a plant, for instance, would find the best-fitted ground more perfectly occupied by distinct plants in one island than in another, and it would be exposed to the attacks of somewhat different enemies. If then it varied, natural selection would probably favour different varieties in the different islands. Some species, however, might spread and yet retain the same character throughout the group, just as we see on continents some species' spreading widely and remaining the same. The really surprising fact in this case of the Galapagos Archipelago, and in a lesser degree in some analogous instances, is that the new species formed in the separate islands have not quickly spread to the other islands. But the islands, though in sight of each other, are separated by deep arms of the sea, in most cases wider than the British Channel, and there is no reason to suppose that they have at any former period been continuously united. The currents of the sea are rapid and sweep across the archipelago, and gales of wind are extraordinarily rare; so that the islands are far more effectually separated from each other than they appear to be on a map. Nevertheless a good many species, both those found in other parts of the world and those confined to the archipelago, are common to the several islands, and we may infer from certain facts that these have probably spread from some one island to the others. But we often take, I think, an erroneous view of the probability of closely allied species invading each other's territory, when put into free intercommunication. Undoubtedly if one species has any advantage whatever over another, it will in a very brief time wholly or in part supplant it; but if both are equally well fitted for their own places in nature, both probably will hold their own places and keep separate for almost any length of time. Being familiar with the fact that many species, naturalised through man's agency, have spread with astonishing rapidity over new countries, we are apt to infer that most species would thus spread; but we should remember that the forms which become naturalised in new countries are not generally closely allied to the aboriginal inhabitants, but are very distinct species, belonging in a large proportion of cases, as shown by Alph. de Candolle, to distinct genera. In the Galapagos Archipelago, many even of the birds, though so well adapted for flying from island to island, are distinct on each; thus there are three closely-allied species of mocking-thrush, each confined to its own island. Now let us suppose the mocking-thrush of Chatham Island to be blown to Charles Island, which has

its own mocking-thrush: why should it succeed in establishing itself there? We may safely infer that Charles Island is well stocked with its own species, for annually more eggs are laid there than can possibly be reared; and we may infer that the mocking-thrush peculiar to Charles Island is at least as well fitted for its home as is the species peculiar to Chatham Island. Sir C. Lyell and Mr. Wollaston have communicated to me a remarkable fact bearing on this subject; namely, that Madeira and the adjoining islet of Porto Santo possess many distinct but representative land-shells, some of which live in crevices of stone; and although large quantities of stone are annually transported from Porto Santo to Madeira, yet this latter island has not become colonised by the Porto Santo species: nevertheless both islands have been colonised by some European land-shells, which no doubt had some advantage over the indigenous species. From these considerations I think we need not greatly marvel at the endemic and representative species, which inhabit the several islands of the Galapagos Archipelago, not having universally spread from island to island. In many other instances, as in the several districts of the same continent, pre-occupation has probably played an important part in checking the commingling of species under the same conditions of life. Thus, the south-east and south-west corners of Australia have nearly the same physical conditions, and are united by continuous land, yet they are inhabited by a vast number of distinct mammals, birds, and plants. The principle which determines the general character of the fauna and flora of oceanic islands, namely, that the inhabitants, when not identically the same, yet are plainly related to the inhabitants of that region whence colonists could most readily have been derived, the colonists having been subsequently modified and better fitted to their new homes, is of the widest application throughout nature. We see this on every mountain, in every lake and marsh. For Alpine species, excepting in so far as the same forms, chiefly of plants, have spread widely throughout the world during the recent Glacial epoch, are related to those of the surrounding lowlands; thus we have in South America, Alpine humming-birds, Alpine rodents, Alpine plants, & c., all of strictly American forms, and it is obvious that a mountain, as it became slowly upheaved, would naturally be colonised from the surrounding lowlands. So it is with the inhabitants of lakes and marshes, excepting in so far as great facility of transport has given the same general forms to the whole world. We see this same principle in the blind animals inhabiting the caves of America and of Europe. Other analogous facts could be given. And it will, I believe, be universally found to be true, that wherever in two regions, let them be ever so distant, many closely allied or representative species occur, there will likewise be found some identical species, showing, in accordance with the foregoing view, that at some former period there has been intercommunication or migration between the two regions. And wherever many closely-allied species occur, there will be found many forms which some naturalists rank as dis-

tinct species, and some as varieties; these doubtful forms showing us the steps in the process of modification.

75 ***Concluding remark***

Darwin looks in hindsight over his theory with is huge explicative power and concludes that:

It is interesting to contemplate an entangled bank, clothed with many plants of many kinds, with birds singing on the bushes, with various insects flitting about, and with worms crawling through the damp earth, and to reflect that these elaborately constructed forms, so different from each other, and dependent on each other in so complex a manner, have all been produced by laws acting around us. These laws, taken in the largest sense, being Growth with Reproduction; inheritance which is almost implied by reproduction; Variability from the indirect and direct action of the external conditions of life, and from use and disuse; a Ratio of Increase so high as to lead to a Struggle for Life, and as a consequence to Natural Selection, entailing Divergence of Character and the Extinction of less-improved forms. Thus, from the war of nature, from famine and death, the most exalted object which we are capable of conceiving, namely, the production of the higher animals, directly follows. There is grandeur in this view of life, with its several powers, having been originally breathed into a few forms or into one; and that, whilst this planet has gone cycling on according to the fixed law of gravity, from so simple a beginning endless forms most beautiful and most wonderful have been, and are being, evolved.

76 ***The merits of Darwin***

What do we ought to Darwin?

Darwin did not invented the idea of scientific evolution, which was due to Lamarck, nor he refutes the theory of the inheritance of acquired characters (all quotations until the end of this section are taken from the Preface of his *Origin*):

Lamarck was the first man whose conclusions on the subject excited much attention. This justly-celebrated naturalist first published his views in 1801; he much enlarged them in 1809 in his "Philosophie Zoologique,' and subsequently, in 1815, in the Introduction to his "Hist. Nat. des Animaux sans Vertébres.' In these works he upholds the doctrine that species, including man, are descended from other species. He first did the eminent service of arousing attention to the probability of all change in the organic, as well as in the inorganic world, being the result of law, and not of miraculous interposition. Lamarck seems to have been chiefly led to

his conclusion on the gradual change of species, by the difficulty of distinguishing species and varieties, by the almost perfect gradation of forms in certain groups, and by the analogy of domestic productions.

Darwin did not invented the concept of natural selection:

In 1813, Dr W. C. Wells read before the Royal Society 'An Account of a White female, part of whose skin resembled that of a Negro'; but his paper was not published until his famous 'Two Essays upon Dew and Single Vision' appeared in 1818. In this paper he distinctly recognises the principle of natural selection, and this is the first recognition which has been indicated; but he applies it only to the races of man, and to certain characters alone. After remarking that negroes and mulattoes enjoy an immunity from certain tropical diseases, he observes, firstly, that all animals tend to vary in some degree, and, secondly, that agriculturists improve their domesticated animals by selection; and then, he adds, but what is done in this latter case 'by art, seems to be done with equal efficacy, though more slowly, by nature, in the formation of varieties of mankind, fitted for the country which they inhabit. Of the accidental varieties of man, which would occur among the first few and scattered inhabitants of the middle regions of Africa, some one would be better fitted than the others to bear the diseases of the country. This race would consequently multiply, while the others would decrease; not only from their inability to sustain the attacks of disease, but from their incapacity of contending with their more vigorous neighbours. The colour of this vigorous race I take for granted, from what has been already said, would be dark. But the same disposition to form varieties still existing, a darker and a darker race would in the course of time occur: and as the darkest would be the best fitted for the climate, this would at length become the most prevalent; if not the only race, in the particular country in which it had originated.' He then extends these same views to the white inhabitants of colder climates. I am indebted to Mr Rowley, of the United States, for having called my attention, through Mr Brace, to the above passage in Dr Wells' work.

Darwin is not responsible for the universalization of the idea of evolution:

From a circular lately issued it appears that Dr Freke, in 1851 ("Dublin Medical Press,' p. 322), propounded the doctrine that all organic beings have descended from one primordial form. His grounds of belief and treatment of the subject are wholly different from mine; but as Dr Freke has now (1861) published his Essay on the 'Origin of Species by means of Organic Affinity,' the difficult attempt to give any idea of his views would be superfluous on my part.

Darwin is not pioneer in given to the theory of evolution a scientific treatment:

Mr Herbert Spencer, in an Essay (originally published in the 'Leader,' March, 1852, and republished in his 'Essays,' in 1858), has contrasted the theories of the Creation and the Development of organic beings with remarkable skill and force.

He argues from the analogy of domestic productions, from the changes which the embryos of many species undergo, from the difficulty of distinguishing species and varieties, and from the principle of general gradation, that species have been modified; and he attributes the modification to the change of circumstances. The author (1855) has also treated psychology on the principle of the necessary acquirement of each mental power and capacity by gradation.

Darwin was not the first to perceive the immense convincing power of biogeography:

Von Baer, towards whom all zoologists feel so profound a respect, expressed about the year 1859 (see Prof. Rudolph Wagner, a "Zoologisch-Anthropologische Untersuchungen,' 1861, s. 51) his conviction, chiefly grounded on the laws of geographical distribution, that forms now perfectly distinct have descended from a single parent-form.

We see that according to his own words and witness all the ingredients of a scientific theory of evolution (plus a vacuum in the theory of heredity) were already known. What is then the merit of Darwin, if any, that has made of him the most prominent scientist over all ages and disciplines until now?

Apart from appropriateness of time and circumstances, the first thing we see is his great insolence: he aims at explaining the great variability, beauty, adaptation, coadaptation and complexity of nature using a very simple mechanism that is exhaustively explained in the title of his book: *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*.

The second point is that his insolence is not one of character but emerges naturally from his encyclopedic account of facts and stands of analysis which are all intertwined with clarity, art and wisdom.

The third point is that he was so busy in building a pattern to organize myriads of scientific facts that he was unable to test it to destruction although he tried (Chapter VI):

To suppose that the eye, with all its inimitable contrivances for adjusting the focus to different distances, for admitting different amounts of light, and for the correction of spherical and chromatic aberration, could have been formed by natural selection, seems, I freely confess, absurd in the highest possible degree. Yet reason tells me, that if numerous gradations from a perfect and complex eye to one very imperfect and simple, each grade being useful to its possessor, can be shown to exist; if further, the eye does vary ever so slightly, and the variations be inherited, which is certainly the case; and if any variation or modification in the organ be ever useful to an animal under changing conditions of life, then the difficulty of believing that a perfect and complex eye could be formed by natural selection, though insuperable by our imagination, can hardly be considered real.

7.5 Wallace

While Darwin worked in his theory for decades, Wallace seemed to arrive all at sudden to the very same conclusion and at the same time when Darwin decided to release his *Origin*. Nevertheless, that was not by casualty but because of his previous hard work (Wallace, 1855):

The great increase of our knowledge within the last twenty years, both of the present and past history of the organic world, has accumulated a body of facts which should afford a sufficient foundation for a comprehensive law embracing and explaining them all, and giving a direction to new researches. It is about ten years since the idea of such a law suggested itself to the writer of this paper, and he has since taken every opportunity of testing it by all the newly ascertained facts with which he has become acquainted, or has been able to observe himself. These have all served to convince him of the correctness of his hypothesis. Fully to enter into such a subject would occupy much space, and it is only in consequence of some views having been lately promulgated, he believes in a wrong direction, that he now ventures to present his ideas to the public, with only such obvious illustrations of the arguments and results as occur to him in a place far removed from all means of reference and exact information.

The following propositions in Organic Geography and Geology give the main facts on which the hypothesis is founded.

Geography.

1. Large groups, such as classes and orders, are generally spread over the whole earth, while smaller ones, such as families and genera, are frequently confined to one portion, often to a very limited district.

2. In widely distributed families the genera are often limited in range; in widely distributed genera, well-marked groups of species are peculiar to each geographical district.

3. When a group is confined to one district, and is rich in species, it is almost invariably the case that the most closely allied species are found in the same locality or in closely adjoining localities, and that therefore the natural sequence of the species by affinity is also geographical.

4. In countries of a similar climate, but separated by a wide sea or lofty mountains, the families, genera and species of the [[p. 186]] one are often represented by closely allied families, genera and species peculiar to the other.

Geology.

5. The distribution of the organic world in time is very similar to its present distribution in space.

6. Most of the larger and some small groups extend through several geological periods.

7. In each period, however, there are peculiar groups, found nowhere else, and extending through one or several formations.

8. Species of one genus, or genera of one family occurring in the same geological time are more closely allied than those separated in time.

9. As generally in geography no species or genus occurs in two very distant localities without being also found in intermediate places, so in geology the life of a species or genus has not been interrupted. In other words, no group or species has come into existence twice.

10. The following law may be deduced from these facts:–Every species has come into existence coincident both in space and time with a pre-existing closely allied species.

In modern times, one would read the tenth point as a declaration of the Evolutionary Theory. Nevertheless, it is curious that Wallace never tries to figure at the same rank with Darwin: he published in 1889 a book with a very interesting title: *Darwinism: An Exposition of the Theory of Natural Selection, with Some of Its Applications.* (Find it as a Google book).

7.6 The creationism of Mendel

The Evolutionary Theory of Darwin was received by the overall people with great enthusiasm in spite of the fact that the laws of inheritance were not known and so people, Darwin included, should had relied on speculations. The door to the grandiose world of genetics, the fully implemented materialist study of inheritance, was open by Mendel, who gave to it a clearly scientific basis by pointing that there are striking regularities in the form as hybrid forms recurrently reappear.

Actually, the opening paragraph of the fundamental article of Mendel (1865) already shows the emphasis he made in the possibility of repeating his experiments:

Künstliche Befruchtungen, welche an Zierpflanzen deshalb vorgenommen wurden, um neue Farben-Varianten zu erzielen, waren die Veranlassung zu den Versuchen, die hier besprochen werden sollen. Die auffallende Regelmässigkeit, mit welcher dieselben Hybridformen immer wiederkehrten, so oft die Befruchtung zwischen gleichen Arten geschah, gab die Anregung zu weiteren Experimenten, deren Aufgabe es war, die Entwicklung der Hybriden in ihren Nachkommen zu verfolgen.

The regularities he speaks about are known in modern times as the two laws of Mendel. They collide head to head with the then accepted theory of blending inheritance, which proposed that the traits of offspring average those of parents. Instead and using modern terminology, Mendel argued that, in first place, biological inheritance is realized through genes that are inherited one from each parent and that are passed on to next generations in separated form (law of segregation).

In second place, when genes come in different chromosomes, they recombine at random (law of independent assortment):

Ausserdem wurden noch mehrere Experimente mit einer geringeren Anzahl Versuchspflanzen durchgeführt, bei welchen die übrigen Merkmale zu zwei und drei Hybrid verbunden waren; alle lieferten annähernd gleiche Resultate. Es unterliegt daher keinem Zweifel, dass für sämmtliche in die Versuche aufgenommenen Merkmale der Satz Giltigkeit habe: die Nachkommen der Hybriden, in welchen mehrere wesentlich verschiedene Merkmale vereinigt sind, stellen die Glieder einer Combinationsreihe vor, in welchen die Entwicklungsreihen für je zwei differirende Merkmale verbunden sind. Damit ist zugleich erwiesen, dass das Verhalten je zweier differirender Merkmale in Hybrider Verbindung unabhängig ist von den anderweitigen Unterschieden an den beiden Stammpflanzen.

Bezeichnet n die Anzahl der characteristischen Unterschiede an den beiden Stammpflanzen, so gibt 3^n *die Gliederzahl der Combinationsreihe,* 4^n *die Anzahl der Individuen, welche in die Reihe gehören, und* 2^n *die Zahl der Verbindungen, welche constant bleiben. So enthält z. B. die Reihe, wenn die Stammarten in 4 Merkmalen verschieden sind,* $3^4 = 81$ *Glieder,* $4^4 = 256$ *individuen und* $2^4 = 16$ *constante Formen; oder was dasselbe ist, unter je 256 Nachkommen der Hybriden gibt es 81 verschiedene Verbindungen, von denen 16 constant sind.*

Alle constanten Verbindungen, welche bei Pisum durch Combinirung der angeführten 7 characteristischen Merkmale möglich sind, wurden durch wiederhorte Kreuzung auch wirklich erhalten. Ihre Zahl ist durch $2^7 = 128$ *gegeben. Damit ist zugleich der factische Beweis geliefert, dass constante Merkmale, welche an verschiedenen Formen einer Pflanzensippe vorkommen, auf dem Wege der wiederholten künstlichen Befruchtung in alle Verbindungen treten können, welche nach den Regeln der Combination möglich sind.*

77 *Rediscovery*

The work of Mendel caused no repercussion in the literature. Precisely, it was because of repeated experiments that Mendel was rediscovered after 30 years of oblivion (Moore, 2001). Anyway, science needed other 30 to 40 years to digest his work. How can we understand this?

Science is the difficult, the very difficult art of fitting theoretical patterns to data. Patterns compete for fitness and successful ones create an inertia that causes every one to get blind against possibly falsifying data or competing schemata. So, science is not the quest for truth. That is why Mendel was impossible to digest during nearly 70 years as the next words of Lönnig (2001) show:

Especially in the decade after the publication of Darwin's ORIGIN (1859) the scientific world was eagerly awaiting the discovery of the laws of heredity

by some experimental or other scientist(s). After two lectures in 1865, Mendel published his famous Pisum-treatise VERSUCHE ÜBER PFLANZEN-HYBRIDEN in 1866. His work was quoted at least 14 times before 1900, the year of its 're-discovery'. There were references in such widely distributed works as Focke's DIE PFLANZEN-MISCHLINGE (1881), THE ENCYCLOPAEDIA BRITANNICA (1881) and the CATALOGUE OF SCIENTIFIC PAPERS OF THE ROYAL SOCIETY (1879). The treatise had been sent to the libraries of some 120 institutions including the Royal and Linnean Society of Great Britain. Moreover Mendel had 40 additional reprints at his disposal, many of which he sent to leading biologists of Europe. In fact, professor Niessl (1903 and 1906) emphasized that Mendel's work was "well known" at his time. So in the face of the expectations just mentioned, - why was the discovery of the laws of heredity ignored by most scientists for more than 35 years, until 1900, and by the "true Darwinians" (Mayr) for another 37 years? That is 72 years in all!

The reasons have been hinted at or clearly stated by several pioneers of genetics as de Vries (1901), Bateson (1904, 1909, 1924), Johannsen (1909, 1926) as well as several historians of biology and/or biologists as Niessl (1903, 1906), Richter (1941, 1943), Stern (1962), Lönnig (1982, 1986, 1995), Callender (1988) and Bishop (1996):

All the evidence points to the main reason as follows: Mendel's ideas on heredity and evolution were diametrically opposed to those of Darwin and his followers. Darwin believed in the inheritance of acquired characters (and tried to back up his ideas with his pangenesis hypothesis, which even Stebbins called an "unfortunate anomaly") and, most important of course, continuous evolution. Mendel, in contrast, rejected both, the inheritance of acquired characters as well as evolution. The laws discovered by him were understood to be the laws of constant elements for a great but finite variation, not only for culture varieties but also for species in the wild (Mendel 1866, pp. 36, 46, 47). In his short treatise EXPERIMENTS IN PLANT HYBRIDIZATION mentioned above Mendel incessantly speaks of "constant characters", "constant offspring", "constant combinations", "constant forms", "constant law", "a constant species" etc. (in such combinations the adjective "constant" occurs altogether 67 times in the German original paper). He was convinced that the laws of heredity he had discovered corroborated Gärtner's conclusion "that species are fixed with limits beyond which they cannot change". And as Dobzhansky aptly put it: "It is...not a paradox to say that if some one should succeed in inventing a universally applicable, static definition of species, he would cast serious doubts on the validity of the theory of evolution".

As the Darwinians won the battle for the minds in the 19th century, there was no space left in the next decades for the acceptance of the true scientific laws of heredity discovered by Mendel and further genetical work was continued mainly

by Darwin's critics among the scientists.

So, the paper was understood by people as contrary to evolution, but Mendel himself understood his work as a scientific study of the evolution of plants but with results that contradict the Evolutionary Theory. Now, given the extreme complexity of life, if a monk denies evolution, what else is he proposing? Without a doubt, he rejoices in his belief that species exist thanks to a direct creation by God, precisely, the point that is made into a mockery by the ideas of Darwin. We see that Darwin and Mendel represent followers of two irreconcilable points of view that are now known under the names of creationism and evolutionism. In **evolutionism** all living beings are related by descend to a common ancestor. By contrast, **creationism** asserts that life in general cannot be scientifically explained and so the faith in a Creator that will judge us cannot be excluded by science.

Let us learn now something from the ethics of this monk:

1. Science provides no proof of religious beliefs, so these must not be mentioned in a scientific work. Of course, every religion has elements that can be subject to scientific scrutiny, but the most important tenets of every religion cannot be scientifically studied. Say, Christianity teaches that Jesus will come and if this is false the whole religion is a nonsense. But how can we prove or refute that?

2. If science seems to refute an important tenet of own religion, it must be carefully investigated. Thus, the paper of Mendel reports a study of evolution with transparent scientific methodologies.

3. Be pure in heart and open your mind to any trace of truth that may be contained in considered attacks. So, Mendel celebrates at the end of his paper the possibility to engineer at will a new species precisely by hybridization and furnishes the needed information about the good species of plants that would serve to verify his claim. Mendel manages to convey the idea that this result is not a success of the Evolutionary Theory but a pitfall: given that the daughter species is always the same, this result unveils the real but very scare potentiality of evolution to create new species, one that would not be enough to explain the immense variability of life in the whole world.

4. Courage and wisdom must be exhibited in the ongoing scientific study but not in the propagation of achieved results. Actually, one must not be tried to defend God: if he is really God, let him defend himself! Rather, it is God that will defend the truth and light that is in you -if any.

Since reproduction is at the very core of the Evolutionary Theory, the scientific basis of the study of inheritance that was fired by Mendel is a must to be pursued relentlessly. The next important detail of that study refers the asymmetric roles in genetics of two types of cells in the body, the germ and the soma lines. The implication is that the doctrine of acquired characters is not supported by science, a conclusion forwarded on by Weismann:

7.7 The germ-plasm theory of Weismann

We all know the case of people that thanks to continued and demanding exercise have gotten very strong but whose children are as feeble as almost every child is. Thus, one has natural and abundant **falsification**s of the theory of inheritance of acquired characters. Nevertheless, these falsifications are not clear cutting because one might invoke diverse factors that regulate the possible underlying learning mechanism, say, its possible dependence of somatic factors. The accepted solution to the ensuing uncertainty was proposed by Weismann (1893, chapter XIII):

By acquired characters I mean those which are not performed in the germ, but which arise only through special influences affecting the body or individual parts of it. They are due to the reaction of these parts to any external influences apart from the necessary conditions for development. I have called them 'somatogenic' characters, because they are produced by the reaction of the body or soma, and I contrast them with the 'blastogenic' characters of an individual, or those which originate solely in the primary constituents of the germ ('Keimesalagen'). It is an inevitable consequence of the theory of the germ-plasm, and of its present elaboration and extension so as to include the doctrine of determinants, that somatogenic variations are not transmissible, and that consequently every permanent variation proceeds from the germ, in which it must be represented by modification of the primary constituents.

I will first attempt to show how this conclusion is arrived at theoretically, and will then proceed to test it by ascertaining how far it is in agreement with actual observation, and whether the theory can be justified by facts.

This work purified the Evolutionary Theory from the doctrine of acquired characters. And so, this theory could found its way to fairer and more robust formulations.

7.8 The modern synthesis

The work of Mendel opened for science the magnificent world of genetics, the scientific study of inheritance, while Weismann purified it from the wrong idea of

inheritance of acquired characters. Nevertheless, according to our interpretation of Mendel, genetics points to creationism. Therefore, to graft genetics into the Evolutionary Theory, the laws of inheritance should had suffered a deep transformation. That grafting work is known as *the modern synthesis* and more rigorously as *the Synthetic Evolutionary Theory*. Its creation can with difficulty be ascribed to some few persons.

At the end, the matter was really simple. In first place, we have that the carriers of inheritance are not constant but may change from one generation to the next. In the words of Morgan in his Nobel Lecture in 1933 (Nobelprize.org, 2011a):

The relative stability of the gene is an inference from genetic evidence. For thousands - perhaps many millions - of subdivisions of its material it remains constant. Nevertheless, on rare occasions, it may change. We call this change a mutation, following De Vries' terminology. The point to emphasize here is that the mutated gene retains, in the great majority of cases studied, the property of growth and division, and more important still the property of stability. It is, however, not necessary to assume, either for the original genes or for the mutated genes, that they are all equally stable. In fact, there is a good deal of evidence for the view that some genes mutate oftener than others, and in a few cases the phenomenon is not infrequent, both in the germ cells and in somatic tissues. Here the significant fact is that these repetitional changes are in definite and specific directions.

In second place, mutations happen at random, i.e., they are statistically uncorrelated with the needs of organisms to improve surviving. This is an experimental generalization that allows a clean and transparent reformulation of Darwinism. This is clearly stated by Muller in his Nobel lecture in 1946 (Nobelprize.org, 2011b):

So far, then, we have no means, or prospect of means, of inducing given mutations at will in normal material, though the production of mutations in abundance at random may be regarded as a first step along such a path, if there is to be such a path. So long as we cannot direct mutations, then, selection is indispensable, and progress in the hereditary constitution of a living thing can be made only with the aid of a most thoroughgoing selection of the mutations that occur since, being non-adaptive except by accident, an overwhelming majority is always harmful. For a sensible advance, usually a considerable number of rare steps must be accumulated in the painful selective process. By far the most of these are individually small steps, but, as species and race crossings have shown, there may be a few large distinctive steps that have been, as Huxley terms it, "buffered", by small changes that readjust the organism to them. Not only is this accumulation of many rare, mainly tiny changes the chief means of artificial animal and plant improvement, but it is, even more, the way in which natural evolution has occurred, under the guidance

of natural selection. Thus the Darwinian theory becomes implemented, and freed from the accretions of directed variation and of Lamarckism that once encumbered it.

These ideas were induced by mathematical studies published by Fisher, Wright and Haldane in the early 1930s (Hey et al., 2005). The concrete step of passing from mutations to species is an idea that was studied and promoted by Dobzhanzky and Mayr (UCMP, 2011a). Precisely, Dobzhanzky (1973) published a paper that was entitled *Nothing in Biology Makes Sense Except in the Light of Evolution*. He is gentle enough to show us in that paper those thoughts that helped him to decide among the origin of species by creation else by evolution:

Antievolutionists fail to understand how natural selection operates. They fancy that all existing species were generated by supernatural fiat a few thousand years ago, pretty much as we find them today. But what is the sense of having as many as 2 or 3 million species living on earth? If natural selection is the main factor that brings evolution about, any number of species is understandable: natural selection does not work according to a foreordained plan, and species are produced not because they are needed for some purpose but simply because there is an environmental opportunity and genetic wherewithal to make them possible. Was the Creator in a jocular mood when he made Psilopa petrolei for California oil fields and species of Drosophila to live exclusively on some body-parts of certain land crabs on only certain islands in the Caribbean? The organic diversity becomes, however, reasonable and understandable if the Creator has created the living world not by caprice but by evolution propelled by natural selection. It is wrong to hold creation and evolution as mutually exclusive alternatives. I am a creationist and an evolutionist. Evolution is God's, or Nature's method of creation. Creation is not an event that happened in 4004 BC; it is a process that began some 10 billion years ago and is still under way.

As an application of the universalization of evolution, we have the synthetic definition of **species: it is the more general population whose members can interbreed and produce fertile offspring. Different species are reproductively, isolated** . Thus, we pay attention to reproduction and forget other observables characteristics, a procedure that is quite different from that of Linnaeus. This shift in concept is directly connected to Mayr and about which he has personally something to say and that touches on the need of getting a science in accordance with common sense specially of that belonging to natives of New Guinea (Academy of Achievement, 2001):

After I'd been a little while in this first collecting place, I realized that every bird the natives saw and that they had shot, they knew the name. So I recorded the Latin name that I knew and the name the natives gave me. Since they knew birds so well – I had three little bird guns – I gave them to the natives and I told

them always when they came with a particularly rare bird, let's say a nieda, I said, "Well, I want more nieda, and then they went out and they brought back nieda and not the common stuff, you know. And at the end, when I finally summarized everything, I found that I had in this locality collected 137 species of birds, and the natives had given me the names of 136 of these. There were only two little nondescript looking little warblers, green warblers, which they had given the same name to the two different species. And I'm sure if I had gotten the right kind of an old man he would have been able to have the name of that 137th species.

`So they had a pretty good species concept themselves, would you say?`

Exactly. The biological species is an absolutely obvious entity to any good naturalist, and they were very clever. In the pidgin English language of Eastern New Guinea, they would name a male bird of paradise for a particular thing, and if we saw a female and I said, "What's that?" they said, "Oh, that's mama belong..." and then the name of the male. They knew perfectly well that they were not two different species, which by the purely typological species concept might have been the case. They knew exactly which bird had only one egg in the nest, which species have two eggs in the nest, and they were superb woodsmen. Very often in certain localities in New Guinea and in the Solomon Islands I would distribute all the guns to the natives and I would go out and collect orchids and other things that the natives weren't interested in collecting.

Therefore, **Speciation** is a process by which a population splits in two portions that get reproductively isolated (de Queiroz, 2005). The **allopatric mechanism** for the arising of new species plays a preferential role to demonstrate that nature might be simple and ease to understand and is therefore in the span of elementary science. That mechanism says that the colonization by part of the population of a new area with a different ecology is ensued by adaptive genetic divergence, which is next followed by a barrier to interbreeding when populations eventually encounter again (Grant and Grant, 2009). With this and other mechanisms, the Modern Synthesis explained the origin of species.

78 ***Evolution for scientists***

The Modern Synthesis was based as in field observations as in huge amount of laboratory work. It is the sort of theory that everyone likes and it is no mystery that it has completely dominated the scientific community even to date - in spite of more or less serious corrections such as that of the horizontal gene transfer. Thus, the Evolutionary Theory is the cornerstone, the sacrosanct dogma of biology. Nevertheless, its priest are very tolerant and with extremely rare exceptions they are incapable to fire heretics.

But a change is occurring: the vision of evolution as a dogma is bit by bit leaving its place to the insight that the Evolutionary Theory is capable of conforming to the ordinary scientific standards according to which science is a whole with two main ingredients: theories and experimental testing.

A theory must be graceful, beautiful, powerful, robust and nevertheless simple. In response to the work of theorists, the first duty of science is to grasp the predictive power of the theory by voicing concrete, interesting and realizable predictions to next put them to test.

In this theory-testing paradigm, field studies are important because they allow the formulation of hypothesis that have high probability of withstanding severe tests but they are anymore sufficient to claim that a theory is correct. By the same token, if a theory succeeds a given test, that theory might acquire more value, i.e., it becomes a candidate for great grants ... to carry on further, more incisive tests.

In the light of the theory-testing-paradigm, the very first important experimental study about the Evolutionary Theory must deal with natural selection. What is it apart from being an abstract term that we humans use to describe trends in death?

The scientific answer given by Endler (1980) has been delightful for everybody. He carried field research together with laboratory and field experiments with guppies and showed, among many other things, that the effects of natural selection in the form of predation on color patterns are predictable and repeatable. This is ordinary science in action: if a researcher is unable to describe predictable and repeatable effects, he or she does not know enough the object of research. Let him or her endure to fit to these normal scientific standards.

All this is OK, but where are genes? The connection of genes with natural selection has been documented for the Bioluminescent color in the Jamaican click beetle (Stolz et al, 2003): *Our results constitute a novel example connecting the selective fixation of specific nucleotides in nature to their precisely determined phenotypic effects.* This is really a very great synthesis that connect genes with function and phenotype and links them with changes in gene frequencies!

Given that we do know that natural selection affects gene frequencies, we can pass to deeper problems: can we reproduce speciation in the lab?

We can use complexity theory to compose a simple but mandatory prediction to welcome the joy of science in regard with speciation:

1. The number of species surpasses two million.
2. Field observations nicely interpreted by the Evolutionary Theory seem to imply that the creation of new species is for nature a game of children: just

pay attention to the approximately 800 native drosophilid species in Hawaii that have probably appeared as the fruit of the evolution of the progeny of a single pregnant female (NAS, 2004).

3. Whence, we predict that it must be quite possible to reproduce speciation in the lab or in controlled field experiments. In any case, and taken into account that scientific projects are financed only if they promise immediate results, it must be realizable to reproduce partial reproductive isolation, i.e., incipient speciation.

Hard as this challenge might be, it has been overtaken with courage and wisdom by many researchers and citations claiming positive scientific results abound.

On first place, we have verified the possibility of engineering a new species by hybridization such as it was accepted by Mendel. The verification has been made, say, upon butterflies (Mavárez et al, 2006).

By the same token, one can learn what is ordinary science in regard with speciation by studying, for instance, the work of Gil et al (2010). There, we can clearly see the relation among theory, experiment and interpretation. Succinctly:

- Speciation is common, so it might depend on very simple, ordinary events. In fact, development of mating preference is a good candidate to be a triggering mechanism of speciation. Therefore, it might be possible to readily develop experimental mating preference.

- Experiment: *In this study, mating preference was achieved by dividing a population of Drosophila melanogaster and rearing one part on a molasses medium and the other on a starch medium. When the isolated populations were mixed, "molasses flies" preferred to mate with other molasses flies and "starch flies" preferred to mate with other starch flies. The mating preference appeared after only one generation and was maintained for at least 37 generations.* Cited authors also present clear ideas about the actual mechanism as this happens to be: symbiont bacteria were found to play the determinant, decisive role.

- Discussion: *How can a bacterially induced mating preference, as described here, contribute to speciation and evolution in nature? One possibility is that, in the natural world, multiple environmental factors act synergistically to differentiate the microbiota and strengthen the homogamic mating preference. For example, it is reasonable to assume that fly populations living on different nutrients will be, at least to some extent, geographically separated. The combination of partial geographic separation and diet-induced mating*

preference would reduce interbreeding of the populations. Slower changes in the host genome could further enhance the mating preference. The stronger the mating preference, the greater the chance that two populations will become sexually isolated, and many evolution biologists have argued that the emergence of sexual isolation is the central event in the evolution of species.

- Implication: statistically significant mating preference of flies was shown to follow adaptation to different substrates of the ensemble conformed by flies and commensal bacteria. This is compatible with and reinforces the belief that speciation is a byproduct of adaptation.

To try a real blending of the ideas of Darwin with those of Mendel in regard with speciation, we need to pursue this discussion down to genes:

1. Speciation is common, so it might depend on very simple, ordinary events. A first option is that speciation depends on some few but powerful changes in DNA. At the other extreme, it might depend on many insignificant changes distributed all over the entire genome.

2. To back these predictions, we have some field observations. In first place, we have that reproductive isolation may hang on a simple locus, as it happens with bindin, a protein mediating recognition and binding between the sperm and the egg once the sperm has penetrated the egg jelly. Recognizing might be species specific. The rate of evolution of bindin is high when related to the formation of many species of sea urchin that occupy the same region but that in contrast is low in contrary case (Palumbi et al, 2005). On the other hand, Michel et al (2010) found in their work with flies that feed on apples else hawthorn that large connected regions of multiple differentiated loci may characterize even the early stages of speciation. It is speculated that this trend might be so determining that it might resists gene flow and be responsible for speciation without geographic barriers but that are due to interactions with the environment and particularly related to feeding material.

Thus, on behalf of science we venture to conclude that evolution is eager and powerful to create speciation:

1. It is very easy to generate experimental incipient sexual isolation in the lab.

2. Speciation may depend on the mutation of just one gene.

3. Whole continents of genes might rapidly get involved in divergence due to ecological factors.

Criticisms may arise that we have too little to claim too much. Our excuse is that evolution for scientists is a successfully running project that cannot be stopped. So, we expect science to satisfactorily fill in this and other criticism although some time will be needed. We conclude that it is a verified fact of science that natural evolution leads to speciation, which must be understood as reproductive speciation. But, beware, science can claim that some species appeared by evolution not that the Evolutionary Theory has been verified. In fact, that theory does not say that evolution have created some species, rather it affirms that all species, including our owns, have appeared by evolution and that therefore God and his judgment are both unnecessary and redundant to explain and orient our existence.

Now that we have agreed on the great power of evolution, a question naturally arises: Is there a limit to evolution? Can we venture, on our own risk, to claim that all species, both extinct and extant, appeared by the creative power of the interrelation of mutation with the environment?

No. We cannot.

The reason is very simple: the very fact that the concept of species is an accepted term in the modern scientific culture shows that the power of evolution is very limited. For suppose that there is no barriers to the creative power of evolution. To fix ideas, let us consider domestic flies. By direct inspection one can see that they are the same all around the world. The wrong explanation would be that they are adapted to an environment created by humans and that this is roughly the same in every part. In short, flies conform an absolute maximum of fitness and evolution is therefore halted. Now, this explanation is wrong because the environment that that flies see is not the human created environment but that created by their siblings and by all other species. And there is plenty of space to mutate in that wide world. The consequence is that every hundred meters one shall expect a new species with peculiar morphological characterization. But that is not observed. Therefore, evolution is obviously limited and it is scientifically unbearable from now on to claim that our existence is explained by the Darwinian Evolutionary Theory.

So, what must we do?

We must arm ourselves with very clear concepts for very hard and long battles. One of them follows.

7.9 The complexity of science: Poe and Ernst

Science is the art of fitting patterns to facts. Science is difficult, very difficult. In effect, science is NP-harsh. One of the best explanations of the meaning of this slogan has been given by Allan Poe (1841) in one absurd fiction about detectives,

an especial type of applied scientists:

79 ***Poe: The Murders in the Rue Morgue***

The purpose of this short story is to show that it is easy to deceive oneself but that wisdom and investigation might lead to the solution of puzzles.

The problem to solve is to find the murder of a woman and her daughter. The police makes a very good work and as result arrests a person. But the hero of the story, which is not a police but is pray of the delight of investigating puzzles just for the joy of finding the truth, arrives with a very contrived explanation that is tested and found to be correct. The culprit is an Ourang-Outang, a being that nobody expects.

The paper is very rich and might give rise to many questions and controversies. In our opinion, the main points to draw are the following:

- The police has a method of investigation that is good and effective for most purposes. Theories and culprits come and go and the deal survives. In fact, the very method is just to investigate long enough to arrive to a hard theory that looks correct and resists scrutiny. Nevertheless, this method is error-prone and thereof innocent people get arrested while true culprits go on free and rejoicing.

- If you plan not to issue a hard theory but to find the truth (Poe considers that the truth can be found by a sufficient level of analytical power, logic and reason), you must begin by taking care of all relevant facts. Perhaps you must create data on your own. Once you get all relevant data, you must work until finding a suitable theory in which every piece softly makes sense.

The way Poe teaches this is as follows:

"If now, in addition to all these things, you have properly reflected upon the odd disorder of the chamber, we have gone so far as to combine the ideas of an agility astounding, a strength superhuman, a ferocity brutal, a butchery without motive, a grotesquerie in horror absolutely alien from humanity, and a voice foreign in tone to the ears of men of many nations, and devoid of all distinct or intelligible syllabification. What result, then, has ensued? What impression have I made upon your fancy?"

I felt a creeping of the flesh as Dupin asked me the question. "A madman," I said, "has done this deed –some raving maniac, escaped from a neighboring Maison de Sante."

"In some respects," he replied, "your idea is not irrelevant. But the voices of madmen, even in their wildest paroxysms, are never found to tally with that peculiar voice heard upon the stairs. Madmen are of some nation, and their language, however incoherent in its words, has always the coherence of syllabification. Besides, the hair of a madman is not such as I now hold in my hand. I disentangled this little tuft from the rigidly clutched fingers of Madame L'Espanaye. Tell me what you can make of it."

Poe designed the truth and so the Reader can make sure that it has been found albeit with a lot of effort and wisdom. But in science, we have no such a guide. We only have robust theories and we must getting alive with them. Now, the next point is that, given the human nature, if you are engaged in devising a theory, you will try to support it and you will close your mind to refutations. But if you dream of truth and have a theory, the least you can do is to encourage others to test it as much as possible.

Thus, it is useful to imagine that science can be considered as the product of the struggle of two teams: the first are the theorists that imagine models and the second consists of persons that are trained in testing theories. So, theorists imagine a model and testers put it to test by calculating predictions and checking for them to be fulfilled. Next, theorists correct their model to fit in the results of the test and the ensuing model is tested anew and so on.

To see what can happen if one fails to encourage testing, let us pay attention to a very interesting case study that belongs in the scientific research of medicine dealing specifically with alternative currents.

80 *Ernst and the complementary medicine*

The next apart from an interview with the renown Edzard Ernst illustrates how painful clear scientific concepts could be (Cressey, 2011):

Have you found that any alternative medicines are useful?

I found that homeopathy is pretty useless. I would have liked the evidence to go the other way because I trained as a homeopath. It would have been quite nice to win a Nobel prize by showing that 'no molecule' can have an effect, but the evidence is clearly against it.

What did you hope to achieve in setting up the complementary-medicine unit at Peninsula College?

I always felt very strongly that I would focus on subjects that are important in the United Kingdom. Therefore we focus on homeopathy, herbal medicine, acupuncture and spinal manipulation. I was also convinced that scientists need to be critical and sceptical, and that if you apply science to any field you don't want to prove that your ideas are correct, you want to test whether they are correct.

That seems like a tiny difference to most people, but in my field it's a very big difference. I think we're the only ones [in our field] who have understood that. Everyone else in alternative-medicine research uses 'science' to prove that their notions are correct and I think that is a very bad starting point.

Let's say that someone has been an acupuncturist, then decides to research the technique. That person, in my experience, [generally] wants to prove the value of acupuncture. This is a misuse of science, perhaps even an abuse of science. In my field, critical [evaluation] is seen as something negative. If somebody tells me 'you're so critical', to me it's a compliment; to them it's an insult.

It might seem that the ideas of Ernst are too deep to be understood. See also Ernst(2000, 2010). That is not the case, rather they represent the ordinary way of life in science, so let us explain the theory beneath them. To fix ideas, let us suppose that someone arrives with a remedy to diminish the negative collateral effects of a powerful drug against mammary cancer. Specifically, the remedy must be administrated with the anticancer drug and is promised to eliminate vomiting 8 hours after the finishing of the treatment. How must we test this theory?

To design the test, we must consider the role of two factors: randomness and variability of self-healing. Randomness refers to the fact that one is ordinarily exposed to many uncontrolled factors both inner and external that may influence the battle of the organism against the collateral effects of the anticancer drug. On the other hand, every organism tends to heal itself against collateral effects, a fact that is no miracle because it may hang on the simple decaying of ingested compounds. Now, persons are all different and so one cannot exclude the possibility that a patient got better not by the remedy but because of these two factors and of their interactions.

Our problem is therefore to neutralize the roles of randomness, self -healing and to separate them from the effects of the remedy:

1. To neutralize randomness we apply the next law: the more randomness is allowed to express itself, the more it self- annihilates. Operationally, all one needs to do is to gather sufficient independent data.

2. To neutralize the effect of the variability of self healing we run a controlled experiment: to a group we administer the remedy in company of the anticancer drug and to another group we administer just the drug or maybe the drug with a sugar pill. Next, we measure the elapsed time from the end of the treatment to the last vomiting. After that, we analyze results using standard statistical methods, say, a t-test. At last, we take a verdict: if there is no evidence of a difference between the two groups, we must conclude that the remedy causes no visible effect upon the noise caused by randomness and

> self-healing or in short that the remedy is useless. But, if we find a difference, then we can claim that the average time of the last vomiting has been statistically diminished beyond the effects of randomness and variation in self-healing and that the remedy is useful such as it was promised.

In short, invent theories and put them to test. In principle, this is all to science. Testing is in general painful. It is thus mandatory to keep in mind that clear scientific concepts awake powerful enemies. Or in plain words, if your words do not awake powerful enemies, they are more of the same else they have not been clear enough. So, expect troubles and therefore be prudent and wise. But if you sincerely consider that it is time to be bold as a lion, then go on. The way as Ernst experiences and teaches this is by opposing directly involved agents and companies that make luxury gains on the needs of people. One of them is the very Prince of Whales, whom Ernst has referred to as a *snake oil salesman* (Sample, 2011). Ernst can say this because he at last has nothing to fear now that he goes to retirement after 18 years of moderated battle, but in general, did I say that you must be prudent and wise?

It maybe difficult to clearly understand the difference between the two methods that Ernst refer to, one that tries to prove that a theory is right and the second that aims at testing that theory. So, let us see how they are implemented in the study of evolution.

81 *Arguing that the theory is correct*

If a person is convinced that the Evolutionary Theory is right, he just shows its explanatory power in contrast with rival theories. So, let us give a concrete example in our field, the study of the Evolutionary Theory. The next quotations, due to a teenager (Bar-Yam, 2011), contrast Lamarck with Darwin and claims for the merits of Darwin:

Darwin believed that the desires of animals have nothing to do with how they evolve, and that changes in an organism during its life do not affect the evolution of the species. He said that organisms, even of the same species, are all different and that those which happen to have variations that help them to survive in their environments survive and have more offspring. The offspring are born with their parents' helpful traits, and as they reproduce, individuals with that trait make up more of the population. Other individuals, that are not so well adapted, die off. Most elephants used to have short trunks, but some had longer trunks. When there was no food or water that they could reach with their short trunks, the ones with short trunks died off, and the ones with long trunks survived and reproduced.

Eventually, all of the elephants had long trunks. Darwin also believed that evolution does not happen according to any sort of plan.

Darwin's theory has been supported by a lot of evidence. Lamarck's Theory of Inheritance of Acquired Characteristics has been disproved. This was done in two major ways. The first is by experiment. We have seen through many real examples and observations that changes that occur in an animal during life are not passed on to the animal's offspring. If a dog's ears are cropped short, its puppies are still born with long ears. If someone exercises every day, runs marathons, eats well, and is generally very healthy, the fitness is not passed on and the person's children still have to work just as hard to get that fit and healthy. These and other examples show that Lamarck's theory does not explain how life formed and became the way it is.

The other way that Lamarck's theory has been proven wrong is the study of genetics. Darwin knew that traits are passed on, but he never understood how they are passed on. During the time when Darwin's first book first came out, Gregor Mendel, who discovered genetics, was just starting his experiments. However, now we know a lot more about genetics, and we know that the only way for traits to be passed on is through genes, and that genes can not be affected by the outside world. The only thing that can be affected is which gene sets there are in a population, and this is determined by which individuals die and which ones live. This is the other way that we have learned that the fruits of an animal's efforts can not be inherited by its offspring.

As we see, convincing arguments may be given to prove that the theory is right. Anyway, we consider that any robust theory for a complex world can be supported as much as desired. Example: the horoscope.

Let us see now how one proceeds if one wants to testing the theory instead of proving it.

82 *Testing the theory*

If a person is not interested in supporting the Darwinian Evolutionary Theory but in testing it, he or she would reason like this in reaction to the ideas expressed in the last subsection:

Trunks of the elephants were not built in one generation but possibly by thousands of generations along which small change was gathered (Dawkins, 1986). Tiny changes were at random but some resulted to be beneficial, increasing the length of trucks. These augmented the possibilities for surviving and would be passed on to the offspring.

Let us think now a bit about the meaning of randomness. Changes could had happen all along the genome and so some changes would cause limbs to be larger.

And other would cause to enlarge the tail. And other would cause to enlarge the dorsal spine. And other would cause the neck to be enlarged. And because all they would be random, we would have individuals with mixed characters, say, over-enlarged limbs and neck.

From this we have the next families of predictions:

1. Some subspecies of elephants must have enlarged necks, other enlarged limbs, others enlarged necks and enlarged limbs, others enlarged necks and enlarged trunks. And so on.

2. The animals that shared with elephants the same stress caused by drought all shall have similar evolutionary fates. So, we must find lions with long trunks in the fossil record. And rhinos with large trunks and necks. And hippos with limbs that must be larger than ordinarily. And giraffes with long prehensile tongues.

Next, we must compare our predictions with nature. This is immediate: it happens that nothing like that is found neither in modern populations nor in the fossil record. Therefore, we must conclude that the Evolutionary Theory is false.

Science, i.e., testing, is a way of being. You must practice enough until you can see **falsifications** of the Evolutionary Theory everywhere, all around.

83 ***Challenge.*** *Devise a Java program to test our predictions. A graphic output could be very advisable.*

Falsifications have different degree of power. Some are innocuous while other might be clear cutting. In regard with our mentioned falsifications, about trunks as adaptation to drought, these need to be fully cooked to be clear cutting. They are as yet half cooked because in a complex world there are too many options to invoke complex interactions and one might bring forth special cases with too many ifs to dismiss falsifications.

We have the power to pursue these discussions to ultimate consequences, most surely with the help of Java, but one must consider the possibility to fall into a never ending argument that apart from frustration nothing else would produce, such as it has happened in the last 150 years.

So, we face a challenge: can we devise a family of **robust falsifications** of the Evolutionary Theory that are both obvious and clear cutting?

84 ***Robust falsifications***

A necessary condition for a mechanistic explanation to be successful and accepted is that it must convert facts in statistically ordinary events. More to the point, reasonable explanations that convert facts in ordinary events are all around preferred to those that rely on randomness and extreme events. Example:

A wife notices that her husband is getting late at home even more than ordinarily. By instinct, she begins to think bad things but when she perceives the pure sight of his eyes, she is filled in peace (one naturally studies other variables and/or make other experiments or gather new observations when an extreme event is registered in regard with a given descriptor). But the husband arrived yesterday two hours late and he did not pass the sight-test and when he argued about the usual jams, she was unable to believe him. In consequence, she passed the night turning around on the bed without getting asleep and she is planning today to ask her friends to keep and eye on her husband because she suspects the worst. (In hindsight, the husband says that his behavior is explained as the result of extreme random events but the wife needs to make sure that it is not explained by infidelity, a situation that will convert the delaying of his arrival in a normal and necessary event.)

Let us transpose this ideas to the discussion of the existence of species, i.e., of our own existence. International Science uses to teach that Darwinian evolution is the correct explanation of our life. This means that, in the light of evolution, life is an ordinary result of the laws of nature. Nevertheless, this is false:

1. It is certain that the genome is software and that evolution is a software developer (Rodríguez, 2010). This is our way of saying that, in first place, the DNA encodes for verbal instructions that are deciphered by the genetic code and that are executed by the ribosomes, which play the role of a processing unit of a digital computer. In second place, mutations can eventually create viable code in anyone evolutionary environment and so *mutation + selection* conform to unity that plays the role of software developer.

2. The experience of the Academic Community is that evolution is very powerful. Our proposal is that your imagination is the only limit to the power of evolution. But powerful as it can be, it has no magic in itself and so it can develop software and build wonders at the price of committing bugs. Some problems of software development might be easy and so they could had been solved by evolution without leaving a trail of bugs in the fossil record. Now, let us keep in mind that life is from the engineering stand the epitome of complexity and perfection. Thus, were evolution the explanation of our existence, the fossil record and extant populations should be filled in many

> diverse types of imperfections of form and function similar to those caused by genetic diseases. But this is openly false: all known genetic imperfections are explained as mutations of the wild, natural type, which is perfect.

The existence of bugs is a necessity of the Evolutionary Theory: variability created at randomness is what selection filters to create perfection. Now, rejected variability are the bugs of our insight, whose registration is immediate as we have seen in this volume. For complex problems, as ordinary problems are, these bugs are seen by intelligent observers as imperfections along the evolutionary process and so, the **Evolutionary Theory predicts that a clear path from imperfection to perfection must exist that must be apparent as in the fossil record as in modern populations. This is a huge family of mandatory predictions that are all falsified by extant life and fossil record**.

The situation is similar to a runner that can run long distances but that must pay a price: to sweat. This is a mandatory prediction that readily filter deceivers: no sweat, no long run. By the same token, evolution must create bugs, whose result are seen by observers as severe imperfections: no imperfections, no evolution.

Our arguments say that life cannot be an ordinary result of the laws of nature. In reaction, science has at present time the option of claiming that life is the result of extreme random events. The social consequence is that science is in the open mark of ideologies weak against creationist claims. Now, our law of the necessity of bugs is applicable even to smart trained human developers, so we bold to claim that the prediction of bugs falsifies as the Darwinian evolution, which is completely natural, as those proposals that pretend that we appear as a result of terrestrial evolutionary experiments set up by aliens.

Science is therefore challenged to remake itself in such spectacular way that life ceases to be a mystery. All it needs to do is to explain on natural grounds how to twist randomness, i.e., how to have perfectly random mutations that are more suitable than otherwise and that accelerate the evolutionary pace towards perfection with the consequence that the quantity and lethality of bugs are strongly diminished. Sufficiently twisted randomness converts life in an ordinary event.

7.10 Twisted randomness

Mutation is a change in DNA during the process of reproduction. Mutations are random (UCMP, 2011b). This implies that in anyone environment, the effect of mutations are in general incapable to establish a direction along the evolutionary process: mutations do not know the way neither to perfection nor to imperfection.

The modern version of the Darwinian Evolutionary Theory is based by definition on random mutations and is moreover plainly operational, so it is very easy to

simulate:

1. Living beings reproduce and so genetic information is copied with random mutation and recombination from parents to offspring.

2. The selective value of a given genetic string is determined a posteriori just when its carrier is exposed to its environment, if only it is viable.

3. Mutation is at random in regard with the reproductive alphabet. This implies that: the procedure that copies the genetic information from parents to offspring determines the alphabet that contains the elementary symbols to be copied. For DNA of living beings, the reproductive alphabet is conformed by A, T, C, G. From the stand of this alphabet, mutation to be Darwinian must be at random, a possibility that does not exclude that different bases might suffer mutation with different probabilities and even with loci dependence. Nevertheless, randomness precludes that mutations could guess the path to perfection. That path must be found on blind trial and error, by natural selection.

4. Selection can be implemented in two versions, positive and negative. The first type awards the fittest individuals and grants to them more opportunities for reproduction. The second assigns a probability of death to every individual that is greater the less fit it is. The first version simulates artificial selection, say by a farmer, that chooses his better exponents to make a new progeny. The second is inspired by creationism, according to which the huge complexity and perfection of living beings cannot be explained naturally and so must be taken as granted. Therefore, mutations are neutral, producing equally fitted individuals, or detrimental or maybe lethal. According to this school, mutation is preferentially seen as an error. Beware: both flavors of selection can propel complexity and perfection beginning from scratch although the negative flavor is easier to simulate.

A successful simulation of evolution that deviates from this protocol is called by us Lamarckian. Our choice is explained by the following facts:

1. Lamarck was the father of modern transformism: species suffer transformations driven by the circumstances to achieve complexity and diversity.

2. Lamarck taught that evolution is actually an accelerated phenomenon, and so it can produce yield just before the eyes of the beholder. (Nevertheless, the mechanism he proposed was shown to be false.)

In spite of the falsification of its main mechanism, a modern reading of Lamarck and the reason because of which he will remembered is that he thinks that it is possible to twist randomness. This is a powerful inspiration for software engineering: twisting randomness is the overall game of applied evolution. Let us see an example:

A very simple form as one can devise a simulation guided by Lamarckism is that changes are not done at random over the reproductive alphabet but are made at random over special super-alphabets that make mutations more successful in regard with an a priori given objective. To fix ideas, let us consider the problem of finding the correct form of a given misspelled word in which letters has been previously disordered. Say:

Lamarkc

If we use Darwinian evolution, a mutation to be at random must change a given letter of that word by another one taken at random from the usual alphabet. So, the following are some random mutations:

Ramarkc, Lamaukc, Lamarkx, Latarkc, Mamarkc

But if we rely on transpositions, which amount to a super-alphabet, that interchange the position of two letters, we might produce from Lamarkc the following strings:

Laamrkc, Kamarlc, Lamarck.

One can guess and verify that transpositions solves the problem of finding the correct spelling faster than Darwinian evolution that relies on random mutation over the reproductive alphabet. So, from the stand of the reproductive alphabet, transpositions amount to correlated changes that accelerate the race for perfection or, in our language, transpositions amount to **twisted randomness**.

How can we infer that a given simulation has been devised using twisted randomness? In that case, the number of trials is diminished, the number of expected bugs is strongly depleted and as consequence, the process is accelerated.

Now, let us translate this discussion into science. Suppose we belief that life on Earth is explained by evolution and we want to know whether it is explained by Darwinian evolution else by some sort of accelerated, Lamarckian evolution.

To decide this question, all we need to do is to observe the quantity and severity of bugs, i.e., of malformations and malfunctions all along the fossil record and in extant populations. In regard with our planet, we find malformations and malfunction but amidst populations whose members are in general whole and perfect so these are not explained by an evolutionary process that goes form imperfection to perfection but as the result of mutation from a whole wild type into defective

mutants. We conclude that if someone needs to choose between Darwinian and Lamarckian, accelerated, evolution, he surely will choose the last one.

Summarizing, the duel between Lamarck and Darwin can be expressed as follows: Darwin says: *pure, straight randomness + selection* explain our existence. Lamarck complaints: to explain our existence by evolutionary means, we must concede that randomness has been very strongly and systematically twisted. Mendel intervenes and proclaims: what else apart from God that will judge us is capable to twist randomness to such a degree as to make of life a must of his plans?

Strong warning: our arguments are compatible with **creationist religions** but they cannot be used to claim that those religions are correct or scientific. A religion to get scientific needs to teach us how to experimentally study its fundamentals tenets. Actually, a **tenet** is a belief without proof that is accepted on faith. So, to pursue the proof of religious tenets is to try to prove the unprovable, it is an oxymoron entailing a purpose that contradicts itself. Curiously enough, hard scientists, say, physicists, are usually pray of the propensity to get oxymoronic: given the impressive fitting of experiments to theoretical models, it is highly probably that one ends believing that models are correct. If this happens, one commits a logical error: from the verification of predictions, one cannot conclude the correctness of the postulates, which must be taken as conforming an effective theory (a good method to summarize observations) else accepted on pure faith as the ultimate truth.

7.11 The concept of evolutionary species

The Grand Synthesis dictated that speciation amounts to reproductive isolation. The advance of science has has shown that this phenomenon may depend on just one gene, as the case of bindin seems to be.

In hindsight, these genetical investigations has shown that the aforementioned concept of speciation is silly because it has nothing to do with the enormous gap of complexity and function that separates a chimp from a human being. Our insight based on the fact that the genome is software and evolution is a software developer allows to propose a solution to this failure:

A **evolutionary species** is conformed by a group of beings that can be joined by common descend given the terrestrial scale of time of some 10^{10} years and the material resources of the planet, some 10^{30} atoms and 10^{50} photons.

The evolutionary theory claims that the earth contains just one evolutionary species. By contrast, our insight proclaims the joy of science along the following directives:

1. The genome is software and evolution is a software developer.

2. Evolution is real (just think of of microbes that evolve defenses against antibiotics). Therefore, evolution contains no magic. Actually, it is easily defeated by complexity as every software designer knows.

3. Therefore and given the apparent extremely high complexity of life, we predict a huge number of evolutionary species, say, more than twelve: man, chimp, horse, dog, cat, mouse, bird, frog, fish, domestic fly, butterfly, plant. Nevertheless, it always will be a glory for science to prove or reject that two given reproductively isolated groups are connected by descend.

4. It is the sole responsibility of the EvolJava Community to elaborate these assertions to achieve any demanded level of clarity, force, power and grace.

7.12 Conclusion

The Evolutionary Theory claims that all living beings are related by descend to a common ancestor. It is an intellectual inheritance due to Lamarck. The very hard work of many, many people beginning with Darwin allows us to enjoy a mechanistic implementation of this theory that is simple, clear, powerful and that have resisted the scrutiny of science in various fundamentals aspects. The theory reads: random mutation is immanent to DNA duplication and reproduction and the ensuing variability is filtered by the environment in such a way that some offspring will produce more descendants than others and so life gets more and more adapted to its milieu and more diverse, complex and beautiful. If we identify speciation with the arising of sexual isolation of two populations, we must predict that incipient speciation shall be very easy to reproduce in the lab, a fact that has been verified with flying colors. While this result is a great advance of science, it is insignificant if we consider it from the stand of competing ideologies, say creationism, an inheritance of Mendel saying that life in general cannot be scientifically explained and so the faith in a Creator that will judge us cannot be excluded by science. In fact, the genetics of sexual isolation may correspond even to 1 gene but a living being can have more than 500. So, scientific evolution is at present not related to the questions that dominate religions and that emphasize the usual extremely high levels of complexity and perfection of extant life. Nevertheless, the general advance of computing science has allowed as to look at the Evolutionary Theory from a perspective that clearly shows that it is obviously false: were Darwinian evolution the explanation of our existence, the fossil record and extant population should be plagued with every sort of malfunctions and malformations. This is of

course, false. So, we contend that the Evolutionary Theory of the modern international literature is a scientific contradiction because it amounts to evolution (of form and function) without evolution (of perfection). Therefore, it is from now on insane to belief and immoral to teach that the Darwinian Evolutionary Theory explains life on Earth. Or, following a modern reading of Lamarck: to explain life with evolution, one needs to admit that randomness has been very strongly twisted much more than any science can resist.

Answers to exercises

Problems of Chapter 1

11. By sampling effects, observed frequencies of a given trait tend to vary, to drift, whenever they are different than zero or than one. But if observed frequencies take on one of these two extreme values, sampling cannot create modifications. So, evolution is halted, a trait gets fixed. Fixation is a rule unless other forces must exist to counterbalance it.

12. This was studied in Vol I of this series, see chapter 20 on genetic drift, say, program 328 and beyond.

13. We can dispense for the use of probabilities from the given program if we replace them by their equivalent operational definition that abstracts a relative frequency: **sample with replacement** an individual from the population (pick at random an individual, measure the desired trait and return it back to its former place), if it is T, generate a T, if it is C generate a C. Here, like begets like without mutation.

17. Testing whether or not the tables of relative frequencies converge to a limit. The program outputs the maximal difference between the corresponding cumulative frequencies to carry on a Kolmogorov-Smirnov test to check whether or not two cumulative tables come as samples from the same distribution. Verdict: one can accept that the tables of relative frequencies converge to a limit with a confidence level a bit lower than 95%. The K-S test is addressed to continuous distributions but our samples show that the distribution of the chosen observable has not long tails and has smooth central tendency, so the abuse is not that dangerous. The code follows:

```
//Program G17 grandFather3
//Calculation of an evolutionary theory.
/*  The  theory represents the usual behavior
 *  of calls from wired telephones (T)  or cell phones (C)
 *  to the grandfather by his family and relatives
 *  before the discovery of the lake. It has
 *  two main elements. First: received number
 *  of calls per week in the last time was  4, 5 or 6
 *  and with equal probability.  Second: people
 *  try to repeat the behavior of the last week,
 *  i.e.,  the  observed frequencies  of T and
 *  C that happened in the previous week served
 *  as probabilities to design the future in
 *  the following week. Moral law and the force
 *  of habits dictate that there must be at least
 *  one T and one C every week.
 *  The initial frequency of T  is 0.4.
 *  Observed facts that must be contrasted with
 *  the theory and that presumably falsify it:
 *  the total number of calls during the last three
 *  weeks has been 13 while in the  last five he
 *  received just 21 calls.
 *  To try to understand whether or not we deal with
 *  a probability density function, we pair
 *  two tables of relative frequencies taken at
 *  different generations.
 *
 */

import java.util.Random;

public class grandFather3
{
 private static int averageNCalls = 5;
 private static int randomShift;
 private static double initialFreqT = 0.4;
 private static int nEvents = 30000;
 private static int RecordCalls[ ] = new int[nEvents];
```

```
 //Tables of relative frequencies
 private static int Dist[ ] = new int[nEvents];
 private static double[] Table1 = new double[100];
 private static double[] Table2 = new double[100];
 //Tables of cumulative frequencies
 private static double[] Cum1 = new double[100];
 private static double[] Cum2 = new double[100];
 private static double[] DiscrepancyAbs = new double[100];
 //Number of groups of weeks
 private static int nGroups;
 //Number of weeks per group
 private static int nWeeks = 5;
 private static int counterCalls;
 //Observed number of calls
 private static int observed;
 private static int min, max;
 private static boolean print;
 //Turn on of the random generator
 private static Random r  = new Random();


//Basic Test of the random generator for random generation
//of 4's, 5's, 6's with equal probabilities
public static void testRandGen1()
{
 System.out.println("Tests the random generator" +
" \nfor random generation of 4's, 5's, 6's " +
"\nwith equal probabilties");
 int counter4 = 0;
 int counter5 = 0;
 int counter6 = 0;
 int nExp = 4000;
 for(int i= 0; i < 3*nExp; i++)
  {
randomShift = r.nextInt(3);
int numberCalls = averageNCalls + randomShift - 1;
System.out.println(numberCalls);
//counters are incremented in one
```

```
if(numberCalls == 4) counter4++;
if(numberCalls == 5) counter5++;
if(numberCalls == 6) counter6++;
  }
 System.out.println("Number of 4's = " + counter4);
 System.out.println("Number of 5's = " + counter5);
 System.out.println("Number of 6's = " + counter6);
 System.out.println("Expected number = " + nExp);
}

//Basic test of the random generator for random generation
//of T's with prob probT;
public static void testRandGen2(double probT)
{
 System.out.println("Tests the random generator" +
" \nfor random generation of T'2" +
"\nwith probability " + probT);
 int counterT = 0;
 int counterC = 0;
 //Total number of chars
 int n = 4000;
 double t;
 for(int i= 0; i < n; i++)
  {
   t = r.nextDouble();
   if (t < probT )
   {
    System.out.println("T");
    counterT++;
   }
  else
   {
System.out.println("C");
counterC++;
   }
 }
 System.out.println("Number of T's = " + counterT);
 System.out.println("Number of C's = " + counterC);
```

```
 System.out.println("Total number of chars = "  + n);
 double expT= n*probT;
 System.out.println("Expected number of T's = "  + expT);
}

public static double generation(double probT)
{
 //generates the number of calls for this week
 randomShift = r.nextInt(3);
 int callsThisWeek = averageNCalls + randomShift - 1;
 if (print)
 System.out.print("Number of calls = "
          + callsThisWeek + ": ");
 counterCalls = counterCalls + callsThisWeek;
 int counterT = 0;
 double t;
 //Generates actual type of calls:
 //Two of them are TC,
 if (print) System.out.print("T");
 counterT++;
 if (print) System.out.print("C");
 //Others calls are
 //generated with  prob of T = probT;
 for(int i = 2; i < callsThisWeek; i++)
  {
   t = r.nextDouble();
   if (t < probT )
{
   if (print) System.out.print("T");
 counterT++;
}
   else
   if (print) System.out.print("C");
  }
 double cT = counterT;
 double freqT = cT/callsThisWeek;
 if (print) System.out.print("    FreqT  = " + freqT);
 if (print) System.out.println();
```

```
 return freqT;
}

public static void evolution(double probT)
{
 int counterRecord = 0;
 for(int i = 0; i < nGroups; i++)
  {
   counterCalls = 0;
   for(int j = 0; j < nWeeks; j++)
    {
     if (print) System.out.print(  i + " " + j);
     probT = generation(probT);
    }
   if (print)
   System.out.println("Number of calls in " + nWeeks
   + " weeks = " + counterCalls);
   RecordCalls[counterRecord] = counterCalls;
   //counter is incremented by one.
   counterRecord++;
  }
}

//The proposed theory is studied:
//First: we find the distribution of calls during
//a group of nWeeks.
//Second: we find the significance of the
//given barriers. First barrier = 13 for groups of 3 weeks
//and the second is 21 for groups of 5 weeks.
//
//Find the distribution of numbers of total
//calls recorded in RecordCalls[]
public static  void findDistribution()
{
 min = nEvents;
 max = 0;
 for(int i = 0; i < nEvents; i++)
  {
```

```
Dist[RecordCalls[i]] ++;
//! means boolean negation
if (!(RecordCalls[i]==0))
 {
  if (RecordCalls[i] > max ) max = RecordCalls[i];
  if (RecordCalls[i] < min ) min = RecordCalls[i];
 }
  }
 System.out.println("Number of groups = " + nGroups);

}

//The pValue of observed value is found, i.e.
//the probability of getting an event as extreme or more
//than the observed value
public static  double[] findSignificance(int observed)
{
  double totalSum = 0;
  double partialSum = 0;
  for(int i = min; i <= max; i++)
   totalSum = totalSum + Dist[i];
  for(int i = min; i <= observed; i++)
  partialSum = partialSum + Dist[i];
  double pValue = partialSum/totalSum;
  double[] Table = new double[100];
  System.out.println("Number of calls" +
   " per week, absolute and relative frequencies");
  for(int i = min; i <= max; i++)
  {
   Table[i-min] = Dist[i];
   Table[i-min] = Table[i-min] / totalSum;
   System.out.println(i + " " + Dist[i] + "  "
                 + Table[i-min] );
  }
  System.out.println("Partial sum until " + observed +
  " =  " + partialSum);
  System.out.println("Total sum = " + totalSum);
  System.out.println("pValue = " + pValue);
```

```
  return Table;
}


//To reject a theory of null hypothesis
//on ordinary statistical grounds, we must make sure
//that we deal with a correctly defined probability
//density function. If that is the case, two
//tables of relative frequencies taken at
//delayed intervals must differ very little.
//A two sample Kolmogorov-Smirnov test is carried out.
public static void studyLimit()
{
 System.out.println("\nPairing of tables of " +
              "absolute frequencies.");
 System.out.println("\ntable1, table2, absolute difference");
 double maxDiff = 0;
 for(int i = 0; i < max-min +1; i++)
  {
   DiscrepancyAbs[i] = Math.abs(Table1[i] - Table2[i]);
   System.out.printf("%11.5f %11.5f %11.5f  %n",
       Table1[i], Table2[i], DiscrepancyAbs[i] );
   if ( DiscrepancyAbs[i] > maxDiff )
         maxDiff = DiscrepancyAbs[i];
  }
 System.out.println("Maximal difference = " + maxDiff);
 System.out.println("\nPairing of cumulative tables.");
 System.out.println("\nCum1, Cum2, absolute difference");

 Cum1[0] = Table1[0];
 Cum2[0] = Table2[0];
 double maxDiffCum = 0;
 for(int i = 1; i < max-min +1; i++)
  {
Cum1[i] = Cum1[i-1] + Table1[i];
Cum2[i] = Cum2[i-1] + Table2[i];
DiscrepancyAbs[i] = Math.abs(Cum1[i] - Cum2[i]);
System.out.printf("%11.5f %11.5f %11.5f  %n",  Cum1[i],
```

```
                    Cum2[i],    DiscrepancyAbs[i] );
   if ( DiscrepancyAbs[i] > maxDiffCum )
         maxDiffCum = DiscrepancyAbs[i];

  }
 System.out.println("Maximal difference in cumulative " +
                  "functions= " + maxDiffCum);
 double criticValue = 1.63 /Math.sqrt(nGroups);
 System.out.println("Critical value for alpha = 0.05 : "
  + criticValue);
 if ( maxDiffCum < criticValue )
 System.out.println("Samples come from the same distribution" +
  "with confidence level 95%");
 else
 System.out.println("Samples come from different" +
  " distribution with confidence level 95%");
}


public static void step1()
{
 print = false;
 boolean testRandGen1 = false;
 if (testRandGen1) testRandGen1();

 System.out.println("Simulation of calls week by week.");
 if (print)
  {
System.out.println("T = call from a wired telephone.");
System.out.println("C = call from a cell phone.");
System.out.println("Initial frequency of T's = "
+ initialFreqT);
}
 boolean testRandGen2 = false;
 if (testRandGen2) testRandGen2(initialFreqT);
}
```

```
public static void main(String[] args)
{
 step1();
 double probT = initialFreqT;
 //Initialization
 nGroups = 10000;
 for(int i = 0; i < nEvents; i++)
 RecordCalls[i] = 0;
 observed = 21;
 System.out.println("\nWeeks are grouped by  " + nWeeks);
 System.out.println("Observed number of calls  "
 + observed);
 evolution(probT);
 findDistribution();
 Table1 = findSignificance(observed);
 //studyTheory();
//Initialization
 for(int i = 0; i < nEvents; i++)
 {
  RecordCalls[i] = 0;
  Dist[i] = 0;
 }
 nGroups = 20000;
 for(int i = 0; i < nEvents; i++)
   {
RecordCalls[i] = 0;
Dist[i] = 0;
   }
 observed = 21;
 System.out.println("\nWeeks are grouped by  " + nWeeks);
 System.out.println("Observed number of calls  "
 + observed);
 evolution(probT);
 findDistribution();
 Table2 = findSignificance(observed);
 studyLimit();

}//End of main method
```

```
}//End of class
```

Problems of Chapter 2

20. We have witnesses that evolution can solve Syntax for the very special case of phrases with 7 words. It is too early to claim that we easily can pass from here to any number of words.

22. At its face value, the slowing of evolution due to recombination shows that the role of recombination has nothing to do with accelerating evolution although in some other evolutionary environments that might happen. To understand this failure, let us consider two chromosomes: 01234567 and 01256734. Recombination might produce 01234734, which has two words repeated and lacks other two. This is a clear hindrance to evolution. That is why evolution is slowed. This also contradict the very extended belief that the main role of recombination is to heal imperfect copies created by mutation. We see in the program that mimics a teacher that teach syntax to her pupils a witness to the contrary: every mutation produces viable products and so mutation is not harmful while recombination is harm prone. All in all, we conclude that evolution is not a panacea but rather a very brilliant idea that is very difficult to implement. That is why it represents a challenge: everyone is invited to become an expert thanks to hard work during many years. If you endure, a new world will unveil itself before your eyes: the existence of biological evolution will get open for you in all its mystery and splendor and you will feel challenged to go beyond cheap scientific explanations.

24. Gene positional selection. The next algorithm solves the *syntax problem* but not at the same pace as the former program. So, this type of selection captures the main ingredient of the organismic selection but not its complete flavor.

```
//Program G24 syntax2
//We mimic a game that a teacher
//proposes to her pupils:
//A disordered phrase is given to pupils.
//A disordered phrase is just a permutation of
//the original one, a variation of order of its words.
```

```
//They must restore from this
//string the original sentence
//with correct meaning and syntax.
//The problem is solved by means of evolution,
//in which GENE SELECTION is the driving force of the
//ensuing evolutionary process.
//In this case, fitness is given by the teacher,
//who already knows the answer and can estimate
//how good are the attempts of the pupils.

//To disorder the original correct phrase,
//a series of swaps or transpositions is done.
//The best way to correctly reorder the
//given disordered phrase is to use also swaps.

import java.util.Random;

public class syntax2
{
 private static String phrase =
  "syntax deals with the formation of correct sentences";
 private static int nLetters = phrase.length();
 private static int nWords;
 //The phrase is decomposed in words
 private static String[] Word = new String[10];
 //Words are numbered. A disordered phrase is encoded
 //in a vector of digits, Permutation[], such that
 //Permutation[i] indicates what word goes
 //at place i of the possibly disordered phrase.
 private static int[] Permutation = new int[10];
 //Turn on of the random generator
 private static Random r  = new Random();

 //===Genetic part of declaration of variables====
 // Individuals are kept in the array
 // Individual[]. It is an array of strings.
 //Each individual encodes
 //the permutation that it represents.
```

```
 //The individual is a string that encodes for
 //a number.


 // The number of individuals must be
 // less than limit.
 static double zMax; static double N;
 static int limit = 50000;
 static double Fitness[];
 static String Individual[ ], Individualc [];
 //Actual number of individuals
 static int nIndiv ;
 static int Order[];
 static String b;
 static int generation;
 static  int  nGen;
 static int ReportMin[], ReportMax[];
 static double oldError, newError;
 static double deltaError;
 static boolean print,testEncoding;
 //Our observable that declares success.
 static boolean done = false;
 static boolean recombination;
 static double ERROR, FITNESS;
 static String SOLUTION;
 static int GENERATION;

//Words are recognized: they are separated by a space.
//They are kept in a vector.
private static void numberWords(String s)
 {
  int counter = 0;
  String f = "";
  for(int i = 0; i < nLetters; i++)
   {
char d = s.charAt(i);
//System.out.println(d);
if (!(d == ' '))  f = f+d;
```

```
else
 {
  Word[counter] = f;
  f = "";
  counter++;
 }
//last word
if (i == nLetters-1) Word[counter] = f;
   }
  //System.out.println( counter);
  System.out.println("\nThe words of the phrase " +
                       "with their indexes: ");
  nWords = counter +1;
  for(int i = 0; i < nWords; i++)
   {
System.out.println(i + "  " +  Word[i]);
Permutation[i] =  i;
   }
 }

 //A transposition  is the interchange of
 //place between two items.
private static void transposition()
 {
  int m = r.nextInt(nWords);
  boolean flag = false;
  //Sites must be different
  int n=0;
  while(!(flag))
   { n = r.nextInt(nWords);
   if (!(n == m)) flag = true;
   }
  if (print) System.out.println("Permuation of "
                              +  m + " and " + n);
  int k = Permutation[m];
  Permutation[m] = Permutation[n];
  Permutation[n] = k;
 }
```

```
//Mixer:  a sequence of transpositions creates variability.
//The output is encoded in a string, a chromosome.
private static String mixer(int numberOfTransp)
 {
  for(int i= 0; i < numberOfTransp; i++ )
 transposition();
  String v = "";
  for(int i = 0; i < nWords; i++)
    {
  Integer e = Permutation[i];
  v = v + e.toString();
    }
  return v;
 }



//Words in the sentence encoded by s are permuted

private static String permutate (String s)
 {
  numberWords(s);
  //Number of transpositions
  int n =  nWords;
  return  mixer(n);
 }

//String number  is decoded into a phrase
private static String stringToPhrase(String number)
  {
   String phrase= "";
   for(int j=0;j  < number.length();j++ )
    {
     char s3  =   number.charAt(j);
    int  l = Character.getNumericValue(s3);
    phrase = phrase + Word[l] + " ";
```

```
    }
   return phrase;
  }

//Declarations and default initializations
// of other arrays.
private static void otherInit( )
{
 Order= new int[limit+1];
 ReportMin= new int[limit+1];
 ReportMax= new int[limit+1];
 Fitness= new double[limit +1];
 for(int i = 0; i< nIndiv; i++)
  {
   Order[i]=0;
   ReportMin[i]=0;
   ReportMax[i]=0;
   Fitness= new double[limit +1];
 }
}

//An individual is a string that is a permutation
//of another one.
private static String generateIndividual( )
{
 //Number of transpositions
 int numberOfTransp = r.nextInt(nWords);
 String ind = mixer(numberOfTransp);
 return ind;
}

/* We generate nIndiv individuals (strings)
 encoding for permutations of 01234567 */
private static void Initialization( )
{
 //Formal declaration of our array.
 Individual= new String[limit];
 if (print)  System.out.println("ORIGINAL POPULATION");
```

```
 for(int i = 0; i< nIndiv; i++)
 {
  if (print) System.out.println( "\ni = " + i);
     Individual[i]="";
 // Individual[i] is a string,
 // it is a genotype that encodes a number,
 //its phenotype.
     Individual[i] = generateIndividual();
     if (print)
      {
       System.out.println(" = " + Individual[i] );
      }
    }
 otherInit();
  }
```

```
//This method decodes the string into the entries
//of permutation[]
private static void stringToVector(String s)
{
 for(int j = 0; j  < nWords;j++ )
  {
char c  =   s.charAt(j);
    int  l = Character.getNumericValue(c);
Permutation[j] = l;
  }
}

//The error of   chromosome s is minus
//the number of wrong pairwise orderings.
//Say, 54 is wrong and obtains a penalty of -1.
private static double error(String s)
{
```

```
 stringToVector(s);
 int counter = 0;
 int a,b ;

 for(int j = 0; j  < nWords-1;j++ )
 {
char c  =   s.charAt(j);
    a = Character.getNumericValue(c);
     c  =   s.charAt(j+1);
    b = Character.getNumericValue(c);
    if (a > b ) counter--;
    //System.out.println( a + " " + b + " " + counter);
 }
 return counter;
}


//This fitness of each individual is found
private static void fitness(int gen)
{
 //We measure the error of individual[i]
 //with respect to the perfect one 01234567.
 //We adopt a gene insight.
 for(int i = 0; i< nIndiv; i++)
  {
   double  error = error(Individual[i]);

   Fitness[i] = error;
   if (print)
  System.out.println("i = " + i
  + " Individual = " + Individual[i]
  + " Error  = "+ error
  + " Fitness"  + Fitness[i]    );
  if (error > - 0.000000000001)
   {
 GENERATION = gen;
 ERROR = error;
 SOLUTION = Individual[i];
```

```
 FITNESS =  Fitness[i];
 done = true;
}
  if (print)  System.out.println();
  }
 }

//Individuals are sorted by fitness
private static void Sorting(int gen)
  {
   fitness(gen);
   if (print) System.out.println("\nSORTING");
   int   Champ;
  //Sorting by fitness.
   //Minimal fitness = - 6. Maximal 0.
  for(int i = 0; i< nIndiv;i++)
{
 Champ = 0;
 for(int j = 0; j< nIndiv;j++)
    if (Fitness[j] >= Fitness[Champ])   Champ = j;
     //The array Order classifies individuals by  fitness
  // by equal or decreasing order.
  Order[i] = Champ;
     if (print)
      {
       System.out.println(  i + "th ind. is  No "
    + Champ + " " + Individual[Order[i] ]
           + " Fitness = "  + Fitness[Champ]);
      }
     Fitness[Champ] = -10;
     }

String t = Individual[Order[0] ];
System.out.println(" Best solution "  + t  +
" -> " + stringToPhrase(t));
  }
```

```
//Each individual of the top ten
// produces 10  copies.
private static void Reproduction()
{
 if (print)
  System.out.println( "Reproduction");
 if (print)
  System.out.println( "The best = " + Order[0]);
 Individualc= new String[limit];
 int counter = 0;
 for (int top = 0; top < 10; top++)
 {
 for(int j = 0; j< 10; j++)
  {
    Individualc[counter] = Individual[Order[top]];
counter = counter +1;
  }
 }
 for(int j = 0; j< counter; j++)
Individual[j] = Individualc[j];

}

//A a sequence of transpositions creates variability.
private static String mixWords(String s,
                               int numberOfTransp)
 {
  //Chromosome s is transformed into a vector
  stringToVector(s);
  //Entries in vector are transposed
  for(int i= 0; i < numberOfTransp; i++ )
 transposition();
  //Vector is encoded into a chromosome, a string.
  String v = "";
  for(int i = 0; i < nWords; i++)
  {
  Integer e = Permutation[i];
```

```
  v = v + e.toString();
  }
  return v;
 }


//A mutation results from a series of transpositions
private static void Mutation()
{
  for(int j = 1; j<  nIndiv; j++)
{
 //Number of transpositions
 int n = r.nextInt(nWords);
 Individual[j] = mixWords( Individual[j],n);
}
}

//Two strings recombine and produce two offspring.
private static void Recombination()
{
 for(int j = 50; j<  nIndiv; j++)
  {
//Define place of recombination
int m =  r.nextInt(nIndiv);
int n = r.nextInt(nIndiv);
String a = Individual[m];
String b = Individual[n];
int placeRec = r.nextInt(nWords);
Individual[m] = a.substring(0, placeRec)
               + b.substring( placeRec);
Individual[n] = b.substring(0, placeRec)
       + a.substring( placeRec);
 }
}

//Master for a generation
private static void dynamics(int gen)
 {
```

```
  //Individuals are sorted by fitness
  Sorting(gen);
  //The top ten are preferentially reproduced
  Reproduction();
  //The new population is subjected to mutation
  Mutation();
  if (recombination) Recombination();
 }

//The original ordered phrase is rebuilt
//by a genetic algorithm.
//This program simulates a teacher that knows
//the answer and guides their students
//for a better solution.
private static void restorePhrase()
 {
  System.out.println("\n\nRunning ");
  nIndiv = 150;
  Initialization( ) ;
  nGen = 200000;
  oldError = 1000;
  for(int gen = 1; ( gen <= nGen) & (!(done)); gen++)
 {
   System.out.print("\nGENERATION = " + gen);
   dynamics(gen);
 }
  if (done)
   {
 System.out.println("\nGENERATION = " + GENERATION +
       " SOLUTION FOUND");
     System.out.println("ERROR = " + ERROR);
     System.out.println("SOLUTION = " + SOLUTION
      + " -> " +  stringToPhrase(SOLUTION) );
     System.out.println("FITNESS = " + FITNESS);
     System.out.println("Number of generations = "
      + GENERATION);
     System.out.println("Number of individuals " +
      "per generation = " + nIndiv);
```

```
     int totalCost = GENERATION * nIndiv;
     System.out.println("NUmber of Generations x " +
      "number of individuals = " + totalCost);
   }
 }

public static void main(String[] args)
 {
  print = false;
  recombination = false;
  System.out.println("This program disorders a sentence"
            +  " \nand reorders it by evolution "
            + "\nusing random transpositions. "
           + "Selection goes down to genes.\n");
  System.out.println("ORIGINAL SENTENCE: ");
  System.out.println(phrase);
  String p = permutate(phrase);
  System.out.println( "\nCHALLENGING DISORDERED SENTENCE" +
   " WITH ITS INDEXATION : \n" );
  System.out.println( p + " -> " + stringToPhrase(p) );
   restorePhrase();
 }//End of main method
}
```

Problems of Chapter 3

33. Mutation in Darwinian evolution is random over the alphabet of the evolutionary environment, which is that dictated by reproduction. For biological evolution of DNA, the alphabet is $\{A,T,C,G\}$ and the elements of the alphabet are called bases. In program G19 we have an evolutionary environment with an alphabet S = $\{0,1,2,3,4,5,6,7\}$. This is the alphabet dictated by reproduction. A random mutation over this alphabet would transform the chromosome 10234567 into, say, 12234567, which is clearly not in the path to perfection, represented by 01234567. Therefore, it is true that random mutations in tandem over alphabet S can generate transpositions but they must be accompanied with a lot of undesirable noise.Thus, Darwinian evolution certainly can produce transpositions but is very expensive. By contrast, transpositions in program G19 happen systematically,

with probability one and without noise. Most applications of evolution use the same idea that in here: instead of using the alphabet dictated by reproduction, suitable super-alphabets are employed. Applied evolution is always Lamarckian.

Problems of Chapter 4

52. The Author considers that the negative implementation is approximately equally good to guide evolution from scratch to perfection as the positive is. But the negative type of selection is easier to implement. Thus, for applied purposes, the negative flavor is better. For this reason, the Author considers that the creationist complaint, saying that evolution can change things to selectively equivalent variants else to the worst, is false. The evolutionary insight is that perfection is a concept that arises from the interrelation of the evolving object with its environment and that a death centered selection is quite good to achieve optima, i.e. perfection. Summing up, creationism to be socially acceptable needs to proclaim the glory of evolution, which is more powerful than anyone can imagine, but that is nevertheless very limited to battle complexity.

53. Twisted randomness in solo is shown to be almighty to solve problems of retro-engineering, in which we know the answer.

```
//Program G53 syntax9
//We mimic a game that a teacher
//proposes to her pupils:
//A disordered phrase is given to pupils.
//A disordered phrase is just a permutation of
//the original one, a variation of order of its words.
//They must restore from this
//string the original sentence
//with correct meaning and syntax.
//The problem is solved by means of TWISTED CHANCE alone.
//To twist randomness, we filter
//generated random numbers and  accept
//a digit only if it fits in the  need of the moment.
//In this case, fitness is given by the teacher,
//who already knows the answer and can decide
//whether or not a proposal is useful
```

```
//for the overall purpose.


import java.util.Random;

public class syntax9
{
 private static String phrase =
  "syntax deals with the formation of correct sentences";
 private static int nLetters = phrase.length();
 private static int nWords;
 //The phrase is decomposed in words
 private static String[] Word = new String[10];
 //Words are numbered. A disordered phrase is encoded
 //in a vector of digits, Permutation[], such that
 //Permutation[i] indicates what word goes
 //at place i of the possibly disordered phrase.
 private static int[] Permutation = new int[10];
 //Turn on  the random generator
 private static Random r  = new Random();
 private static boolean print;



//Words are recognized: they are separated by a space.
//They are kept in a vector.
private static void numberWords(String s)
 {
  int counter = 0;
  String f = "";
  for(int i = 0; i < nLetters; i++)
   {
char d = s.charAt(i);
//System.out.println(d);
if (!(d == ' '))  f = f+d;
else
 {
  Word[counter] = f;
```

```
  f = "";
  counter++;
 }
//last word
if (i == nLetters-1) Word[counter] = f;
   }
  //System.out.println( counter);
  System.out.println("\nThe words of the phrase " +
                       "with their indexes: ");
  nWords = counter +1;
  for(int i = 0; i < nWords; i++)
   {
System.out.println(i + "  " +  Word[i]);
Permutation[i] =  i;
   }
 }

 //A transposition  is the interchange of
 //place between two items.
private static void transposition()
 {
  int m = r.nextInt(nWords);
  boolean flag = false;
  //Sites must be different
  int n=0;
  while(!(flag))
   { n = r.nextInt(nWords);
   if (!(n == m)) flag = true;
   }
  if (print) System.out.println("Permuation of "
                               +  m + " and " + n);
  int k = Permutation[m];
  Permutation[m] = Permutation[n];
  Permutation[n] = k;
 }

//Mixer:  a sequence of transpositions creates variability.
//The output is encoded in a string, a chromosome.
```

```
private static String mixer(int numberOfTransp)
 {
  for(int i= 0; i < numberOfTransp; i++ )
 transposition();
  String v = "";
  for(int i = 0; i < nWords; i++)
    {
  Integer e = Permutation[i];
  v = v + e.toString();
    }
  return v;
 }



//Words in the sentence encoded by s are permuted

private static String permutate (String s)
 {
  numberWords(s);
  //Number of transpositions
  int n =  nWords;
  return  mixer(n);
 }

//String number  is decoded into a phrase
private static String stringToPhrase(String number)
  {
   String phrase= "";
   for(int j=0;j  < number.length();j++ )
    {
     char s3  =   number.charAt(j);
    int  l = Character.getNumericValue(s3);
    phrase = phrase + Word[l] + " ";

    }
   return phrase;
  }
```

```
//Digits that are generated at random
//are filtered and are admitted only those
//that fit the precise need.
//So, the first organism is made perfect
//from the pure beginning.
//This is twisted randomness at its maximal expression:
//all one needs is one individual to create perfection.
//The random generator is supposed
//to represent random events in a real world.
//The twisting procedure only has the information
//to regulate things.
private static void matchPhrase()
 {
  System.out.println("\n\nRunning ");
  boolean go = true;

  Integer n = 1234567;
  //target = 01234567
  String target = n.toString();
  target = "0" + target;
  String trial = "";
  int k1;
  for(int i = 0; i < nWords; i++)
   {
 while(go)
   {
 //Generate digits at random
 k1 = r.nextInt(nWords);
 //Digits are filtered before
 //they go into a new organism
 if ( k1 == i )
  {
   //if the digit is needed right now,
   //let it be part of the new chromosome.
   Integer k = k1;
       trial =  trial + k.toString();
       go = false;
```

```
  }
   }
  // System.out.print("\n i  = " + i);
    go = true;
   System.out.println(" phrase in construction  = " + trial
 +  " -> " + stringToPhrase(trial) );
  }
   if (trial.equalsIgnoreCase(target))
   {
   System.out.println("\nAttempt = " + 1 +

       " SOLUTION FOUND");

     System.out.println("Number of individuals " +
      "per attempt = 1 " );
   }
 }


public static void main(String[] args)
 {
  print = false;
  System.out.println("This program disorders a sentence"
         + " \nand reorders it by twisted chance. ");
  System.out.println("ORIGINAL SENTENCE: ");
  System.out.println(phrase);
  String p = permutate(phrase);
  System.out.println( "\nCHALLENGING DISORDERED SENTENCE" +
   " WITH ITS INDEXATION : \n" );
  System.out.println( p + " -> " + stringToPhrase(p) );
   matchPhrase();
 }//End of main method
}
```

Problems of Chapter 5

54. Let us imagine for a moment that we indeed appeared as a fruit of evolution on Earth:

1. We are not against the belief that one can achieve perfection in Darwinian evolution. But a price in bugs must be paid. The trace of their existence must be apparent as in the fossil record as in extant populations through every kind of malformations and genetic malfunctions not by disease but by faulty design. In Lamarckian evolution there is possibly a path from scratch to perfection but it goes so fast that we cannot demand the existence of malfunctions or malformations neither in extant populations nor in the fossil record. Therefore and in the light of complexity, Lamarckian evolution seems more suitable to explain our existence that Darwinian one.

2. The most plausible implementation of Lamarckian evolution might be by claiming the existence of aliens that converted our planet in a lab of directed evolution. Nevertheless, we are very far from achieving an agreement on whether or not they exist or existed. By the same token, in spite of the fact that the international news reveal videos of ghosts in action, science is as yet not ready to accept that as a scientific fact. Therefore, spiritual claims do not belong in science. At the same time, most scientists consider at present that Darwinian evolution on earth is all we need to explain life. Summing up: Lamarckian evolution has not gained a place in science.

3. Modern International Literature sees how well the fossil record and taxonomic data are fit by a genetic tree dominated by a trend towards complexity and hominization and then teaches everyone that life is obviously explained by evolution. Nevertheless, the envisaged type of evolution cannot be Darwinian since it provides no cue in regard with the mandatory price that Darwinian evolution must pay: bugs followed by corrections that create more bugs. The evolution of the Modern International Literature cannot be Lamarckian either because there is no mention to external agents or forces that could help evolution to defeat complexity. Rather, Modern International Literature is pray of a lack of clear concepts. Clarity springs as follows:

   (a) The genome is software and Darwinian evolution is really a software developer.
   (b) Evolution according to Darwinian evolution is a real natural phenomenon that must obey the laws of nature, the very first of which is: there is no software development without bugs followed by corrections that generate more bugs. Bugs must reveal

themselves through malformations and malfunctions all through the fossil record and extant populations.

(c) If a robust evidence of a path towards perfection is lacked, as it indeed happens, Darwinian evolution cannot be invoked to explain the existence of the various millions of species that inhabit the Earth nowadays.

4. Because the International Literature unduly teaches that we are explained by Darwinian Evolution, it forces everyone to ignore the universal and trans-cultural witnesses in favor of the existence of a spirit that lives after the death of the body together with the many religions that point to a Creator. Nevertheless, our discussion by no means entails a scientific creationism or the like:

   (a) While all religions tolerate and even favor some scientific studies and discussions, the core of some religions is definitively not scientific. Say, who can bring forth the risen Jesus to see how tall he is? Therefore, all efforts, if any, to make of Christendom a scientific doctrine is based on misconceptions. Rather, let the believer to apply his or her religion to his or her life and let him or her reap and taste the fruit of his or her deeds to see how well the vacuum for eternity that exists in his or her soul has been filled.

   (b) To say that reality has things that are outside the scope of science and experimentation is to claim that our physics is and must be incomplete. This postulate is a regulative one and does not belong in science. Some people will accept it, while other would reject it, and never will be agreement between the two parties. Nevertheless, if you clearly understand all these points, you will have no need to kill someone that thinks not like you.

Problems of Chapter 6

61. The problem is that we have two alphabets. The first is $\{0,1,2,...,13\}$, in which each number represents a word. The second is $\{0,1...,9\}$ which the alphabet we use to represent numbers in base 10. Which is the correct alphabet? Which one is dictated by reproduction? The answer is this: the alphabet of a given evolutionary environment is not dictated by the used

codons but by the letters that compose them. To get convinced of this, let us think of the DNA string ATTTAGT. This string contains two codons but the concept of codon is unknown to reproduction and mutation. The only important thing for mutation is that this string contains 6 sites where a random change can be made. So, the alphabet to decide whether or not evolution is Darwinian is uniquely defined by the way as natural mutation works. For DNA, it is $\{A,T,G,C\}$. For our program, the alphabet dictated by reproduction is $\{0,1,2,...,9.\}$ The implemented mutation in the program was random over $\{0,1,2,...,13\}$ but it cannot be random over $\{0,1,2,...,9\}$ since executed mutations contains correlations if they are seen from the stand of the alphabet $\{0,1,2,...,9\}$. Specifically, the char 0 is more probably to appear at even places because it must accompany all digits, say, as in 05.

62. Darwinian evolution follows.

```
//Program G62 syntaxb2
//This program shows one very simple and
//inexorable law: if evolution can indeed find the answer,
//a price must be paid: a lot of bugs must be made,
//and therefore,
//a clear path towards perfection must exist.
//We show here
//how long one is frozen in a local minimum and
//how many bugs evolution makes.
//We have a BUG each time that appears
//an offspring that is less fitted than their parents.
//
//We define fitness as minus the number of mismatches
//of  given chromosome with respect to the optimal one.
//
//Selection is negative: the probability of dead
//before reproduction is the higher the
//less is the fitness of the individual.
//
//Problem to solve:
//We mimic a game that a teacher
//proposes to her pupils:
//A phrase is subject to a series of
//permutations of its words to end with a string
```

```
//with no meaning. Pupils must produce from this
//string the original sentence
//with correct meaning and syntax.
//
//The problem is solved by means of
//DARWINIAN EVOLUTION,
//in which selection is the driving force of the
//ensuing evolutionary process and
//mutation is random over the alphabet.
//In this case, selection is given by the teacher,
//who already knows the answer and can estimate
//how good are the attempts of the pupils.
//
//Robustness of the code against perturbations
//of the input  is enhanced to deal
//with sentences with 100  words or less.
//This is done trying to enable EVOLVABILITY,
//i.e., facilitating the future adaptation of
//the code to digest phrases with more than 100 words.

import java.util.Random;

public class syntaxb2
{
 private static String phrase =
  "one can use evolution to solve a problem " +
  "but without a trace of evolvability";
 private static String target ="";
 private static int nLetters = phrase.length();
 private static int nWords;
 //The phrase is decomposed in words
 private static String[] Word = new String[100];
 //A disordered sentence is just a permutation of
 //the original one:
 private static String[] Permutation = new String[100];
 //Turn on of the random generator
 private static Random r  = new Random();
 //Mutation rate per symbol per generation
```

```
private static double mutRate;

//======Genetic part of declaration of variables====
// Individuals are kept in the array
// Individual[]. It is an array of strings.
//Each individual encodes
//the permutation that it represents.
//The individual is a string that encodes for
//a number.


// The number of individuals must be
// less than limit.
static double zMax; static double N;
static int limit = 50000;
static int[] Fitness;
static double[]  FitnessCopy;
static String[] Individual, Individualc ;
//Actual number of individuals
static int nIndiv ;
static int[] Order;
static String b;
static int generation;
static  int  nGen;
static int[] ReportMin, ReportMax;
static double oldError, newError;
static double deltaError;
static boolean print,testEncoding;
//Our observable that declares success.
static boolean done = false;
static boolean recombination;
static double ERROR, FITNESS;
static String SOLUTION;
static int GENERATION;
//Number of generation with no progress
//because  of a local minimum.
static int nEcstasy = 1;
static int oldBestFitness = -100;
```

```
 static int newBestFitness = 0;
 //Gained fitness
 static int gainedFitness;
 //Number of bugs during a generation
 //(individuals whose performance is worst
 //than that of their parents).
 static int nBugs = 0;
 //Number of bugs while expecting a jump
 static int nTotalBugsEcstasy = 0;
 static int bigTotalBugs = 0;

//Words are recognized: they are separated by a space.
//They are kept in a vector.
private static void numberWords(String s)
 {
  int counter = 0;
  String f = "";
  for(int i = 0; i < nLetters; i++)
   {
char d = s.charAt(i);
//System.out.println(d);
if (!(d == ' '))  f = f+d;
else
 {
  Word[counter] = f;
  f = "";
  counter++;
 }
//last word
if (i == nLetters-1) Word[counter] = f;
   }
  //System.out.println( counter);
  System.out.println("\nThe words of the phrase " +
                       "with their indexes: ");
  nWords = counter +1;
  for(int i = 0; i < nWords; i++)
   {
System.out.println(i + "  " +  Word[i]);
```

```
Integer e = i;
String h = e.toString();
  if (h.length() == 1) h = '0'+h;
Permutation[i] =  h;
target =target + h;
   }
  System.out.println("Target = phrase is " +
            "encoded in a chromosome = " + target);
 }

 //A transposition  is the interchange of
 //place between two items.
private static void transposition()
 {
  int m = r.nextInt(nWords);
  boolean flag = false;
  //Sites must be different
  int n=0;
  while(!(flag))
   { n = r.nextInt(nWords);
   if (!(n == m)) flag = true;
   }
  if (print) System.out.println("Permuation of "
                               +  m + " and " + n);
  String k = Permutation[m];
  Permutation[m] = Permutation[n];
  Permutation[n] = k;
 }

//Mixer:  a sequence of transpositions creates variability.
//The output is encoded in a string, a chromosome.
private static String mixer(int numberOfTransp)
 {
  for(int i= 0; i < numberOfTransp; i++ )
 transposition();
  String v = "";
  for(int i = 0; i < nWords; i++)
    {
```

```
  String e = Permutation[i];
  v = v + e.toString();
    }
  return v;
 }

//Words in the sentence encoded by s are permuted

private static String permutate (String s)
 {
  numberWords(s);
  //Number of transpositions
  int n =  nWords;
  return  mixer(n);
 }

//String number  is decoded into a phrase
private static String stringToPhrase(String s)
 {
  String phrase= "";
  for(int j=0;j  < nWords;j++ )
   {
    char s1 =   s.charAt(2*j);
    int  p1 = Character.getNumericValue(s1);
  s1  =   s.charAt(2*j+1);
  int  p2 = Character.getNumericValue(s1);
  int number = p1*10 + p2;
  if (print) System.out.println( "number = " + number);
  String d = "";
  if ( number < nWords ) d = Word[number];
  else d = "HHH";
  phrase = phrase + d + " ";
   }
  return phrase;
 }


//Declarations and default initializations
```

```
// of other arrays.
private static void otherInit( )
 {
  Order = new int[limit+1];
  ReportMin = new int[limit+1];
  ReportMax = new int[limit+1];
  Fitness = new int[limit +1];
  FitnessCopy = new double[limit +1];
  for(int i = 0; i< nIndiv; i++)
  {
   Order[i] =0;
   ReportMin[i] =0;
   ReportMax[i] =0;
   Fitness[i] = 0;
   FitnessCopy[i] = 0;
 }
 }

//An individual is a string that is a permutation
//of another one.
private static String generateIndividual( )
{
 //Number of transpositions
 int numberOfTransp = r.nextInt(nWords);
 String ind = mixer(numberOfTransp);
 return ind;
}

/* We generate nIndiv individuals (strings)
 encoding for permutations of 01234567 */
private static void Initialization( )
{
 //Formal declaration of our array.
 Individual = new String[limit];
 Individualc = new String[limit];
 if (print)  System.out.println("ORIGINAL POPULATION");
 for(int i = 0; i< nIndiv; i++)
  {
```

```
   if (print) System.out.println( "\ni = " + i);
   Individual[i]="";
   Individualc[i]="";
   // Individual[i] is a string,
   // it is a genotype that encodes a number,
   //its phenotype.
   Individual[i] = generateIndividual();
   if (print)
      System.out.println(" = " + Individual[i] );
  }
  otherInit();
}


//This method decodes the string s into the entries
//of permutation[]. Letters are taken by pairs.
private static void stringToVector(String s)
{
 if (print) System.out.println("s = " + s);
 for(int j = 0; j  < nWords;j++ )
  {
String t = s.substring(2*j, 2*j + 2);
Permutation[j] = t;
if (print)  System.out.println("t = " + t);
  }
}


//The fitness of each individual is found
private static void fitness(int gen)
{
 //We measure the error of individual[i]
 //with respect to the perfect one 01234567.
 for(int i = 0; i< nIndiv; i++)
  {
   int  error = 0;
   String s = Individual[i];
   if (print) System.out.println( "s = " + s);
```

```
   for(int j = 0; j  < nWords;j++ )
   {
String a = s.substring(2*j, 2*j + 2);
String b = target.substring(2*j, 2*j + 2);
if (print) System.out.println( "a = " + a + " b = " + b);
    if (!(a.equals(b))) error++;
   }

   //The error defines the fitness: less error,
   //more fitness.
   //Maximal fitness = 0; minimal = -10.
   Fitness[i] = - error;
   FitnessCopy[i] = - error;
   if (print)
  System.out.println("i = " + i
  + " Individual = " + Individual[i]
  + " Error  = "+ error
  + " Fitness"  + Fitness[i]    );
  if (error  == 0)
   {
 GENERATION = gen;
 ERROR = error;
 SOLUTION = Individual[i];
 FITNESS =  Fitness[i];
 done = true;
}
  if (print)  System.out.println();
  }
 }



//Individuals are sorted by fitness
private static void sorting(int gen)
{
 fitness(gen);
 nBugs = 0;
 if (print) System.out.println("\nSORTING");
```

```
 int   Champ = 0;
 //Sorting by fitness.
 for(int i = 0; i< nIndiv;i++)
  {
   //The i-th individual is picked out
   for(int j = 0; j< nIndiv;j++)
    if (Fitness[j] >  Fitness[Champ])  Champ = j;
   //The array Order classifies individuals by  fitness
   Order[i] = Champ;
   if (print)
    {
     System.out.println(  i + "th ind. is  No "
    + Champ + " " + Individual[Order[i] ]
           + " Fitness = "  + Fitness[Champ]);
    }
   if (i == 0) newBestFitness = Fitness[Champ];
   Fitness[Champ] = -10;
   Champ = 0;
  } //end of for i
  //Bug's section
  for(int j = 0; j< nIndiv;j++)
  if (FitnessCopy[j] <  FitnessCopy[Champ])  nBugs++;
  if (print) System.out.println( "Number of bugs  = "
                       + nBugs);
  nTotalBugsEcstasy  = nTotalBugsEcstasy  + nBugs;
  bigTotalBugs = bigTotalBugs + nBugs;
  String t = Individual[Order[0] ];
  gainedFitness = newBestFitness - oldBestFitness;
  //If an evolutionary  jump, an improvement in fitness,
  //is detected...
  if (gainedFitness > 0)
   {
System.out.println("\nGENERATION = " + gen);
System.out.println(" Best solution "  + t  +
" -> " + stringToPhrase(t));
System.out.print("Generations needed for" +
" a new jump = " +  nEcstasy);
System.out.println(", with " + nTotalBugsEcstasy
```

```
                      + " bugs");
System.out.println("Old best fitness = "
+ oldBestFitness);
System.out.println("New best fitness = "
+ newBestFitness);
System.out.println("Gained fitness = " +
 gainedFitness);
oldBestFitness = newBestFitness;
nEcstasy = 1;
nTotalBugsEcstasy  = 0;
   }
 else
  {
   nEcstasy++;
  }
} //End of sorting


//Negative selection:
//The probability of dead is higher,
//the lower is the fitness
private static void death()
{
 //Fitness is rescaled
 double min = 100;
 double max = -100;
 for(int j = 0; j< nIndiv;j++)
  {
   if (FitnessCopy[j] > max ) max = FitnessCopy[j];
   if (FitnessCopy[j] < min ) min = FitnessCopy[j];
  }

 //
 for(int j = 0; j< nIndiv;j++)
  {
   double probDeath = -FitnessCopy[j]/(max-min) + max/(max-min);
   //System.out.println("ProbDeath = " + probDeath);
   double p = r.nextDouble();
```

```
   if (p <=probDeath) Individual[j] = "";
  }
}

//A random not nil individual is chosen.
private static int  randomIndiv()
{
 boolean cloned = false;
 int k = 0;
 while (cloned == false)
  {
k = r.nextInt(nIndiv);
if (!(Individual[k] == ""))
cloned = true;
  }
  if (print) System.out.println("Cloned from = " + k);
 return k;
}

//The pre-population of the new generation
//is built with clones
//of living individuals  sampled at random
//with replacement.
private static void Reproduction()
{
 if (print)
 System.out.println( "Reproduction");
 if (print)
 System.out.println( "The best = " + Order[0]);
 for(int j = 0; j<  nIndiv; j++)
 Individualc[j] = Individual[randomIndiv()];
  for(int j = 0; j< nIndiv; j++)
Individual[j] = Individualc[j];
}

//String s is subject to mutation
private static String mutate(String s)
{
```

```
 int nLetters = s.length();
 String v = "";
 //Mutate or not mutate
 double p = r.nextDouble();
 //Entries in vector are subject to mutation
 if ( p < mutRate )
  {
//Site of mutation is chosen at random
int site = r.nextInt(nLetters);
//Effect of mutation is chosen at random
Integer change = r.nextInt(10);
  v =  s.substring(0, site) + change.toString()
      + s.substring( site+1 );
  }
 else v = s;
 if (print)
System.out.println( "old = " + s + "\nnew = " + v);
 return v;
}


//Mutation is Darwinian:
//The alphabet consists in the letter of codons.
//Both the site of mutation and the change
//are defined by randomness.
private static void Mutation()
{
  for(int j = 1; j<  nIndiv; j++)
{
 Individual[j] = mutate( Individual[j]);
}
}



//Two strings recombine and produce two offspring.
private static void Recombination()
{
```

```
 for(int j = 50; j<  nIndiv; j++)
  {
//Define place of recombination
int m =  r.nextInt(nIndiv);
int n = r.nextInt(nIndiv);
String a = Individual[m];
String b = Individual[n];
int placeRec = r.nextInt(2*nWords);
Individual[m] = a.substring(0, placeRec)
               + b.substring( placeRec);
Individual[n] = b.substring(0, placeRec)
       + a.substring( placeRec);
 }
}

//Master for a generation
private static void dynamics(int gen)
 {
  //Individuals are sorted by fitness
  sorting(gen);
  //Differential death
  death();
  //The top ten are preferentially reproduced
  Reproduction();
  //The new population is subjected to mutation
  Mutation();
  if (recombination) Recombination();
 }

//The original ordered phrase is rebuilt
//by a genetic algorithm.
//This program simulates a teacher that knows
//the answer and guides their students
//for a better solution.
private static void restorePhrase()
 {
  System.out.println("\n\nRunning ");
  nIndiv = 5;
```

```
  Initialization( ) ;
  nGen = 5000000;
  oldError = 1000;
  for(int gen = 1; ( gen <= nGen) & (!(done)); gen++)
 {

   dynamics(gen);
 }
  if (done)
   {
 System.out.println("\nGENERATION = " + GENERATION +
       " SOLUTION FOUND");
     System.out.println("ERROR = " + ERROR);
     System.out.println("SOLUTION = " + SOLUTION
      + " -> " +  stringToPhrase(SOLUTION) );
     System.out.println("FITNESS = " + FITNESS);
     System.out.println("Number of generations = "
      + GENERATION);
     System.out.println("Number of individuals " +
      "per generation = " + nIndiv);
     long totalCost = GENERATION * nIndiv;
     System.out.println("Total cost = " +
      "Number of Generations x " +
      "number of individuals = " + totalCost);
     System.out.println("Big Total  of bugs "
                              + bigTotalBugs);
     double tC = totalCost;
     double tB = bigTotalBugs;
     System.out.println("Normalized cost = " +
      "Big Total of bugs/ Total cost = "
             + tB/tC);
   }
 }

public static void main(String[] args)
 {
  print = false;
  recombination = true;
```

```
  mutRate = 0.5;
  System.out.println("This program disorders a sentence"
     + " (with the help of transpositions), \n"
         + "and reorders it by evolution \n"
         + "using random mutations over the \n"
         + "alphabet 0,1,2,3,...,n, \n"
         + "where n is the number of words \n"
         + "of the sentence. \n"
         + "\nEvolution is Darwinian.\n");
  System.out.println("ORIGINAL SENTENCE: ");
  System.out.println(phrase);
  String p = permutate(phrase);
  System.out.println( "\nCHALLENGING DISORDERED SENTENCE" +
   " WITH ITS INDEXATION : \n" );
  System.out.println( p + " -> " + stringToPhrase(p) );
   restorePhrase();
 }//End of main method
}
```

63. The next code resolves syntax for sentences with a number of words less than 1000. This program is a modification of program G59 syntaxb1, page 128. The modification was of a degree of complexity that was just enough to generate fun and a bit of stress. So, we consider that program G59 indeed awakes an evolvable branch.

```
//Program G63 syntaxb3
//This program shows one very simple and
//inexorable law: if evolution can indeed find the answer,
//a price must be paid: a lot of bugs must be made,
//and therefore,
//a clear path towards perfection must exist.
//We show here
//how long one is frozen in a local minimum and
//how many bugs evolution makes.
//We have a BUG each time that appears
//an offspring that is less fitted than their parents.
//
```

```
//We define fitness as minus the number of mismatches
//of  given chromosome with respect to the optimal one.
//
//Selection is negative: the probability of dead
//before reproduction is the higher the
//less is the fitness of the individual.
//
//Problem to solve:
//We mimic a game that a teacher
//proposes to her pupils:
//A phrase is subject to a series of
//permutations of its words to end with a string
//with no meaning. Pupils must produce from this
//string the original sentence
//with correct meaning and syntax.
//The problem is solved by means of
//LAMARCKIAN EVOLUTION,
//in which evolution is helped with suitable cues.
//In this case, selection is given by the teacher,
//who already knows the answer and can estimate
//how good are the attempts of the pupils.
//Style= global variables.
//
//Robustness of the code against perturbations
//of the input  is enhanced to deal
//with sentences with 1000  words or less.
//This is done trying to enable EVOLVABILITY,
//i.e., facilitating the future adaptation of
//the code to digest phrases with more than 100 words.


import java.util.Random;

public class syntaxb3
{
 private static String phrase =
  "one can use evolution to solve a problem " +
  "but without a trace of evolvability";
```

```
//A long phrase is the result of nCopies
//of the original one.
private static int nCopies;
private static String target ="";
private static int nLetters;
private static int nWords;
//The phrase is decomposed in words
private static String[] Word = new String[1000];
//A disordered sentence is just a permutation of
//the original one:
private static String[] Permutation = new String[1000];
//Turn on of the random generator
private static Random r  = new Random();
//Mutation rate per symbol per generation
private static double mutRate;

//====Genetic part of declaration of variables====
// Individuals are kept in the array
// Individual[]. It is an array of strings.
//Each individual encodes
//the permutation that it represents.
//The individual is a string that encodes for
//a number.


// The number of individuals must be
// less than limit.
static double zMax; static double N;
static int limit = 50000;
static int[] Fitness;
static double[]  FitnessCopy;
static String[] Individual, Individualc ;
//Actual number of individuals
static int nIndiv ;
static int[] Order;
static String b;
static int generation;
static  int  nGen;
```

```
 static int[] ReportMin, ReportMax;
 static double oldError, newError;
 static double deltaError;
 static boolean print,testEncoding;
 //Our observable that declares success.
 static boolean done = false;
 static boolean recombination;
 static double ERROR, FITNESS;
 static String SOLUTION;
 static int GENERATION;
 //Number of generation with no progress
 //because  of a local minimum.
 static int nEcstasy = 1;
 static int oldBestFitness = -1000;
 static int newBestFitness = 0;
 //Gained fitness
 static int gainedFitness;
 //Number of bugs during a generation
 //(individuals whose performance is worst
 //than that of their parents).
 static int nBugs = 0;
 //Number of bugs while expecting a jump
 static int nTotalBugsEcstasy = 0;
 static int bigTotalBugs = 0;

//Words are recognized: they are separated by a space.
//They are kept in a vector.
private static void numberWords(String s)
 {
  nLetters = s.length();
  int counter = 0;
  String f = "";
  for(int i = 0; i < nLetters; i++)
   {
char d = s.charAt(i);
//System.out.println(d);
if (!(d == ' '))  f = f+d;
else
```

```
 {
  Word[counter] = f;
  f = "";
  counter++;
 }
//last word
if (i == nLetters-1) Word[counter] = f;
   }
  //System.out.println( counter);
  System.out.println("\nThe words of the phrase " +
                       "with their indexes: ");
  nWords = counter +1;
  for(int i = 0; i < nWords; i++)
   {
System.out.println(i + "  " +  Word[i]);
Integer e = i;
String h = e.toString();
if (h.length() == 1) h = "00" + h;
  if (h.length() == 2) h = '0'+h;
Permutation[i] =  h;
target =target + h;
   }
  System.out.println("Target = phrase is " +
            "encoded in a chromosome = " + target);
 }

 //A transposition  is  the interchange of
 //place between two items.
private static void transposition()
 {
  int m = r.nextInt(nWords);
  boolean flag = false;
  //Sites must be different
  int n=0;
  while(!(flag))
   { n = r.nextInt(nWords);
   if (!(n == m)) flag = true;
   }
```

```
  if (print) System.out.println("Permuation of "
                                +  m + " and " + n);
  String k = Permutation[m];
  Permutation[m] = Permutation[n];
  Permutation[n] = k;
 }

//Mixer:  a sequence of transpositions creates variability.
//The output is encoded in a string, a chromosome.
private static String mixer(int numberOfTransp)
 {
  for(int i= 0; i < numberOfTransp; i++ )
 transposition();
  String v = "";
  for(int i = 0; i < nWords; i++)
    {
  String e = Permutation[i];
  v = v + e.toString();
    }
  return v;
 }

private static String generateLongSentence(String s)
{
  String v = "";
  for(int i = 0; i < nCopies; i++)
v = v + s;
  return v;
}


//Words in the sentence encoded by s are permuted
private static String permutate (String s)
 {
  s = generateLongSentence(s);
  System.out.println("Long phrase = " + s);
  numberWords(s);
  //Number of transpositions
```

```
  int n =  nWords;
  return  mixer(n);
 }


//String number  is decoded into a phrase
private static String stringToPhrase(String s)
 {
  String phrase= "";
  for(int j=0;j  < nWords;j++ )
   {
    char s1 =   s.charAt(3*j);
    int  p1 = Character.getNumericValue(s1);
  s1  =   s.charAt(3*j+1);
  int  p2 = Character.getNumericValue(s1);
  s1  =   s.charAt(3*j+2);
  int  p3 = Character.getNumericValue(s1);
  int number = p1*100 + p2*10 + p3;

  if (print) System.out.println( "number = " + number);
  phrase = phrase + Word[number] + " ";
   }
  return phrase;
 }


//Declarations and default initializations
// of other arrays.
private static void otherInit( )
{
 Order = new int[limit+1];
 ReportMin = new int[limit+1];
 ReportMax = new int[limit+1];
 Fitness = new int[limit +1];
 FitnessCopy = new double[limit +1];
 for(int i = 0; i< nIndiv; i++)
  {
   Order[i] =0;
```

```
   ReportMin[i] =0;
   ReportMax[i] =0;
   Fitness[i] = 0;
   FitnessCopy[i] = 0;
 }
}

//An individual is a string that is a permutation
//of another one.
private static String generateIndividual( )
{
 //Number of transpositions
 int numberOfTransp = r.nextInt(nWords);
 String ind = mixer(numberOfTransp);
 return ind;
}

/* We generate nIndiv individuals (strings)
 encoding for permutations of 01234567 */
private static void Initialization( )
{
 //Formal declaration of our array.
 Individual = new String[limit];
 Individualc = new String[limit];
 if (print)  System.out.println("ORIGINAL POPULATION");
 for(int i = 0; i< nIndiv; i++)
  {
   if (print) System.out.println( "\ni = " + i);
   Individual[i]="";
   Individualc[i]="";
   // Individual[i] is a string,
   // it is a genotype that encodes a number,
   //its phenotype.
   Individual[i] = generateIndividual();
   if (print)
      System.out.println(" = " + Individual[i] );
  }
  otherInit();
```

```
}


//This method decodes the string s into the entries
//of permutation[]. Letters are taken by pairs.
private static void stringToVector(String s)
{
 if (print) System.out.println("s = " + s);
 for(int j = 0; j  < nWords;j++ )
  {
String t = s.substring(3*j, 3*j + 3);
Permutation[j] = t;
if (print)  System.out.println("t = " + t);
  }
}


//The fitness of each individual is found
private static void fitness(int gen)
{
 //We measure the error of individual[i]
 //with respect to the perfect one 01234567.
 for(int i = 0; i< nIndiv; i++)
  {
   int  error = 0;
   String s = Individual[i];
   if (print) System.out.println( "s = " + s);
   for(int j = 0; j  < nWords;j++ )
   {
String a = s.substring(3*j, 3*j + 3);
String b = target.substring(3*j, 3*j + 3);
if (print) System.out.println( "a = " + a + " b = " + b);
    if (!(a.equals(b))) error++;
   }

   //The error defines the fitness: less error,
   //more fitness.
   //Maximal fitness = 0; minimal = -10.
```

```
   Fitness[i] = - error;
   FitnessCopy[i] = - error;
   if (print)
  System.out.println("i = " + i
  + " Individual = " + Individual[i]
  + " Error  = "+ error
  + " Fitness"  + Fitness[i]    );
  if (error  == 0)
   {
 GENERATION = gen;
 ERROR = error;
 SOLUTION = Individual[i];
 FITNESS =  Fitness[i];
 done = true;
}
  if (print)  System.out.println();
  }
 }



//Individuals are sorted by fitness
private static void sorting(int gen)
{
 fitness(gen);
 nBugs = 0;
 if (print) System.out.println("\nSORTING");
 int   Champ = 0;
 //Sorting by fitness.
 for(int i = 0; i< nIndiv;i++)
  {
   //The i-th individual is picked out
   for(int j = 0; j< nIndiv;j++)
    if (Fitness[j] >  Fitness[Champ])  Champ = j;
   //The array Order classifies individuals by  fitness
   Order[i] = Champ;
   if (print)
    {
```

```
     System.out.println(  i + "th ind. is  No "
    + Champ + " " + Individual[Order[i] ]
            + " Fitness = "  + Fitness[Champ]);
    }
   if (i == 0) newBestFitness = Fitness[Champ];
   Fitness[Champ] = -1000;
   Champ = 0;
  } //end of for i
  //Bug's section
  for(int j = 0; j< nIndiv;j++)
  if (FitnessCopy[j] <  FitnessCopy[Champ])  nBugs++;
  if (print) System.out.println( "Number of bugs  = "
                        + nBugs);
  nTotalBugsEcstasy  = nTotalBugsEcstasy  + nBugs;
  bigTotalBugs = bigTotalBugs + nBugs;
  String t = Individual[Order[0] ];
  gainedFitness = newBestFitness - oldBestFitness;
  //If an evolutionary  jump, an improvement in fitness,
  //is detected...
  if (gainedFitness > 0)
   {
System.out.println("\nGENERATION = " + gen);
System.out.println(" Best solution "  + t  +
" -> " + stringToPhrase(t));
System.out.print("Generations needed for" +
" a new jump = " +  nEcstasy);
System.out.println(", with " + nTotalBugsEcstasy
                     + " bugs");
System.out.println("Old best fitness = "
+ oldBestFitness);
System.out.println("New best fitness = "
+ newBestFitness);
System.out.println("Gained fitness = " +
 gainedFitness);
oldBestFitness = newBestFitness;
nEcstasy = 1;
nTotalBugsEcstasy  = 0;
   }
```

```
 else
  {
   nEcstasy++;
  }
} //End of sorting


//Negative selection:
//The probability of dead is higher,
//the lower is the fitness
private static void death()
{
 //Fitness is rescaled
 double min = 100;
 double max = -100;
 for(int j = 0; j< nIndiv;j++)
  {
   if (FitnessCopy[j] > max ) max = FitnessCopy[j];
   if (FitnessCopy[j] < min ) min = FitnessCopy[j];
  }

 //
 for(int j = 0; j< nIndiv;j++)
  {
   double probDeath = -FitnessCopy[j]/(max-min) + max/(max-min);
   //System.out.println("ProbDeath = " + probDeath);
   double p = r.nextDouble();
   if (p <=probDeath) Individual[j] = "";
  }
}

//A random not nil individual is chosen.
private static int  randomIndiv()
{
 boolean cloned = false;
 int k = 0;
 while (cloned == false)
  {
```

```
k = r.nextInt(nIndiv);
if (!(Individual[k] == ""))
cloned = true;
  }
  if (print) System.out.println("Cloned from = " + k);
 return k;
}

//The pre-population of the new generation
//is built with clones
//of living individuals  sampled at random
//with replacement.
private static void Reproduction()
{
 if (print)
 System.out.println( "Reproduction");
 if (print)
 System.out.println( "The best = " + Order[0]);
 for(int j = 0; j<  nIndiv; j++)
 Individualc[j] = Individual[randomIndiv()];
  for(int j = 0; j< nIndiv; j++)
Individual[j] = Individualc[j];
}

//A a sequence of transpositions creates variability.
private static String mutate(String s)
{
 //Chromosome s is transformed into vector Permutation[]
 stringToVector(s);
 //Mutate or not mutate
 double p = r.nextDouble();
 //Entries in vector are subject to mutation
 if ( p < mutRate )
   {
//Site of mutation is chosen at random
int site = r.nextInt(nWords);
//Effect of mutation is chosen at random
Integer change = r.nextInt(nWords);
```

```
String h = change.toString();
if (h.length() == 1) h = "00" + h;
if (h.length() == 2) h = '0'+h;
Permutation[site] = h;
   }
 //Vector is encoded into a chromosome, a string.
 String v = "";
 for(int i = 0; i < nWords; i++)
  {
  String h = Permutation[i];
  v = v + h;
  }
 return v;
}

//A mutation is Darwinian:
//both the site of mutation and the change
//are defined by randomness.
private static void Mutation()
{
  for(int j = 1; j<  nIndiv; j++)
{
 Individual[j] = mutate( Individual[j]);
}
}



//Two strings recombine and produce two offspring.
private static void Recombination()
{
 for(int j = 50; j<  nIndiv; j++)
  {
//Define place of recombination
int m =  r.nextInt(nIndiv);
int n = r.nextInt(nIndiv);
String a = Individual[m];
String b = Individual[n];
```

```
if (print)
 System.out.println("a = " + a + " \nb = " + b);
int placeRec = r.nextInt(3*nWords);
Individual[m] = a.substring(0, placeRec)
               + b.substring( placeRec);
Individual[n] = b.substring(0, placeRec)
       + a.substring( placeRec);
 }
}

//Master for a generation
private static void dynamics(int gen)
 {
  //Individuals are sorted by fitness
  sorting(gen);
  //Differential death
  death();
  //The top ten are preferentially reproduced
  Reproduction();
  //The new population is subjected to mutation
  Mutation();
  if (recombination) Recombination();
 }

//The original ordered phrase is rebuilt
//by a genetic algorithm.
//This program simulates a teacher that knows
//the answer and guides their students
//for a better solution.
private static void restorePhrase()
 {
  System.out.println("\n\nRunning ");
  nIndiv = 10;
  Initialization( ) ;
  nGen = 5000000;
  oldError = 1000;
  for(int gen = 1; ( gen <= nGen) & (!(done)); gen++)
 {
```

```
   dynamics(gen);
 }
  if (done)
   {
 System.out.println("\nGENERATION = " + GENERATION +
       " SOLUTION FOUND");
     System.out.println("ERROR = " + ERROR);
     System.out.println("SOLUTION = " + SOLUTION
      + " -> " +  stringToPhrase(SOLUTION) );
     System.out.println("FITNESS = " + FITNESS);
     System.out.println("Number of generations = "
      + GENERATION);
     System.out.println("Number of individuals " +
      "per generation = " + nIndiv);
     long totalCost = GENERATION * nIndiv;
     System.out.println("Total cost = " +
      "Number of Generations x " +
      "number of individuals = " + totalCost);
     System.out.println("Big Total  of bugs "
                           + bigTotalBugs);
     double tC = totalCost;
     double tB = bigTotalBugs;
     System.out.println("Normalized cost = " +
      "Big Total of bugs/ Total cost = "
            + tB/tC);
   }
 }

public static void main(String[] args)
 {
  print = false;
  recombination = true;
  mutRate = 0.5;
  nCopies = 2;
  System.out.println("This program disorders a sentence"
     + " (with the help of transpositions), \n"
         + "and reorders it by evolution \n"
```

```
        + "using random mutations over the \n"
        + "alphabet 0,1,2,3,...,n, \n"
        + "where n is the number of words \n"
        + "of the sentence. \n"
        + "\nEvolution is Darwinian.\n");
  System.out.println("ORIGINAL SHORT SENTENCE: ");
  System.out.println(phrase);
  System.out.println("Number of copies: " + nCopies);
  String p = permutate(phrase);
  System.out.println( "\nCHALLENGING DISORDERED SENTENCE" +
   " WITH ITS INDEXATION : \n" );
  System.out.println( p + " -> " + stringToPhrase(p) );
   restorePhrase();
 }//End of main method
}
```

Bibliography

[1] Academy of Achievement. (circa 2000) Interview: Ernst Mayr. The Darwin of the 20th Century
http://achievement.org/autodoc/printmember/may1int-1
Verified 2/XII/2011

[2] Bar-Yam S (2011) Lamarck vs. Darwin
http://necsi.edu/projects/evolution/lamarck/intro./lamarck_intro.html
Verified 2/XII/2011

[3] Cressey D (2011) Alternative medicine and science. A legacy of scepticism. Edzard Ernst, the world's first professor of alternative medicine, is stepping down. Published online 30 May 2011, Nature, doi:10.1038/news.2011.322
//http://www.nature.com/news/2011/110530/full/news.2011.322.html
Verified 2/XII/2011

[4] Darnell J, Lodish H, Baltimore D (1986) *Molecular cell biology*. Scientific American Books.

[5] Darwin C (1859) *The origin of species* Murrap, Londres.

[6] Dawkins R (1976). The Selfish Gene. Oxford University Press, Oxford, UK.

[7] Dawkins R (1986) The Blind Watchmaker. W.W.Norton & Company, NY, London.

[8] de Queiroz K (2005) Ernst Mayr and the modern concept of species. PNAS May 3, 2005 vol. 102 no. Suppl 1 6600-6607
http://www.pnas.org/content/102/suppl.1/6600.full
Verified 2/XII/2011

[9] Dobzhansky T (1973) Nothing in Biology Makes Sense Except in the Light of Evolution The American Biology Teacher, March 1973 (35:125-129).

http://people.delphiforums.com/lordorman/light.htm
Verified 2/XII/2011

[10] Endler J (1980) Natural Selection on Color Patterns in Poecilia reticulata Evolution, Vol. 34, No. 1. (Jan., 1980), pp. 76-91.
http://links.jstor.org/sici?sici=0014-3820%28198001%2934%3A1%3C76%3ANSOCPI%3E2.0.CO%3B2-X
Verified

[11] Ernst E (2000) The role of complementary and alternative medicine, PMC 321(7269): 1133–1135. PMCID: PMC1118903
http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1118903/
Verified 2/XII/2011

[12] Ernst E (2010) Die Homöopathie ist ein Dogma. Homöopathie: Der Streit über die angeblichen Wundermittel. Spiegel Online.
http://www.spiegel.de/wissenschaft/medizin/0,1518,706257,00.html
Verified 2/XII/2011

[13] Fauci A,- (1984) Immune system.
http://www.niaid.nih.gov/topics/immunesystem/pages/default.aspx
Verified 2/XII/2011

[14] Gil S, Segal D, Ringo J, Hefetz A, Zilber-Rosenberg I, and Rosenberg E (2010) Commensal bacteria play a role in mating preference of Drosophila melanogaster. PNAS — November 16, 2010 — vol. 107 — no. 46 — 20051–20056
www.pnas.org/cgi/doi/10.1073/pnas.1009906107
Verified 2/XII/2011

[15] Gould S (2002) The Structure of Evolutionary Theory. Belknap, Harvard.

[16] Grant P, Grant R (2009) The secondary contact phase of allopatric speciation in Darwin's finches. PNAS , December 1, 2009 , vol. 106, no. 48, 20141–20148
www.pnas.org/cgi/doi/10.1073/pnas.0911761106
Verified 2/XII/2011

[17] Hey J, Fitch W, Ayala F (2005) Systematics and the origin of species: An introduction PNAS , May 3, 2005 , vol. 102 suppl. 1 6515– 6519
http://www.pnas.org/content/102/suppl.1/6515.full
Verified 2/XII/2011

[18] Lamarck J B (1794) Recherches sur les causes de principaux faits physiques, Tome premier. Maradan, Paris.
http://www.lamarck.cnrs.fr/
Verified 2/XII/2011

[19] Lamarck J B (1809) Philosophie zoologique, ou Exposition des considérations relatives à l'histoire naturelle des animaux... Dentu , Paris.
http://www.lamarck.cnrs.fr/
Verified 2/XII/2011

[20] Lavoisier A (1789) Traité élémentaire de chimie. Les Oevres de Lavoisier
http://www.lavoisier.cnrs.fr/livres.html
Verified 2/XII/2011

[21] Lienhard J (1997) Engines of Our Ingenuity. No 76: About Alchemists.
http://www.uh.edu/engines/epi76.htm
Verified 2/XII/2011

[22] Lönnig W (2001) Johann Gregor Mendel: Warum seine Entdeckungen 35 (72) Jahre ignoriert wurden (with an English summary)
http://www.weloennig.de/mendel.htm
Verified 2/XII/2011

[23] Martufi G (2007) Software evolvability: an industry's view.
http://www.resist-noe.org/DOC/2nd_OpenWorkshop/slides/Responder-Evolvability.pdf
Verified 2/XII/2011

[24] Mavárez J, Salazar C, Bermingham1 E, Salcedo C, Jiggins C, Linares M (2006) Speciation by hybridization in Heliconius butterflies. Nature 441, 868-871 (15 June 2006) doi:10.1038/nature04738;
http://www.nature.com/nature/journal/v441/n7095/abs/nature04738.html
Verified 2/XII/2011

[25] Mayr, E. (1997) The objects of selection Proc. Natl. Acad. Sci. USA 94 (March): 2091-2094.
http://www.pnas.org/content/94/6/2091.full
Verified 2/XII/2011

[26] Mendel, J.G. (1866). Versuche über Pflanzenhybriden. Verhandlungen des naturforschenden Vereines in Brünn, Bd. IV für das Jahr, 1865 Abhandlungen:3–47.
http://www.mendelweb.org/MWGerText.html
Verified 2/XII/2011

For the English translation, see: Druery, C.T and William Bateson (1901). ”Experiments in plant hybridization”. Journal of the Royal Horticultural Society 26: 1–32.
http://www.esp.org/foundations/genetics/classical/gm-65.pdf
Verified 2/XII/2011

[27] Michel A, Sim S, Powell T, Taylor M, Nosil P, Feder J (2010) Widespread genomic divergence during sympatric speciation. 9724–9729 — PNAS — May 25, 2010 — vol. 107 — no. 21
www.pnas.org/cgi/doi/10.1073/pnas.1000939107
Verified 2/XII/2011

[28] Moore R (2001) ”The ”Rediscovery” of Mendel’s Work”. Bioscene. Vol 27(2), pp 13-26.
http://amcbt.indstate.edu/volume_27/v27-2p13-24.pdf
Verified 2/XII/2011a

[29] Morgan T (1933) ”Thomas H. Morgan - Nobel Lecture”. Nobelprize.org. 22 Sep 2011a
http://www.nobelprize.org/nobel_prizes/medicine/laureates/1933/morgan-lecture.html
Verified 2/XII/2011

[30] Muller (1946) ”Hermann J. Muller - Nobel Lecture”. Nobelprize.org. 22 Sep 2011b
http://www.nobelprize.org/nobel_prizes/medicine/laureates/1946/muller-lecture.html
Verified 2/XII/2011

[31] NAS (2004) Evolution in Hawaii: A Supplement to Teaching About Evolution and the Nature of Science
http://www.nap.edu/openbook.php?record_id=10865&page=1
Verified 2/XII/2011

[32] Palumbi S, Lessios H (2005) Evolutionary animation: How do molecular phylogenies compare to Mayr’s reconstruction of speciation patterns in the sea? 6566 – 6572 , PNAS, May 3, 2005, vol. 102, suppl. 1
http://www.pnas.org/content/102/suppl.1/6566.full
Verified 2/XII/2011

[33] Poe E (1841)The Murders in the Rue Morgue
http://poestories.com/text.php?file=murders
Verified 2/XII/2011

[34] Rens I (2009) Lamarck, fondateur de la biologie et précurseur du transformisme par Ivo Rens dans Le Courrier (quotidien Suisse) du mardi 7 juillet 2009.
http://www.lecourrier.ch/index.php?name=NewsPaper&file=article&sid=442909
Verified 2/XII/2011

[35] Rodriguez J (2010) The genome is software and evolution is a software developer. arxiv.org/abs/1003.0575
http://arxiv.org/abs/1003.0575
Verified 2/XII/2011

[36] Sample I (2001) Prince Charles branded a 'snake oil salesman' by scientist. guardian.co.uk, Monday 25 July 2011 18.05 BST Article history.
http://www.guardian.co.uk/science/2011/jul/25/prince-charles-snake-oil-salesman
Verified 2/XII/2011

[37] Stolz U, Velez S, Wood K, Wood M, Feder† J (2003) Darwinian natural selection for orange bioluminescent color in a Jamaican click beetle PNAS, December 9, 2003 , vol. 100 , no. 25 , 14955–14959.
www.pnas.org/cgi/doi/10.1073/pnas.2432563100
Verified 2/XII/2011

[38] The Alchemy Web Site (2011) Emerald Tablet of Hermes
http://www.levity.com/alchemy/emerald.html
Verified 2/XII/2011

[39] The Three Initiates (1908) The Kybalion,
www.hermetics.org/pdf/kybalion.pdf
Verified 2/XII/2011

[40] UCMP (2011a) The history of evolutionary thought. Understanding Evolution. University of California Museum of Paleontology.
http://evolution.berkeley.edu/evolibrary/article/history_19
Verified 2/XII/2011

[41] UCMP (2011b) Understanding evolution for teachers. Mutations.
http://evolution.berkeley.edu/evosite/evo101/IIIC1Mutations.shtml
Verified 2/XII/2011

[42] Wallace A R (1855) On the Law Which Has Regulated the Introduction of New Species (S20: 1855) Annals and Magazine of Natural History, pp 184-194.

http://people.wku.edu/charles.smith/wallace/S020.htm
Verified 2/XII/2011

[43] Weismann A (1893) Germ-Plasm, a theory of Heredity. On-line Electronic Edition: Electronic Scholarly Publishing. Prepared by Robert Robbins
http://www.esp.org/books/weismann/germ-plasm/facsimile/title3.html
Verified

Index