SAPIENZA
UNIVERSITÀ DI ROMA

DEPARTMENT OF COMPUTER SCIENCE

The Battle of Sexes

A Java Simulation implementing Game Theory

PROGRAMMING 2

Academic year 2021/2022

Professor

Pietro Cenciarelli

Students

Ramil Leonard Vincent

Gai Annafabia

Lakasz Cristina

Contents

1 Introduction

This project is inspired by Chapter 9 of The Selfish Gene, a famous book by Richard Dawkins that builds upon the *Adaptation and Natural Selection* theory as a way of expressing the **gene-centered** view of evolution.
The chapter analyzes the evolution of a population sample and the study lead to the observation of the so called "the battle of the sexes". The population model features two male types, the faithful (F) and the philanderers (P), and two female types, the coy (C) and the fast (S).
The latter are the players of Dawkins' *Game Theory* based study in which he articulates a mathematical model of conflict and cooperation between rational decision-makers.

Each type of Individual has different strategy of survival:

– **faithful men:** they are willing to engage in a long courtship and participate in rearing their children;
– **philanderer man**: reckless men, they don't waste time in courting women: if not immediately accepted, they move away and try somewhere else; moreover, if accepted,they mate and then leave anyway, ignoring the destiny of their children;
– **coy women**: they accept a partner only after a long courtship;
– **fast women**: if they feel so, they don't mind copulating with whoever they like,even if just met.

The evolutionary payoffs involved in the study are the following:

a : the evolutionary benefit for having a baby

b : the cost of parenting a baby

c : the cost of courtship

Next we represent the payoffs resulting from a woman of type X engaging with a man of type Y by means of a pair (x, y), where x is the payoff of X, and y that of Y . According to the characters described above, the battle of the sexes can be formalized as in the following matrix, which may be used in defining the evolution rules.

	F	P
C	$(a - b/2 - c, a - b/2 - c)$	$(0, 0)$
S	$(a - b/2, a - b/2)$	$(a - b, a)$

Table 1: Evolutionary rules set in the payoff matrix

The goal of the project is to perform non-deterministic simulations that are parametric to the values a, b, c, that take into account the behavior of male and female types, and return the percentages of the four categories of people at the end of the simulations, analyzing the system's equilibrium state.

2 Design

The concept of this Battle of Sexes' simulation is to create a population of the 4 different types of individuals: coys, fasts, faithfuls and philanders, each with different evolutionary payoffs, make them interact with each other and observe the population trend during each generation and the progressive approach to the stability of the population, the ultimate goal of this project.

2.1 Generations

The analysis of generations gives the possibility to evaluate the number of children procreated after each round of matches. It gives interesting insights on the trend of the population and the allows to see the number of individual of each category and the number of faithful that have decided to rebel from their current type and become philanders instead. The matching of man and women, (where the man is the one to approach the opposite sex), the procreation and the influence of each relationship in the life of each individual are the core properties on which each generation relies on. Relationships have influence on the individuals involved as each meeting has a defined pay-off that dictates the ultimate evolution trend of the population and the relative approach to the stability of it.

2.2 Rebels

Each society has individuals that don't accept the rules they have to adapt to. Many are the ones that decide to eradicate themselves for the social constructs they don't accept, and many are the ones that change their life style, form what had been imposed to them, to what their truly nature is. Those are the people that are considered rebels, those that don't fit in, but

don’t disown their nature either. Our simulation takes act of this real life casuistry, and implements rebels as well. In our model some faithful man understand that long courtship and taking care of their children is not really what they strive for in their lives, they then decide to rebel from the social status that had been imposed to them and to become philanders instead, as a life of recklessness and courting women is more suitable for their attitude and is what they are most drawn to. The same concept applies to coy women, that, tired of waiting a long courtship before accepting a partner, they decide to change their morality and live a more thoughtless life as fast women.

3 Implementation

3.1 Human package

3.1.1 Human abstract class

The abstract class Human extends the class Threads and provides a template for the future Man and Woman classes. It sets as fields the default score and life of each individual. The methods that the class provides are: SetScore, which will be used to set the scores of each person, that depends on the tendency for a human to procreate. The less the tendency, the less the score. SetLife, that will be used to decrement the life of each person based on its score. If the score of a person is low then the life will be decremented more compared to the decrement of a person with higher score. IsDead checks, after we take off the years in SetLife, if the person is still alive or already is dead.

3.1.2 Man class

The Man class extends class Human. The constructor of this class extends the father’s constructor (class’s Human), and allows to set the type of the man: Philander or Faithful. The method in the class provides the implementation of the Thread method Run. Each man will be a thread and in order to execute the method we have to check if the thread (the person) is still alive, if it is then we insert the person inside of the dating queue; then the man will extract the woman and itself from the two different queues and make the meeting happen. In the meeting method we calculate the evolution rules given by the Dawkins’ Game Theory. Depending on the result we set the score and then we update the life in SetLife. After all this procedures we create the baby by using a random bound in order to determine the type.

3.1.3 Woman class

The woman class extends class Human. The constructor of this class extends the father’s constructor (class’s Human), and allows to set the type of the man: Fast or Coy. The methods provided in this class are method wakeUp, which allows to call a woman in the dating list when a man is available for a possible match, and method run (implementing the Thread Run method), which allows a woman to enter the dating list, where she will be waiting until an available man approaches her for a match. Women, as men, will then be regarded as a thread.

3.2 Match package

3.2.1 Dance class

The dance class allows men and women, that are collected in a queue, to be extract at the moment of the meeting between the two opposite sex. The attributes provided are dancing, which indicate the opening of the "dancing floor" where men and women can meet each other, and synchro which is the data structure used to collect the men and women that will then be extracted upon call of the former.

3.2.2 Queue class

The queue class provides a simple implementation of the queue data structure, through liked lists, which will be broadly used throughout the project. Each element of the queue has a value element and a next element (also called head and tail). In the head we'll find the actual value of the object, whereas in the tail we will find the reference to the next element. If there is no reference the value of the tail will be equal to null, since it would be the last element.

3.2.3 Synchro class

The synchro class takes usage of the queue structure implemented in the queue class in order to collect man and women separately and in a synchronized way. Here we implement the main methods that every queue must have in order to work. Insert, is a generic void method, that will put in the specific queue (either man or woman) the element that we passed as a parameter. Methods isEmpty, used to check whether the queue are empty returns true if the value of the element is equal to null. Extract, used to extract a person from the collections, it will return the element only if the queue is not empty.

3.3 Planet earth package

3.3.1 Evolution class

The Evolution class allows to take note of the generations of the population throughout the simulation. The attributes provided are counters to collect the numbers of children of each type of people that had been born. Two methods are implemented here: evoRules and baby. The former sets the evolutionary rules that the simulation will follow, the same as those involved in Dawkins's study, explained in the introduction chapter, and the consequent scores that each type of individual will receive during the population growth. The method baby provides the implementation to give birth to the children and will set the babies types randomly after each successful match; the counters of each type will increment accordingly to track the population trends after each generation.

3.3.2 Population class

The population class has as fields the three important parameters a, b, c, the peculiarity of this study. Other attributes of this class are the number of people of each type at the

beginning of the simulation, the queues of the four types of people and the queues of dead individuals. Methods getMin and getMax are the methods that regulate the life of each individual based on the evolutionary rules that are the base of the simulation. Method creation is used in order to insert into the queues and into the population the new individuals of the four different types. A general queue called population is created in order to contain all individuals. The extractWoman and extractMan methods are used to extract men and women from the specific queues, when it is needed. Methods isEmptyWoman and isEmptyMen are used to check whether the two queues are empty. At last, methods clearQueue and clearPopul are used to clear the queues in order to move to the next generation.

3.3.3 Rebellion class

The rebellion class implements what had been explained in the Rebels section. This class extends the class Evolution and regulates, through mathematical computations, the number of individuals of each type, that will change their connotations, this amount is stored and printed after each generation. The important methods present are:

Check_rebel_men through the appropriate calculations we observe that 63% of Faithfull, hence stability, corresponds to having the following expressions:

NumFaithfull = 1.7 * NumPhilander.

Check_rebel_women through the appropriate calculations we observe that 83% of Coy, corresponds to having the following expressions: NumCoy = 5 * NumFast.

Rebels_men and Rebels_women is a method where an ArrayList is created. At index zero is put the number of faithful men and coy women that we currently have. At index one is stored the number of man and women that want to rebel.

Rebellion_men and Rebellion_women clear the original ArrayList with the number of types and updates it with the number obtained in rebels_men and rebels_women.

3.4 Simulation package

3.4.1 Graphics

The implementation of graphics was done straight forward by importing the open source library provided by XChart – Knowm.org. The type of chart used is QuickChart, and it has been used for displaying both the population trend chart and the ratio trend chart of the simulation. The data taken into account is, on the x-axis the number of the generations, on the y-axis the population, or the ratio, of each of the 4 types: coy, fast, faithful and philander. Some reference of the charts will be provided in the results and evaluation chapter.

3.4.2 Simulation

The Simulation class is the one where the main method is provided. Here the interactions between men and women begin to take place, by creating the population through the call of the Population class's constructor. The population's evolution then starts to take place and, after each generation the details of such, are displayed.

4 Results and evaluations

The completion of the simulation gives us the possibility to evaluate the output of such, and gives us a deep understanding of the

4.1 Graphical visualization

As a visualization aid we implemented two graphical plot charts to help with the understanding of the population trends and the ratio of each individual type of the men population in relation to the complete men population and each individual type of the women in relation to the complete women population. The former chart allows to see the growth of each population type during every generation, whereas the latter allows to visualize the trend towards stability.

The library being used in order to implement the charts is: XChart. QuickChart is the type of chart we decided to use, as it provides a simple implementation and is clear and effective in our purpose.

4.2 Analysis of the outcome

We consider the population to be stable when the ratio of each type of Individual is growing asymptotically and when the ratios and percentages of Dawkins are respected.
The main key to reach stability in our program is the manipulation of data inside of the rebellion class.

Base Case

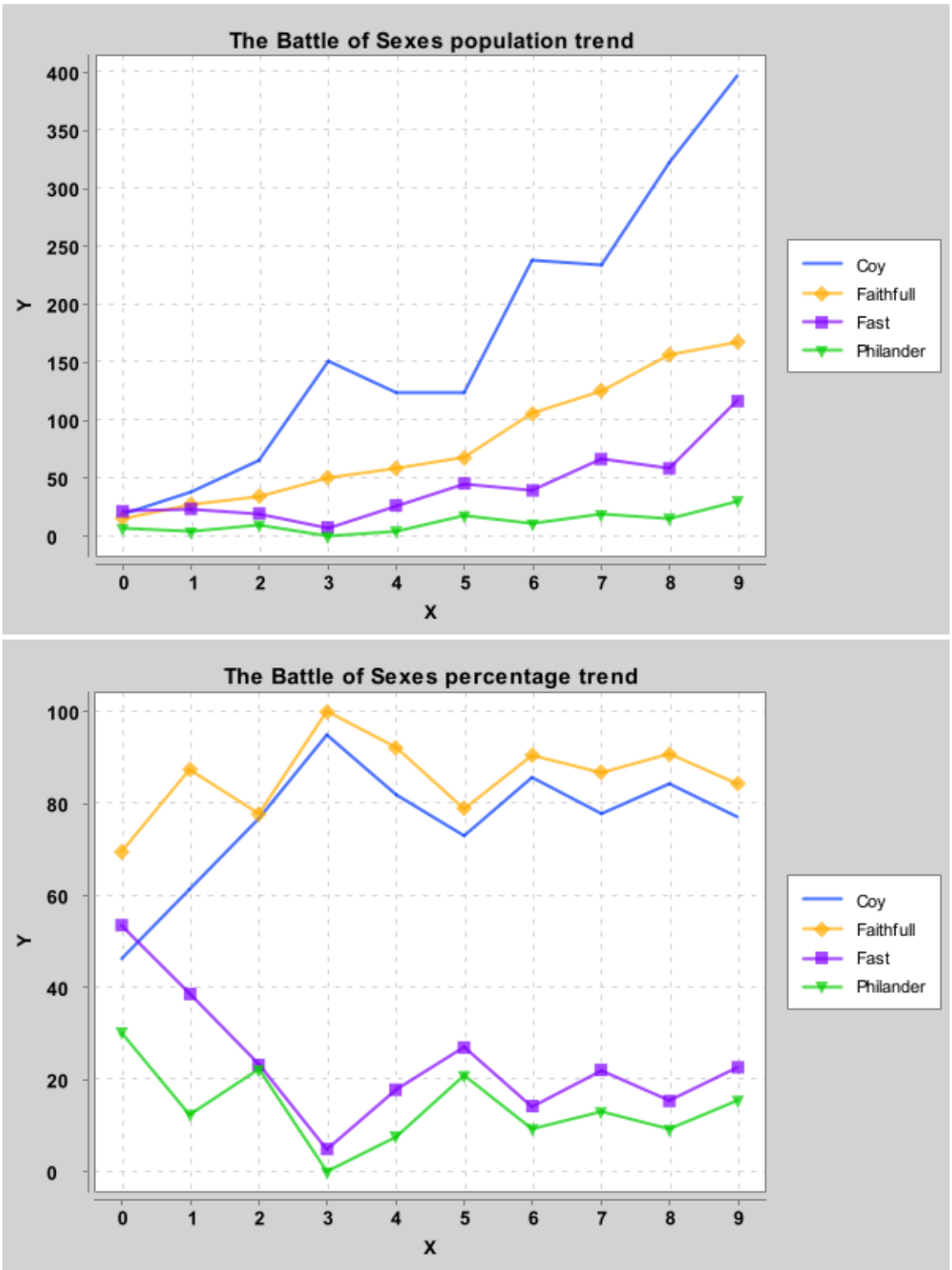

Figure 1: Philanders 10 - Faithful 10 - Fast 10 - Coy 10
a = 15 b = 20 c = 3

GENERATION	Philander	Faithful	Coy	Fast
0	7	16	19	22
1	4	28	38	24
2	10	35	66	20
3	0	51	152	8
4	5	59	124	27
5	18	68	124	46
6	11	106	239	40
7	19	126	235	67
8	16	157	323	59
9	31	168	397	117

Faithful	63% of men
Coy	77% of women

C parameter equal to 1

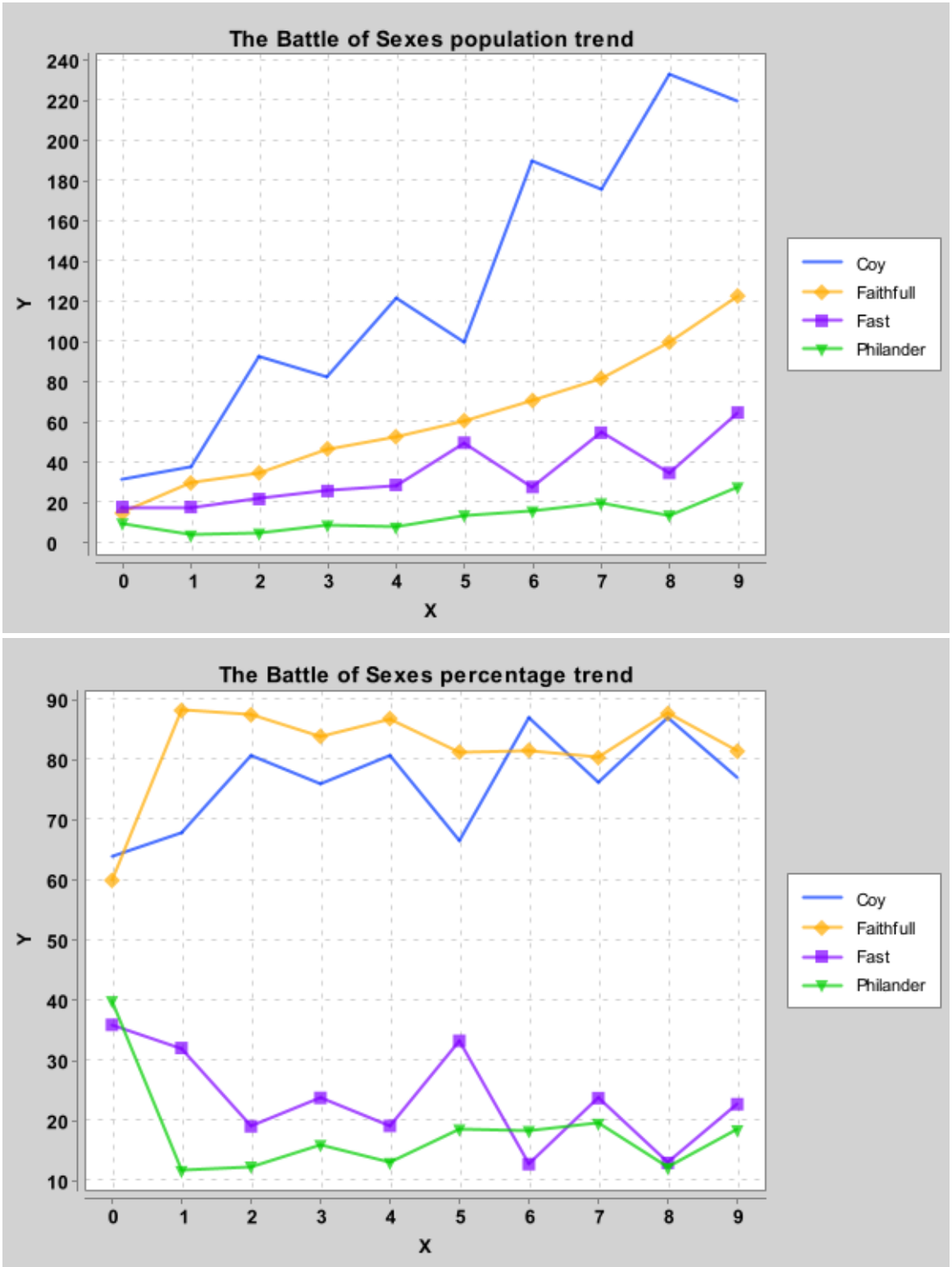

Figure 2: Philanders 10 - Faithful 10 - Fast 10 - Coy 10
a = 15 b = 20 c = 1

GENERATION	Philander	Faithful	Coy	Fast
0	10	15	32	18
1	4	30	38	18
2	5	35	93	22
3	9	47	83	26
4	8	53	122	29
5	14	61	100	50
6	16	71	190	28
7	20	82	176	55
8	14	100	233	35
9	28	123	220	65

Faithful	61% of men
Coy	77% of women

0 Faithful and 0 Coy

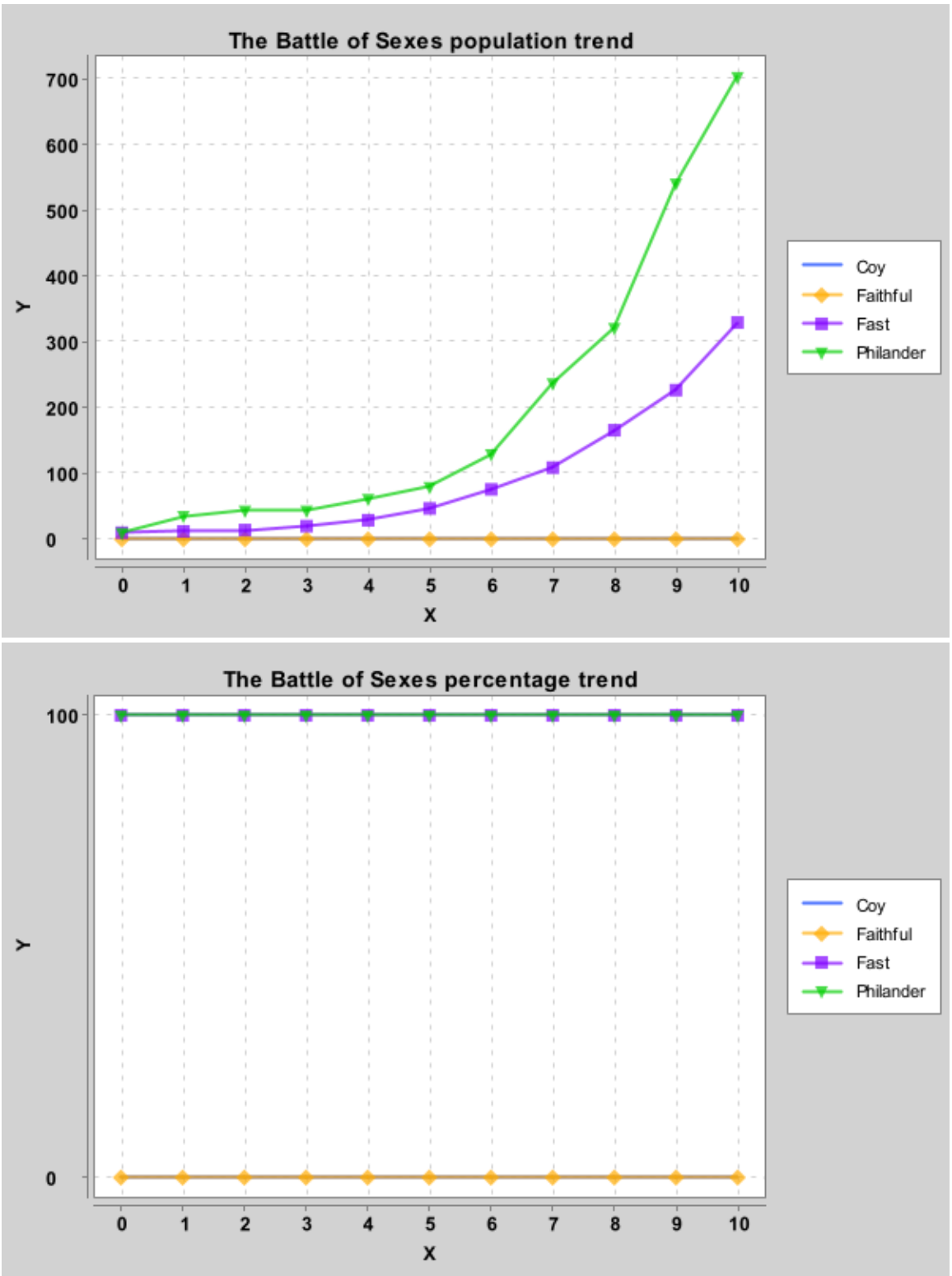

Figure 3: Philanders 10 - Faithful 0 - Fast 10 - Coy 0
a = 15 b = 20 c = 1

GENERATION	Philander	Faithful	Coy	Fast
0	10	0	0	10
1	36	0	0	12
2	44	0	0	14
3	44	0	0	20
4	62	0	0	29
5	80	0	0	47
6	130	0	0	76
7	238	0	0	109
8	322	0	0	166
9	542	0	0	227

Faithful	0% of men
Coy	0% of women

It is noticeable how in the first two cases studied the population always reaches stability, after already few generations. The designed rebels above extensively explained make it so that instable cases are impossible to achieve. Coy women are always around 5/6 of the female population and Faithful men are around 5/8 of the male population. The only case in which stability is not reached is the last case that analyzes the simulation with a starting population of zero coys and zero faithfuls. The unsuccess in reaching stability is due to the lack of these two types, the creation of which is not possible neither through the rebellion peculiarity of this simulation.

It can be then concluded that the evolutionary rules established in this simulation are so that stability is always reached.

5 Conclusions

Simulations are the re-creation of a real world process in a controlled environment. They involve creating laws and models to represent the world, and then running those models to observe the system's behavior, revealing the underlying mechanisms of such. More effectively, simulation may be used to predict a system's future behavior and figure out what you can do to alter it. Modeling a population and analyzing its behavior based on evolutionary rules was the goal of this project. With programming knowledge and the understanding of the conceptual characteristics of the Java programming language we have been able to successfully design a simulation that would allow us to fully understand the evolutionary behavior of a population.

This group project allowed us to gain communication and organization skills, which helped us in the understanding of each individual point of views and perfectionate our programming skills thanks to the knowledge of each of us. This project gave us the possibility to gain substantial improvement in our team work capabilities. We grew familiarity in the usage of important software that had a major role in the realization of the project, (such as GitHub.com, Latex, IntelliJ…), and with both working simultaneously both in presence and remote, and in assigning tasks to each one of us, to be completed individually. One of the most effective methods to learn about how the world works is to work in groups. The experiences gained through this project help us become better in the understanding of the mechanisms underlying the importance of team work.