The Battle Of Sexes

Benelli Alessandro

Bianchi Christian

Ciampricotti Francesca

Sannino Siria

Applied Computer Science and Artificial Intelligence

Professor Pietro Cenciarelli

Sapienza University

I. Introduction

- This simulation is based on Chapter 9 of The Selfish Gene, a 1976 book by Richard Dawkins. The chapter describes "the battle of the sexes", where a model is provided of a population, featuring two male types, the faithful (F) and the philanderers (P), and two female types, the coy (C) and the fast (S).

  In this documentation, we discuss the implementation and the logic on which the project is based.

II. Implementation

I. Class Flirt

- This class is a subclass of Thread, in which, through the method run, the payoff of each type is computed.

Field	Access Modifier and Type	Description
payoffFaithCoy	package private, int	it contains the value of the payoff of Faithful with Coy
payoffFaithCoy	package private, int	it contains the value of the payoff of Coy with Faithful
payoffFaithFast	package private, int	it contains the value of the payoff of Faithful with Fast
payoffFaithFast	package private, int	it contains the value of the payoff of Fast with Faithful
payoffFast	package private, int	it contains the value of the payoff of Fast with Philander
payoffPhil	package private, int	it contains the value of the payoff of Philander with Fast
girlfriend	package private Woman	it is a variable of type Woman
boyfriend	package private Man	it is a variable of type Man

Constructors	Description
Flirt(Man boyfriend, Woman girlfriend, Population population)	it updates the population's score

Methods	Access Modifier	Description
run()	public	it makes couples and increases type's point according to the payoff

II. Class Population

- The aim of this class is to create the starting population and the next generations.

Field	Access Modifier and Type	Description
deathAge	package private, int	it represents the age of death
countFast	package private, int	it contains the starting population of type Fast
countCoy	package private, int	it contains the starting population of type Coy
countPhil	package private, int	it contains the starting population of type Philander
countFaith	package private, int	it contains the starting population of type Faithful
pointFast	package private, int	it contains the score for the next generation of Fast
pointPhil	package private, int	it contains the score for the next generation of Philander
pointCoy	package private, int	it contains the score for the next generation of Coy
pointFaith	package private, int	it contains the score for the next generation of Faithful
men	package private ArrayList<Man>	an array of objects of type Man
women	package private ArrayList<Woman>	an array of objects of type Woman
a	package private final int	the evolutionary benefit for having a baby
b	package private final int	cost of parenting a child
c	package private final int	cost of courtship

Constructors	Description
Population(int starterP, int a, int b, int c, int deathAge)	it creates the population

Methods	Access Modifier and Type	Description
nextGeneration()	package private void	it creates the next Generation
addMan(boolean faith)	package private void	syncronized add method for ArrayList men
addWoman(boolean coy)	package private void	syncronized add method for ArrayList women
removePerson(Man man)	package private void	syncronized remove method
removePerson(Woman woman)	package private void	syncronized remove method
pickGirl()	package private Woman	syncronized get method for ArrayList women
growing()	package private void	it updates ages and it removes dead people
info()	package private void	it displays the current data for each type

III. Class Person

- Person is an abstract class and it gives the based methods for class Man and Woman.

Field	Access Modifier and Type	Description
age	package private int	it represents the age

Methods	Access Modifier and Type	Description
getAge()	package private void	syncronized get method of the age of one person
getAge(int age)	package private void	syncronized get method basing on a particular age

IV. Class Man

- Man is a subclass of Person.

Field	Access Modifier and Type	Description
faithful	package private boolean	it represents a faithful with a true type

Constructors	Access Modifier	Description
Man(boolean Faithful)	package private	constructor to instantiate a man

V. Class Woman

- Woman is a subclass of Person.

Field	Access Modifier and Type	Description
coy	package private boolean	it represents a coy with a true type

Constructors	Access Modifier	Description
Woma(boolean coy)	package private	constructor to instantiate a woman

VI. Class StartSimulation

- Through the method start() of this class, the simulation begins.

Methods	Access Modifier and Type	Description
start()	package private void	it calls the methods necessary for the life of types
main(String[] args)	public static void	it is the main method of the project

VII. Exceptions

- The exceptions used are InterruptedException, ReachedStabilityException, NoSuchSexException and TypeHasDiedException.

Exception	Description
NoSuchSexException	it is thrown when women's array is empty
InterruptedException	it is thrown to interrupt a thread waiting or sleeping
TypeHasDiedException	it is thrown when the counter of a type is zero at the end of a year
ReachedStabilityException	it is thrown when the stability is reached

III. Main Concepts

I. Payoffs

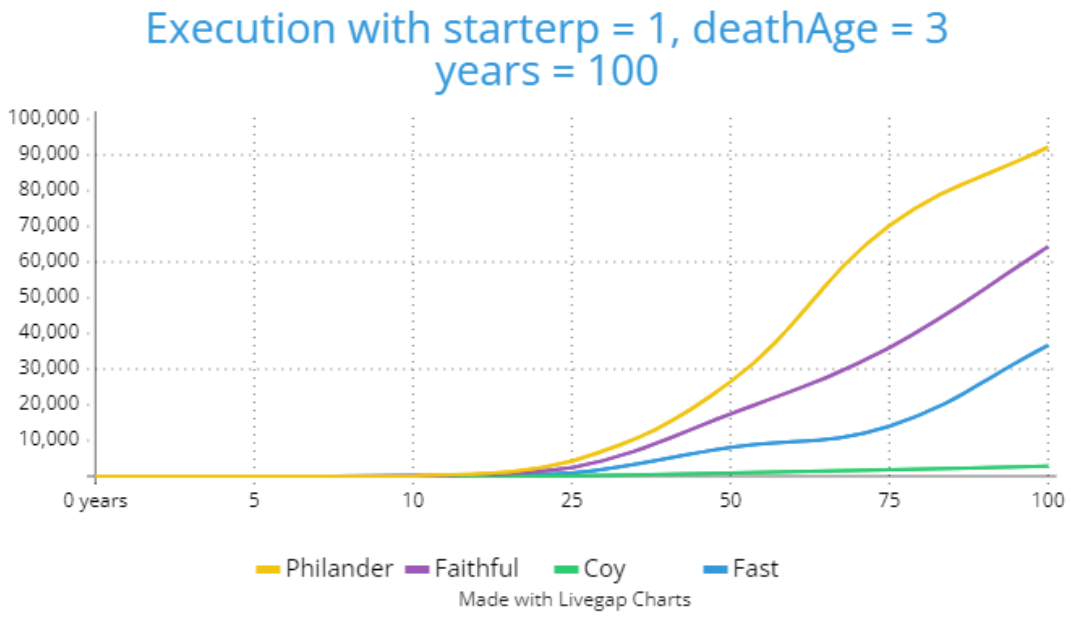
- The payoffs,according to the Richard Dawkins' table should have been eight (two for each couple)
- However, we noticed that the payoff for Faithful and Coy is equal (as well as for the couple composed by a Fast with a Faithful, so not to be redundant we decided to use only one variable for each payoff. In addition we do not include the payoff for the couple Coy and Philander, given that they are always equal to zero.

.

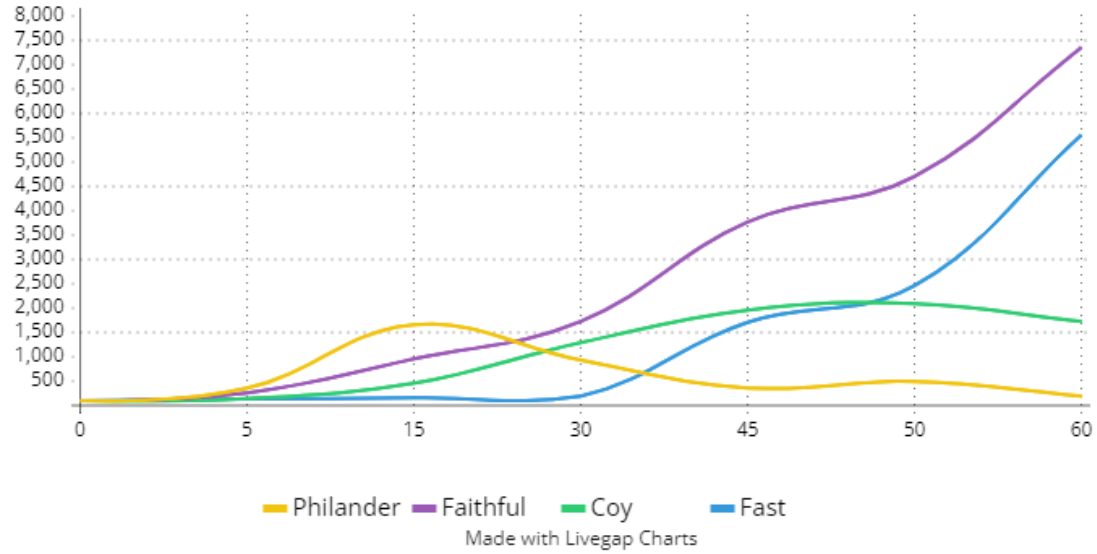
II. Stability...

At the beginning of the project, we focus our mind to the analytical part, asking about the concept of stablity. The stability can be reached when a value, we can call as the calculus convection wants, Δ, which represent the difference between the ratio between two different types of the same sex of the previous generation and the next generation.

$$\Delta \equiv \frac{1typeonesex}{2typeofonesex} = \frac{1typeofonesexnextgen}{2typeofonesexnextgen)} \quad (1)$$



Execution with starterp = 100, deathAge = 20, years = 60



Execution with starterp = 100, deathAge = 90, years = 250