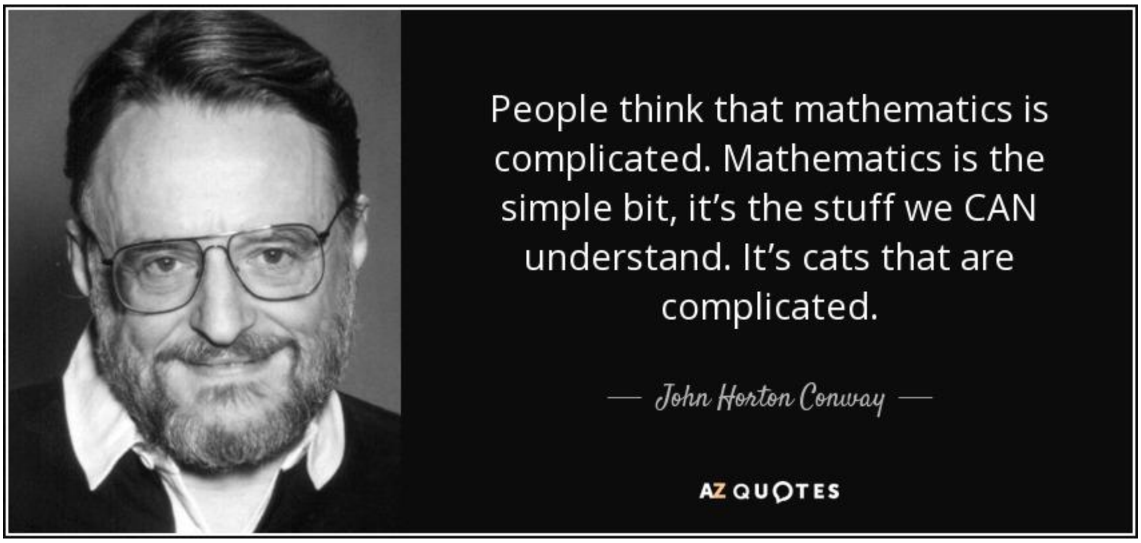
BBC Software Engineering Graduate Trainee Scheme

‘Game of life’

David Méndez Merino



People think that mathematics is complicated. Mathematics is the simple bit, it’s the stuff we CAN understand. It’s cats that are complicated.

— John Horton Conway —

06/02/2019

Table of Content

Introduction

This project aims to implement the famous game invented by John Horton Conway. To carry out this game, it has been used the environment **Eclipse IDE 2018-12.** The chosen language has been **Java** because **OOP** (Object Oriented Programming) has been considered the best option to develop the project. This choice is due to a range of advantages such as modular structure, easily adapted software and personal knowledge and practice in OOP projects.

Programming concepts

In this section, the bases of programming of the project will be explained without many details. For a more technical explanation, the code is carefully commented.

The project consists of 3 Java classes called "**Cell**", "**Board**" and "**GUI**".

The first one describes a **Cell** object which contains the required properties to be a single cell of the game. The second represents a **Board** object which has a matrix of **Cell** objects (among other features and variables). About the **GUI** class, it holds the main part of the code, including the logical and graphical part.

The libraries used to build the graphics of the game are the **AWT** (Abstract Windows Toolkit) and **Swing**.

The **Javadoc** has been made to generate the documentation of the classes, methods and functions of the entire project in HTML format. This documentation is in the folder *"Game_of_Life_DavidMéndez\doc".*

To understand in a more detailed way the structure of the program the following picture (figure 1) shows the class diagram of the project. Taking a closer look at this diagram we can appreciate the composition of the game as it was previously described.

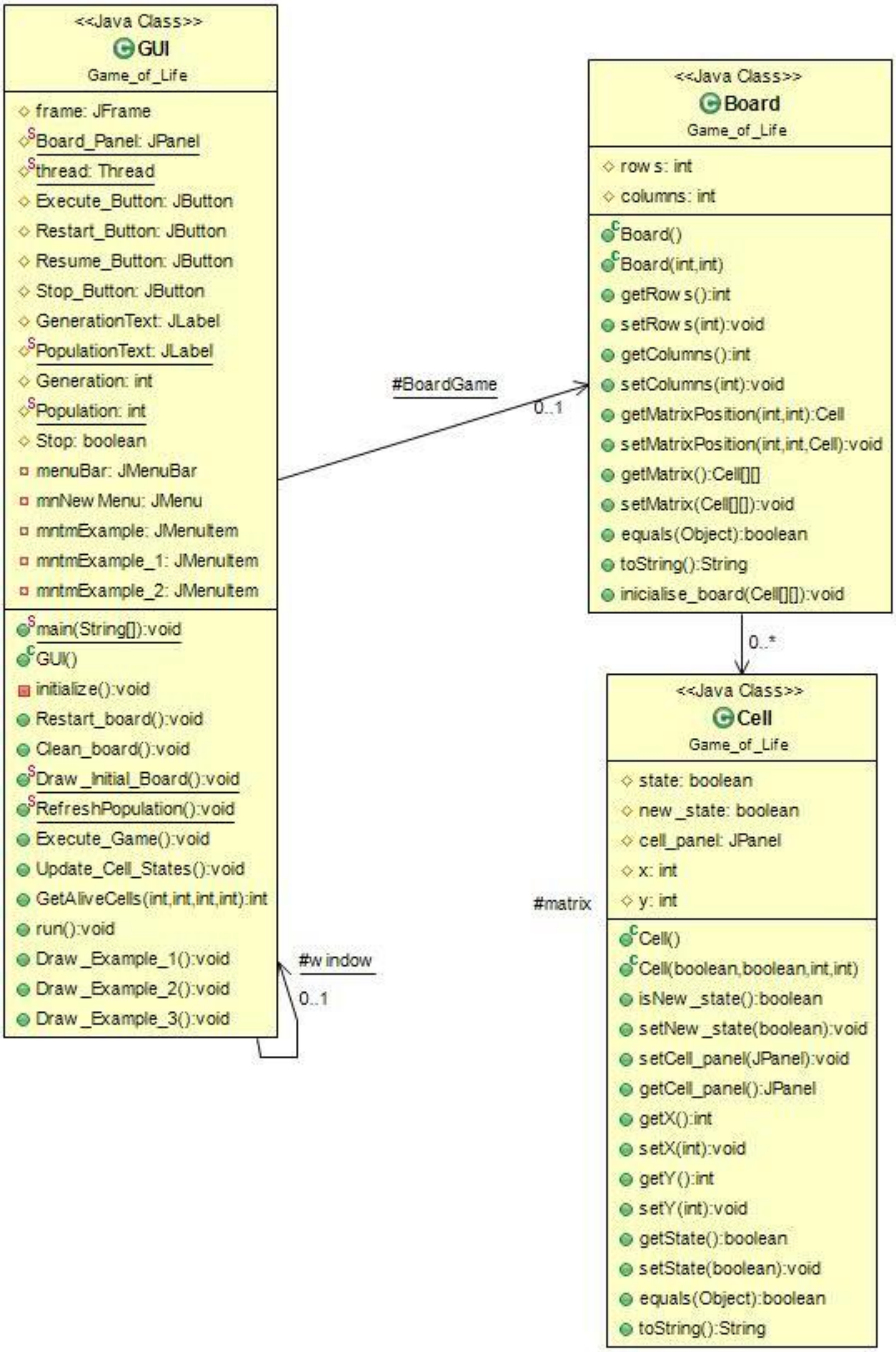


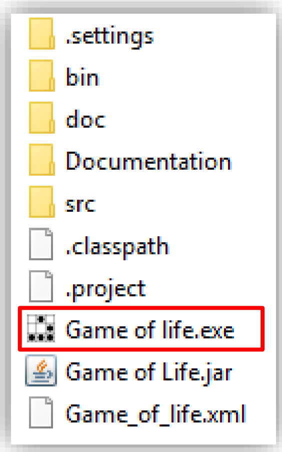
Figure 1. Class diagram of the project

User Manual

The user manual provides the information the user must know to make full use of the program. Guided by several images, this manual contains details about how to run the game as well as how to interact with the program interface.

How to run the program

To run the program there is an executable file (.exe) called "Game of life" (Figure 2). Double click on the file and the game will start.



.settings
bin
doc
Documentation
src
.classpath
.project
Game of life.exe
Game of Life.jar
Game_of_life.xml

Figure 2. Game of life.exe

Initial interface

When the program is run, the initial interface will look like figure 3. The game board is made up of 60 rows and 120 columns, with a total of 7.200 cells. In this step, the user is able to create the initial distribution of cells, modifying the state of the cells (from dead cell to alive cell and vice versa) by clicking on them. When the board distribution is finished, to execute the game the user will click on the button "**- - EXECUTE - -** ".

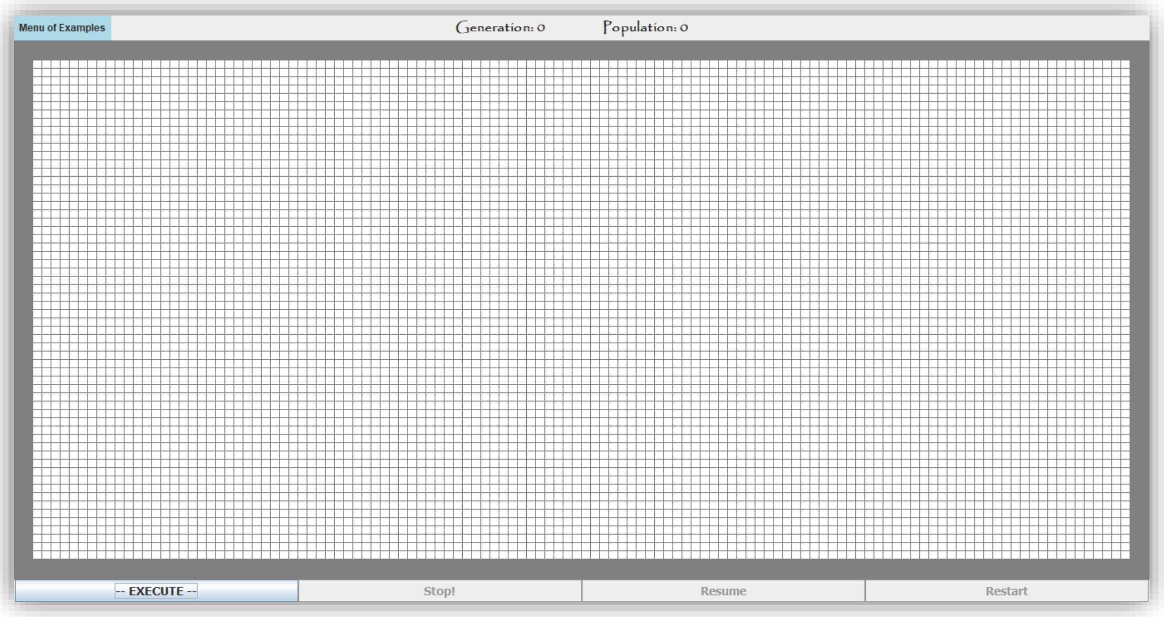


Figure 3. Initial interface

Top of the screen

On the top of the interface (figure 4), there are some components: a **menu** to choose distribution example, a label “**Generation**” to indicate the number of the current generation and another label “**Population**” to show the number of living cells in every moment.

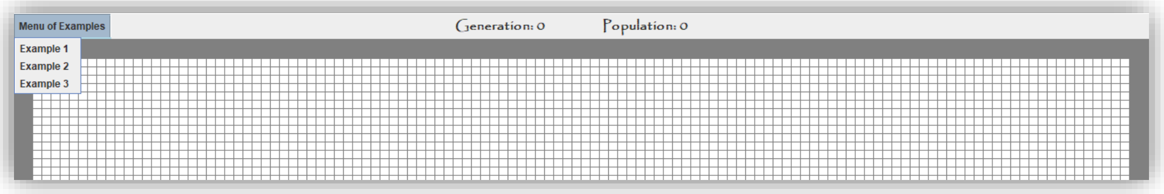


Figure 4. Top of the interface

Examples

The game provides **3 examples** of cell distribution that the user will find on the top-left menu. Figures from 5 to 7 displays these cell setups. The reason why the project presents these interesting examples is to show to the user the power of this game and to give some ideas in order to help the user to create new cell distributions.

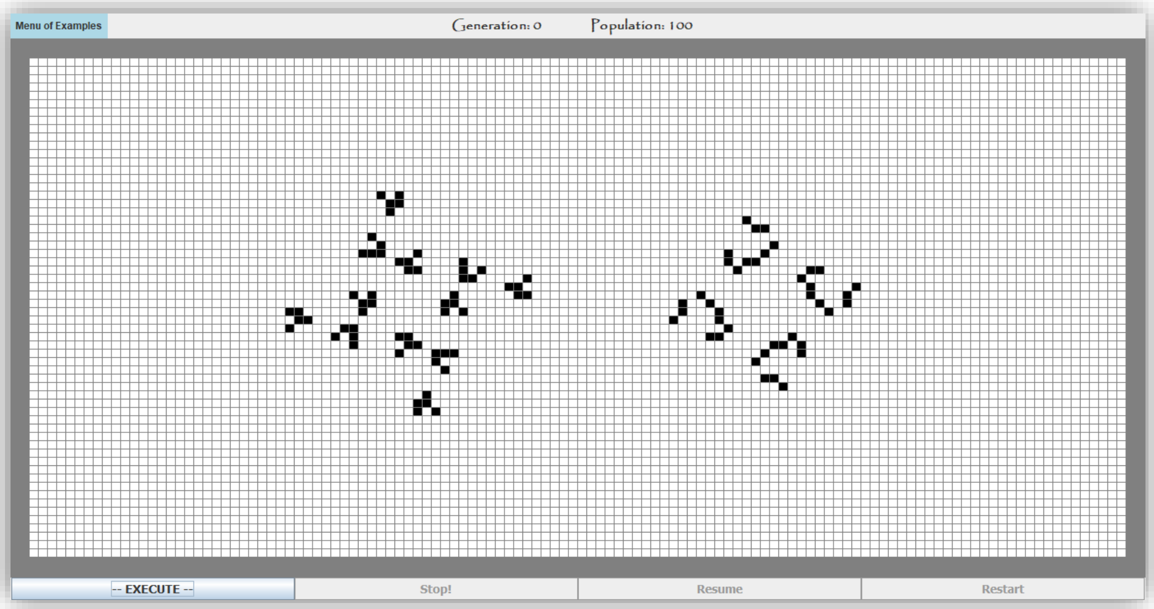


Figure 5. First example

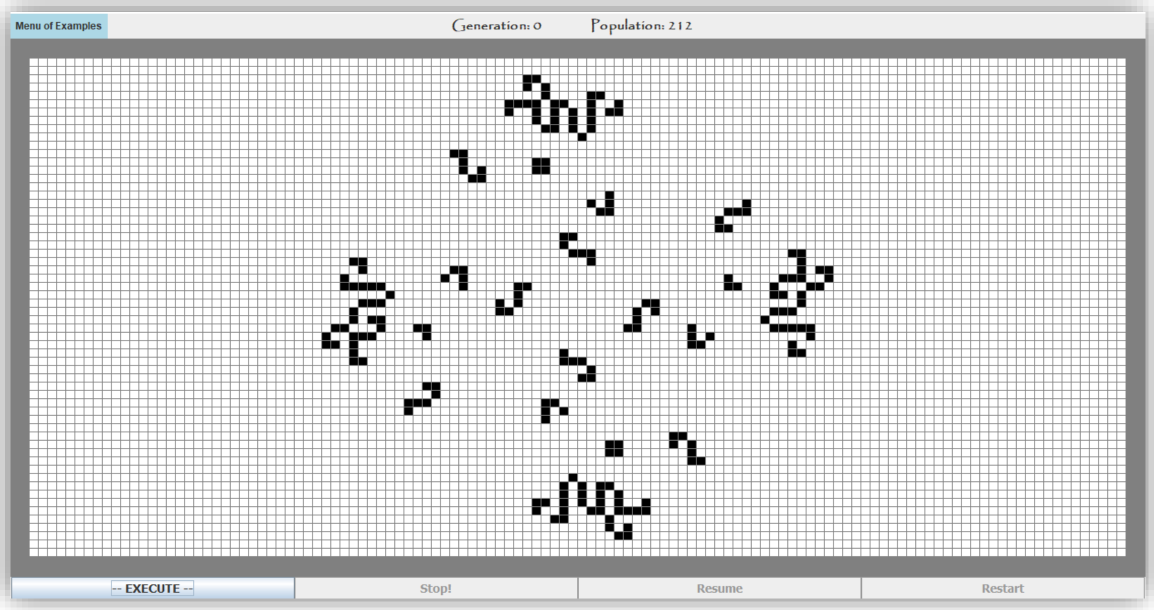
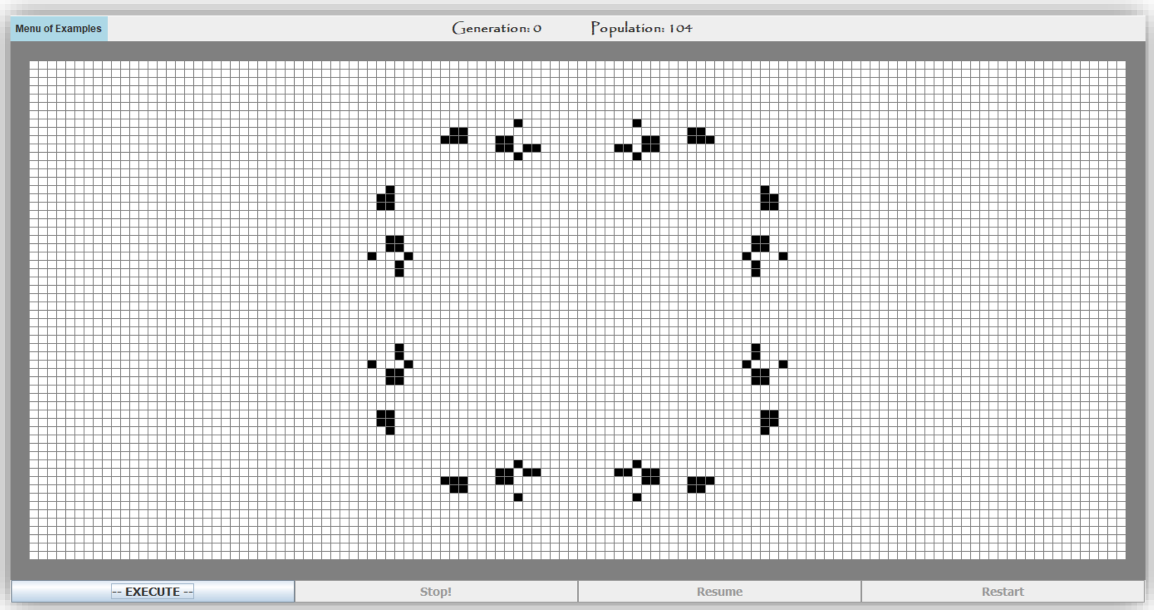


Figure 6. Second example



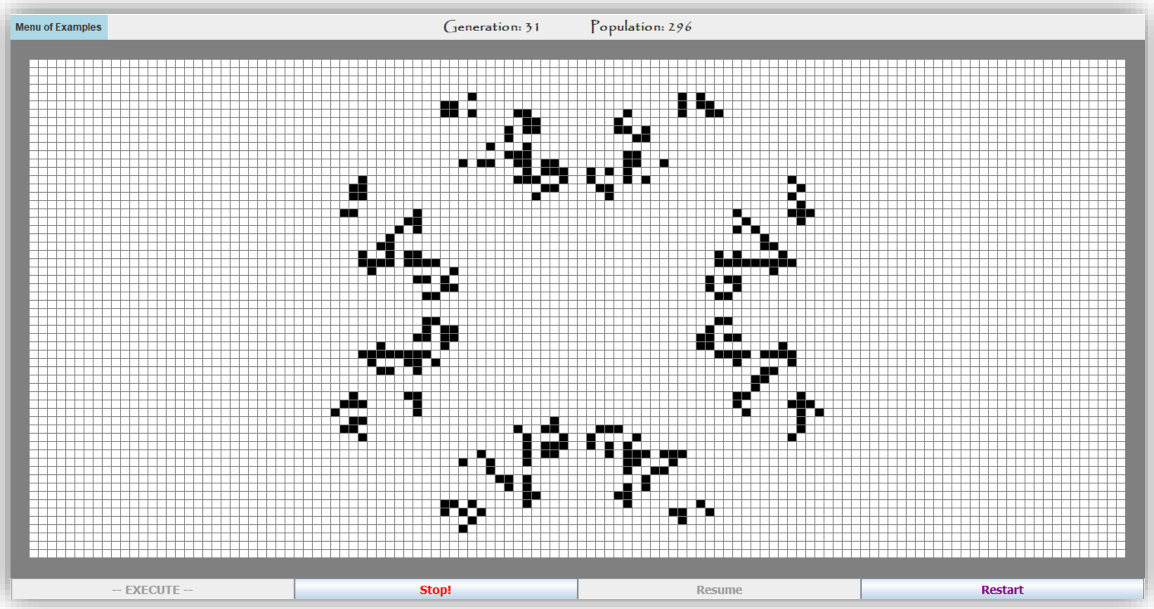
Generation: 0

Population: 104

Figure 7. Third example

Bottom of the screen

Once the game is running, the buttons from the bottom will be enabled (figure 8). These buttons are: **Stop** button, which allows to stop the execution; **Resume** button, to continue the execution with the current cell distribution and **Restart** button, which allows restarting the board, cleaning it to return to the initial board.



Generation: 31

Population: 296

Figure 8. Example 3 execution