

Game of Life

Coursework 3

Leila Flynn & Jay Shah | PPA | Febuary 2024

Contents

[Base Tasks 1](#_Toc160099278)

[Updating mycoplasma 1](#_Toc160099279)

[Two new life forms 1](#_Toc160099280)

[Colour changing - Rainbow 1](#_Toc160099281)

[Time Progression - Timecells 2](#_Toc160099282)

[Challenge Tasks 2](#_Toc160099283)

[Non-deterministic cells 2](#_Toc160099284)

[Symbiosis 3](#_Toc160099285)

[Disease 3](#_Toc160099286)

[Reset Button 3](#_Toc160099287)

# Base Tasks

## Updating mycoplasma

The Mycoplasma class was updated in the act() method though a series of if statements. If the cell is alive, if it has fewer than two or more than three alive neighbours, it dies. Otherwise, it lives on. If a dead Mycoplasma cell has exactly three alive neighbours, it will be alive in the next generation.

In addition, the Mycoplasma cells can develop a disease and pass it on to other cells. The way that developing the disease is determined is random, with each Mycoplasma cell having a 0.05% chance of becoming diseased every generation.

## Two new life forms

### Colour changing - Rainbow

In order to create a new life form, we created a new class called Rainbow which extends the class Cell. We initially copied the Mycoplasma class in order to get started making the code. We then defined the abstract method act() making the criteria for changing colours be if the number of neighbours is even it turns green and if odd it turns blue. We added a few more criteria, so if the number of neighbours is greater the three the cell dies.

In order to have the cells Mycoplasma and Rainbow coexist, we had to edit the Simulator class. We created a variable called RAINBOW\_ALIVE\_PROB, which will determine the likelihood of the cell initially spawning into the grid. In the populate method, we created an else if statement, checking if the probability is met and also creating and adding the cell to the grid. The initial colour of these cells is blue.

### Time Progression - Timecells

This cell exhibits different behaviours as time progresses and has a mutualistic relationship with the Nondeterministic cells (see the symbiosis section below). The time cells act according to four rules. Between generation zero and ten, the cell will be alive if it has more than one alive neighbour. Between generation ten and twenty-five, the cell will be alive if it has more than two alive neighbours. Between generation twenty-five and fifty, the cell will be alive if it has more than three alive neighbours. From generation fifty onwards, the cell will be alive if it has more than four neighbours. If it does not, there is a 25% chance that it will be alive. If none of these conditions are met, the cell will be dead in the next generation.

As with the Rainbow cells, adding the time cells required the Simulator class to be changed. We did this by setting a constant value for the probability of a particular cell being a time cell and then setting the colour of the cell to pink.

# Challenge Tasks

## Non-deterministic cells

This cell behaves in a non-deterministic manner and has a mutualistic relationship with the time cells (see the symbiosis section below). The non-deterministic cells act according to four rules. At the start of the act() method a random double value between 0 and 1 is generated. If this value is less than 0.2, the cell will be alive if it has more than two alive neighbours. If the value is between 0.2 and 0.5, there is a fifty-fifty chance of the cell being alive or dead. If the value is between 0.5 and 0.7, the cell will be alive if the current generation number is even, and it will be dead if the generation number is odd. If the value is between 0.7 and 1, the cell’s current state will be reversed i.e. if it is currently alive, it will be dead and if it is currently dead, it will be alive.

Similarly to the previous cells, adding the non-deterministic cells required the Simulator class to be changed. We did this by setting a constant value for the probability of a particular cell being a non-deterministic cell and then setting the colour of the cell to light grey.

## Symbiosis

The cells that act in a symbiotic manner are the time cells and the non-deterministic cells. They have a mutualistic relationship where they can grant each other immunity form disease if certain conditions are met. The TimeCell class has a method that returns the number of non-deterministic cells it has as neighbours. The Nondeterministic class has a method that returns the number of time cell it has as neighbours. In both of classes’ respective act methods, if the value returned from this method is greater than two and the cell is alive and not currently diseased, there is a 50% chance that the cell will be granted immunity for the current generation. This is done using the setImmune() method from the Cell class. The cell is then set back to not immune using the setNotImmune() method from the Cell class.

## Disease

## In order to make the cells diseased, we created a method in the Cell class called makeDiseased(). This method is called when an alive cell has a diseased cell as its neighbour, or if a Mycoplasma develops the disease. This method sets the cell colour to black, sets the variable diseased to true and sets the diseaseTimer variable to 3. Setting diseaseTimer to 3 means that the cell will be diseased for 3 generations before going back to normal. The diseaseTimer is decremented when a cell is diseased in the actDiseased() method which changes the behaviour of diseased cells so that they live on if they have two or more alive neighbours and die otherwise.

## When the diseaseTimer reaches 0, the cell becomes immune to disease for 5 generations. This is done by setting the variable immune to true and the immuneTimer variable to 5 and this is decremented when the cell is immune and acting normally. Once the immuneTimer reaches 0, the cell is no longer immune so can be diseased again.

## The way that the disease enters the game is only through the Mycoplasma cells. Each Mycoplasma cell has a fixed probability (0.05%) of becoming diseased every generation. The Mycoplasma cells do not pass the disease onto each other in the same way that the other cells do. The other cells call the diseaseChecks() method which, if it returns false, it makes them infected if any of their neighbours have the disease and they are alive and not immune. If it returns true, the cell acts like normal.

## Reset Button

For the self-chosen challenge task, we chose to implement a button using Java FX. Firstly, we added the action event and the button import statements. Then, we created a button with a responding label. In order to arrange the field, another border pane was added, outsidePane. We placed the border pane which included the text at the bottom (population, etc) into the outside plane in the centre section. Then added the button to the bottom of the outsidePane (borderpane). In order to implement the button, through a lambda statement the buttonClick() method was called, which called the reset() method.

The placement of the button when the application is run was too low on the stage, so we changed the fieldCanvas in order to make the button more visible to users.