Game Of Life

PPA COURSEWORK 3

Iffat Siddick: K220311129

Ekaterina Hunter: K22019809

Iffat Siddick: k220311129
Ekaterina Hunter: k22019809

Life forms:

Mycoplasma rules:

- If the cell is alive is alive and has more than 3 living neighbours, it will be dead in the next generation.
- If it is alive and has more than 1 living neighbour, it will live on for the next generation.
- If it is alive and has less than 2 living neighbours, it will die in the next generation.
- If the cell is dead and has 3 living neighbours, it will be alive in the next generation.
- For all other conditions the cell will be dead in the next generation
- Mycoplasma's original colour is very light orange.

Salmonella rules:

- Whether or not the cell is alive or dead, if it has more than 6 living neighbours it will be dead in the next generation.
- Whether or not the cell is alive or dead, if it has more than 4 and less than 7 living neighbours it will be alive in the next generation.
- Whether or not the cell is alive or dead, if it has more than 2 and less than 5 living neighbours it will be dead in the next generation.
- Whether or not the cell is alive or dead, if it has less than or equal to 2 living neighbours it will be alive in the next generation.
- If the cell is dead but has 3 living neighbours, it will be alive in the next generation.
- If the cell is dead but has at least one alive neighbour of Listeria (irrespective of whether it's diseased or not) it will be alive in the generation and its colour will be set to its original colour.
- If the cell is dead and does not meet either or the 2 previous conditions, it will be dead in the next generation.

Colour rules:

- It inherits the abstract changeColour() method from the abstract class CellChangeColour
- This ensures all Cells which are subclasses of CellChangeColour will have the changeColour method and this makes the program more easily extendible if further lifeforms were to be added in future.
- Salmonella's original colour is very light red.
- If the cell has been alive for more than 10 consecutive generations, the cell's colour will be set to red.
- If the cell has been alive for more than 20 consecutive generations, the cell's colour will be set to dark red.
- If the cell has been alive for more than 30 consecutive generations, the cell's colour will be set to very dark red.

Listeria rules:

- If the cell is alive and has either 8 or 0 living neighbours, then it will be dead in the next generation.

- If the cell is dead and has more than or equal to 3 and less than or equal to 5 living neighbours, it will be alive in the next generation.
- If the cell is dead but has at least one alive Salmonella neighbour (irrespective of whether it's diseased or not) it will be alive in the generation and its colour will be set to its original colour very light blue.
- If the cell is dead and does not meet either or the 2 previous conditions, it will be dead in the next generation.

Colour change rules:

- It inherits the abstract changeColour() method from the abstract class CellChangeColour.
- If the cell is alive and has more than 4 living neighbours, then its colour will change to very dark blue.
- If the cell is alive and has 3 or 4 living neighbours, then it's colour will change to blue.

Cholera rules:

- If the cell is alive and its lifespan is a multiple of 10 and the random number generated is less than 0.2 the cell will become diseased
- If the cell is alive and its lifespan is a multiple of 10 and the random number generated is less than 0.3 the cell will be dead in the next generation
- If the cell is alive and doesn't meet either of the 2 previous conditions, it will be alive in the next generation.

- If the cell is dead and its lifespan is a multiple of 3 and the random number generated is less than 0.4 the cell will be alive in the next generation
- If the cell is dead and its lifespan is a multiple of 7 and the random number generated is less than 0.6 the cell will be diseased in the next generation
- If the cell is dead and doesn't meet either of the 2 previous dead conditions, it will be alive in the next generation.

Challenge tasks:

Disease implementation:

- We decided to implement this in the class Cell as a feature each cell could have.
- For this we created the boolean instance field isDiseased which indicated if a cell was diseased or not and a static final field that held the probability of the cell becoming diseased
- We created methods for populating the field with diseased cells once it was created and spreading the disease in each generation.
- The rules that govern whether a cell is dying of disease or becoming cured are in in the actDiseased(), cured() and dies() method in class Cell.
- If a diseased cell survives 20 generations, it will die.
- An abstract method is used to set the formerly diseased cell back to its original colour so what if it were to be brought back to life, it would no longer be diseased.

Iffat Siddick: k220311129
Ekaterina Hunter: k22019809

Nondeterministic cell:

- For this we created a new subclass of Cell, Cholera, which would implement the non-deterministic rules.
- We decided for each rule a new random double would be generated and compared to our chosen probability to decide which rule would be executed.
- This would keep the double being compared different for each rule which in turn makes the implementation of the rules more random than using the same generated probability for each comparison.
- Each rule is dependent on how many generations the cell had been alive for.

Mutualistic symbiotic relationship:

- We chose Salmonella and Listeria to be in a mutualistic relationship.
- The relationship states that when one type of cell is dead, if it had at least 1 living neighbour of the other type, it would become alive in the next generation.
- We decided that whether the neighbour was diseased wouldn't matter as the main point was that the two types of cells supported each other by increasing each other's population.

Pause button:

- A pause button is located on the left-hand side of the pop-up window.
- When pressed, it temporarily prevents the simulation from continuing until the user presses the button again and the simulation will continue where it left off.
- This allows users to study individual relationships in greater detail.
- Implementation is in the SimulatorView alongside the rest of the gui code.
- SimulatorView calls the paused() method which is responsible for changing the pausedSim variable. PausedSim holds the Boolean value to indicate if the simulation is currently paused.
- PausedSim is used in the simulate() method to determine whether the next generation should run using an if else statement.