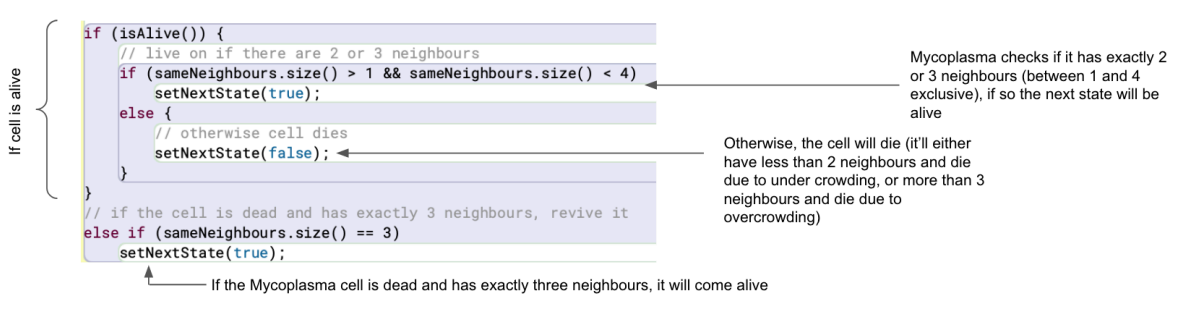
4CCS1PPA Programming Practice and Applications

Coursework 3: Life

Ishab Ahmed (K22009721) and Harshraj Patel (K22018200)

Base Tasks Completed (2/2)

1. The base code comes with a simple life form, the Mycoplasma. Your first task is to make it more interesting. Modify its rule set so that [it follows Game of Life rules].



2. Using the Mycoplasma as an example, develop at least two new forms of life. Their rule set should be different from the Mycoplasma.

Two new types of cells have been created: Isseria and Helicobacter. Mycoplasma's default colour is orange, Isseria's default colour is magenta and Helicobacter's default colour is cyan.

Helicobacter follows these rules:

- If the cell has exactly one live neighbours, it will die
- If the cell has exactly three live neighbours, it will live on to the next generation
- If the cell has more than three live neighbours, it will die due to overcrowding
- Lastly, any dead cell will come alive if it has exactly three live neighbours

Isseria follows these rules:

- If the cell has exactly two live neighbours, it has a chance of living on to the next generation
- If the cell has exactly four live neighbours, it has a chance of living on to the next generation
- Otherwise, it will die

– At least 1 life form should incorporate colour in its rule set, i.e., its colour may change while alive
– At least 1 life form should exhibit different behaviours as time progresses

The colour of the Helicobacter cell is dependent on the current generation. A method `darkenHeliColour(generation)` is used to decrease the RGB value of the Helicobacter's colour, based on the current generation. This gives a slow gradient of Helicobacter cells becoming darker and darker as the simulation goes on. This darkened colour is used to signify a change in behaviour - the Helicobacter's increased resistance to infected cells. As the colour darkens, the probability that a Helicobacter cell gets infected when surrounded by infected cells decreases, simulating increasing immunity to infection as time progresses.

Challenge Tasks Completed (4/4)

1. Non-deterministic cells

The rules that the Isseria cells follow are triggered based on a probability. Given the rule "If the cell has exactly two live neighbours, it has a chance of living on to the next generation", there is a 60% chance that this rule will execute. Furthermore, given the rule "If the cell has exactly four live neighbours, it has a chance of living on to the next generation", there is a 50% chance that this rule will execute. Non-determinism is implemented by using the `Random.getNextDouble()` method where an if-statement ensures that the next line of code is changed only if below the desired decimal value (e.g. `Random.getNextDouble() < 0.6` for the former rule).

Non-determinism is also present in the symbiosis and disease challenge tasks. They are implemented in the same way as described above and are used to provide a probability of changing to a new cell, for the disease to spread, etc. Whenever non-determinism is implemented and implicitly mentioned in the rest of the report, *(non-determinism)* will be present next to the explanation to make it more explicit.

2. Symbiosis

There is a parasitic relationship between Helicobacter and the other two types of cells, Isseria and Mycoplasma, where Helicobacter benefits, and all others are harmed.

If an Isseria or Mycoplasma cell has between one and three neighbours that are Helicobacter, there is a probability *(non-determinism)* that the cell will change its colour to the Helicobacter colour and thus become Helicobacter cell (no object replacement). This state change to becoming a Helicobacter cell is solely determined by the number of Helicobacter neighbours, which causes it to change its colour to the Helicobacter colour.

There is also a mutualistic relationship between Mycoplasma and Isseria cells.

If a dead Mycoplasma cell has *at least* one living Mycoplasma neighbour, and *at least* one living Isseria neighbour, the dead cell will come alive in the next generation as an Isseria cell. This simulates the "breeding" of neighbouring Mycoplasma and Isseria cells. This is implemented by getting neighbours of the current cell. If there is at least one Mycoplasma neighbour and at least one Isseria neighbour, the cell being processed changes its colour and thereby its state (no object replacement). This benefits Mycoplasma as its dead cells can come alive, and it benefits Isseria as the dead cell comes alive as an Isseria cell.

However, there is a chance *(non-determinism)* that when the two cells "breed", the new cell comes alive, but as an infected. This simulates a sexually transmitted disease between the cells. This is harmful to all life forms as the infected cell can infect any cell.

3. Disease

As mentioned before, when a Mycoplasma and Isseria cell are the neighbours of a dead Mycoplasma cell, there is a chance that the dead Mycoplasma changes to an infected state. Once the cell has an infected state, in addition to turning red, its behaviour is changed by no longer being able to breed, and for other cells to be infected if it has a certain number of infected neighbours. This is achieved by the cell getting all of its living neighbours which are infected. If it has at least one infected neighbour, there is a chance *(non-determinism)* that cell's state also becomes infected. This simulates a highly contagious infection, where only one neighbour needs to be infected. Dead cells can also have a chance *(non-determinism)* to come alive as an infected cell if three of its neighbours are infected.

Infected Mycoplasma cells are made to be even more contagious as an extra rule is added for infected Mycoplasma cells. Mycoplasma cells live on to the next generation if they have exactly two or three neighbours, as described in the rules mentioned before. However, if they are infected, they have a chance *(non-determinism)* to live on to the next generation no matter how many neighbours they have. This probability of living on is dependent on the current generation (more infectious at smaller generations when the infection is new, and less infectious as time progresses).

4. Simulation Controls

To the GUI, three components have been added:

1. The ability to change playback speed

A slider at the bottom of the GUI has been added to change how fast the simulation is ran. This is implemented by using the `delay()` method. The slider gives a value between 0 and 100. This is converted to a percentage and multiplied by 300 ms. For example, if the delay slider is set all the way to the left, there will be a 300ms delay, but if the delay slider is set all the way to the right, there will be no delay. All slider position in between are a percentage of 300ms.

2. Toggle simulation button

This button toggles whether the simulation is running or not. The text on the button changes depending on the state of the simulation. If the simulation has not been started yet, the text is set to "Start". If the simulation is running, the text is set to "Pause". If the simulation has been paused, the text is set to "Resume". The changing of the state of the simulation is implemented using a `paused` field. If the paused field is false, the simulation runs as usual. However, if paused is true, the `simOneGeneration()` method which progresses the simulation is not called in the `simulate()` method.

3. Reset simulation button

This button resets both the simulation and the simulation controls pane. To reset the simulation, the generation is set to 0, all cells are removed from the field, and the field is repopulated. The simulations control pane is reset by changin the text of the toggle button to "Start" and resetting the value of the playback speed slider to 50%.