Game of Life

I - Description of the life forms

- Mycoplasma description : (colors = light blue, dark blue)

The Mycoplasma cells that are alive and have 2 or 3 living neighbours remain alive, whilst the others die. Cells that were deemed dead, but have exactly 3 neighbours come back to life. These cells can be infected by a disease with an infection probability of 0.08. If the cell is infected, it changes color to black and dies if its number of living neighbours is fewer than 4. Finally, these cells exhibit different behaviors during the day and at night. During the day, they behave as previously described, and at night their color becomes darker and they show no activity.

```
public void act() {
    List<Cell> neighbours = getField().getLivingNeighbours(getLocation());
    setNextState(false);

    double disease = rand.nextDouble();

    if (Simulator.isDay()) {
        if (isAlive() && !isInfected()) {
            setColor(Color.LIGHTBLUE);
            if (disease <= PROB_DISEASE) {
               setNextState(true);
               setInfected();
               setNextInfected(true);
               setColor(Color.BLACK);
            } else if (neighbours.size() < 2 || neighbours.size() > 3) {
               setNextState(false);
            } else {
               setNextState(true);
            }
        } else if (isInfected()) {
            if (neighbours.size() < 4) {
               setNextState(false);
            } else {
               setNextState(true);
            }
        } else if (!isAlive()){
            if (neighbours.size() == 3) {
               setNextState(true);
            }
        }
    } else {
        if (isAlive() && !isInfected()) {
            setNextState(true);
            setColor(Color.DARKBLUE);
        } else if (isInfected()) {
            setNextState(true);
        }
    }
}
```

- Clostridium description : (colors = magenta, red, orange)

```
public void act() {
    List<Cell> neighbours = getField().getLivingNeighbours(getLocation());
    setNextState(false);

    double disease = rand.nextDouble();

    if (isAlive() && !isInfected()) {
        if (disease <= PROB_DISEASE) {
           setNextState(true);
           setInfected();
           setNextInfected(true);
           setColor(Color.BLACK);
        } else if (neighbours.size() == 3) {
           setNextState(true);
           setColor(Color.RED);
        } else if (neighbours.size() == 1) {
           setNextState(true);
        }
    } else if (isInfected()) {
        if (neighbours.size() < 4) {
           setNextState(false);
        } else {
           setNextState(true);
        }
    } else if (!isAlive()) {
        if (neighbours.size() == 2) {
           setNextState(true);
           setColor(Color.ORANGE);
        }
    }
}
```

Clostridium cells that are alive remain alive unless they have exactly 1 neighbour, which results in the death of the cell. However, if the cell has exactly 3 neighbours, it turns red and remains alive. On the other hand, if Clostridium cells are dead, but have exactly 2 living neighbours it revives and turns orange. These cells can also be infected by a disease with an infection probability of 0.08. If the cell is infected, it changes color to black and dies if its number of living neighbours is fewer than 4.

- Pseudomonas description : (colors = purple, yellow)

Pseudomonas cells that are alive show no activity until generation 15. After this, only the cells that are already alive and have more than 2 living neighbours remain alive. Moreover, after 25 generations, all living Pseudomonas cells change color and become yellow. Finally, these cells can only be infected if one (or more) of their neighbors is an infected cell.

```
public void act() {
    List<Cell> neighbours = getField().getLivingNeighbours(getLocation());
    setNextState(false);

    gen++ ;
    boolean isNeighbourDisease = false;

    if (isAlive() && !isInfected()) {
        for (Cell neighbour : neighbours) {
            if (neighbour.isInfected()) {
                isNeighbourDisease = true;
                break;
            }
        }
        if (isNeighbourDisease) {
            setNextState(true);
            setInfected();
            setNextInfected(true);
            setColor(Color.BLACK);
        } else if (gen <= 15) {
            setNextState(true);
        } else if (gen > 15 && gen <= 25) {
            if (neighbours.size() > 2) {
                setNextState(true);
            }
        } else {
            setNextState(true);
            setColor(Color.YELLOW);
        }
    }else if (isInfected()) {
        if (neighbours.size() < 4) {
            setNextState(false);
        } else {
            setNextState(true);
        }
    }
}
```

- Bacillus description : (colors = brown, grey, pink)

```
public void act() {
    List<Cell> neighbours = getField().getLivingNeighbours(getLocation());
    setNextState(false);

    double prob = rand.nextDouble();
    boolean isNeighbourDisease = false;

    if (isAlive() && !isInfected()) {
        for (Cell neighbour : neighbours) {
            if (neighbour.isInfected()) {
                isNeighbourDisease = true;
                break;
            }
        }
        if (isNeighbourDisease) {
            setNextState(true);
            setInfected();
            setNextInfected(true);
            setColor(Color.BLACK);
        } else {
            if (prob <= R1) {
                setNextState(true);
            } else if (prob <= R2) {
                setNextState(true);
                setColor(Color.GREY);
            } else if (prob <= R3) {
                setNextState(true);
                setColor(Color.PINK);
            } else {
                setNextState(true);
            }
        }
    }else if (isInfected()) {
        if (neighbours.size() < 4) {
            setNextState(false);
        } else {
            setNextState(true);
        }
    }
}
```

Bacillus cells remain alive in subsequent generations, but they follow three rules based on different probabilities. The first rule has a probability of 0.3 and the cell remains the same color. The second rule has a probability of 0.5 and changes the cell's color to grey. The third rule has a probability of 0.8 and changes the cell's color to pink. Finally, these cells can also only be infected if one (or more) of their neighbors is an infected cell.

- Staph description : (colors = light green, dark green)

Staph cells that are alive remain alive throughout the generations. However, the cells that were deemed dead, but have exactly 6 neighbours come back to life. Finally, in the same way as Mycoplasma cells, Staph cells exhibit different behaviors during the day and at night. During the day, they behave as previously described, and at night their color becomes darker and they show no activity.

```
public void act() {
    List<Cell> neighbours = getField().getLivingNeighbours(getLocation());
    setNextState(false);

    if (Simulator.isDay()) {
        if (isAlive()) {
            setNextState(true);
            setColor(Color.LIGHTGREEN);
        } else if (neighbours.size() == 6) {
            setNextState(true);
            setColor(Color.LIGHTGREEN);
        }
    } else {
        if (isAlive()) {
            setNextState(true);
            setColor(Color.DARKGREEN);
        }
    }
}
```

- Ecoli description : (color = bisque)

```
public void act() {
    List<Cell> neighbours = getField().getLivingNeighbours(getLocation());
    boolean isStaphNeighbour = false;

    for (Cell neighbour : neighbours) {
        if (neighbour instanceof Staph) {
            isStaphNeighbour = true;
            break;
        }
    }

    if (isStaphNeighbour) {
        setNextState(false);
    } else {
        setNextState(true);
    }
}
```

Ecoli cells remain alive through generations unless one of their neighbours cells is a Staph cell.

II - Completion of challenge tasks

- Challenge task 1 : 'Non-deterministic cells' :

For this first challenge task, we have created a new form of life, Bacillus cells. These cells have the ability to react to their environment in various ways based on predefined probabilities, thus exhibiting non-deterministic behavior. Within the method act() of this class, we employ a Random object to generate a random number that dictates the cell's behavior for each generation. Three probability thresholds (R1, R2 and R3) are defined to represent the different rules that the cell might take. The first rule has a probability of 0.3 and the cell remains the same color. The second rule has a probability of 0.5 and changes the cell's color to grey. The third rule has a probability of 0.8 and changes the cell's color to pink.

```
private static final double R1 = 0.3;
private static final double R2 = 0.5;
private static final double R3 = 0.8;
private Random rand = new Random();
```

```
if (prob <= R1) {
    setNextState(true);
} else if (prob <= R2) {
    setNextState(true);
    setColor(Color.GREY);
} else if (prob <= R3) {
    setNextState(true);
    setColor(Color.PINK);
} else {
    setNextState(true);
}
```

- Challenge task 2 : 'Symbiosis - parasitic relationship' :

To complete this second task, we have chosen the parasitic relationship and created two forms of life, Staph and Ecoli cells. In our model, Ecoli cells represent the host organisms, while Staph cells embody the parasites. The act() method in the Ecoli class begins by scanning the immediate neighbors of an Ecoli cell to detect the presence of any Staph cells. If a Staph cell is found among the neighbors, the method concludes that the Ecoli cell is under parasitic attack. Then, the Ecoli cell is marked to not survive to the next generation, symbolizing the harmful impact of the Staph parasite.

```
boolean isStaphNeighbour = false;

for (Cell neighbour : neighbours) {
    if (neighbour instanceof Staph) {
        isStaphNeighbour = true;
        break;
    }
}

if (isStaphNeighbour) {
    setNextState(false);
} else {
    setNextState(true);
}
```

- Challenge task 3 : 'Disease' :

```
public void setNextInfected(boolean value) {
    nextInfected = value;
}
```

```
private static final double PROB_DISEASE = 0.08;
private Random rand = new Random();
```

```
if (disease <= PROB_DISEASE) {
    setNextState(true);
    setInfected();
    setNextInfected(true);
    setColor(Color.BLACK);
```

```
for (Cell neighbour : neighbours) {
    if (neighbour.isInfected()) {
        isNeighbourDisease = true;
        break;
    }
}
if (isNeighbourDisease) {
    setNextState(true);
    setInfected();
    setNextInfected(true);
    setColor(Color.BLACK);
```

For this task, we have chosen two forms of life that can be afflicted with the disease (Mycoplasma and Clostridium) and two forms of life that can be contaminated by it (Pseudomonas and Bacillus). On the one hand, for the cells that can be infected by this disease, we proceeded in the same way for both. We first created several methods (boolean…) in the Cell class to be able to mark a cell as infected or not, or even mark the cell of the next generation as infected or not. Then, as with the first challenge task, we used the random number generator, as well as a contamination probability so that the cell would be contaminated by the disease based on a probability. On the other hand, for the two cells that can be contaminated, we used the same functionalities as in challenge task 2 to detect if one of the neighboring cells is an infected cell. Finally, all contaminated cells have a lower chance of survival ,indeed we added the condition that these cells die if their number of neighbors is less than 4.

- Challenge task 4 : 'Day and Night' :

To complete this last task, that we have invented, we have chosen two forms of life (Mycoplasma and Staph) that exhibit different behaviors during the day and at night. During the day, they behave normally, and at night their color becomes darker and they show no activity. We have implemented in the Simulator class static variables and a condition in the simOneGeneration() method to create a day and night cycle. The static variables include a boolean isDay to indicate the current phase, a dayNightCycle defining the length of a complete day-night cycle in terms of generations, and a counter dayNightCounter to track the progression through the current cycle. With each generation in the simOneGeneration() method, dayNightCounter increments, and upon reaching the threshold defined by dayNightCycle, the day-night phase toggles, resetting the counter for the next cycle. Finally, for the cells to change behavior according to the current phase, we have added an isDay condition in the act() method of the respective cells.

```
private static boolean isDay = true;
private static final int dayNightCycle = 5;
private static int dayNightCounter = 0;
```

```
dayNightCounter++;

if (dayNightCounter >= dayNightCycle) {
    isDay = !isDay;
    dayNightCounter = 0;
}
```