Object-oriented Modelling and Programming in Engineering

Homework 2

1. Problem

We want to simulate a simple population. This population consists of cellular automats, with each cell having two states: alive or dead. Every cell has 8 neighbors (Moore-Neighborhood). The state of the cell depends on it's current state and the number of living and dead neighbors. The goal is, to simulate the evoluion of an initial population. The subsequent generation is a result of the prior one. Hence, care for not mixing the states between the generations. Develop a software, which models the evolution of the population and visualizes the result as a grid with colored squares.

2. Evolution rules

I. Dead automat
   a. Cell reincarnates if it has got exact 3 living neighbors

II. Living automat
   a. Cell dies if it has got less than 2 living neighbors (loneliness)
   b. Cell stays alive if it has got 2 or 3 living neighbors
   c. Cell dies if it has got more than 3 living neighbors (overpopulation)

3. Example

As an example we will take the pattern shown in figure 1.

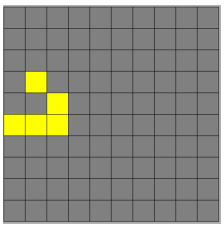
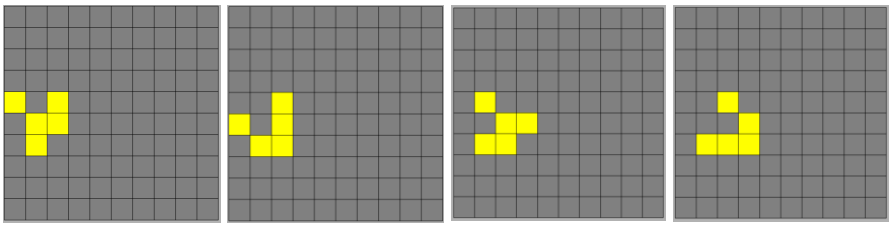


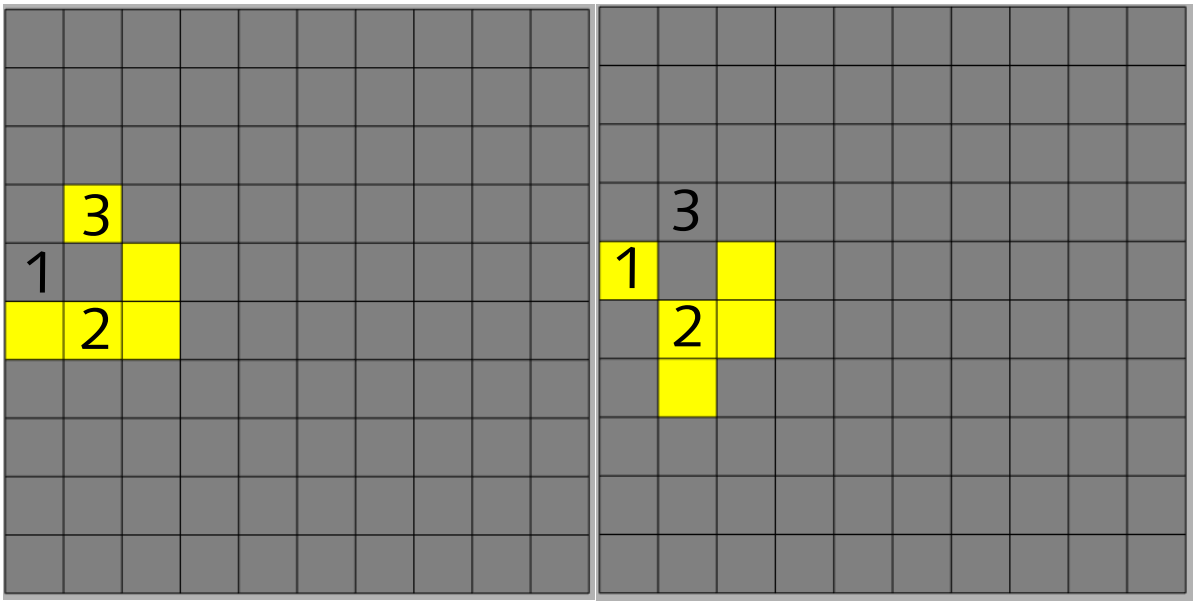
figure1: example pattern

This pattern will evolute like shown in the figures below



evolution 1 *evolution 2* *evolution 3* *evolution 4*

Let's have a look at the first evolution.



Cell one is getting alive, because there are 3 alive neigbours (top right, bottom and bottom right. Cell 2 stays alive because there are 2 alive neighbors (left and right). Cell three dies, because it has less than 2 neighbors (only bottom right). You can observe the effect of rule IIc in evolution 3 to 4.

4. Hints

- You **don't** need a dynamic animation
- Consider to split up your algorithms into 3 parts: Data model, view, and controller.
- Use one of the rectangle classes you have already written to visualize a single cell/automat
- Visualize the state of a rectangle by changing its color
- Change the color of the mesh within the rectangle class, to change the visualization of the rectangle, according to the state of the corresponding automat

5. Hand in

Hand in is only accepted via moodle with the file formats .pdf and .java.

Hand in the following elements:

A .pdf-document with the following content:

- Graph to represent the behaviour of a single automat via Nassi-Shneiderman, a mealy machine https://en.wikipedia.org/wiki/Mealy_machine, or an UML state machine
- The Nassi-Shneiderman diagram for the calculation of the whole field
- The Nassi-Shneiderman diagram for the visualization of the field
- Class diagrams for the implemented classes
- Screenshots of the evolution results

And your source code (Remember to **send all** files – especially if you used a class from a seminar or lecture):

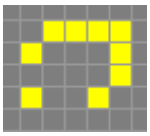
- Java class(es) with implementations
  - Data representation of the population
  - Managing the control flow for evolution
  - Visualizations for the population
- Java class(es) for testing

6. Tasks

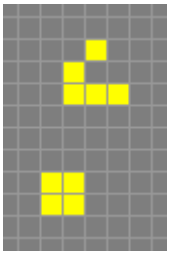
Analyze the behavior of the patterns below. Place each pattern in the middle of a 40 x 40 field and run the evolution. Take screenshots from the whole field

- At the 1st generation (Initial)
- At the 5th generation
- At the 10th genertation
- At the 20th generation
- At the 50th generation

Pattern 1



Pattern 2



Pattern 3