Project Description

DT124G Java for User Interfaces and Networking HT2018/2

Uwe Köckemann (uwe.kockemann@oru.se)
Franziska Klügl (franziska.klugl@oru.se)

Summary

The objective of this project is to implement Conway's Game of Life as a client-server application with a Graphical User Interface (GUI). The project has to be done *individually*.

Conway's Game of Life

Conway's Game of Life (WikiPedia) is one of the most famous Cellular Automata (CA). It uses a 2-dimensional grid representation in which each cell of the grid has one of two states: alive or dead. In each round, the state of a cell is re-evaluated according to a fixed set of rules. Hereby the state of a cell depends on how many of the 8 neighbours ("Moore Neighbourhood") are in the state alive or dead. The following table lists the standard rules with **s** as the current state of a cell, **n** is the number of alive neighbors, and **s'** is the resulting state.

s	**n**	**s'**	**comment**
alive	<2	dead	Death because of loneliness
alive	2-3	alive	Right number to stay alive
alive	>3	dead	Death because of overcrowdedness
dead	3	alive	Right number to become alive

Initialization of the grid is random: Each cell is set to alive with a small probability, for example 5%. Cells at the boundaries have either a reduced neighbourhood or are connected with the other boundaries to form a torus.

Requirements

Your solution needs to fulfill the following requirements:

- ***Client-server application:*** GUI and game logic must be able to run on different machines and communicate through a socket based connection. It is up to you how to assign client and server roles.

- ***Simulation Control:*** The GUI must at least allow to start, pause, and reset a running simulation. It is possible to advance the simulation by a chosen number of steps. It should be also possible to set particular pattern of alive and dead cells.
- ***Configuration:*** The GUI must allow changing parameters of the game. There are quite a lot of parameters that one might want to experiment with: The initialization probability, the map size, the numbers used in the rules, the number and content of the rules, the map size, the way how the boundaries are treated, etc.
- ***Pattern Library:*** The GUI should may also offer functionality to create and store a library of patterns that can be used or adds to the initial state or to a paused simulation.

Your code must also fulfill a number of quality requirements.

- ***Test Cases:*** JUnit test cases are used to assure that the simulation works as expected with the default set of rules. Each test case creates a pattern on a small map, advances the simulation by a single step and tests if the expected correct outcome occurs. Good candidates for tests can be found under *Example Patterns* on the Wikipedia page.
- **JavaDoc**: We expect every major class/method to be accompanied by JavaDoc documentation.

Milestones

1. Submit a project plan (First lab, Nov. 13th) that answers the following questions:
    a. What work packages do you plan for your implementation?
    b. How do you intend to schedule the work? How long do you expect each step to take?
    c. What possible problems could come up?
2. Submit an outline of your GUI (Deadline: Nov, 20th, Beginning of 2nd lab)
3. Submit a UML diagram outlining the design of implementation (Deadline: Nov 27, Beginning of 3rd lab)
4. For each of the lab times no 4 (Dec, 4h), no 5 (Dec. 11th) and no 6 (Dec. 18th) hand-in a short report about the status of your project. This report can be done orally during the lab times, you can also send an email to Uwe about your advances.

Final Hand-in and Grading

For finalizing the project part of the DT124G course, you need to

1) Hand-in a report (send as an attachment to a message to Uwe and Franziska via Blackboard)
2) After we checked the report, you will get access to a doodle at which you may select a time for demonstration - if the report is sufficiently good
3) You demonstrate your program

4) You revise your report according to the feedback that you received during the demo and send it together with you code and all requested materials to Uwe and Franziska via the messaging function at blackboard..
5) After we received your updated report (if necessary), Uwe and I will evaluate all material that you handed in, including your source code and grade it.

The deadline for handing in you report is **Monday 7th of January**.
If you do not manage to finish your report in time, there will be other time slots for demonstrations around the re-exams. Be aware that the course is given for the last time now. It will be phased out after this edition. Hand-ins later than August 19, will be difficult.

The Report

Each student must submit a short report written in English or Swedish containing the following elements

- Introduction to the project: clearly write a description of what you are asked to do and what you wanted to achieve
- Design of the simulation platform: describe your idea of how to implement basic server and client and support your idea with UML class and sequence diagrams.
- Discuss your design and implementation: mention any problems or pitfalls that you met during the implementation. What would you have done differently, if you knew from the beginning. Give some information on how you tested your platform and describe its performance,

Grading

The project is the only mandatory element of the practical part of the JAVA course. It determines your grade – the exam is just pass or not. **Projects are individual! No copying of code is allowed, also not from the internet.** Of course you may discuss solutions among your friends, so some structural similarities are not a problem, but when it comes to producing the code, you have to do it yourself. We will use (software) plagiarism checkers to determine whether you tried to cheat and will register all cheating attempts that we encounter.

We will apply the following schema for determining the grade given to the project. The project must fulfill the minimum requirement at all elements. So, a lousy report can fail a wonderfully programmed system or a wonderful report does not help if the source code is super chaotic.

	3 (minimum)	4 (good)	5 (excellent)
Report	The text describes all components of the software and follows the given structure. A UML Class diagram is contained, but does not go beyond was has been shown in the lectures.	Like 3 plus: The text is well written, there is a clear distinction between design and implementation. A UML class diagram clearly shows the design. The report contains screenshots and a discussion how realistic the simulation is.	Like 4 plus: The text is easy to understand and well structured with a good line of argumentation. It discusses why particular options have been chosen and gives clear information about simulation results. The report contains UML sequence diagrams.
Demonstration	Server and client work and the overall requested functionality is there for at least the configurations that the student gives. The game of life appears to work correctly..	Like 3 plus: Demonstration works smoothly, no exceptions happen for all fair configurations. The Game of Life is properly updated. Overall user interface is suitable for the tasks.	Like 4 plus: demonstration without flaw. The student has implemented more than the required: more than one client, really nice GUI.
Source Code	Code works, but one may discover some striking code smells. It is almost fully tested, some exceptions may happen. Documentation is scarce.	Code works well, program runs efficiently, all exceptions are properly caught. Code is well structured, but uses traditional old-fashioned constructs. Code is well documented, also with JavaDoc.	Code uses interfaces, high-level interaction constructs and clear design patterns. Code is very well-structured and readable. Hardly any code smells can be discovered. Documentation is well done, including JavaDoc.

Again: Projects are individual. Without handing in a report, it is not possible to demonstrate. During the demonstration you will not be examined in a harsh way, we just want to make sure that you understood what you did. We might request updates for the report during demonstration.

Do not hesitate to ask if you have questions, if something is unclear or you are not sure whether your approach may lead to some result.