**Exercise 2: A Two-Player version of Conway’s Game of Life Implementation**

The implementation of a Two-Player version of Conway’s Game of Life followed a similar approach as in the last two exercises. Once again we tried to create only as many classes as necessary and as few as possible in order to avoid God classes but still keep a manageable number of classes. With this approach we want to keep the project slim and clear so that it is easily understandable. The Classes are grouped in four main categories, which we call from now on “folders” for the sake of simplicity. We use this terminology as parallel to the filesystem in which the project is stored. The folders consist of “game”, “gui”, “library” and “listeners”. Each containing multiple classes. This answer sheet is structured the same way as the folders (in bold letters) are structured to be more understandable.

Any Methods mentioned in this answer sheet will be using a simplified representation without their parameters. For example a method add(int i, int j), will simply be represented by add(). This decision was taken to not overcomplicate and clutter the answer sheet

**Folder Game**

The folder Game contains all the different classes regarding the Two-Player version of Conway’s Game of Life. Excluded are the graphical representation which is represented in the folder gui. The folder Game consists of the glasses Cell, Controller, GameOfLife, Grid, Player, and Ruler.

**Class Cell**

The Class cell handles the methods and Objects roe the representation of a cell in the Conway’s Game of Life. Objects of the Class Cell have 4 attributes. A Boolean if the cell is alive, a Colour a Boolean regarding a new status and another colour. These Attributes will be assigned the value false for the Booleans and null for Colour upon creation of said objects by the in the class cell defined constructor. The constructor is public.

The Class holds multiple methods which either return the colour or a Boolean value. It also handles the checking if the cell will have a new status by the method setNextStatus(), which determines if a cell will change it´s state

**Class Controller**

The Class controller is the most central class in our construction of the Game of Life. It is responsible for the “handling” of the game

Object of the class Controller have the following attributes:

A private and final **Grid** named gamegrid;

A private and final **GameOfLifeGUI** named gameOfLifeGUI

A public two dimensional array list of **Cell** named survivalMatrix

A private and final **Ruler** named ruler

A private and final **List** of Players named playerList

A private **Player** named activePlayer

A private **int** named generation

A private **boolean** cellKilled with initial value false

A private **boolean** cellBorn with initial value false

A private static and final **Controller** named **controllerInstance**

To follow the principles learned in the lecture the attributes were set as private and final if possible, to follow the Designpattern of encapsulation and to protect the attributes from Unnecessary Access from client code.

Additionally, the Class Controller contains a private Constructor, which assigns upon the creation of a controller Object values/instances to its attributes.

The class contains a multitude of methods from returning attribute values to manipulating the Grid and cells to manage the game. The methods are once again set as private if possible, in order follow the learnt design patterns.

Example of methods are clear() which clears the survival matrix and creates new objects of the Class cell for each coordinate in the matrix, or the method stepForward() which moves the game to the next round by checking setting new colors according to the rules.

**Class GameOfLife**

The class GameOfLife is the main Class and “starts” the game by creating a new Controller object by calling the in the Controller Class implemented Method getControllerInstance() which in turn calls for the Constructor with the preassigned instances as arguments.

**Class Grid**

The Class Grid is responsible for the representation of the Grid in the game of Life. Objects of the Class Grid have the attributes

public int sizeX

public int sizeY

public int tileSize

private final Color colorBG

private final Color colorGrid

private static final Grid gridInstance = new Grid()

The role of these Attributes should be self-explanatory. Once again the attributes and the soon to be introduced methods are set to private if possible.

The constructor sets to the attributes pre-determined values (x = 600, y = 400, tilesize = 20 and colours), which are required for the graphical representation of the game.

The class extends JPanel and therefore als calls the constructor of its parentclass with super().

The methods getsize(), setSize() and paintComponent() have been overwritten with @Override as learnt in the lecture.

T

**Class Player**

The Class player is responsible for the creation of Objects of the Class Player, which are then Attributes to Objects of the Class Controller.

A Object of Class player has the following attributes: A private name (string) a private and final Color and a integer value of cells that are alive.

The Attributes are set to private in accordance with the learnt design patterns and the Class contains several public methods to return the values of Player Attributes as well as change them

**Class Ruler**

The Class Ruler is responsible for handling the rules of the game by checking how many neighbours are alive (methodconwaysRules) and returning said number aswell as counting the numbers of cells of each colour (red or blue).

Objects of the class Ruler have one attribute. A Object of the Class Controller

**Folder GUI**

The Folder GUI is responsible for the graphical representation of the game. Due to it´s nature most attributes are public. The colours however are protected and final. Each of the components of the Game have fixed positions.

**Folder Library**

The folder Library contains **external** Libraries/Code (Class Species and class Species Library) which was **not** created by us but is required for the GameOfLife. These libraries are containing the different shapes/Species that can be used in the game of Life.

**Folder Listeners**

The Folder listeners contains two classes a Button listener and a CellTogglelistener. They are responsible for Checking for user input by mouse. These Listeners are once again **external** code, that has been adapted for this use case.