CS2401 – Week 3-4

With this lab assignment, you are going to get a chance to practice some more on 2D arrays and refresh your memory on object-oriented programming, with the definition and manipulation of new types. We hope this lab helps you feel more comfortable manipulating again some concepts from CS1 and embarking on CS2 material. Let's get started!

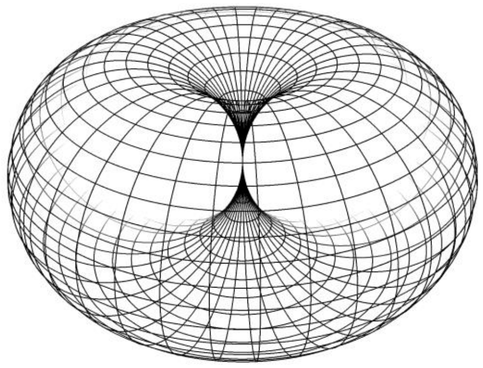
What is the goal of this lab?

We expect that, by the time you complete this lab you will:

- Be more confident about using arrays;
- Remember essential elements of java programming; and
- Feel more comfortable about approaching and solving problems.

What problem will you be addressing in this lab? In this lab, you are going to implement Conway's Game of Life, with a twist. Let us explain.

In case you are not familiar with Conway's Game of Life, please visit the following page for full details of all rules: https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life. The traditional game of life is "played" on a flat board (flat 2D array). In this lab, we will implement the game of life to be played on a torus-like board; see picture below:



Your mission is that, given any torus-shaped game-of-life, your implementation should allow any user to initialize the torus with any population and make it evolve over any given number of time steps.

Tip: a torus-like game of life follows the same rules as a flat-board game of life. The only difference between a flat-board and a torus-like game of life is what we consider a neighbor (since a torus is wrapped around itself).

What are you asked to do? You have 3 main tasks to complete in this lab.
1/ You are first going to design a new type GameOfLife (defined in a java file called GameOfLife.java). This new type is described as follows:

- Attributes:
    - Size: int

    - Board: int[][] ← this is a square board, same number of rows and columns
    - Previous: int[][] ← this 2D array will store a copy of the game of life board.
- Constructors:
    - Default constructor
    - Constructor that takes a size as input (this will be the number of rows as well as the number of columns of your board and previous)
    - Constructor that take a 2D array as an input (its size will be the size of the board and previous; its content is to be copied into previous)
- Getter:
    - getBoard → returns the current board
- oneStep:
    - Takes no parameter and is a void method
    - It consists in transforming the current board into its next shape (board at next time stamp).
- neighbors:
    - Takes two indices (representing a row index and a column index).
    - Computes the number of neighbors the corresponding cell on the board has.
- evolution:
    - Takes an integer n, which represents the number of evolution steps we need to conduct.
    - Transforms the board into the board after n steps of evolution (i.e., n successive calls to oneStep).
- Note: you can use more methods (as helper methods for the above methods). However, you need to stick at least to the ones above.

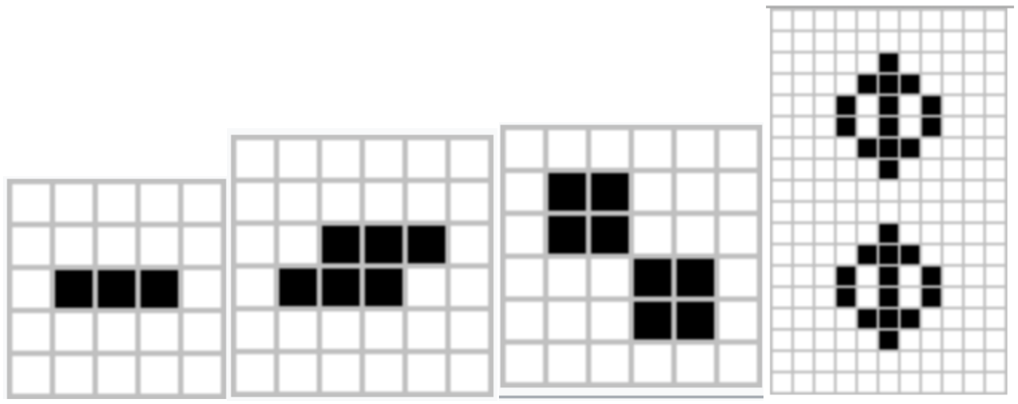
2/ You are then going to design a new type TorusGameOfLife (defined in a java file called TorusGameOfLife.java), which inherits from the type GameOfLife (keyword: extends). This new type is described as follows:

- Attributes:
    - None: all are inherited from GameOfLife
- Constructors:
    - All constructors defined as for GameOfLife
- neighbors:
    - Re-implement this method at the TorusGameOfLife level since it is the only difference between a flat-board game of life and a torus game of life.

3/ You have to test that your implementation of the flat-like as well as of the torus-shaped games of life are correct. You are expected to do so using JUnit testing, in a file called GOLTester.java.
In this file, you have to test the following class methods of the GameOfLife type for GameOfLife and TorusGameOfLife objects (5 tests at least per method):

- neighbors(int, int)
- oneStep()
- evolution(int)

In particular, you should test the behavior of your two versions of the game of life on at least the following boards:



What should you turn in?

You should submit your 3 java files: GameOfLife.java, TorusGameOfLife.java, and GOLTester.java.

How should you submit your work?

You will submit your assignment via your commits to your GitHub repository. Be sure your work shows on your repository. Failing to follow submission instructions and guidelines given by your respective TA will result in up to 15 points off your overall grade in this lab. So please pay attention.

Additionally, your java files are expected to be neat and clear (indentation and clear variable naming for the java file). Failing to do so will result in up to 15 points off. On the other hand, extra neat and clear work will be rewarded by up to 10 extra points.

By when should you submit your work?

Due date: Friday February 14 at 11:59pm

Lateness rule: -10 pts for 1 day of lateness / - 20 pts for 2 days of lateness / 0 after that, but you still have to turn in your work

---

Grading:

40 pts	GameOfLife.java
30 pts	TorusGameOfLife.java
30 pts	TorusGOLTester.java