Assignment 1 (Worth 4%) Fall 2012

Given: Friday, September 14, 2012
Electronic Copy Due: Thursday, September 27, 2012 (Midnight)

Objectives:

- To review using Java and object oriented design.
- Working with ArrayLists
- Figuring out and working with the provided code

The Problem

You are to write a simple simulator for a basic 2D cellular automata program. A 2D cellular automaton consists of a two dimensional grid of cells. Each cell either has a living creature in it or is empty. Through each iteration/generation rules are applied to determine which creatures die or are born into the next generation based upon how many neighbors are around each cell. Over time interesting patters can emerge when looking at the transition in the 2d grid over time as the rules are applied.

http://en.wikipedia.org/wiki/Cellular_automaton

Your program will be given a file that contains the rules that should be applied, the starting world and the number of generations that the program should execute.

An example of a specific set of rules is Conway's Game of Life. There are many websites devoted to describing and discussing the game of life. One good source in the Wikipedia page:
http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

Here: the rules are:

1. Any live cell with fewer than two live neighbors dies, as if caused by under-population.
2. Any live cell with two or three live neighbors lives on to the next generation.
3. Any live cell with more than three live neighbors dies, as if by overcrowding.
4. Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

<u>The Program</u>

You will be provided with code that will read in the configuration of the rules and the staring seed (or initial state) of the world. The program them must step through the requested number of generations.

When your program begins, it should prompt the user for a file name that contains the rules and initial world. After reading the file, it should then display the world at time step zero. After that, the program should pause and ask the user to hit enter before continuing on to the next generation. For bonus you could allow the user to have the program pause after each generation and continue automatically after a slight delay.

The basic steps of the program are as follows

1. Load the rules and world
2. Setup your own structures to represent information
3. Display the current world to the screen and pause
4. Start stepping through each generation
    a. For each cell in the world determine how many neighbors it has (ie the cells around it that are alive)
    b. Now go through each cell and apply any rules that match
        i. E.g. if the cell is alive then first find all the rules that apply to alive cells and then within these rules find the ones that match based upon the number of neighbors. **Note:** You will never be a case were more than one rule applies to the current situation. So you can stop when you find the first rule that applies.
    c. Display the new generation to the screen and pause
    d. Repeat steps a to d until the required number of generations have been executed.

Bonus

If the rest of the program is working, you can choose to perform one or both of the following bonus ideas (or any other instructor approved ideas): In your submission clearly indicate if you have attempted ay of the challenges.

- Make the world toroidal. This means that the world wraps around top to bottom and left to right. This has to work for determining whether a cell survives and whether a new cell is born/generated.
- Have the program pause for a short time each generation and continue on after a short period (without having to wait for user input)

Design & Implementation

The program must use an object oriented design. The comments for each class should include a short description of why you created this class and where it fits into your design.

Coding Style and Documentation

You must document your code. This documentation **must** include your name in every file you submit! In addition, basic documentation for methods and at the top of your main file is essential. Finally, poor style (e.g. incorrect indentation) will be taken into account in the grading.

Submission

Submit everything to the I Drive in a folder with your name and assignment 1 (eg JordanKidney_Asst1). Incorrect submissions will result in the loss of marks.