

CS1322 Homework 9: Game of Life

Due: Tuesday, November 22 @ 6:00PM

The "Game of Life" (not to be confused with the Parker Brothers edition) is a well-known mathematical simulation that can mimic some very complex behavior. The game tries to mimic the life cycle of cells.

The rules for the game are really very simple:

- The game is played on a rectangular board.
- The board is made up of squares.
- Each square can be either empty or occupied (i.e., contain a cell).
- To start the game, the user selects which squares are initially occupied. This is the 0th (start) generation.
- For each generation, the next generation is computed. The transition from one generation to another is called a turn.
- In each turn the next generation is computed as follows:
 - If an empty square is surrounded by exactly 3 neighbors, a new cell is born (i.e., that square becomes occupied).
 - A cell dies of overcrowding if it is surrounded by 4 or more neighbors (i.e., that square becomes empty).
 - A cell dies of loneliness if it has zero or one neighbors (i.e., that square becomes empty).
 - If a cell has 2 neighbors it remains as it is (i.e., square stays either empty or occupied)

Important: All births and deaths occur at the very end of the turn (i.e., cells selected to die would still be counted as neighbors and cells to be born are not counted as neighbors).

Neighbors are defined as any adjacent square left, right, up, down or diagonally. The diagram below depicts all the neighbors of a center cell. Cell 5's neighbors are 1, 2, 3, 4, 6, 7, 8, 9.

1	2	3
4	5	6
7	8	9

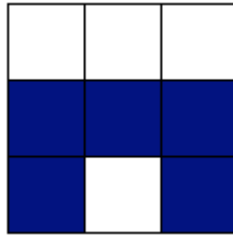
If a cell is at a border, then treat all locations off-board as empty.

Here are a few sample illustrations to help demonstrate the rules.

1)

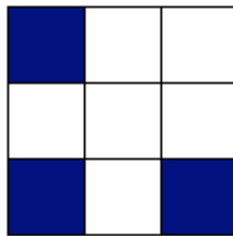
The center cell would die (become an empty square) in the next generation since it only has 1 neighbor.

2)



The center cell would die (become an empty square) in the next generation since it has 4 neighbors.

3)



The center cell would be born (become an occupied square) in the next generation since it has exactly 3 neighbors.

Program Requirements

Your assignment is to program the Game of Life subject to the rules described above. Your program should run in two modes: single step and animate. In single step mode, the game advances one turn per button press. In animate mode the game advances each turn automatically based upon the timer delay.

User Interface

Your program should run as a GUI application in a JFrame. You should have a JPanel-derived class containing 4 buttons, a JLabel, a text entry field and a drawing area.

The buttons consist of:

- Clear: Clear all cells and make all squares unoccupied. When the Run button has been pressed, this button has no effect until the Stop button has been pressed. (You cannot clear the cells while an animation is in progress)
- Step: Advance the game one turn, calculate and show the next generation. This operates the simulation in single step mode. This button has no effect in animate mode.
- Run: Run the game animated. Calculate and show each generation automatically based upon the time delay. (You will need to use a Timer to implement the delay.)
- Stop: Stop the game when running in animate mode. Has no effect if in single step mode.

For **extra credit**, you can display the current generation number (turn number) in a JLabel.

The delay between turns in animate mode should be 100ms. For **extra credit**, you may allow users to enter a custom delay.

In the drawing area, you should display the grid representing the squares that hold the cells. The default grid size should be 20 x 20. For **extra credit**, you may allow user-specified grid sizes to be entered.

When starting the game, the user should be allowed to click with the mouse inside each cell to make that cell occupied or empty. Clicking on an empty cell makes it occupied. Clicking on an occupied cell makes it empty. Once the user presses the Step or Run button, mouse clicks in the grid area are ignored until the clear button is pressed.

When a square is occupied it should be filled in with the color of your choice. All the squares should be the same color. For **extra credit**, newly born cells in each generation can be colored with a different color from the older cells in a generation.

Deliverables

- The class used to run your program (i.e., the class containing main) must be named `LifeGame`.
- Include all of the Java source code files that are required to compile and run your project.
- Remember to suitably Javadoc your code
- Remember to retrieve your submission after uploading it.

Extra Credit Opportunities

See the text in **bold** above

Some special start cell sequences

Here is an interesting starting sequence called “the glider.” The glider is cool because it repeats itself every few generations and seems to move across the screen. It will move down and right every four generations. You can use this sequence to test your application. The following snapshots show the first four generations of the glider. The pattern repeats itself on the 5th, 9th, etc. generations.

