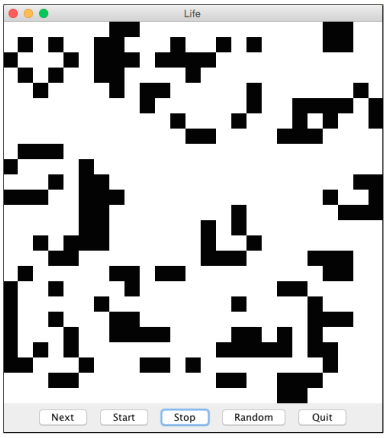
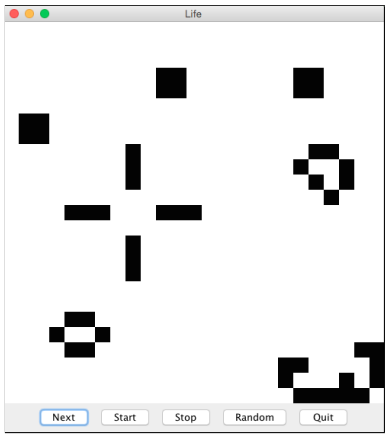
CS161: Introduction to Computer Science
Lab Assignment 10

Today you and your partner will be implementing *The Game of Life* – not the board game but the cellular automata formulated by John Conway in the 1970s.

The game is represented using a two-dimensional array of cells. Each cell can either be dead or alive. In the pictures below, alive cells are black and dead cells are white. Depending upon the state of its neighbors, a cell may either die or come to life at each generation.





Getting Started

Download the `lab10` starter code. Inside are 3 classes: `Life`, `LifePanel`, and `LifeFrame`.

- The `Life` class implements the actual game.
- The `LifePanel` class creates the panel (i.e. the window you see when you run the program)
- The `LifeFrame` class adds the interactive functionality – i.e. the buttons and responding when you push the buttons

Your job is to implement the methods in the `Life` class.

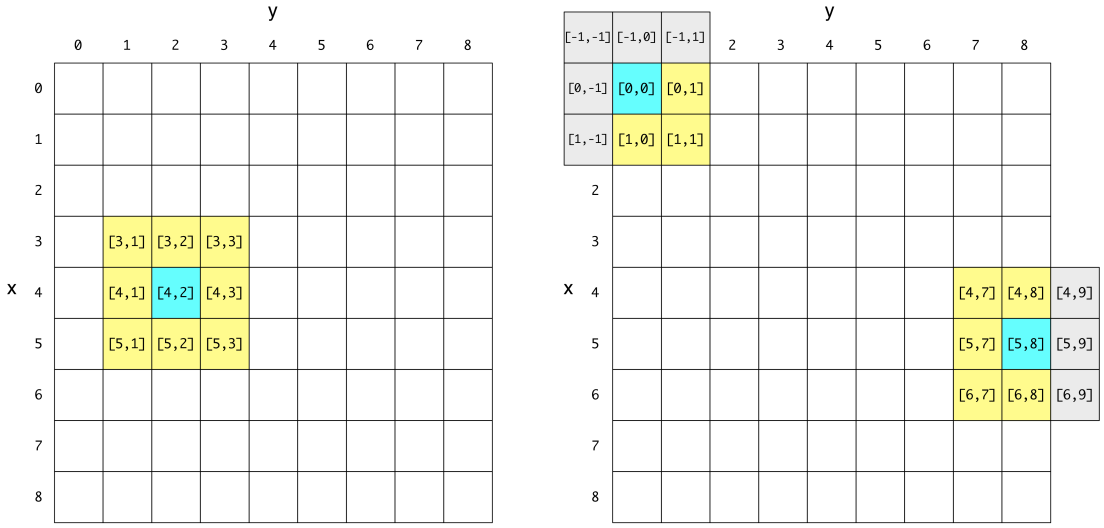
Tip: When you're debugging, it will be easier to work with a smaller board. To change the number of rows and columns in the board you can change the `BOARD_WIDTH` (i.e. columns) and `BOARD_HEIGHT` (i.e. rows) variables in the `LifePanel` class.

Rules of the Game

In the game of life, you begin with a random configuration of the board. That is, you randomly initialize each cell to dead or alive. The game is then run for a number of iterations (i.e. "generations"). For each iteration, you determine whether a cell is dead or alive using the following rules:

- Any living cell with fewer than two living neighbors dies due to underpopulation or loneliness.
- Any living cell with more than three living neighbors dies due to overcrowding.
- (By inference) Any living cell with exactly two or three living neighbors stays alive.
- Any dead cell with exactly three living neighbors becomes alive.

A cell's neighbors are the cells that are horizontally, vertically, or diagonally adjacent. In other words, a cell's neighbors are the other cells that it touches at edges or corners.



As the picture above shows, most cells have 8 neighbors. The blue cell at `(4,2)` in the first image has exactly 8 neighbors shown in yellow. However, the cells along the border have fewer than 8 neighbors. For example, `(0,0)` only has 3 neighbors and `(5,8)` only has 5 neighbors.

Writing the `Life` Class

Here is a recommended strategy for implementing the methods in the `Life` class:

1. Start with the constructor and initialize your instance variables. At the very least, you'll need a 2-dimensional array. What should be the type of the elements stored in this array? In your constructor, call the `fillRandom()` method to initialize the board.
2. The `isAlive()` and `fillRandom()` methods are the easiest to implement after the constructor.
3. Next implement the `countLivingNeighbors()` method. You'll need to handle the cases where a cell has fewer than 8 neighbors. Using a smaller board size and print statements is a great way to help you debug your method.
4. Finally, implement the `nextGeneration()` method. **Tip:** you don't want to update/modify the board itself since you need the board to count the number of neighbors. Instead, create a new 2-dimensional array to hold the cells in the next generation. After you're done filling this new array, update your instance variable to point to it.

Running the Game

To run the game, right-click on the `LifeFrame` class and select the constructor (`new LifeFrame()`). Hit "Ok" and a new window should open up for you.

You can push "Start"/"Stop" to start or pause the game. You can push "Next" to manually step through the game one generation at a time. And you can select "Random" to restart with a new random board.

Although it's random, if your board always stops changing after only 4-5 generations, something is probably wrong in your `countLivingNeighbors()` or `nextGeneration()` methods. My solution consistently either never converges to a steady state or takes over 20 generations to converge.

Extensions

The most interesting extension to the game of life is making the board "wrap around" so that every cell has 8 neighbors. In this case, the top row is wrapped around so it touches the bottom row. And the leftmost column is wrapped around so that it touches the rightmost column.

Consider the cell at `(0,0)`. In a wrapped board, this cell has 8 neighbors: `(8,8)`, `(8,0)`, `(8,1)`, `(0,8)`, `(0,1)`, `(1,8)`, `(1,0)`, `(1,1)`.

Or consider the cell at `(5,8)`. It's neighbors are: `(4,7)`, `(4,8)`, `(4,0)`, `(5,7)`, `(5,0)`, `(6,7)`, `(6,8)`, `(6,0)`.

Go back and modify your `Life` class to simulate a wrapped board where each cell has 8 neighbors.

Submitting Your Lab

Rename your folder with both of your names and then upload it to Moodle.