Life, Liberty and the Pursuit of Computer Science

This lab is meant to introduce you to the concept of animation. Nowadays we are so used to having animations on our screen we hardly give any thought as to how it is done. This is **very** new and has only happened in the last 30 years. What you are going to do is implement John Conway's program "Life".

You will need to go to the following web page:
https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

to get the gory details but I will cover the basics here and why this is so important.

One of the things computers do is enable us to **simulate** systems. You hear all the time about the different weather models on television. You also hear about nuclear simulations. All science fields use simulations. Even closer to home for our northwest silicon forest, we use simulations to simulate the running of a computer chip. My career for the last twenty years before teaching involved programming systems of millions of lines of code that allowed engineers at all the major semiconductor manufacturers (Intel, AMD, Samsung, Texas Instruments etc) the ability to simulate their chips as a software **model** long before they actually produced the silicon.

So, what's involved? What we have is a **system**. We have everything we need to know about that system and keep track of that knowledge with variables. We have rules that say how the variables should interact at any given instant of time (and this is important). So, we set up the system and then click an instant of time. We go through every variable and update its value based on the rules. Then we click the clock again and reevaluate. This works really well for "discrete" things. Computer chips really do run on a discrete clock clicking. For more complex analog systems (like weather) our models are a bit of guesswork and are only approximations.

OK. So, you are doing the game of life by Conway. Please read the above wiki link. Read the rules and history. After you have done that, return here for the details on our program.

You will need to create a two-dimensional array. The elements of the array are integers and the array should be 150X150 cells. Now, there is a little trick here. The cells are either one or zero. One means alive, zero means empty. When you print it out to the screen you want to see rectangles. But your screen is working off of pixels. Pixels are pretty tiny, so I made each element that I print out be a rectangle that was 3x3. This means your display grid is really 450X450.

Here was the declaration for the Canvas I used. Arraysize is 150, recsize is 3.

```
Canvas myCanvas = new Canvas(arraysize*recsize, arraysize*recsize);
```

So, through all my logical operations on the array I only care about is that cell alive or dead. When I print it out **then** I worry about making a 3X3 pixel rectangle.

How do we do time-sliced simulations in Javafx?

The best way to do this is using an object called a **KeyFrame.** This object allows us to tick an internal clock and update all of our objects. Once they are updated they will all display again. I have attached to these files a "ball" program. It displays a bouncing ball. The key thing here is notice that we set up everything we want about the ball, then create a KeyFrame that manages the simulation. Inside the KeyFrame we update the new x and y coordinates of the ball. The KeyFrame runs 10 milliseconds and then updates the ball coordinates and redisplays.

Program Requirements:

1) When your program starts up it should have a window (not the cmd window!) with a question of how full to fill your Life array randomly. Let's say I ask for 1000 cells to be set alive. You will get a random value for x and a random value for y. Then, in your array, see if [x,y] is 0. If so, set it to 1, bump your count up and get a new pair of random numbers. If the cell is already 1, get another pair of random numbers but don't bump your count up. When you finally get to 1000, you're ready to go.
2) Once you have the array "full" start your simulation in a **new** window like the one I've been showing in class. The simulation should run indefinitely till the window is closed.

Some suggestions for doing this. Baby steps!!!! First, write a program which just fills the array and displays it. You should run it several times till you can fill it with different densities. Once you have this done, everything else is easy to add the simulation.