Conway’s Game of Life

## From <http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life>:

The universe of the Game of Life is an infinite two-dimensional [orthogonal](http://en.wikipedia.org/wiki/Orthogonal) grid of square cells, each of which is in one of two possible states, live or dead. Every cell interacts with its eight neighbours, which are the cells that are directly horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur:

Any live cell with fewer than two live neighbours dies, as if by needs caused by underpopulation.

Any live cell with more than three live neighbours dies, as if by overcrowding.

Any live cell with two or three live neighbours lives, unchanged, to the next generation.

Any dead cell with exactly three live neighbours becomes a live cell.

The initial pattern constitutes the 'seed' of the system. The first generation is created by applying the above rules simultaneously to every cell in the seed — births and deaths happen simultaneously, and the discrete moment at which this happens is sometimes called a tick. (In other words, each generation is a pure function of the one before.) The rules continue to be applied repeatedly to create further generations.

## Requirements:

For this assignment, you will provide a way for a user to define the size of a grid on your window. Each dimension has a minimum of 10 and a maximum of 100. Each grid cell will have some representation (a Button, Label, Canvas or something else) of a single Cell. Initially all the Cells will be dead, but you will provide a way for the user to populate the grid with a random distribution of live Cells. Additionally, if the user clicks the Cell in the UI, it will change the state of the cell from live to dead or vice versa.

You will provide a way for the user to step ahead a single generation, and a way to press “play” that will progress through the generations at a maximum speed of 15 generations per second.

You must create a Cell class that holds at least the information of whether the cell is alive or dead. The Cell’s representation must be data-bound to instances of the Cell class. The representations’ change in appearance must be done by some kind of ValueConverter. Any logic done for the game in your application must be done on the Cell objects, not the on-screen representation.

## Rubric:

* (30 pt.) Game of Life is implemented with the correct rules, and it functions correctly.
  + If you want to test your implementation, draw a straight line (horizontal or vertical, not diagonal) with 10 live Cells in a row, then run it. You should see it morph into a repeating pattern. Alternately, you can try any of the patterns documented in the Wikipedia article.
* (10 pt.) User can create a grid of any size between 10x10 to 100x100. The grid can be rectangular instead of only square. You may allow the user to create smaller or larger grids, but the program must allow at least these sizes.
* (5 pt.) Button to step ahead one generation is correctly implemented.
* (15 pt.) Button to “play” is correctly implemented, including speed throttle.
  + Some people like to let their simulations run at maximum speed. If you’re one of those people, be sure to include some sort of throttle option that the grader can enable so that they can test if you’ve implemented the 15 generations / second requirement.
* (10 pt.) Provide a way to populate the grid randomly with living cells.
* (10 pt.) Cell representations on the grid use data binding to update themselves.
* (10 pt.) Cell representations can be clicked to toggle a cell’s state between alive and dead.
* (10 pt.) User interface is easy to use, code is clean and well-documented.