Conway's Game of Life

Let the fun begin! Your real assignment centers on the use of a two-dimensional grid as a data structure in a cellular simulation. It will give you practice with control structures, classes and their responsibility, and even a bit of string and file processing. More importantly, the program will exercise your ability to decompose a large problem into manageable bits. What's especially nice about this program is that you can construct a beautiful and elegant solution if you take the time to think through good design. It can be as lovely to look at the code as it is to watch it run.

The Problem

Your mission is to implement a hyped-up version of the game of Life, originally conceived by the British mathematician J.H. Conway in 1970 and popularized by Martin Gardner in his Scientific American column. The game is a simulation that models the life cycle of bacteria. Given an initial pattern, the game simulates the birth and death of future generations using simple rules. The game is played on a two-dimensional grid. Each grid location is either empty or occupied by a single cell (X). A location's neighbors are any cells in the surrounding eight adjacent locations. In the following example, the shaded middle location has three "live" neighbors

The Rules

The simulation starts with an initial pattern of cells on the grid and computes successive generations of cells according to the following rules:
1. A location that has zero or one neighbor will be empty in the next generation. If a cell was in that location, it dies of loneliness.
2. A location with two neighbors is stable—that is, if it contained a cell, it still contains a cell. If it was empty, it's still empty.
3. A location with three neighbors will contain a cell in the next generation. If it was unoccupied before, a new cell is born. If it currently contains a cell, the cell remains. Good times.
4. A location with four or more neighbors will be empty in the next generation. If there was a cell in that location, it dies of overcrowding.
5. The births and deaths that transform one generation to the next must all take effect simultaneously. Thus, when computing a new generation, new births and deaths in that generation don't impact other births and deaths in that generation. To keep the two generations separate, you will need to work on two versions of 3 the grid—one for the current

generation, and a second that allows you to compute and store the next generation without changing the current one.

Configuration

To get the colony underway, your program should offer the user a chance first define the number of rows and column in grid and then seed the grid manually or request a random start. When every cell is stable then stop the simulation and print the resultant colony.

Coding rules

Push your code on Github repo and share it with us @ priyanka@walkover.in, hatim@walkover.in
You can use any preferable language, make sure to write a proper structure code following concepts of OOPS, with unit test (TDD).

Happy coding :)