# Problem

This Kata is about calculating the next generation of Conway’s game of life, given any

starting position. See http://en.wikipedia.org/wiki/Conway%27s\_Game\_of\_Life for

background.

## Description

You start with a two dimensional grid of cells, where each cell is either alive or dead. In this

version of the problem, the grid is finite, and no life can exist off the edges. When calculating

the next generation of the grid, follow these rules:

1. Any live cell with fewer than two live neighbours dies, as if caused by under population.

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

3. Any live cell with two or three live neighbours lives on to the next generation.

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

You should write a program that can accept an arbitrary grid of cells, and will output a similar

grid showing the next 5 generations.

Make a beautiful interface to render grid of any size.

Make sure you have enough coverage of edge cases - where there are births and deaths at

the edge of the grid.

# Requirements

Task: Write a program that can accept an arbitrary grid of cells, and will output a similar

grid showing the next 5 generations.

1. Use a two dimensional grid of cells where the grid is finite.
2. The calculation of the next generation must follow the following rules:
3. Any live cell with fewer than two live neighbours dies, as if caused by underpopulation.
4. Any live cell with more than three live neighbours dies, as if by overcrowding.
5. Any live cell with two or three live neighbours lives on to the next generation.
6. Any dead cell with exactly three live neighbours becomes a live cell.
7. Make a beautiful interface to render grid of any size.

# Problems / Questions

1. How is the start defined? How does Gen-0 looks like?
2. How is the end defined? Will the algorithm die after some generations or does it needs to be restricted?
3. How are the edges calculated properly eg. a corner like position [0][0]?

# Assumptions

1. Gen 0 is generated randomly.
2. Calculation stops after given amount of generations.
3. Edges are calculated so that the “missing” neighbours are all 0.

# Milestones

1. First working version in console
2. First working version in GUI
3. Nice working version in GUI