Exercise: Concurrency Patterns – Dependencies

In this exercise you will become acquainted with the handling of dependencies in concurrent programs and use one or more of the strategies we have learned to work around them and still gain a performance increase through parallelization

---

Conway's Game Of Life is an old "zero-player" game which simulates the life (and death) of cells on a square grid. You can read about the rules for the game on Wikipedia, but basically, a cell on a 2D grid can be in one of two states: living or dead. The simulation progresses in steps, and for each step each cell on the grid is evaluated for its state in the next step, based on the number of live neighbors.

Exercise 1:
Implement a serial version of the Game Of Life on a 1000x1000 grid on which each cell initially has a 25% chance of being alive (you can play with the numbers as you like). Run the simulation a number of times (e.g. a 100 times or a 1000 times) and see how long it takes.

Exercise 2:
Obviously, the individual steps of the calculation are dependent on each other, but the calculations within each step is not. Create a parallel version of the Game Of Life, run the serial and parallel versions on the same initial grid configuration and compare their execution times.
Hint: Use Parallel.For or Parallel.ForEach (or try both)

Exercise 3:
Create a parallel version of the game, which uses a *task barrier* as presented in the lecture. Compare this implementation's execution time to the other two. Which one is more efficient?

Exercise 4:
Are you sure your parallel implementations are correct?
Run your sequential and parallel versions side-by-side and verify that the same number of cells are alive in both implementations in each iteration.

Exercise 5 – optional
Yes, you can add a GUI to the game. Go ahead, have fun!