


Course	<b>Design of Software Systems</b>	 The Mærsk Mc-Kinney Møller Institute
Lecturer	Marco Kuhrmann	
Kind	Assignment 2	
Title	<b>Software Test and Test-driven Development</b>	
Hand in	2015-11-15, 23:59 (strict); i.e., end of week 46	

## Preamble and Instructions

In this assignment, you work in small teams (up to 3 students). The main topics of this assignment are:

- Software Development
- Software Test, in particular: Test-driven Development

In this task, you will have to develop 3 small software solutions and have to provide proper test environment, in particular, the test suites and documented test runs.

### Expected Deliverables:

Your team's submission consists of a single zip file containing the following folder structure:

- group-[login1]-[login2]-[login3].zip
  - [login1]-[login2]-[login3]-ass02.pdf [DOCU]
  - [tennis] [TENNISGAME]
  - [gol] [GAMEOFLIFE]
  - [mars] [MARSROVER]

In the assignment document [DOCU], you have to name your team again (full name, numbers, etc. for the BB evaluation tools), and you have to briefly describe your implementation and testing approach. Notably, describe how you organized and implemented the development and testing approach, i.e., how did you find test cases, how did you develop the software solutions? Present the protocols generated from your test suites. Furthermore, please include a small section with instructions of how to run your software.

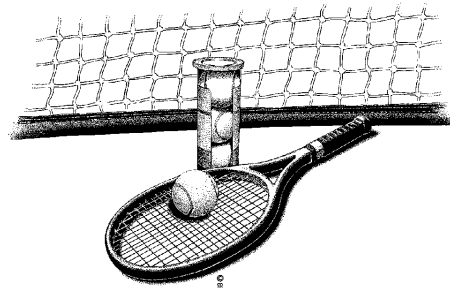
The folders [TENNISGAME, GAMEOFLIFE, MARSROVER] contain the source codes of your applications. The folders must contain the source code (as Visual Studio project) for inspection. The project must compile and run. The folders must also contain the unit tests, which have to be executable as well.

**In order to pass this assignment, the delivery must be complete! The strict deadline for submitting your solution is end of week 46, in particular: 2015-11-15, 23:59.**

**Recommendation: Start work as soon as possible!**

# 1 Let's play: Tennis...

In this task, you'll implement a small tennis game.



Here are the rules:

- Rule 1: A game is won by the first player to have won at least four points in total and at least two points more than the opponent.
- Rule 2: The running score of each game is described in a manner peculiar to tennis: scores from zero to three points are described as "love", "fifteen", "thirty", and "forty" respectively.
- Rule 3: If at least three points have been scored by each player, and the scores are equal, the score is "deuce".
- Rule 4: If at least three points have been scored by each side and a player has one more point than his opponent, the score of the game is "advantage" for the player in the lead.

## Task:

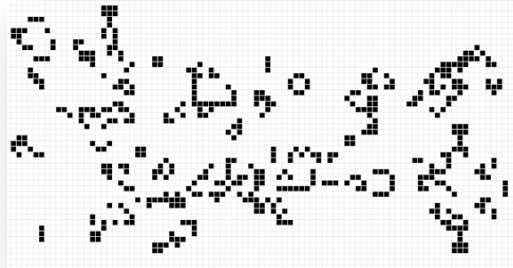
Your task is to implement a small C# console program, which simulates a tennis game. For this, the program uses a random number generator to decide which player scored. The program shall print the current scores and points to the console.

## Expected Deliverables:

- A Visual Studio solution containing your allocation and the tests.

## 2 Let's play: Conway's Game of Life...

"The Game of Life" takes place on a two-dimensional, finite, and 2D square grid of cells. Each cell is in one of two possible states: alive or dead. The grid is initialized randomly and the game progresses in discrete steps called "tick's". At each tick, existing snapshot of the grid is used for each cell to interact with its neighbors based on four basic rules to determine the next state of the grid. The state of the grid is referred to as "Life", which is a representation of the dead and alive cells in a grid of a fixed size.



The following rules are applied for "Life" to move towards a new state:

- Rule 1:** Any live cell with fewer than two live neighbors dies.
- Rule 2:** Any live cell with two or three live neighbors lives.
- Rule 3:** Any live cell with more than three live neighbors dies.
- Rule 4:** Any dead cell with exactly three live neighbors becomes alive.

Game-specific requirements:

- A cell shall be in two possible states: Dead or Alive
- The grid shall be of any size that is composed of a square arrangement of cells in 2D, e.g., 10x10, 50x50, etc. Therefore, each cell shall be accessible by a unique (x, y) coordinate.
- The states of the cells in the grid shall be initialized uniformly random at the beginning, i.e., number of alive cells should be roughly equal to the number of dead cells.
- Neighbors of a cell are defined as the ones, which are in the immediate vicinity that surround the cell (including the diagonals). For instance, any cell inside the grid has eight neighbors, whereas cells in the borders have five and cells in the corners have 3 neighbors. A cell is not its own neighbor.
- At each tick, the state of the grid shall be updated according to the four neighborhood-based rules above. These rules shall be applied to all cells simultaneously, i.e., the order of cells to which the rules are applied should not affect the outcome.
- The state of the grid is called "Life" and shall be represented as a string, where dead cells are represented with a "-" (dash) and alive cells are represented with a "\*" (star) character. When printed, this string should be displayed in N lines (rows of the grid), each consisting of N characters (columns of the grid), where NxN is the size of the grid.
- You shall throw "CustomLifeException" for all error cases including but not limited to null pointer errors and attempts to access a cell that is outside the edges of the grid (e.g., when checking for neighbors).

### Task:

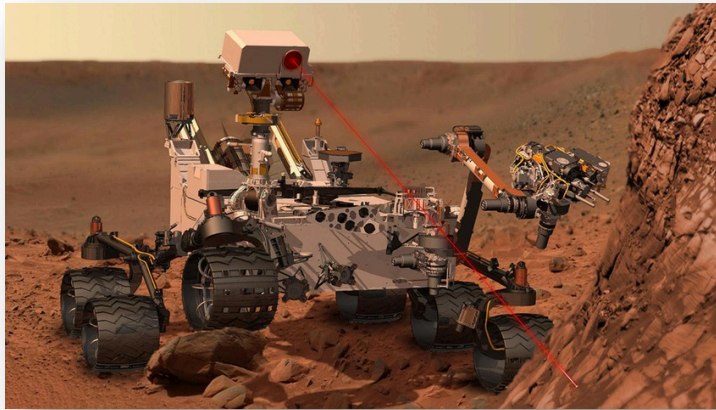
Your task is to implement "The Game of Life" as a C# text console program. Right after the start, the program shall ask for the field size and the number of ticks to be played. Then a randomly generated start population is created, before the game starts. Print out all ticks to the console window.

### Expected Deliverables:

- A Visual Studio solution containing your allocation and the tests.

### 3 Where no one has gone before: Mars Rover API

Your task is to develop and test a controlling API for a Mars rover...



Your overall task is to develop an API that moves a rover around on a grid. For this, find the following conditions and have to fulfill the following tasks:

- You are given the initial starting point (x, y) of a rover and the direction (N, S, E, W) it is facing.
- The rover receives a character array of commands. Commands are handled as follows:
  - Implement commands that move the rover forward/backward (f, b).
  - Implement commands that turn the rover left/right (l, r).
- Implement wrapping from one edge of the grid to another. (Planets are spheres after all...)
- Implement obstacle detection before each move to a new square. If a given sequence of commands encounters an obstacle, the rover moves up to the last possible point and reports the obstacle.

**Example:** The rover is on a 100x100 grid at location (0, 0) and facing NORTH. The rover is given the commands "ffrff" and should end up at (2, 2)

#### Task:

In order to provide you with a framework, you receive a scaffold application that visualizes your moves. Your task is therefore to develop a *Controller* component that delivers the move sequences to the rover. In particular, your task is:

- a) Take the scaffold application.
- b) Use TDD as development approach to develop the *Controller* component.

#### Expected deliverables:

- A Visual Studio solution containing the TDD-version of the *Controller*, i.e., the *Controller* class(es) and their tests only.
- A completed scaffold application that includes your (tested) controller.