E.1 – Mars Rover (Classicist TDD)

Objective

- Evolve the Mars Rover behaviour one test at a time.
- Continuously refactor the Mars Rover internal design

Requirements

- Mars Rover starts at the "0-0- N" position
    - 0,0 are X,Y coordinates on a 10 by 10 grid
    - 0,0 is at the bottom left corner and 9,9 is at the top right corner.
    - "N" is the direction the rover is facing (i.e. N,S,E,W).
- Mars Rover must received a string of characters with instructions. Valid characters are:
    - "L": rotates rover left
    - "R": rotates rover right
    - "M": moves rover one cell forward in the direction it is facing
- The rover should return its current position after executing the sequence of commands. Example:
    - "RMMLM": Rover rotates right (North->East), move 2 cells, rotate left (East->North), and move another cell. Rover returns its current position at "2-1- N"
- The rover wraps around if it reaches the end of the grid. E.g. when it gets to y:9 or x:9 it moves to y:0 or x:0 respectively.
- The grid may have obstacles. If a given sequence of commands encounters an obstacle, the rover stops at the last possible move and reports the obstacle and the position where it stopped, e.g. O-2-2- N

Extra requirements

- Make size of the grid configurable
- (advanced) Make grid infinite

Exercise rules

- Start with a tests for the Rover class.
- Write one test at a time, following Classicist TDD
- Keep the behaviour inside the Rover class to a minimum.
- Extract classes in the refactoring phase.
- Unit test other classes, if necessary.
- Each class should have a single responsibility.
- Use data structures that are related to the Rover domain.