Assignment: Mars Rover

By: Behroz Sikander

Assumptions:

1- No collision between rovers.
2- There can be more than 1 plateau. Due to this reason, the Plateau class is not static.
3- Currently, the rover can spin along 4 cardinal points (N,E,S,W) but it is assumed that in future more intermediate cardinal points can be added (NE, SE, SW etc). So, an interface *CardinalManager* has been provided which manages the cardinal points for rover. *SimpleCardinalManager* is the concrete implementation. This interface also helps to avoid direct dependency on concrete class. In future, a Context class can be added to select at runtime which cardinal manager is needed (simple, intermediate, others). Strategy pattern is applied here. Further, North/South/East/West have been represented by their own classes so that in future if we have a *IntermediateCardinalManager* class, it can be easily reuse the already implemented directions.
4- 'L', 'R' and 'M' actions have been separated out using Strategy pattern and it is assumed that in future more actions can be added.
5- As soon as the commands are sent to the rover from *MissionControl*, it is assumed that the rover starts the movement.
6- This implementation might seem to be over-engineered but It was made this way due the following line in the assignment.

"code that you'd be able to run, maintain, and evolve"

Approach:

TDD has been used to implement this assignment. Approach was RED-GREEN-REFACTOR.

1- Write testcase for a requirement
2- Add enough implementation code to compile the code
3- Refactor to add proper implementation (while considering clean code)
4- Handle corner cases by writing test case followed by implementation.
5- Repeat

P.S. I am very new to TDD and I have applied all of my knowledge that I have learnt through meetups and internet. It was quite fun and interesting way to develop softwares.

Running the code:

The application has been developed in Java. Open the project in Eclipse, build and execute the test suite.

"TestRoverSquadNavigation" test case contains the example given in the assignment.

Design:

1- *CardinalDirections* is an abstract class representing different directions. Currently North, South, East and West derived classes have been provided and in future more can be added.
2- *CardinalManager* is an interface which provides an abstraction on top of rovers cardinal directions. *SimpleCardinalManager* is a concrete implementation which uses the *CardinalDirections* to specify the cardinal system consisting of N,S,E and W.
3- *Commands* interface is used to abstraction on top of rovers actions (L,R,M).
4- *Plateau* is a simple class holding the lower and upper coordinates of a plateau and a "contains" method which can tell if a rover is within boundaries or not.
5- *Point* class is just a simple 2dPoint representation.
6- *Rover* is the main class which accepts a string of actions and performs them using the above mentioned classes.
7- *MissionControl* is a class representing NASA and current mission. This class can deploy rovers on plateaus and can send commands to the active rover.

Big picture: