Candidate Name: Samuel de Souza Lopes
Position: Legacy Java Developer

Problem Two (MARS Rovers)

1. Objective

This document intends to give a brief description of the design and assumptions along with my code. Section 3 shows instructions on how to run my application.

2. Design Description

To solve this problem, I developed an application with the following classes:

- **Input**: The objects of this class receive the canonical path of an input file *.txt* and convert this file to supply the objects of the classes Plateau, Rover, and Squad. About the input, the application ends its execution and print an error message if (i) the upper-right coordinate of the plateau is less or equal to zero, (ii) the initial position of a rover is out of the plateau zone, (iii) the initial direction of a rover is different of N, S, E and W, (iv) the letter that represents the rover movement has at least one letter different from L, M or M, and (v) the plateau or rover coordinates are not integer numbers;
- **Plateau**: This class represents a plateau through its lower-left and upper-right coordinates. Such coordinates are needed to guide the rover movement. The class **Coordinate** represents such coordinates;
- **Rover**: This class represents the rovers and associates them with a plateau and a squad. The method *finalPosition()* calculates the final position of the rover. Collision among the rovers and leaving the plateau are critical situations. For each movement, the method checks if the rover can collide with another one that already performed its moves or if the rouver can leave the plateau. If at least one of such situations occurs, the method cancels the move and skips to the next one;
- **Squad**: Here, an ArrayList of Rover class objects represents the squad;
- **Output**: This class receives a squad and print the final result in the defined format;
- **Main**: Main class call the methods needed to execute the aplication properly.

I also created five test cases using the JUnit framework to check if the results are correct. I took into account the cases where rovers might collide with another one and if the rover tries to leave the plateau.

3. Instructions to run the application

To run the application, you need to open the file *input.txt* and insert the input. The input format is the same as in the problem specification. To represent the rover direction, you need to use the characters *N* (North), *S* (South), *E* (East) and *W* (West). The *input.txt* has the same example

showed in the problem description. The first line is the upper-right plateau coordinates, where must have integer numbers greater than 1. The remaining lines are the rovers input data, where the first line is the initial coordinate (integer numbers greater than 1) and direction (N, S, E or W), and the second line is the sequence of moves (L, M, or R). After this, save *input.txt*, open the class Main.java and run it.