Mars Rover

ThoughtWorks Coding Test

Alex Olivier

# Outline

The solution to this test is made up of three main sections. The first is the simple UI that is presented to the user with input and output text boxes and a button to run the system.

The parser class handles the interpreting of the input instruction string and expanding it into the individual elements. A plateau object is created and its dimensions set based on the input. Following this, for each pair of robot instructions, a robot object is created (with the plateau passed to it) and the robot’s initial position and orientation set. Then the parser runs through the instruction set to run on each robot and returns a collection of robots back to the UI which reads their final position and displays them to the user.

The robot class itself keeps track of a robots position and orientation and handles the commands from the instruction updating its location as each one is called. It also checks that it doesn’t go out of the bounds of the plateau.

# Structure

|  |  |
| --- | --- |
| File | Description |
| MarsRoverUI.cs | The UI for the system which presents the user with the input and output boxes as well as the button to start the parsing |
| Parser.cs | The input string parser which validates the input and then creates the robot objects and runs the instructions on the robot |
| Models/Location.cs | The data structure for a robots location contain X, Y and Orientation (enum) |
| Models/Plateau.cs | The data structure for defining a plateau that contains the methods to test if a location is contained within it. |
| Models/Robot.cs | Represents a robot on the defined plateau, its location and the movement and rotation methods. |

# Testing

Visual Studio’s internal Unit Testing system was used to test the system. A separate test project is included in the solution.

# Assumptions

The developed programme makes the following assumptions:

* The first line of the input will always be 2 integers to specify the size of the plateau. All conversions are done with Int32.convert() so floats would not be acceptable
* This line is then stripped off, and it is assumed the rest of the input with contain line pairs for each robot. The system will display an error message to the user if the format of the these line pairs is not as expected
* The program assumes a robot cannot go outside of the plateau and checks for this before making a move.
* In any error case, an exception is thrown with the relevant error message. These can be handled in many ways. For the sake of this test, the message is written out to the output UI text box to inform the user.