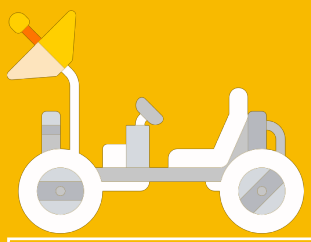


Mars Rover Code Challenge

The diagrams below is not UML diagram. They just demonstrate the classes implemented in this challenge and the relationship between the classes. It looks something close to UML, but is not a UML diagram.

NOTE: All constructors are private. Objects are instantiated using a *From* method. Why? It just reads better to me(*i.e make Position.From(x, y, direction)*).

ASSUMPTIONS: It was not stated what should happen when the Driver of the rover tries to move outside the bounds of the rover surface. Lastly, it was not stated what happens when the rover tries to lend outside the bounds of the surface defined(i.e. Surface = 8 10, Initial position = 10 10 E).
Since these was not stated, the following assumptions where made:

- The bounds of the surface are similar to a **wall**. When a Rover/robot/anything hits the wall/ surface-boundry and the driver tries to move forward, ***the rover will remain in the same position***.

- When the rover tries to lend outside the defined surface, ***it will calibrate its lending position to lend within the defined surface***. Thus for a Rover with *Surface* = 8 10, *Initial position* = 10 10 E, the initial position will be calibrated to *Initial Position* = 7 9 E (See a 8 10 surface in *figure 2*).

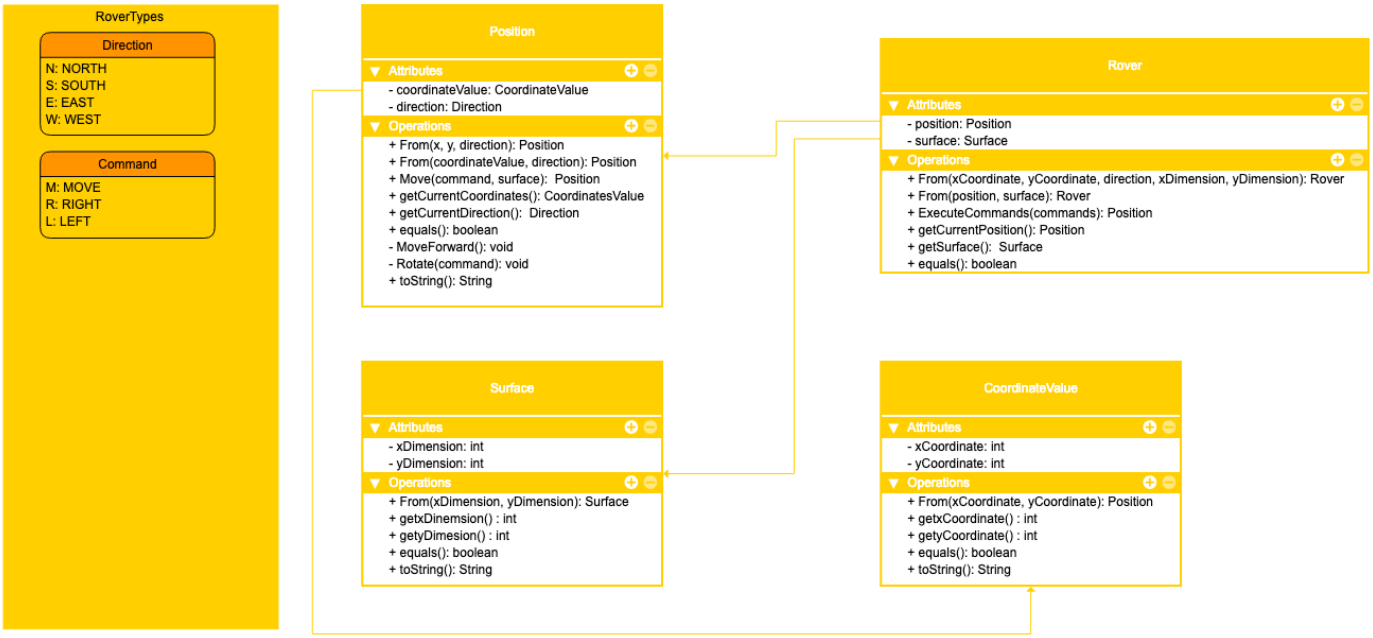
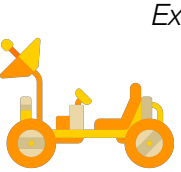


Figure 1: Relationship between all components that make up the rover

Rover class

The whole challenge is about a *Rover*, thus it make sense to have a class called **Rover**. Rover class has the **surface** on which it will be moving on, and its current **position**.

Besides the basic methods that the Rover has, there is a method for executing commands - *ExecuteCommand.* This method takes a list of commands and executes them.



Position class

This class is self explanatory. This class contains the position of the Rover and this position consists of both the Rover **Coordinates** and **Direction**. This attributes are described in the following section.

This class contains the basic house keeping methods. It also contains a method that changes the position itself - *Move*. This method takes the *command* and the *surface* where the rover will be moving. The *command* is used to decide how to change the position, while the *surface* is used to ensure that the rover does not move outside the surface.

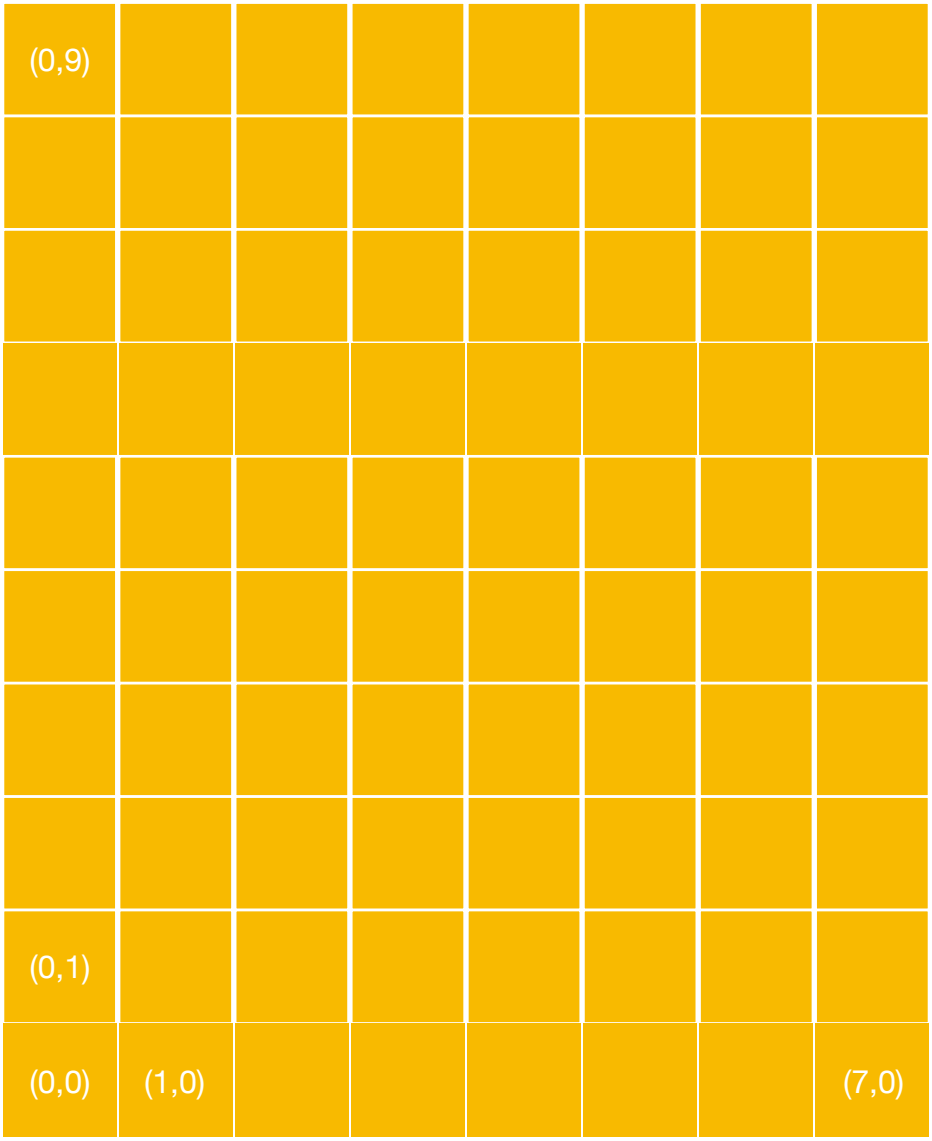
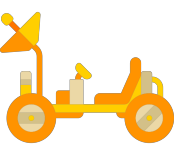


Figure 2: Example of a Surface with 8 10 dimensions

Surface class

This class defines the surface where the rover will be moving. This is just a class with dimensions. In fact, this class should have been in the **ValueObjects** Package.



CoordinateValue Class

Why Value Objects? Value Objects increase the conceptual cohesiveness of objects, readability of our code and the overall suppleness of our design. When I started implementing the Rover, I thought it was going to be complex, but then it was not. For a complex model, value objects are the best way to represent the real life values instead of primitive values. We can add logic inside value objects to ensure that the value follows the real life rules. SADLY THIS SOLUTION WAS NOT COMPLEX AND WE ONLY HAVE ONE/TWO VALUE OBJECTS.

Coming back to the **CoordinateValue** class, it is a basic representation of the coordinates. Nothing fancy.

RoverTypes

Rover has two types which are Enums.

COMMAND
This defines all commands understood by the Rover as stated in the challenge.

DIRECTION
This defines all direction a rover can be facing as stated in the challenge.

RoverDriver class

This is literally the Driver of the Rover :). This class contains the main method. The solution is ran from the terminal(See README file).

This Driver is very strict, meaning if you are not nice to it, it will not be nice to you. What does it mean to not *be nice:*

- Passing commands that the Rover cannot understand.
- Passing *non-integer* or *negative* position values.
- Passing *non-integer* or *negative* surface values.
- Passing direction the Rover cannot understand.
- Not giving the Driver enough information to drive the Rover.

So please be nice and enjoy driving this Rover. :)