Notes on revised UML for Melody Mars Rover App

Please see UML on next page

MarsRoverManager is the main service level componant of the Mars Rover sub-system

Plateau is an abstract class representing a plateau object that implements the ICheckPosition interface to check whether a given **Position** is within the boundary. This is to allow for different shaped plateaus to be accomodated

RectangularCartesianPlateau represents a rectangle plateau with X,Y coordinate system and is the only derived class of Plateau implemented

Mover is a class instanced from **MarsRoverManager** and is responsible for Moving a **MarsRover** on the plateau. This class does performs validation of each **Position** in the move to ensure it is available before moving the **Vehicle** according to the instructions presented to it

VehicleManager manager a list of **Vehicles**

Vehicle is an abstract class for moveable vehicles from which other vehicles can be derived

MarsRover is derived from **Vehicle** and represents a Mars Rover class. Not sure whether I'll have time to implement all the functionality shown though!

ICheckPosition is a general interface that defines a 'canUse (**Position**)' method. This is implemented in the classes **Plateau** (to check within the boundary), **VehicleManager** (to check the position is not occupied by another Vehicle) and **MarsObjectManager** (to check that there is not an obstruction in a given position

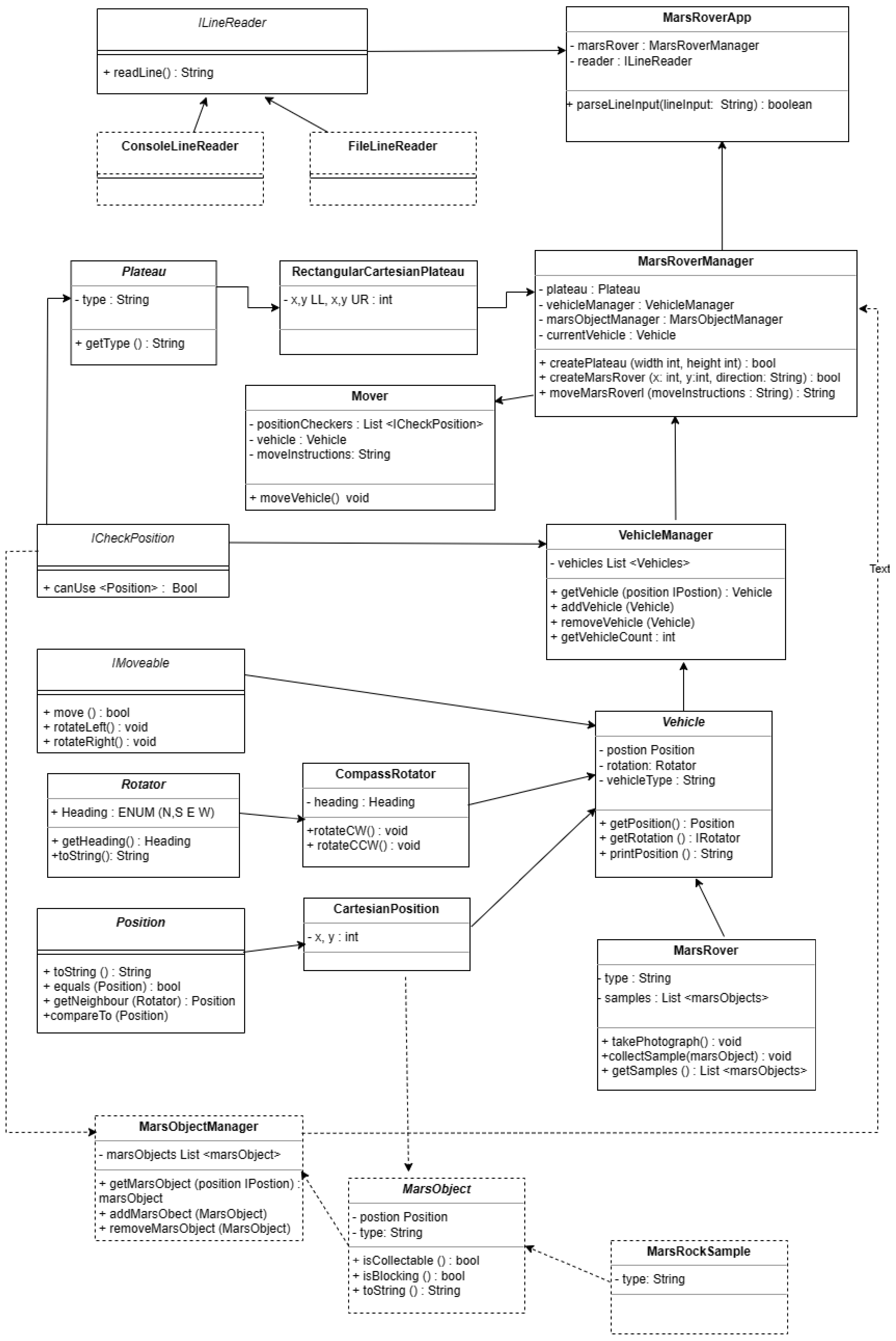
IMoveable defines the functionalty for a Moveable object

Rotator is an abstract class for dealing with rotations, whilst **CompassRotator** is the implemented version of this. I was trying to abstract the rotation functionality to maybe support other methods for example degrees or radians or some other method

Position is an abstract class to represent a Position on the grid, and **CartesianPosition** is the extended implermentation. The idea was to allow for other positional systems in the future e.g lat, long, 3 words etc

The items shown in dashed lines (**MarsObjectManager**, **MarsObject**, **MarsRockSample**) are to manage other object in the plateau, such as rocks, obstructions, samples etc that don't move but could be placed into the grid prior to running say a Move on a **MarsRover.** The Move instruction set could then be expanded to allow the rover to pick up samples in a basked for example or take photos. Although this is outlined on on the UML I'm probably not going to have time to implement this. Would also need some additional methods in **MarsRoverManager** to access and create objects etc

I havn't really looked at the UI aspect yet, I'll probably need to update the UML when I've done some Googling around this!


ILineReader
+ readLine() : String
MarsRoverApp
- marsRover : MarsRoverManager
- reader : ILineReader
+ parseLineInput(lineInput: String) : boolean
ConsoleLineReader
FileLineReader
Plateau
- type : String
+ getType () : String
RectangularCartesianPlateau
- x,y LL, x,y UR : int
MarsRoverManager
- plateau : Plateau
- vehicleManager : VehicleManager
- marsObjectManager : MarsObjectManager
- currentVehicle : Vehicle
+ createPlateau (width int, height int) : bool
+ createMarsRover (x: int, y:int, direction: String) : bool
+ moveMarsRoverl (moveInstructions : String) : String
Mover
- positionCheckers : List <ICheckPosition>
- vehicle : Vehicle
- moveInstructions: String
+ moveVehicle() void
ICheckPosition
+ canUse <Position> : Bool
VehicleManager
- vehicles List <Vehicles>
+ getVehicle (position IPostion) : Vehicle
+ addVehicle (Vehicle)
+ removeVehicle (Vehicle)
+ getVehicleCount : int
Text
IMoveable
+ move () : bool
+ rotateLeft() : void
+ rotateRight() : void
Vehicle
- postion Position
- rotation: Rotator
- vehicleType : String
+ getPosition() : Position
+ getRotation () : IRotator
+ printPosition () : String
Rotator
+ Heading : ENUM (N,S E W)
+ getHeading() : Heading
+toString(): String
CompassRotator
- heading : Heading
+rotateCW() : void
+ rotateCCW() : void
Position
+ toString () : String
+ equals (Position) : bool
+ getNeighbour (Rotator) : Position
+compareTo (Position)
CartesianPosition
- x, y : int
MarsRover
- type : String
- samples : List <marsObjects>
+ takePhotograph() : void
+collectSample(marsObject) : void
+ getSamples () : List <marsObjects>
MarsObjectManager
- marsObjects List <marsObject>
+ getMarsObject (position IPostion) : marsObject
+ addMarsObect (MarsObject)
+ removeMarsObject (MarsObject)
MarsObject
- postion Position
- type: String
+ isCollectable () : bool
+ isBlocking () : bool
+ toString () : String
MarsRockSample
- type: String