Robot Game

By

Santosh Yamarthi

Table of Contents

1. Requirements

1.1. Description

Toy Robot Simulator
==========

Description

- The application is a simulation of a toy robot moving on a square tabletop, of dimensions 5 units x 5 units.
- There are no other obstructions on the table surface.
- The robot is free to roam around the surface of the table, but must be prevented from falling to destruction. Any movement that would result in the robot falling from the table must be prevented, however further valid movement commands must still be allowed.

Create an application that can read in commands of the following form:

PLACE X,Y,F
MOVE
LEFT
RIGHT
REPORT

- PLACE will put the toy robot on the table in position X,Y and facing NORTH, SOUTH, EAST or WEST.
- The origin (0,0) can be considered to be the SOUTH WEST most corner.
- The first valid command to the robot is a PLACE command, after that, any sequence of commands may be issued, in any order, including another PLACE command. The application should discard all commands in the sequence until a valid PLACE command has been executed.
- MOVE will move the toy robot one unit forward in the direction it is currently facing.
- LEFT and RIGHT will rotate the robot 90 degrees in the specified direction without changing the position of the robot.
- REPORT will announce the X,Y and F of the robot. This can be in any form, but standard output is sufficient.
- A robot that is not on the table can choose the ignore the MOVE, LEFT, RIGHT and REPORT commands.
- Input can be from a file, or from standard input, as the developer chooses.
- Provide test data to exercise the application.

1.2. Constraints

- The toy robot must not fall off the table during movement. This also includes the initial placement of the toy robot.
- Any move that would cause the robot to fall must be ignored.

1.3. Example Input and Output

Example a

```
PLACE 0,0,NORTH
MOVE
REPORT
```

Expected output:

```
0,1,NORTH
```

Example b

```
PLACE 0,0,NORTH
LEFT
REPORT
```

Expected output:

```
0,0,WEST
```

Example c

```
PLACE 1,2,EAST
MOVE
MOVE
LEFT
MOVE
REPORT
```

Expected output

```
3,3,NORTH
```

1.4. Deliverables

The Ruby, JavaScript or Java source files, the test data and any test code.

It is not required to provide any graphical output showing the movement of the toy robot.

2 Solution Design

2.1 Design

The Robot Board game solution has been kept to minimal implementation without any graphical representation and defined in a self-explanatory package and class structure.

The Application has been split into 5 different packages based on the functionality.

a) com.myarcade.robotgame.gamesetup - In this package, Board edges have been defined, robot is initiated and position set with location object (which contains x and y coordinates) and direction (north, east, west, south) object. Robot State is also set via IRobot and IOperation interfaces.
b) com.myarcade.robotgame.interfaces – All interfaces specific to robot and operations are defined in this package
c) com.myarcade.robotgame.instructions – The classes defined in this package reads input instructions either from flat files or stdin. Then it converts instructions into game specific operations implementing IRobot and IOperation interfaces and apply to RobotState
d) com.myarcade.robotgame.operations – The classes defined in this package perform various operations like right, move, left, place and report
e) com.myarcade.robotgame – Main class

2.2 Decisions

- JUNIT version 4.12 has been used for unit testing instead of JUNIT Jupiter API (5.1) because of the known eclipse bug (below). This decision was due to time constraint in debugging the issue.
  https://bugs.eclipse.org/bugs/show_bug.cgi?id=538956

- Interfaces have been combined into single package because, as an extension to future, new default methods can be easily incorporated inside interfaces without impacting any abstract classes that extend them or classes that implement them.

- Unit tests have been covered to test Robot initialisation, Read Instructions, Execute Instructions and Operations like right, left, move, report

- Unit tests also covers valid and invalid robot moves

2.3 Deliverables

The application has been uploaded to GitHub and can be downloaded at below link
https://github.com/syamarthi/robot-game

I have also included zip version of the application and sent via email.

2.4 Build

- Download application code from github or download zip file from email
- Unzip to a location
- Open command prompt
- Go to project root where pom.xml file exists.
- Use '***mvn clean install'*** to build package (jar) in ***'./target'*** folder

2.5 How to Run

Command Line:

1) Java –jar robotgame-<version>.jar [instructions file]

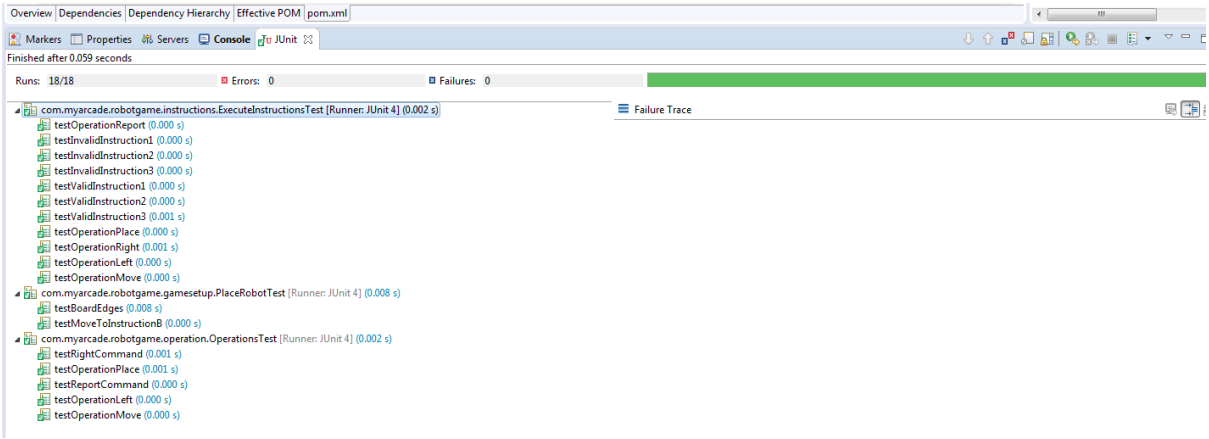
Note: The Robot game can be invoked by either instruction file as an input argument or empty arguments. When there are no arguments, application reads instructions from stdin.

3 Test Coverage

3.1 Unit test cases

Package	Class	Testcases	Result	Description
com.myarcade.robotgame.instructions	ExecuteInstructions	testInvalidInstruction1()	Pass	The test case tests invalid input with value "PLACE4,2,EAST"
		testInvalidInstruction2()	Pass	The test case tests invalid input with value "PLACE 4, 2,EAST"
		testInvalidInstruction3()	Pass	The test case tests invalid input with value "PLACE 4,2,LUNAR"
		testOperationRight()	Pass	The test case tests RIGHT operation
		testValidInstruction1()	Pass	The test case tests valid input "PLACE 0,0,NORTH"
		testValidInstruction2()	Pass	The test case tests valid input PLACE 2,3,SOUTH
		testValidInstruction3()	Pass	The test case tests valid input PLACE 2,3,SOUTH
		testOperationLeft()	Pass	The test case tests LEFT operation
		testOperationMove()	Pass	The test case tests MOVE operation
		testOperationPlace()	Pass	The test case tests PLACE operation
		testOperationReport()	Pass	The test case tests REPORT operation
com.myarcade.robotgame.gamesetup	PlaceRobot	testBoardEdges()		The test case tests board edges and robot placement
		testMoveToInstruction()		The test case tests robot movements
com.myarcade.robotgame.operation	Operations (a generic test class to test operations)	testOperationLeft()		The test case tests LEFT operation
		testOperationMove()		The test case tests MOVE operation
		testOperationPlace()		The test case tests PLACE operation
		testReportCommand()		The test case tests REPORT operation
		testRightCommand()		The test case tests RIGHT operation

3.2 Test Summary Report



4 Known Issues

In some of the older versions of java runtime, scanner methods that reads input from stdin can take longer time to exit. This causes program not to respond to ‘CNTR + z +enter’ on windows or ‘CNTR + D’ on unix.

Solution: As a future extension to the robot game application, a unique key like ‘##’ or ‘END’ can be given as an input to exit the stdin. This is not implemented in this solution.