|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **LEARNING PROFILE FOR ASSIGNMENT#\_2\_\_\_\_\_ AND QUESTION#\_9\_\_\_\_\_\_\_** | | | | | |
| *Name* | *:* | *Steven Morrissey* | *Due Date* | *:* |  |
| *Student ID* | *:* | *3300222* | *Submission Date* | *:* |  |

**1. Problem Statement:**

Create a 10x10 matrix as a 2D array. Assume that a robot is placed in position [0, 0]. Now randomly generate a move. The move could take the robot to one of the eight possible adjacent slots – {up, down, left, right, left-upcorner, left-down-corner, right-up-corner, and right-down-corner} – these slots are represented by {1, 2, 3, 4, 5, 6, 7, 8}. However, at [0, 0], the robot only has three possible slots to m ove to – right, down, right-down-corner. Create another robot called R2 and place it on [9, 9]. Now randomly generate an integer in the range of [1 to 8]. This first random integer corresponds to a possible move for Robot R1. If the move is valid, then move R1 to its new slot. A move is invalid if it takes the robot out of bounds of the [10x10] matrix. If the move is invalid, then keep generating random integers until a valid move is found. Repeat this procedure for the second Robot R2. If both R1 and R2 are in the same slot, then stop, print the final slot, print the sequence of random numbers that led R1 to this slot, and the print the sequence of random numbers that led R2 to the same slot. Implement this program with a Robot class and a MovingRobot subclass.

**2. Description of the Code:**

**Consists of 2 classes: Robot, and MovingRobot which extends the former. MovingRobot contains most of the operational logic, which consists of generating moves for a robot, validating the legality of the move, then moving the robot. Then we check to see if the robot occupies the same space as another robot.**

**3. Errors and Warnings:**

Table 1: List of Errors and Warnings Encountered in the Program

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Errors / Warnings** | **Details** | **How I solved them** |
| 1 | W1 | Created a block inside a functional call instead of using just straight lambda function | Replaced block with lambda function which also made it easier to read |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |

**4. Sample Input and Output:**

**I wrote a few junit tests for this exercise as it can easily be tested by unit tests. They include:**

**-a subset of the validateNextMove paths to be tested**

**-sameSlot test for both happy and negative path**

**-printMoves test with 2 moves in the list (none more needed)**

**-move() test that verifies after a move is performed that the coordinates have been updated**

**5. Discussion:**

One part I was unsure about was printMoves(). The method is called “print moves”, but it has a return value of String… which made me think ok, well dont “print” the moves, just return them as a string and System.out.println them inside of main() when needed. It also made me think that it would be easier that way as you can’t identify which robot is which from the object; so identifying them inside main() and then just appending the returned moves list after seemed logical.