PROJECT ROBOT MOTION

Task 1: Development and Verification Report

COEN 6761 - Software Testing and Validation

Team Code Blooded

Rajat Rajat - 40160245
Sharul Dhiman - 40195730
Rohan Kodavalla - 40196377
Rishi Murugesan Gopalakrishnan - 40200594

Department of Electrical and Computer Engineering

Gina Cody School of Engineering and Computer Science

Winter 2023

Introduction

This project focuses on developing an application that simulates a robot which can move around an NxN floor. The robot has certain functionalities. The Robot can move around the floor based on the commands received from the user. The Robot holds the pen in two positions, up and down. When the pen is down the robot will trace the path when it moves around the floor and when the pen is up the robot moves freely without tracing anything. Initially, the robot's position is [0,0] of the floor, the robot's pen will be up and the robot will be facing north.
The following are the commands that the robot will respond to.

Command	**Use**
I n | i n	Initializes the n x n array floor, n >0. This command will also set the robot's initial position to (0,0) and will set the pen's position to up and the robot's direction as north.
U | u	Pen Up
D | d	Pen Down
R | r	Turn Right
L | l	Turn Left
M s | m s	This command is to make the robot move forward 's' spaces where 's' is a non-negative number.
P | p	Prints the NxN array and displays the indices in the console.
C | c	This command will print the current position of the pen and whether it is up or down and which direction it is facing.
Q | q	Terminate the program

Github URL

https://github.com/Rishi-M-G/COEN6761-Code-Blooded

Requirements

The requirements for the robot simulation project are listed below, each with a unique identifier:

- Initialize the system (R1)

The program should be able to initialize the system with a specified size of the floor array, N. The array values should be set to zeros and the robot should be positioned at [0, 0], with the pen up, and facing north.

- Change pen position (R2)

The program should be able to change the position of the pen from up to down or from down to up.

- Turn the robot (R3)

The program should be able to turn the robot in either direction (left or right) depending on the user's command.

- Move the robot (R4)

The program should be able to move the robot forward a given number of spaces, s. The spaces should be non-negative integers. If the pen is down while the robot moves, the appropriate elements of the floor array should be set to 1.

- Print the floor (R5)

The program should be able to display the N-by-N array, where wherever there is a 1 in the array, an asterisk should be displayed; wherever there is a zero, a blank should be displayed.

- Current position (R6)

The program should be able to display the current state of the robot including the position of the pen and whether it is up or down and its facing direction.

- Display asterisks and blanks (R7)

The program must display the floor as the N*N array with the asterisk where the robot has traced and blanks where it has not.

- End program (R8)

The program should be able to stop running when the user gives the "Q" or "q" command.

Screenshots

```
public void initializeArrayFloor(int n) {
    // Initializing the array
    N = n;
    if( N < 0 )
    {
        System.out.println("Invalid Number. Array Dimension should be a Positive Value");
    }
    else
    {
        floor = new int[N][N];
        penStatus = "up";
        //Initializing array values to 0
        for (int i = 0; i<N; i++)
        {
            for (int j = 0;j<N;j++)
            {
                floor[i][j]= 0;
            }
        }
        //Setting Robot Position
        x = y = 0;
        floor[x][y]=0;
        //Setting Robot Direction
        Direction = "north";
    }
}
```

Figure 1. Function for Initializing the Floor and Robot - R1

```
public void displayFloor() {
    for(int i = N-1;i>=0;i--)
    {
        System.out.println();
        System.out.print(i+"  ");
        for (int j = 0;j<N;j++)
        {
            if(floor[j][i] == 1)
                System.out.print("*  ");
            else
                System.out.print("   ");
        }
        System.out.println();
    }
    System.out.println(" ");
    System.out.print("   ");
    for(int k = 0;k<N;k++)
    {
        System.out.print(k+"  ");
    }
}
```

Figure 2. Function for displaying the floor - R5 and R8

```
public void currentPosition() {
    System.out.println("Position: "+x + "," + y + " - Pen: " + penStatus+" - Facing: "+Direction);
}
```

Figure 3. Function for displaying the current position of the Robot - R6

```
public void penUp() {
    penStatus = "up";
}


public void penDown() {
    penStatus = "down";
}
```

Figure 4. Functions for changing the pen's position - R2

```
public void turnRight() {
    if(Direction == "north")
    {
        Direction = "east";
    }
    else if(Direction == "south")
    {
        Direction = "west";
    }
    else if(Direction == "west")
    {
        Direction = "north";
    }
    else
    {
        Direction = "south";
    }
}
```

Figure 5. Function to make robot turn right - R3

```
public void turnLeft() {
    if(Direction == "north")
    {
        Direction = "west";
    }
    else if(Direction == "south")
    {
        Direction = "east";
    }
    else if(Direction == "west")
    {
        Direction = "south";
    }
    else
    {
        Direction = "north";
    }
}
```

Figure 6. Function to make robot turn left - R3

```
public void quit() {
    // Program Termination
    System.out.println("ROBOT MOTION TERMINATED");
    System.exit(0);
}
```

Figure 7. Function for terminating the program - R7

```
public boolean moveForward(int S) {
    s = S;
    if((s<0 || s>=N) == true)
    {
        System.out.println("'s' should be a positive number and should be within the Floor");
        return false;
    }
    else
    {
        if(Direction == "north" && (y+s < N))
        {
            int m = y;
            y = y + s; // Setting new coordinates
            if(penStatus == "down")
            {
                for(int l=m;l<=y;l++)
                {
                    floor[x][l] = 1;
                }
            }
        }
        else if(Direction == "east" && (x+s < N))
        {
            int m = x;
            x = x + s;
            if(penStatus == "down")
            {
                for(int l=m;l<=x;l++)
                {
                    floor[l][y] = 1;
                }
            }
        }

            else if(Direction == "west" && (x-s > 0))
            {
                int m = x;
                x = x - s;
                if(penStatus == "down")
                {
                    for(int l=m;l<=y;l++)
                    {
                        floor[l][y] = 1;
                    }
                }
            }
            else if(Direction == "south" && (y-s > 0))
            {
                int m = y;
                y = y - s;
                if(penStatus == "down")
                {
                    for(int l=m;l<=y;l++)
                    {
                        floor[x][l] = 1;
                    }
                }
            }
            else {
                System.out.println("Robot Cannot leave the floor");
                return false;
            }
        }
        return true;
    }
}
```

Figure 8. Function for moving the robot - R4

Test Cases

- **Test Case ID** : 1
  Tester's Name : Rishi Murugesan Gopalakrishnan
  Date : 02/11/2023
  Test Type : Unit Testing
  Test Function Name : testTurnRight()
  Test Input Data: 'R' (or) 'r'
  Test Case Description : Program to test turning right functionality of the robot.
  Expected Output: The robot will turn right

```
@Test
public void testTurnRight()
{
    robot.Direction = "north";
    robot.turnRight();
    assertEquals("east",robot.Direction);

    robot.Direction = "south";
    robot.turnRight();
    assertEquals("west",robot.Direction);

    robot.Direction = "west";
    robot.turnRight();
    assertEquals("north",robot.Direction);

    robot.Direction = "east";
    robot.turnRight();
    assertEquals("south",robot.Direction);
}
```

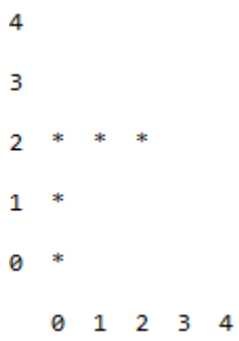
- **Test Case ID** : 2
  Tester's Name : Rishi Murugesan Gopalakrishnan
  Date : 02/11/2023
  Test Type : Unit Testing
  Test Function Name : testTurnLeft()
  Test Input Data:'L' (or) 'l'
  Test Case Description : Program to test turning left functionality of the robot.
  Expected Output: The Robot will turn to its left

```
@Test
public void testTurnLeft()
{
    robot.Direction = "north";
    robot.turnLeft();
    assertEquals("west",robot.Direction);

    robot.Direction = "south";
    robot.turnLeft();
    assertEquals("east",robot.Direction);

    robot.Direction = "west";
    robot.turnLeft();
    assertEquals("south",robot.Direction);

    robot.Direction = "east";
    robot.turnLeft();
    assertEquals("north",robot.Direction);
}
```

- **Test Case ID** : 3
  Tester's Name : Rishi Murugesan Gopalakrishnan
  Date : 02/11/2023
  Test Type : Unit Testing
  Test Function Name : testFloorValues()
  Test Input Data: "I 5", 'd',"m 2",'r', "m 2"
  Test Case Description : Program to test the whole floor to check whether the robot is tracing the floor when the pen is down by assigning 1 and 0 in the array locations.
  Expected Output:
    - 1 in array locations to where the robot has moved
    - 0 where the robot has not moved

```
@Test
public void testFloorValues() {
    robot.initializeArrayFloor(n);
    robot.penDown();
    robot.moveForward(s);
    robot.turnRight();
    robot.moveForward(s);

     assertEquals(1,robot.floor[0][0]);
     assertEquals(1,robot.floor[0][1]);
     assertEquals(1,robot.floor[0][2]);
     assertEquals(1,robot.floor[1][2]);
     assertEquals(1,robot.floor[2][2]);
     assertEquals(0,robot.floor[0][3]);
     assertEquals(0,robot.floor[0][4]);
     assertEquals(0,robot.floor[1][0]);
     assertEquals(1,robot.floor[1][2]);
     assertEquals(0,robot.floor[1][3]);
     assertEquals(0,robot.floor[1][4]);
     assertEquals(0,robot.floor[2][0]);
     assertEquals(0,robot.floor[2][1]);
     assertEquals(0,robot.floor[2][3]);
     assertEquals(0,robot.floor[2][4]);
     assertEquals(0,robot.floor[3][0]);
     assertEquals(0,robot.floor[3][1]);
     assertEquals(0,robot.floor[3][2]);
     assertEquals(0,robot.floor[3][3]);
     assertEquals(0,robot.floor[3][4]);
     assertEquals(0,robot.floor[4][0]);
     assertEquals(0,robot.floor[4][1]);
     assertEquals(0,robot.floor[4][3]);
     assertEquals(0,robot.floor[4][4]);
}
```

- **Test Case ID** : 4
  Tester's Name : Rishi Murugesan Gopalakrishnan
  Date : 02/11/2023
  Test Type : Unit Testing
  Test Function Name : testFloorOutput()
  Test Input Data: "I 5", 'd',"m 2",'r', "m 2"
  Test Case Description : Program to test whether * (asterix) and " "(a blank space)
  Expected Output:




```
@Test
public void testFloorOutput() {
    robot.initializeArrayFloor(n);
    robot.penDown();
    robot.moveForward(s);
    robot.turnRight();
    robot.moveForward(s);
    ByteArrayOutputStream output = new ByteArrayOutputStream();
    System.setOut(new PrintStream(output));

    robot.displayFloor();
    String expectedOutput = "4                \n" +
                            "\n3                \n" +
                            "\n2  *  *  *       \n" +
                            "\n1  *             \n" +
                            "\n0  *             \n \n" +
                            "   0  1  2  3  4";

    //assertEquals("\n4                \n3                \n2  *  *  *       \n1  *             \n0  *
    assertEquals(expectedOutput.trim().replaceAll("\\s",""),output.toString().trim().replaceAll("\\s",""));
}
```

- **Test Case ID** : 5
  Tester's Name : Rohan Kodavalla
  Date : 02/11/2023
  Test Type : Unit Testing
  Test Function Name : testPenDownAndTracing()
  Test Input Data: "I 5", 'd',"m 2"
  Test Case Description : Program to test whether the robot is tracing the floor when the pen is down.
  Expected Output: Position: 0,2 - Pen: down - Facing: north

```
@Test
public void testPenDownAndTracing() {
    robot.initializeArrayFloor(n);
    robot.penDown();
    robot.moveForward(s);
    assertEquals(1,robot.floor[0][2]);
}
```

- **Test Case ID** : 6
  Tester's Name : Rohan Kodavalla
  Date : 02/11/2023
  Test Type : Unit Testing
  Test Function Name : testMovingOutOfBounds()
  Test Input Data: "I 5"
  Test Case Description : Program to test whether the robot is tracing only inside the floor when the pen is down and does not go out of boundaries.
  Expected Output: 's' should be a positive number and should be within the Floor (if False)

```
@Test
public void testMovingOutOfBounds() {
    robot.initializeArrayFloor(n);
    robot.penDown();
    assertEquals(true, robot.moveForward(n-1)); /
    assertEquals(false, robot.moveForward(n)); //
}
```

- **Test Case ID** : 7
  Tester's Name : Rishi Murugesan Gopalakrishnan
  Date : 02/11/2023
  Test Type : Unit Testing
  Test Function Name : testCurrentPosition()
  Test Input Data: "I 5"
  Test Case Description : Program to test whether the program displays the current output of the robot
  Expected Output: Position: 0,0 - Pen: up - Facing: north

```
@Test
public void testCurrentPosition() {
    ByteArrayOutputStream output = new ByteArrayOutputStream();
    System.setOut(new PrintStream(output));

    robot.initializeArrayFloor(n);

    robot.currentPosition();
    assertEquals("Position: 0,0 - Pen: up - Facing: north", output.toString().trim());
}
```

- **Test Case ID** : 8
  Tester's Name : Rajat Rajat
  Date : 02/11/2023
  Test Type : Unit Testing
  Test Function Name : testPenUp()
  Test Input Data: "I 5", 'd', 'u'
  Test Case Description : Program to test whether the program changes pen's position to "up" when command 'u' is entered
  Expected Output: up

```
@Test
public void testPenUp() {
    robot.penUp();
    assertEquals("up",robot.penStatus);
}
```

- **Test Case ID** : 9
  Tester's Name : Rajat Rajat
  Date : 02/11/2023
  Test Type : Unit Testing
  Test Function Name : testPenDown()
  Test Input Data: "I 5", 'd'
  Test Case Description : Program to test whether the program changes pen's position to "down" when command 'u' is entered
  Expected Output: down

```
@Test
public void testPenDown() {
    robot.penDown();
    assertEquals("down",robot.penStatus);
}
```

- **Test Case ID** : 10
  Tester's Name : Sharul Dhiman
  Date : 02/11/2023
  Test Type : Unit Testing
  Test Function Name : testInitializeSystem()
  Test Input Data: "I 5"
  Test Case Description : Program to test whether the program initializes the floor and the robot when command i is used
  Expected Output: up, 0, Position: 0,0 - Pen: up - Facing: north

```
@Test
public void testInitializeSystem() {
    ByteArrayOutputStream output = new ByteArrayOutputStream();
    System.setOut(new PrintStream(output));

    robot.initializeArrayFloor(n);
    assertEquals("up",robot.penStatus);
    assertEquals(0, robot.floor[0][0]); //return true if y,x values = [x],[y]
    robot.currentPosition();
    assertEquals("Position: 0,0 - Pen: up - Facing: north", output.toString().trim());
}
```

- **Test Case ID** : 11
  Tester's Name : Sharul Dhiman
  Date : 02/11/2023
  Test Type : Unit Testing
  Test Function Name : testMoveForward()
  Test Input Data: "I 5", 'd',"m 2"
  Test Case Description : Program to test whether the move forward functionality works perfectly
  Expected Output: Position: 0,2 - Pen: down - Facing: north , 1

```
@Test
public void testMoveForward() {
    ByteArrayOutputStream output = new ByteArrayOutputStream();
    System.setOut(new PrintStream(output));

    robot.initializeArrayFloor(n);
    robot.penDown();
    robot.moveForward(s);
    robot.currentPosition();
    assertEquals("Position: 0,2 - Pen: down - Facing: north", output.toString().trim());
    assertEquals(1,robot.floor[0][2]);
}
```

Below is the table mapping requirements and unit test cases

Requirement ID	**Requirement**	**Test Case Scenario**	**Unit Test Case Function**	**Execution Result (Pass/Fail)**
R1	Initialize the system	Test that the floor array is correctly initialized to zeros, that the robot is positioned at [0, 0], with the pen up, and facing north.	Test Case ID : 10	Pass
R2	Change pen position	Test that the pen can be changed from up to down and from down to up.	Test Case ID : 8 Test Case ID : 9	Pass
R3	Turn the robot	Test that the robot can be turned in either direction (left or right) depending on the user's command.	Test Case ID : 1 Test Case ID : 2	Pass
R4	Move the robot	Test that the robot can move forward a given	Test Case ID : 11	Pass

		number of spaces and that the appropriate elements of the floor array are set to 1 when the pen is down while the robot moves.		
R5	Print the floor	Test that the N-by-N array is displayed correctly with asterisks where there is a 1 in the array and blanks where there is a zero.	Test Case ID : 3 Test Case ID : 5	Pass
R6	Display current position	Test that the current position of the pen, whether it is up or down, and its facing direction can be displayed correctly.	Test Case ID : 7	Pass
R7	Display asterisks and blanks	Test that the program can display * and blank space to show the path that the robot has traced	Test Case ID : 4	Pass
R8	End program	Test that the program stops running when the user gives the "Q" or "q" command.	Test Case ID : 12	Pass

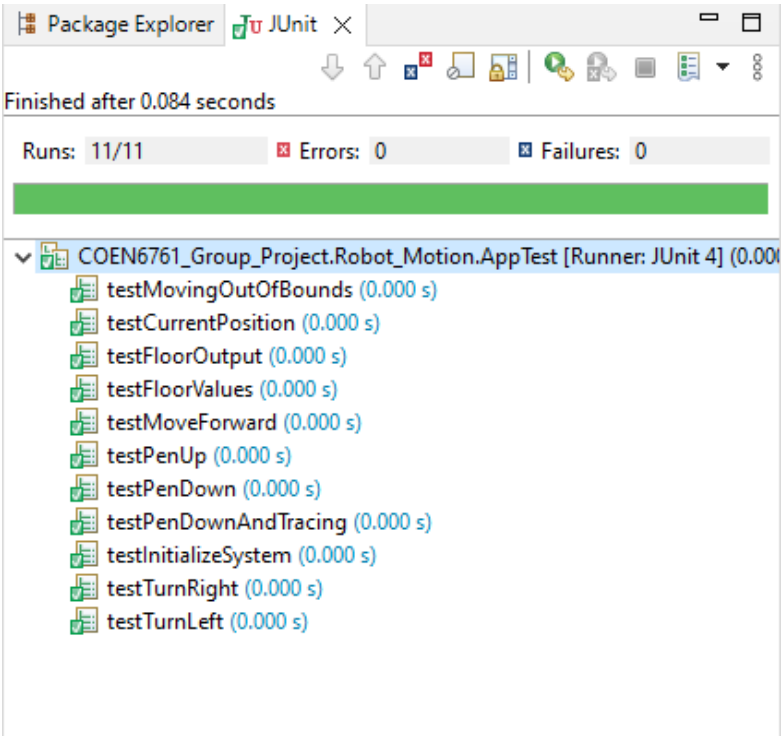


Figure 9. JUnit Test Result Window

Conclusion

In conclusion, the robot simulation project requires the development of a Java program that can simulate the movement of a robot as it moves through a floor represented by an N-by-N array. The program should be able to initialize the system, change the pen position, turn the robot, move the robot, display the floor, display the current position of the robot and pen, and end the program. These requirements provide a comprehensive outline of the functionality expected in the final product.