Testing Report: MapDraw class testing report

Prepared by Yifei Pei , ID : 1611648

1 Introduction

This testing session focuses on an important part of the GUI, which draws the map on the GUI. The class to do this functionality of GUI is callled MapDraw. In MapDraw, a map data structure that has already been read from the textual formatted .XML data will be implemented in the Map display panel of the GUI. The class itself extends JPanel in Java code so it can manipulate features of JPanel to add the visualised map on the map display panel on GUI. To display the representations of different strucutres inside the map data structure, the MapDraw class implements paintComponent function to activate the painting machanism provided by Java API, Graphics2D, visualising the map into painted components in the panel. To simplify and beautify the painting mechanism, the unexplored zone, the closure, the obstacle, the road, the intersection, and the robot are represented by images that acquired by the group. The the beginning of the class, it will try to read all the images in the class. If any exception occurs in this process, it will be reported to the user.

The purpose of this unit test is to test if the MapDraw class can correctly turn the map data structure into visualised components on GUI. Here the correctness has two major attributes, the correct position of the representation, thus, the coordinates for the representation, and the correct image painted for the representation. By nature, the representations of different components in the map data structure have been splited into eight parts, the Boundary and Size of the map, the Roads, the Intersections of the roads, the Obstacles, the Closures, the Disaster zones, the Unexplored zone, and the Robot. This testing session is going to propose the test for the correctness of these eight components of the map visualisation.

To test the correctness of the components that organise the whole visualisation of the map data structure, both white box and black box testing are used. Due to the usage of GUI, the main purpose of the MapDraw class is to show the visualised map to the user, so it is important for the tester to give a black box test to check the mechanism of MapDraw in a user's perspective. Furthermore, the tester has freely access to the source code of this class. A white box that has the depth into the source code is also necessary to make the process smooth, Especially when the positions of different component cannot be identified by human eyes in a very precise context. However, due to the design of the MapDraw class, there are no explicit variables in the class that can be tested by using JUnit. All the information of positions are parameters that are transfered into Graphics2D to draw the components, which are unable to be accessed by external classes. Therefore, this testing session decided to not use JUnit tests for the Mapdraw class but to write ''System.out.print" functions inside the class to explicitly print out the data required for the test. The ''print out" method is not a perfect way to test variables communicated inside the class, but due to the constraints of this testing session, this method can easily show the effects inside the class in a very short time. Developers are also able to modify the source code depending on the information provided by ''print out" functions.

2 Testing Cases

The testing cases of this testing session read an example map provided by the group named ‘’map5.xml”. The steps for the testing cases for the eight components are: 1. write down the ‘’print out” functions for the testing part so it can show the information later in the implementation 2. Open the GUI 3. Read the map 4. check the image of different components: whether they are represented by correct images 5. check the information printed out: if the positions of the components are correct.

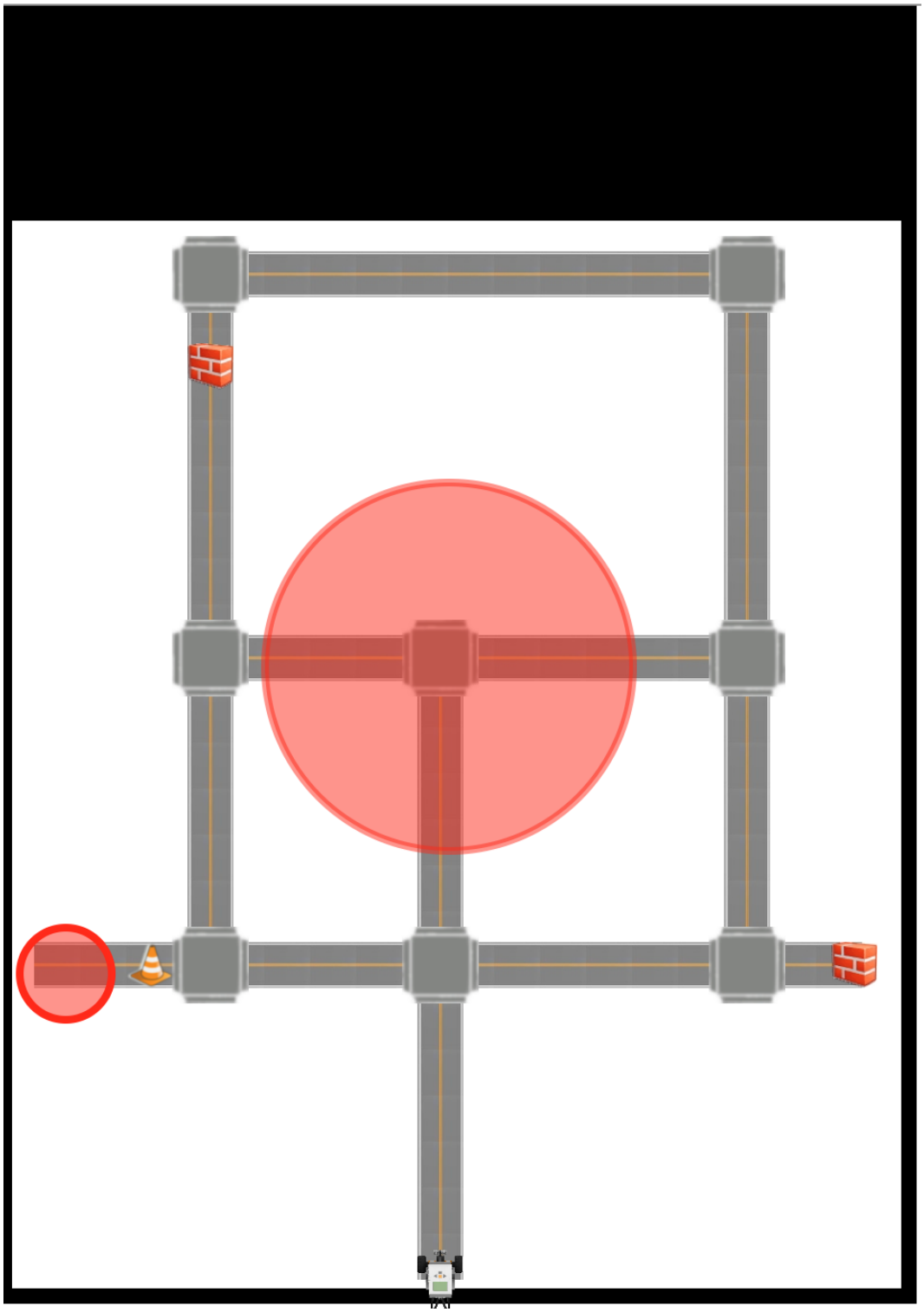
2.1 Test Case: The map boundary and size

Numerical data in white box testing: The set size for the map displayed on GUI is 900x690. The MapDraw class provided a scale mechanism for the painting function to display the map of any size to be 900x690 large. If the GUI size is adjusted by the user in using, the size of map will also fit the new scale. The size of the map showed in map5.xml is 59x84. The information that it prints out when the GUI read the map is:

```
scale = 8
The width of the actual map is 59
The Height of the actual map is 84
The width of the drawn map is 897
The Height of the drawn map is 646
```

It seems the width and height of the map is not quite precise, but the error is in acceptable level.

The visualised component in black box testing: The whole map looks like this.



2.2 Test Case: The roads

Numerical data in white box testing: There are 6 roads prepared in map5.xml. They are all presented by two points, which are [(29,1)(29,41)], [(2,21)(56,21)], [(14,21)(14,66)], [(49,21)(49,66)], [(14,66)(49,66)], [(14,41)(49,41)]. The information that it prints out when the GUI read the map is:

```
the Vertical roads: the x is 29, the y is 1, the length is 40
the Horizontal roads: the x is 2, the y is 21, the length is 54
the Vertical roads: the x is 14, the y is 21, the length is 45
the Vertical roads: the x is 49, the y is 21, the length is 45
the Horizontal roads: the x is 14, the y is 66, the length is 35
the Horizontal roads: the x is 14, the y is 41, the length is 35
```

The MapDraw only read the head of the roads and calculate the length of the road to draw the roads on map panel. So far, all the information provided is precise and correct. The calculating method of length is enclosed in the map data structure. The tests for the data structure will show if the calculation method is correct. If the length calculation method has some bugs, the images here will also be displayed incorrectly.

The visualised component in black box testing: The visualisation of the roads can be seen in the picture of last section.

2.3 Test Case: The intersections

Numerical data in white box testing: There are 8 intersections prepared in map5.xml. They are all presented by points, which are (29,21), (14,21), (14,41), (29,41), (49,21), (49,41), (14,66), (49,66). The information that it prints out when the GUI read the map is:

```
The intersections:x is 29.0, y is 21.0
The intersections:x is 14.0, y is 21.0
The intersections:x is 14.0, y is 41.0
The intersections:x is 29.0, y is 41.0
The intersections:x is 49.0, y is 21.0
The intersections:x is 49.0, y is 41.0
The intersections:x is 14.0, y is 66.0
The intersections:x is 49.0, y is 66.0
```

The positions are precise.

The visualised component in black box testing: The visualisation of the intersections can be seen in the picture of the first section.

2.4 Test Case: obstacles

Numerical data in white box testing: There are 2 obstacles prepared in map5.xml. They are both presented by points, which are (56,21) and (14,60). The information that it prints out when the GUI read the map is:

```
The obstacles: x is 56.0, y is 21.0
The obstacles: x is 14.0, y is 60.0
```

The positions are precise.

The visualised component in black box testing: The visualisation of the obstacles can be seen in the picture of the first section.

2.5 Test Case: closures

Numerical data in white box testing: There is only one closure prepared in map5.xml. It is presented by point, which is (10,21). The information that it prints out when the GUI read the map is:

```
The closures: x is 10.0, y is 21.0
```

The positions are precise.

The visualised component in black box testing: The visualisation of the closures can be seen in the picture of the first section.

2.6 Test Case: disaster zones

Numerical data in white box testing: There are two disaster zones prepared in map5.xml. It is presented by points and radius, which are (4,21 : 3) and (29,41 : 12). The information that it prints out when the GUI read the map is:

```
The disaster zones: x is 4.0, y is 21.0, and the radius is 3
The disaster zones: x is 29.0, y is 41.0, and the radius is 12
```

The positions are precise.

The visualised component in black box testing: The visualisation of the disaster zones can be seen in the picture of the first section.

2.7 Test Case: unexplored zones

Numerical data in white box testing: There is one unexplored zone prepared in map5.xml. The starting point of it is (0,70) and the width is 59 while the height is 14. The information that it prints out when the GUI read the map is:

```
The unexplored zones: x is 0.0, y is 70.0, the width is 27, the height is 14
The unexplored zones: x is 27.0, y is 70.0, the width is 27, the height is 14
The unexplored zones: x is 54.0, y is 70.0, the width is 5, the height is 14
```

The unexplored zones are splited into small pieces in the MapDraw so when the robot is exploring the map, it can be explored by small chunks but not the whole unexplored zone is unveiled. From the results, the small pieces can be organised to be the large chunk of unexplored zone and the positions are precise.

The visualised component in black box testing: The visualisation of the unexplored zones can be seen in the picture of the first section.

2.8 Test Case: robot

Numerical data in white box testing: The robot status prepared in map5.xml are that: it is facing 90 angles to the left edge of the whole map and its position is (29,1). The information that it prints out when the GUI read the map is:

```
The robot is facing 90, and its position is x: 29.0, y: 1.0
```

The position is precise.

The visualised component in black box testing: The visualisation of the robot can be seen in the picture of the first section.

3 Conclusion

From the results presented above, the MapDraw can precisely read all the map information and draw a correct visualised representation of the map data structure. Only one .xml file is not enough for this test. The test also tested another xml file called map4.xml. The correctness of reading that file was the same as this test. Due to the time and work constraints, the test of another file will not be presented in detail here.