THE UNIVERSITY of ADELAIDE

Software Engineering and Project

Archaeology Robot Test Report

Jun Chen 1206265	

October 22, 2012

Contents

Chapter 1

Introduction

In this test plan, I am doing some tests about the map structure. The tests I make will test the functionalities of map. The functionalities include:

1. Find any pixels in the map.
2. set a NoGoZone Circle.
3. get the length of the map.
4. get the width of the map.
5. set the length of the map.
6. set the width of the map.
7. get the ID of the map.
8. set the ID of the map.
9. get the Start xPos.
10. get the Start yPos.
11. set the Start xPos.
12. set the Start yPos.
13. get the current pixel.
14. get the start pixel.
15. get the goal pixel.
16. check two maps are the same or not.
17. create a clone of the map.

There are many methods in the Map class, each of them represents a function of the map. The tests will ensure all the methods is working properly.

Chapter 2

Test Cases

2.1 Testing Environment

Testing tool:
Eclipse SDK for Java Developers: Indigo Version: 3.7.0
with JUnit 3.11
Java version:
Java(TM) SE Runtime Environment (build 1.7.0 07-b11)
Operating System:
Windows 7

2.1.1 Test 1

Test Case Name : findPixeltest

Type : JUnit Test

Rationale :findPixel is used to find a specifical pixel on the map. Its the basic method of this map, so we should ensure it works properly.

Test Objective : Test the findPixel method of the map. The map should be able to find the specifical pixel on the map.

Test method :

1. Create a new map.
2. Find a pixel on the map and then store it into p1.
3. Check the xPos and yPos of p1, see if its value is equal to the one map finds.

Pass or Fail : Pass.

2.1.2 Test 2

Test Case Name : setNoGoZoneCircletest

Type : JUnit Test

Rationale :set No-go zone is a requirement of this project so we need to test it to ensure it works.

Test Objective : Test and figure out the function pass or not.

Test method :

1. Create a new map.
2. set a no-go zone.
3. set a pixel in the no-go zone
4. check the pixel's value. It should contain no-go zone value if it is in no-go zone.

Pass or Fail : Pass.

2.1.3 Test 3

Test Case Name : getLengthtest

Type : JUnit Test

Rationale :Testing getLengthtest method is useful for other methods which will use this method.

Test Objective : Test and figure out the function pass or not.

Test method :

1. Create a new map.
2. get length of the map and store it in a variable x.
3. check the variable value is same as map's length or not

Pass or Fail : Pass.

2.1.4 Test 4

Test Case Name : getWidthtest

Type : JUnit Test

Rationale :Testing getWidthtest method is useful for other methods which will use this method.

Test Objective : Test and figure out the function pass or not.

Test method :

1. Create a new map.
2. get width of the map and store it in a variable x.
3. check the variable value is same as map's width or not

Pass or Fail : Pass.

2.1.5 Test 5

Test Case Name : setLengthtest

Type : JUnit Test

Rationale :Testing setLengthtest method is useful for other methods which will use this method. With setLength method we can create the map with the size we want.

Test Objective : Test and figure out the function pass or not.

Test method :

1. Create a new map.
2. set a new length value of this new map
3. get length of the map and store it in a variable x.
4. check the variable value is same as map's new length or not

Pass or Fail : Pass.

2.1.6 Test 6

Test Case Name : setWidththtest

Type : JUnit Test

Rationale :Testing setWidththtest method is useful for other methods which will use this method. With ssetWidththtest method we can create the map with the size we want.

Test Objective : Test and figure out the function pass or not.

Test method :

1. Create a new map.
2. set a new width value of this new map
3. get width of the map and store it in a variable x.
4. check the variable value is same as map's new width or not

Pass or Fail : Pass.

2.1.7 Test 7

Test Case Name : getIDtest

Type : JUnit Test

Rationale :Testing setWidththtest method is useful for other methods which will use this method. With getID method we can get the map ID.

Test Objective : Test and figure out the function pass or not.

Test method :

1. Create a new map.
2. get ID of the map and store it in a variable x.
3. check the variable value is same as map's ID or not

Pass or Fail : Pass.

2.1.8 Test 8

Test Case Name : setIDtest

Type : JUnit Test

Rationale :Testing setWidththtest method is useful for other methods which will use this method. With setID method we can set the map ID when we call the method.

Test Objective : Test and figure out the function pass or not.

Test method :

1. Create a new map.
2. set a new ID for this map.
3. get new ID of the map and store it in a variable x.
4. check the variable value is same as map's new ID or not

Pass or Fail : Pass.

The other test cases for getXXX methods and setXXX methods are same as above. They test the functionalities of the map.

2.1.9 Test 9

Test Case Name : compareTotest

Type : JUnit Test

Rationale :compareTo method is used to check if two maps are same or not.

Test Objective : Test and figure out the function pass or not.

Test method :

1. Create three new maps, two of them are same(map1, map2), one of them is different(map3).
2. compare map1 to map2 and then store the boolean value into b1.
3. compare map1 to map3 and then store the boolean value into b2.
4. check the variable b1 and b2 value is true or false.

Pass or Fail : Pass.

2.1.10 Test 10

Test Case Name : clonetest

Type : JUnit Test

Rationale :clone method is used to make a copy of a map.

Test Objective : Test and figure out the function pass or not.

Test method :

1. Create a new map.
2. call clone method and store the new map to map2.
3. compare map to map2 and then store the boolean value into b.
4. check the variable b value is true or false.

Pass or Fail : Pass.

Appendix A

JUnit Test Cases

```
/**
 * MapStructure Map class Unit Test
 * author Jun Chen (1206265)
 */
package Tests;

import static org.junit.Assert.*;
import java.util.ArrayList;
import MapStructure.Map;
import MapStructure.Pixel;
import MapStructure.RobotStatus;
import org.junit.Test;

public class MapStructureTest {

        Map map;
//      Pixel p1 = new Pixel(5, 9, 0, map);

@Test
public void findPixeltest(){
        Map map = new MapStructure.Map("MapNo1", 10, 10, 0, 0, 1);
        Pixel p1 = map.findPixel(5, 9);
        int x = p1.getxPos();
        int y = p1.getyPos();
        assertTrue(x==5);
        assertTrue(y==9);
}

//@Test
//public void setNoGoZonetest(){
//      Map map = new MapStructure.Map("MapNo1", 10, 10, 0, 0, 1);
//      map.setNoGoZone(1, 5, 5, 5);
//      Pixel p = map.findPixel(2, 3); //it should be in the no-go zone
//      Pixel p2 = map.findPixel(4, 4); //it should be in the no-go zone
//      Pixel p3 = map.findPixel(6, 3); //it should not in the no-go zone
//      int x = p.getValue();
//        int y = p2.getValue();
//        int z = p3.getValue();
//        assertTrue(x == Integer.MAX_VALUE);
//        assertTrue(y == Integer.MAX_VALUE);
//        assertTrue(z == 0 || z == 1 || z == 2 || z == 999 || z == 500);
//}

//@Test
//public void setNoGoZonetest2(){
//        Map map = new MapStructure.Map("MapNo1", 10, 10, 0, 0, 1);
//        map.setNoGoZone(1, 5, 5, 5);
//        Pixel p = map.findPixel(5, 5); // in the no-go zone (border)
//        Pixel p2 = map.findPixel(1, 1); // in the no-go zone (border)
//        Pixel p3 = map.findPixel(1, 5); //in the no-go zone (border)
//        Pixel p4 = map.findPixel(5, 1); // in the no-go zone (border)
//        int x = p.getValue();
//        int y = p2.getValue();
//        int z = p3.getValue();
//        int r = p4.getValue();
//        assertTrue(x == Integer.MAX_VALUE);
//        assertTrue(y == Integer.MAX_VALUE);
//        assertTrue(z == Integer.MAX_VALUE);
//        assertTrue(r == Integer.MAX_VALUE);
//}

@Test
public void setNoGoZoneCircletest(){
        Map map = new MapStructure.Map("MapNo1", 10, 10, 0, 0, 1);
        map.setNoGoZoneCircle(5, 5, 2);
        Pixel p = map.findPixel(6, 6); //it should be in the no-go zone
        Pixel p2 = map.findPixel(4, 4); //it should be in the no-go zone
        int x = p.getValue();
        int y = p2.getValue();
        assertTrue(x == Integer.MAX_VALUE);
        assertTrue(y == Integer.MAX_VALUE);
}

@Test
public void getLengthtest(){
        Map map = new MapStructure.Map("MapNo1", 10, 10, 0, 0, 1);
        int x = map.getLength();
        assertTrue(x == 10);
}

@Test
public void getWidthtest(){
        Map map = new MapStructure.Map("MapNo1", 10, 10, 0, 0, 1);
        int x = map.getWidth();
        assertTrue(x == 10);
```

```
}

@Test
public void setLengthtest(){
        Map map = new MapStructure.Map("MapNo1", 10, 10, 0, 0, 1);
        map.setLength(8);
        int x = map.getLength();
        assertTrue(x == 8);
}

@Test
public void setWidthtest(){
        Map map = new MapStructure.Map("MapNo1", 10, 10, 0, 0, 1);
        map.setWidth(8);
        int x = map.getWidth();
        assertTrue(x == 8);
}

@Test
public void getIDtest(){
        Map map = new MapStructure.Map("MapNo1", 10, 10, 0, 0, 1);
        String x = map.getId();
        assertTrue(x == "MapNo1");
}

@Test
public void setIDtest(){
        Map map = new MapStructure.Map("MapNo1", 10, 10, 0, 0, 1);
        map.setId("Maplol");
        String x = map.getId();
        assertTrue(x == "Maplol");
}

@Test
public void getStartxPostest(){
        Map map = new MapStructure.Map("MapNo1", 10, 10, 0, 0, 1);
        int x = map.getStartxPos();
        assertTrue(x == 0);
}

@Test
public void getStartyPostest(){
        Map map = new MapStructure.Map("MapNo1", 10, 10, 0, 0, 1);
        int x = map.getStartyPos();
        assertTrue(x == 0);
}

@Test
public void setStartxPostest(){
        Map map = new MapStructure.Map("MapNo1", 10, 10, 0, 0, 1);
        map.setStartxPos(1);
        int x = map.getStartxPos();
        assertTrue(x == 1);
}

@Test
public void setStartyPostest(){
        Map map = new MapStructure.Map("MapNo1", 10, 10, 0, 0, 1);
        map.setStartyPos(2);
        int x = map.getStartyPos();
        assertTrue(x == 2);
}

//@Test
//public void getUnexplorePixelstest(){
//      Map map = new MapStructure.Map("MapNo1", 10, 10, 0, 0, 1);
//      int x = map.getUnexplorePixels();
//      assertTrue(x == 0);
//}

@Test
public void getCurrentPixeltest(){
        Map map = new MapStructure.Map("MapNo1", 10, 10, 0, 0, 1);
        Pixel p = new Pixel(5, 10, 0, map);
        map.setCurrentPixel(p);
        Pixel p2 = map.getCurrentPixel();
        int x = p2.getxPos();
        int y = p2.getyPos();
        assertTrue(x == 5);
        assertTrue(y == 10);
}

@Test
public void getStartPixeltest(){
        Map map = new MapStructure.Map("MapNo1", 10, 10, 0, 0, 1);
        Pixel p = new Pixel(1, 1, 0, map);
        map.setStartPixel(p);
        Pixel p2 = map.getStartPixel();
        int x = p2.getxPos();
        int y = p2.getyPos();
        assertTrue(x == 1);
        assertTrue(y == 1);
}

@Test
public void getGoaltest(){
        Map map = new MapStructure.Map("MapNo1", 10, 10, 0, 0, 1);
        Pixel p = new Pixel(10, 10, 0, map);
        map.setGoal(p);
        Pixel p2 = map.getGoal();
        int x = p2.getxPos();
        int y = p2.getyPos();
        assertTrue(x == 10);
        assertTrue(y == 10);
}

//@Test
//public void outofBoundstest(){
//      Map map = new MapStructure.Map("MapNo1", 10, 10, 0, 0, 1);
//      Pixel p = new Pixel(15, 18, 0, map);
//      Pixel p2 = new Pixel(2, 2, 0,map);
//      Pixel p3 = new Pixel();
//      boolean out = map.outOfBounds(p);
//      boolean out2 = map.outOfBounds(p2);
//      boolean out3 = map.outOfBounds(p3);
//      assertTrue(out);
//      assertFalse(out2);
//      assertFalse(out3);
//}

@Test
public void compareTotest(){
        Map map = new MapStructure.Map("MapNo1", 10, 10, 0, 0, 1);
        Map map2 = new MapStructure.Map("MapNo1", 10, 10, 0, 0, 1);
        Map map3 = new MapStructure.Map("MapNo3", 8, 8, 0, 0, 1);
        boolean b1 = map.compareTo(map2);
        boolean b2 = map.compareTo(map3);
        assertTrue(b1);
        assertFalse(b2);
}

@Test
public void clonetest(){
        Map map = new MapStructure.Map("MapNo1", 10, 10, 0, 0, 1);
        Map map2 = map.clone();
        boolean b = map.compareTo(map2);
        assertTrue(b);
}
}
```