IndividualTesting Report

Yufeng Bai

ID: 1600095

20 October 2012

Contents

Chapter 1

Introduction of the Test Plan

In this project, I am be chosen to do the GUI button, speed slider tests and some physical tests. The GUI is used on the host system side, which is used to implement the manual control, map operation and robot status displaying. The scope of my GUI test covers these parts below:

1.1 GUI Button test introduction

My GUI button test is mainly used to test the buttons of the GUI in different situation.

1. The default button. When the GUI is running, if the user does not do the further operation, some buttons are not allowed to be pressed.
2. The new map initial button. When the map is created, there are some buttons are active, but some others are still inactive.
3. The bluetooth connection button. When the bluetooth connection is successful, there are some buttons are active, but some others are still inactive. The bluetooth connection button include the mode change button. When the mode change button is active, the current mode button is not allowed to be pressed again.

1.2 Speed Slider test

The speed slider test is used to test whether the value of the speed slider matches the value of the current speed. Meanwhile, it is also test whether or not this value is able to demonstrate on the information board.

1.3 Physical test

The physical test is used to test the accuracy of the robot. In physical test, i mainly test whether the current program parameters are accurate to control the robot. In my physical test case, I mainly include three movement operation of the robot.

1. Forward Operation. Using GUI to control the forward movement(it means move to the north of the map) of the robot, mainly check whether or not the robot is able to move one pixel when user presses forward button every one time.
2. TurnLeft Operation. Using GUI to control the turnLeft movement of the robot. When user uses the GUI to let the robot turn left, I need to make sure the robot is able to turn left 90 degrees.

Chapter 2

Test Plan and Specific Test Case

2.1 The function or class I need to test in different situation

1. Since the GUI is an entirety, it is hard to test one method, I plan to test the all relevant methods for one button. The test functions include All functions and class about the Jbutton auto, manual, addNew, loadMap, up, down, left, right, turnLeft, turnRight, bluetoothConnection in the FIle LEGOGUI.
2. Test the functions and class which is relevant to the Slider. Testing the value of the speed slider and the value of the current speed.
3. For the Physical test, I presume i do not know the specific program of the robot, what i need to do is to test the specific location of the robot after operating by GUI.

2.2 Testing Tool and Operating System

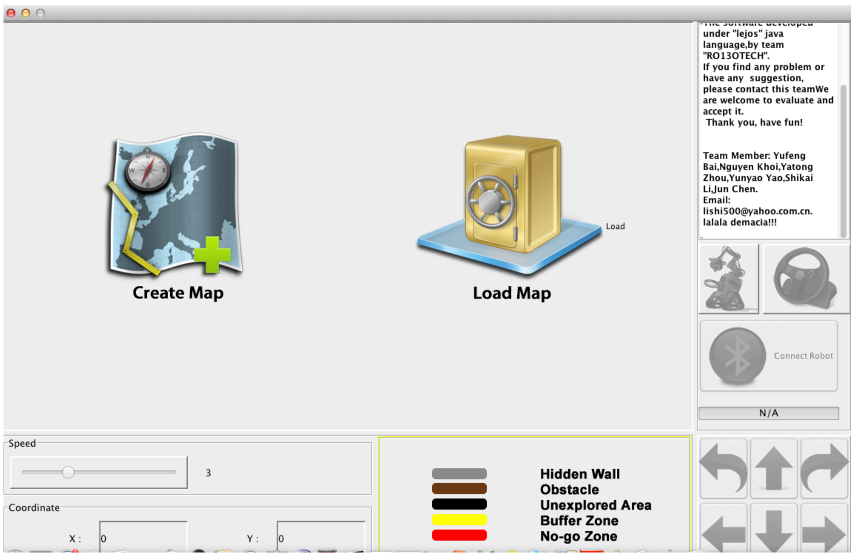
- Eclipse SDK for java developer with JUnit test
- Version: 3.7.2
- Build id: M20120208-0800
- Operating System: OS X 10.8.2(12C60)

2.3 Test1: Default Button

2.3.1 Test method

2.3.2 Description:

When the GUI is running, there are some buttons allowing to press and some buttons not allowing to press. I plan to use Jbutton.isEnabled method to test whether not these buttons are able to work correctly. The button with colour tray is not allowed to be pressed.The initial GUI displays below:



under "lejos" java language,by team "RO13OTECH".
If you find any problem or have any suggestion, please contact this teamWe are welcome to evaluate and accept it.
 Thank you, have fun!

Team Member: Yufeng Bai,Nguyen Khoi,Yatong Zhou,Yunyao Yao,Shikai Li,Jun Chen.
Email:
lishi500@yahoo.com.cn.
lalala demacia!!!

▬	Hidden Wall
▬	Obstacle
▬	Unexplored Area
▬	Buffer Zone
▬	No-go Zone

2.3.3 Pass criteria

- The Jbutton.isEnable() methods for addNew, loadMap are true, which means these two buttons are able to be pressed.
- The Jbutton.isEnable() methods for up, down, left, right, turnLeft, turnRight, auto, manual and bluetoothConnection are false, which means these buttons are not allowed to be pressed.

2.3.4 Test Type

JUnit test

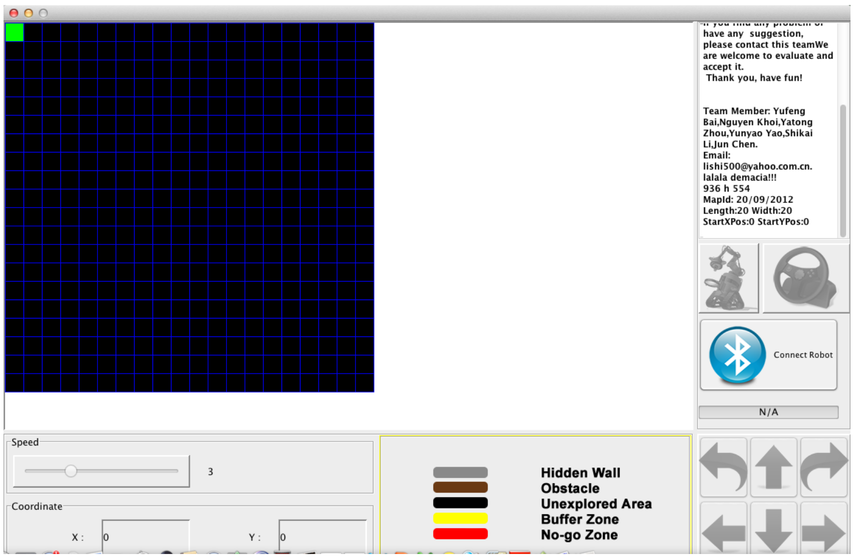
2.3.5 Pass or Fail

Pass

2.4 Test2: New map initial button

2.4.1 Description:

I plan to use Jbutton,doClick() method to implement the operation of creating new map. After confirming the doClick operation, then using Jbutton.isEnabled() method to test the buttons on the GUI screen. The GUI after creating the map shows below:


have any suggestion,
please contact this teamWe
are welcome to evaluate and
accept it.
Thank you, have fun!

Team Member: Yufeng
Bai,Nguyen Khoi,Yatong
Zhou,Yunyao Yao,Shikai
Li,Jun Chen.
Email:
lishi500@yahoo.com.cn.
lalala demacia!!!
936 h 554
MapId: 20/09/2012
Length:20 Width:20
StartXPos:0 StartYPos:0

2.4.2 Pass criteria

- The addNew.doClick() method implements successfully.
- The Jbutton.isEnable() methods for bluetoothConnection, new map panel are true.
- The Jbutton.isEnable() methods for loadMap, auto, manual, up, down, left, right, turn-Left, turnRight are not false.

2.4.3 Test Type

JUnit Test

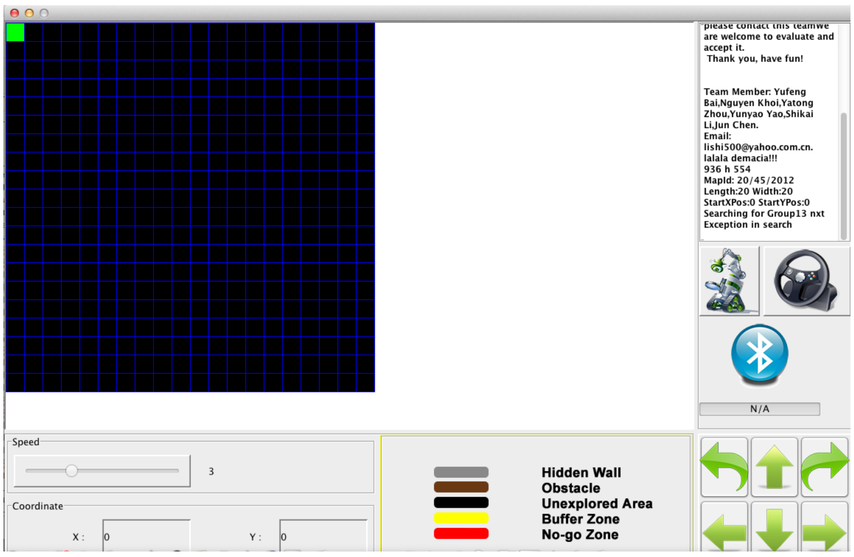
2.4.4 Pass or Fail

Pass

2.5 Test3: Bluetooth connection button

2.5.1 Description

After implementing the addNew.doClick() and bluetoothConnection.doClick() methods, then using Jbutton.isEnabled() method to test the buttons. In addition, if mode change button is pressed, the current mode button is not allowed to be pressed again. The GUI after implementing press the blueConnection button display below:



```
please contact this teamwe
are welcome to evaluate and
accept it.
 Thank you, have fun!


Team Member: Yufeng
Bai,Nguyen Khoi,Yatong
Zhou,Yunyao Yao,Shikai
Li,Jun Chen.
Email:
lishi500@yahoo.com.cn.
lalala demacia!!!
936 h 554
MapId: 20/45/2012
Length:20 Width:20
StartXPos:0 StartYPos:0
Searching for Group13 nxt
Exception in search
```

▬	Hidden Wall
▬	Obstacle
■	Unexplored Area
▬	Buffer Zone
🟥	No-go Zone

2.5.2 Pass criteria

- The addNew.doClick() and bluetoothConnection.doClick() implements successfully.
- The Jbutton.isEnabled() methods for auto, manual, up, down, left, right, turnLeft, turnRight are all true
- After auto.doClick() implements successfully, the Jbutton.isEnabled() method for auto is false
- After manual.doClick() implements successfully, the Jbutton.isEnabled() method for manual is false

2.5.3 Test Type

JUnit Test

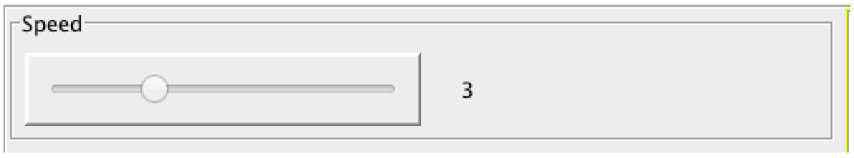
2.5.4 Pass or Fail

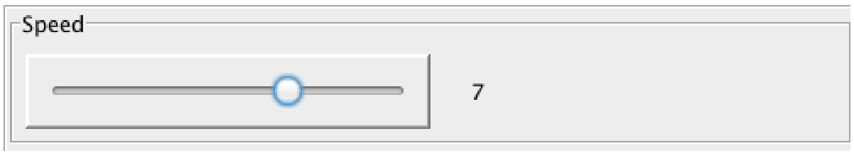
Pass

2.6 Test4: SpeedSlider test

2.6.1 Description

The initial speed of the robot is 3, when we change the speed of the robot by changing the value on the speed slider, the value beside the speed also will change and this value will display on the robot information board. The speed slider display below:





```
_Speed set to be: 7
```

2.6.2 Pass criteria

1. The initial speed is 3, if user changes the speed, the value beside the speed slider will change immediately.
2. The changed speed will display on the information board.

2.6.3 Test Type

Junit test

2.6.4 Pass or Fail

Pass

2.7 Physical Test1: Forward

2.7.1 Description:

When the Robot is in manual control mode, when user clicks the arrow which is point to up on GUI one time. The robot is required to move to the north of the map one pixel(the size of pixel is 25mmx25mm).

2.7.2 Test Specification

In my test plan, we decide to paint two pixels(25mmx25mm) on the map by ruler(the sequence of two pixels is up and down). And then place the robot on the paper and make sure the two wheels on the top of line of one pixel. Press the Forward Button one time to make the robot does forward movement. Release the button and record the wheels position on the paper.

2.7.3 Pass Criteria

- After the forwarding movement operation, the wheels of the robot is on the top of line of another pixel.

2.7.4 Testing Type

Blackbox

2.7.5 Pass or Fail

Pass

2.8 Physical Test2: TurnLeft

2.8.1 Description:

In manual mode, when user press the turnLeft button, the robot will turn left 90 degrees. This test is to make sure the robot is able to rotate 90 degrees accurately.

2.8.2 Test Specification

In my test plan, I draw a Rectangular Plane Coordinate System on one paper, then place the robot's two wheels on the top of the positive X-coordinate. Pressing the turnLeft button to let the robot implement turn left operation. finally, test whether nor not the robot's wheels on the negative y-coordinate.

2.8.3 Pass Criteria

- After turning left, the two wheels on the negative y-coordinate of the Rectangular Plane Coordinate System.

2.8.4 Testing Type

Blackbox

2.8.5 Pass or Fail

Pass

Appendix A

the GUI and SpeedSlider test

The link to the GUI test case is:

`https://version-control.adelaide.edu.au/svn/sep2012-13/Code/Working%20Copy/Version%200.3/src/Tests/GUITestCase.java`

```
package Tests;

import static org.junit.Assert.*;

import java.awt.AWTException;
import java.awt.Robot;
import java.awt.event.KeyEvent;

import GUI.*;
import org.junit.Test;

public class GUITestCase {
        LEGOGUI lego;
        @Test
        public void testDefaultButtonEnabled(){
                LEGOGUI lego = new LEGOGUI();
                //test addNew, loadMap initial statues
                assertTrue(lego.addNew.isEnabled());
                assertTrue(lego.loadMap.isEnabled());
                //test auto, manual,bluetoothConnection button initial statues
                assertFalse(lego.auto.isEnabled());
                assertFalse(lego.manual.isEnabled());
                assertFalse(lego.bluetoothConnection.isEnabled());

                //test 6 control button initial statues
                assertFalse(lego.up.isEnabled());
                assertFalse(lego.down.isEnabled());
                assertFalse(lego.left.isEnabled());
                assertFalse(lego.right.isEnabled());
                assertFalse(lego.turnLeft.isEnabled());
                assertFalse(lego.turnRight.isEnabled());
                //lego = null;
        }

        @Test //(expected = NullPointerException.class)
        public void testNewMapInitialedButtonEnabled(){

                // when a new map initialed
                // test the statues of each button
                 lego = new LEGOGUI();


                // set addNew map clicked;
                // set new map panel confirm button clicked
                lego.addNew.doClick ();
                lego.nms.mi.Confirm.doClick();

                // addNew and loadMap disabled
                assertFalse(lego.loadMap.isEnabled());
                assertTrue(lego.nms.mi.Confirm.isEnabled());

                // bluetoothConnection button is enabled
                assertFalse(lego.auto.isEnabled());
                assertFalse(lego.manual.isEnabled());
                assertTrue(lego.bluetoothConnection.isEnabled());

                // new map panel is enabled
                assertTrue(lego.nms.mi.isEnabled());
                lego = null;

        }
        @Test //(expected = NullPointerException.class)
        public void testBlueToothConnectedButtonEnabled(){
                lego = new LEGOGUI();
                lego.addNew.doClick ();
                lego.nms.mi.Confirm.doClick();
                //precondition tested through above
                // start connect robot, expected NullPointerException
                lego.bluetoothConnection.doClick();
                // auto and manual Mode enabled
                assertTrue(lego.auto.isEnabled());
                assertTrue(lego.manual.isEnabled());
                // 6 control button enabled
                assertTrue(lego.up.isEnabled());
                assertTrue(lego.down.isEnabled());
                assertTrue(lego.left.isEnabled());
                assertTrue(lego.right.isEnabled());
                assertTrue(lego.turnLeft.isEnabled());
                assertTrue(lego.turnRight.isEnabled());

                //if mode changed
        lego.auto.requestFocus();
                new Thread(){
                public void run(){
                        int i = 0;
                        while(i< 10){
                                lego.robot.keyPress(KeyEvent.VK_ENTER);
```

```
                                        try {
                                                Thread.sleep(100);
                                        } catch (InterruptedException e) {
                                                e.printStackTrace();
                                        }
                                        i++;
                                }
                        }
                }.start();

                        lego.auto.doClick();
                        assertFalse(lego.auto.isEnabled());
                        assertTrue(lego.manual.isEnabled());
                        lego.manual.doClick();
                        assertTrue(lego.auto.isEnabled());
                        assertFalse(lego.manual.isEnabled());}


        @Test
        public void testSpeedSlider() {
                LEGOGUI lego = new LEGOGUI();

                //test speed slider bar reaction and relate label display
                assertEquals(String.valueOf(lego.slider.getValue()), lego.speedLab

                //test when speed slider is changed
                //if the label is change with it or not

                lego.slider.setValue(8);
                assertEquals(String.valueOf(lego.slider.getValue()), lego.speedLab

                // test when speed is changed, if the textField update the change
                int value = 5;
                String text = lego.text.getText();
                String str = "Speed set to be: " + String.valueOf(value);
                lego.slider.setValue(value);
                assertEquals(String.valueOf(lego.slider.getValue()), lego.speedLab
                assertEquals(text+"\n"+str,lego.text.getText());
        }

}
```

Appendix B

Glossary and Reference

B.1 Glossary

- Blackbox: tests the functionality of an application as opposed to its internal structures or workings.
- GUI: Graphical User Interface.
- JRE: Java Runtime Environment
- Junit: is a unit testing framework for the Java programming language.

B.2 Reference

```
http://en.wikipedia.org/wiki/JUnit
http://en.wikipedia.org/wiki/Black-box_testing
http://docs.oracle.com/javase/1.4.2/docs/api/javax/swing/JButton.html
```