IndividualTesting Report

Li Shikai

ID: a1214223

21 October 2012

Contents

Chapter 1

Introduction of the Test Plan

This test project will concentrate on Menu testing, meanly test the availability of Menu Items in different states, include default state, New Map Initial state, and Robot Connected state. Also there will be two test cases about text field and robot current coordinate displayed on GUI.

1.1 Introduction of Menu Test

1. The default is when the GUI is start to run and without any further interruption. Most of Menu item operation is not available to to select. I am plan to test, whether the Menu item has been set correctly.

2. The new map initial state is when addNew button has been clicked and new map has been initialed. Some Menu item will unlock from disable statue. i will test if these menu unlocked as we expected.

3. The Robot Connected state is When the robot is connected though Menu or connect button. Most of Menu will enable, but few Menu will disable when connected. Also, mode change test will include in this test case for testing availability status change of some Menu items.

1.2 Introduction of TextField Test

System log is a text field located on the top right of GUI panel. It return system information from robot, XML file and user operation. This test will checking the correctness of text display, and Print Stream from system.

1.3 Introduction of Robot Current Coordinate Test

Robot coordination labels are located on the bottom left of GUI panel. It will performance robot coordination real time updating after connection is established. This test case will test accuracy of coordination when map initialed and position altered.

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 Test different GUI states

1. The function of GUI is build a simple and easily understood interface, let user give instructions to system, and system will reaction in a proper way. So this test will concentrate on the different constraints are set to Menu of GUI in each state. And how user can interact with it.

2. System log is the primary way user can get information return from system, and confirm the correctness of current operation. So all the information should be catched completely. In this case, i will test three ways to update system log, include two functions and default System.out.print().

3. Although user can get robot position just by looking at map panel, robot coordination is still a good way to get precise robot current position. So this test case mainly test the accurate and real time update when map has been changed.

2.2 Testing Tool and Operating System

- Eclipse SDK
- Version: 4.2.0
- Build id: I20120608-1400
- JUnit
- Operating System: Windows 7

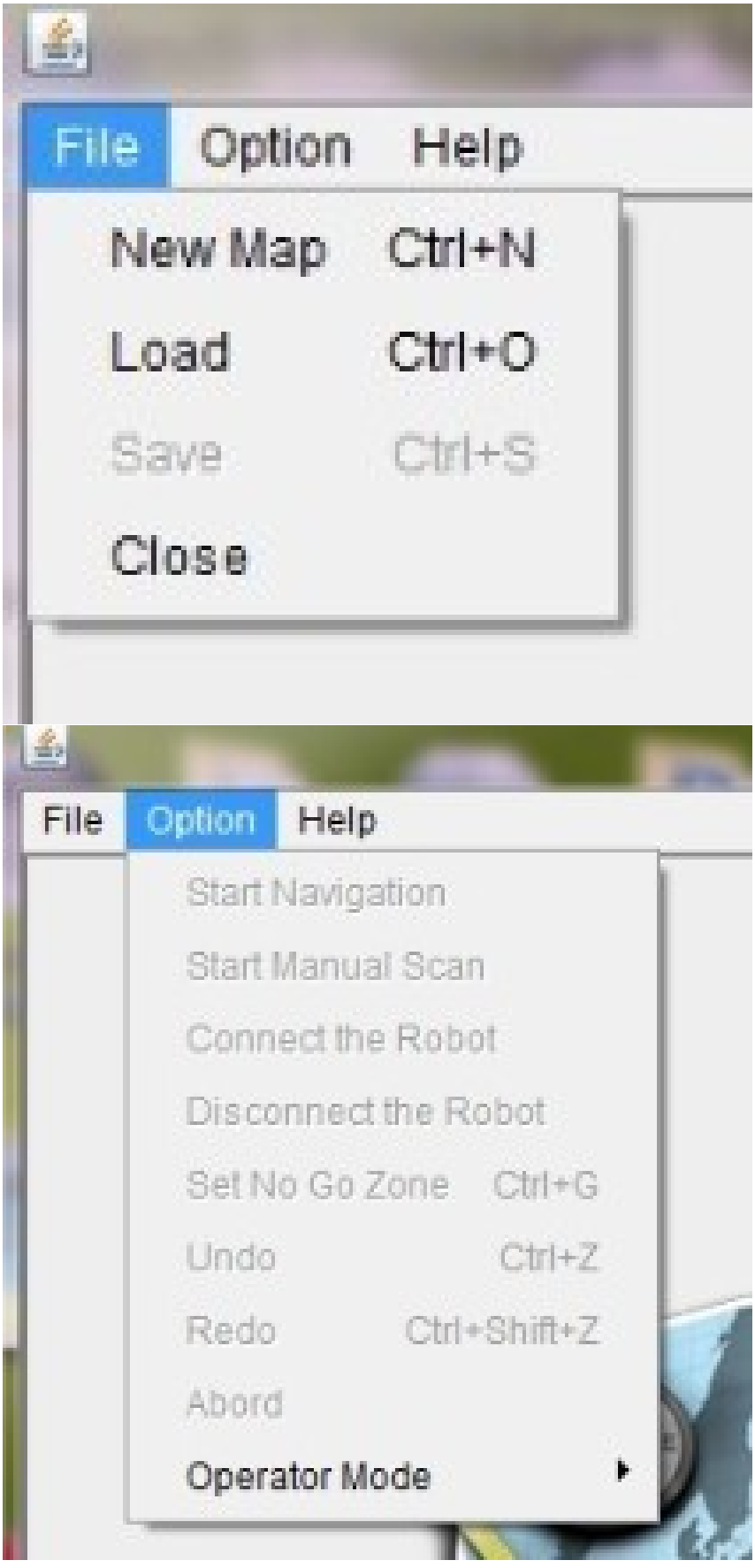
2.3 Test1: Default Menu State

2.3.1 Test method

- Method: MenuBarInitail(). This method is called when a new Map is Initialed.

2.3.2 Description:

When the program is started. MenuBarInitail() will be called by initial() function to build menu part of GUI. This function will give the default constraint of each menu item, like user are not expected to save a map if then did not even initial a new map. So i will use java.awt.Component.isEnabled() to test whether the Menu items are disabled as we expect.


File
Option
Help
New Map Ctrl+N
Load Ctrl+O
Save Ctrl+S
Close
File
Option
Help
Start Navigation
Start Manual Scan
Connect the Robot
Disconnect the Robot
Set No Go Zone Ctrl+G
Undo Ctrl+Z
Redo Ctrl+Shift+Z
Abord
Operator Mode

2.3.3 Pass criteria

- As the picture shows, most items in File menu are available in default stage, except Save map option, which are not permit to select without precondition new map or load map.
- For Option menu, except change operator mode, all the other items are not able to be select. Those items are all about instruction to robot explore, so we want user to use it after the connection has been established.

2.3.4 Test Type

JUnit test

2.3.5 Pass or Fail

Pass

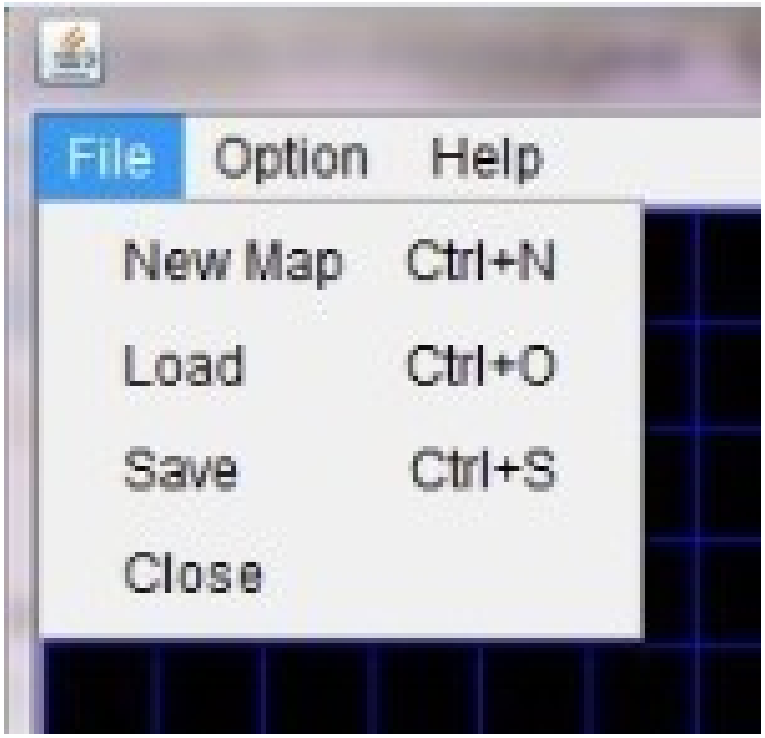
2.4 Test2: New map initial State

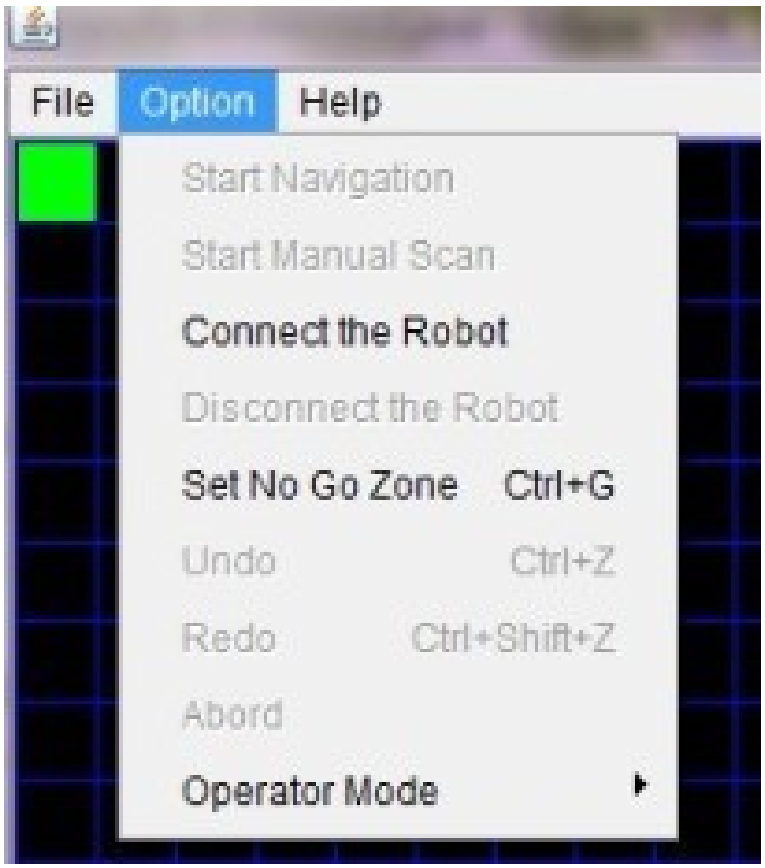
2.4.1 Method:

- class NewMapStart()

2.4.2 Description:

The second state is New map initial, i plan to use javax.swing.AbstractButton.doClick() method to simulate a mouse click to addNew button then do the same operation to Confirm button to accept default map arguments and then Initial a new map. Enter the second state, we do the same test as default state, but change some expecting result to test the Menu item constraints.





2.4.3 Pass criteria

- As we can see all the items are enabled in File menu. So we use AssertTrue to test those items.
- After a map has been initial, we expect user are able connect robot, also no go zone can be set before exploring. AssertTrue are used to measure those two items.

2.4.4 Test Type

JUnit Test

2.4.5 Pass or Fail

Pass

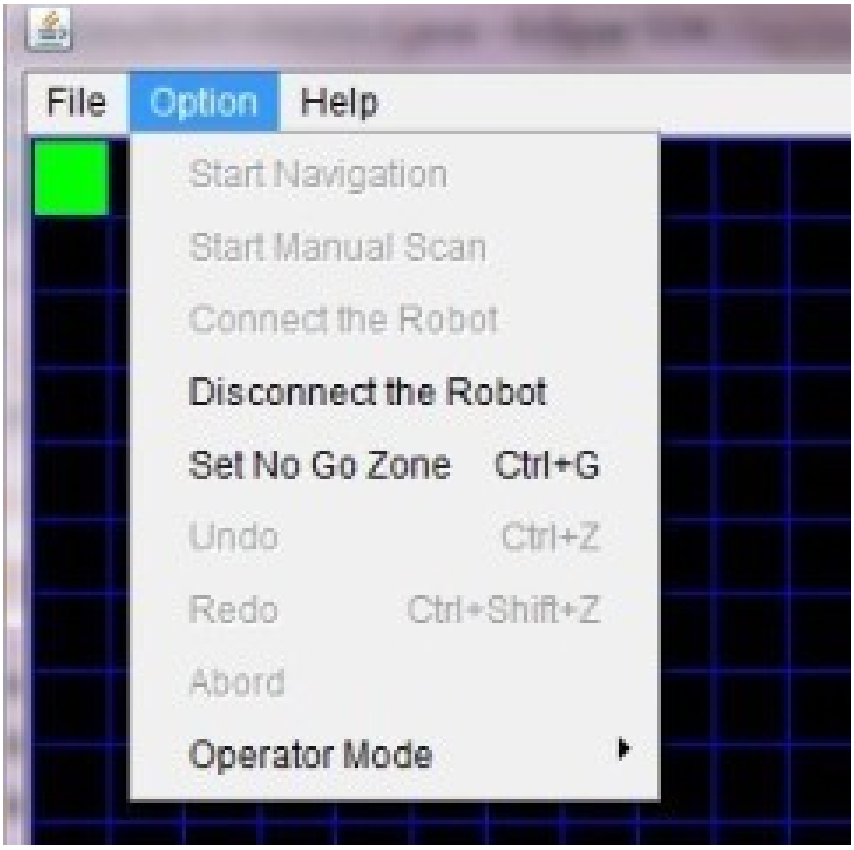
2.5 Test3: Robot connection State

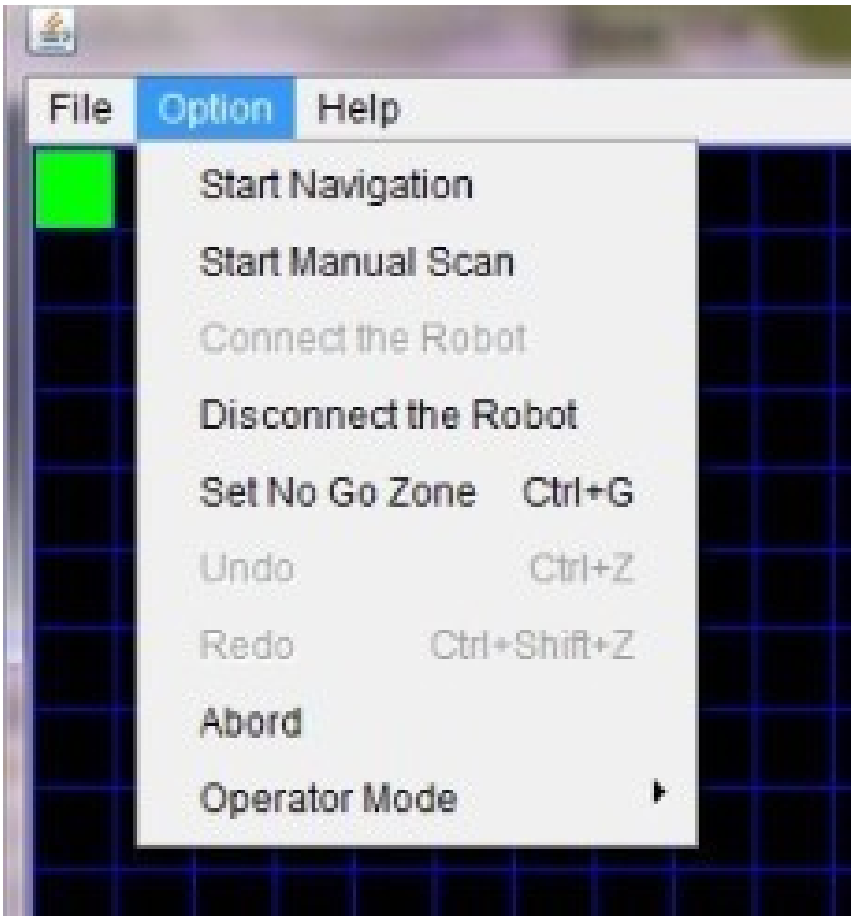
2.5.1 Test method

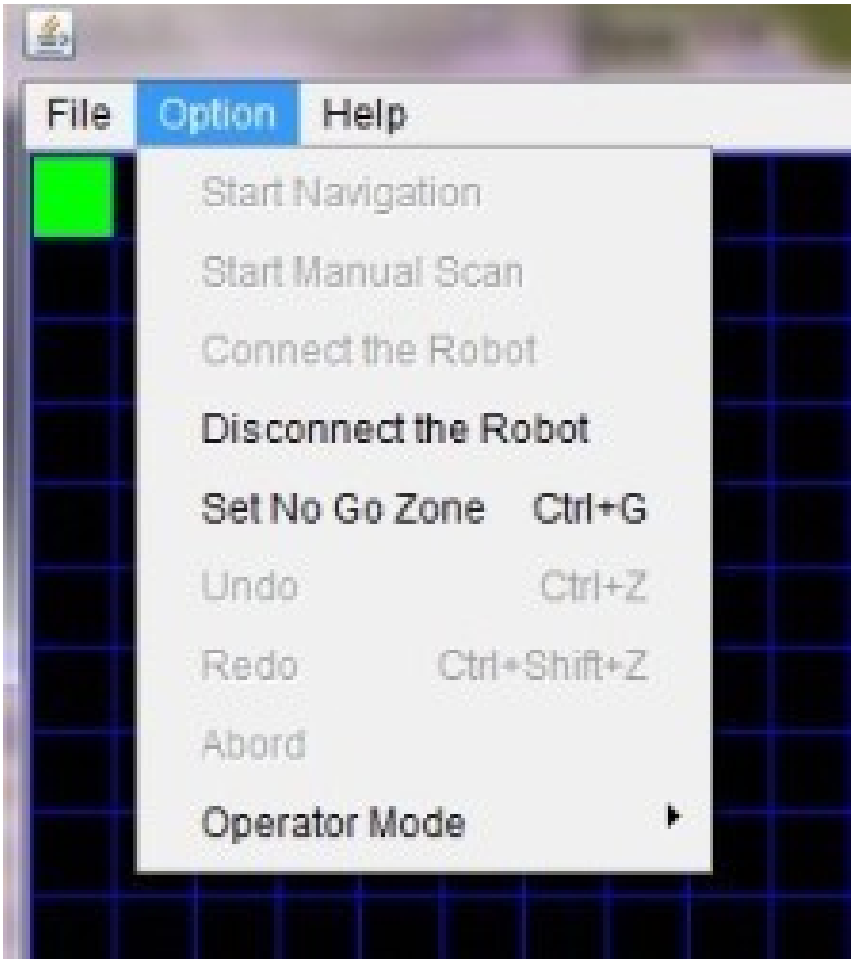
- listener ConnectionListener().

2.5.2 Description

After initial a new map, we do the same operation to button bluetoothConnection for connecting the robot. Because we are not really establish the connection, nullPointerException is expected in this test. After test connection state menu items constraint, i will test the change when control mode switch.


File
Option
Help
Start Navigation
Start Manual Scan
Connect the Robot
Disconnect the Robot
Set No Go Zone Ctrl+G
Undo Ctrl+Z
Redo Ctrl+Shift+Z
Abord
Operator Mode


File
Option
Help
Start Navigation
Start Manual Scan
Connect the Robot
Disconnect the Robot
Set No Go Zone Ctrl+G
Undo Ctrl+Z
Redo Ctrl+Shift+Z
Abord
Operator Mode



2.5.3 Pass criteria

- After robot connected, disconnect menu items will unlock. Meanwhile connect option will disable. Until next map initialed.
- To automatic test mode change i implement a thread to run Java.awt.Robot.keyPressed() to accept JOptionPane mode switch option.
- In auto mode, we assertTrue() start navigation, start manual scan and abord. On the contrary we use assertFalse to meansure those three items in manual mode.

2.5.4 Test Type

JUnit Test

2.5.5 Pass or Fail

Pass

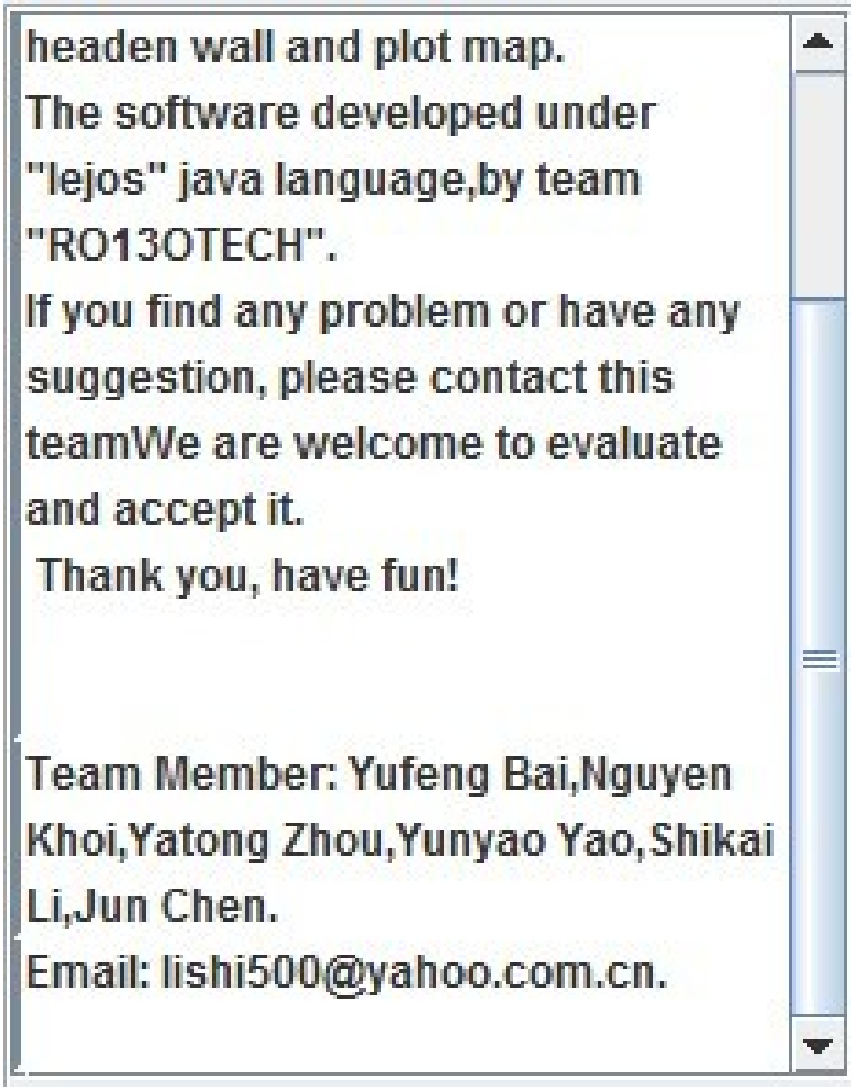
2.6 Test4: TextField Test

2.6.1 Test method

1. public void LEGOGUI.setText(String str).
2. public void LEGOGUI.append(String str)

2.6.2 Description

There are three ways to update text field. Two are through function call. One is print stream from System.out.print();



2.6.3 Pass criteria

1. First test append(String) function. This function will append the input String with a new line on the end of existing String. So we first get the current text from text field though textField.getText(), then call append function to append a string on text Field. After that we assertEquals() old text plus a new line plus text we append with current text field text.

2. Similar operation, but setText(String) is set text field as the given String, so we just assertEquals() given text with getText();

3. We transfer the output of System.out.print() from terminal to text field, in TextAreaPrintStream class. It perform just as append(String) function, so the testing method is mostly same.

2.6.4 Test Type

Junit test

2.6.5 Pass or Fail

Pass

2.7 Robot Coordination Test

2.7.1 Description:

When the Robot position changed, the coordinate of robot will change with it, So test if the current position of robot(map.getCurrentPixel().getxPos()/.getyPos()) is same with map panel display.

2.7.2 Pass Criteria

- First i will test if the coordination display correctly when the new map is initialed. With the default setting robot start position (0,0).
- Then i will test if the map changed, the robot current position has been altered, whether the coordination panel change with it.

2.7.3 Testing Type

Junit test

2.7.4 Pass or Fail

Pass

Appendix A

the GUI test

```
package Tests;

import static org.junit.Assert.*;
import java.awt.event.KeyEvent;

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



public class MenuTestCase {

        @Test
        public void testDefaultMenuEnabled(){

                LEGOGUI lego = new LEGOGUI();

                //test FIleMenu load ,save statues
                assertTrue(lego.file_new.isEnabled());
                assertTrue(lego.file_open.isEnabled());
                assertFalse(lego.file_save.isEnabled());
                assertTrue(lego.file_close.isEnabled());

                //test OptionMenu initial statues
                assertFalse(lego.start_navigation.isEnabled());
                assertFalse(lego.manual_scan.isEnabled());

                assertFalse(lego.connect_connectRob.isEnabled());
                assertFalse(lego.connect_disconRob.isEnabled());

                assertFalse(lego.file_setNoGoZone.isEnabled());
                assertFalse(lego.file_UndoNoGoZone.isEnabled());
                assertFalse(lego.file_RedoNoGoZone.isEnabled());
                assertFalse(lego.abordMisson.isEnabled());

        }
        @Test
        public void testNewMapInitialedMenuEnabled(){
                LEGOGUI lego = new LEGOGUI();
```

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

            //test FIleMenu load ,save statues
            assertTrue(lego.file_save.isEnabled());// file is enabled

            //test OptionMenu initial statues
            assertFalse(lego.start_navigation.isEnabled());
            assertFalse(lego.manual_scan.isEnabled());

            assertTrue(lego.connect_connectRob.isEnabled()); // connect_robot
            assertFalse(lego.connect_disconRob.isEnabled());

            assertTrue(lego.file_setNoGoZone.isEnabled());  //set no go zone i
            assertFalse(lego.file_UndoNoGoZone.isEnabled());
            assertFalse(lego.file_RedoNoGoZone.isEnabled());
            assertFalse(lego.abordMisson.isEnabled());

    }

    @Test//(expected = NullPointerException.class , timeout=1000)
    public void testConnectedMenuEnabled() {
            final LEGOGUI lego = new LEGOGUI();
            // set addNew map clicked;
            // set new map panel confirm button clicked
            lego.addNew.doClick ();
            lego.nms.mi.Confirm.doClick();
            lego.bluetoothConnection.doClick();

            //test FIleMenu load ,save statues
            assertTrue(lego.file_save.isEnabled());// file is enabled

            //test OptionMenu initial statues
            assertFalse(lego.start_navigation.isEnabled());
            assertFalse(lego.manual_scan.isEnabled());

            assertFalse(lego.connect_connectRob.isEnabled()); // connect_robot
            assertTrue(lego.connect_disconRob.isEnabled());

            assertTrue(lego.file_setNoGoZone.isEnabled());  //set no go zone i
            assertFalse(lego.file_UndoNoGoZone.isEnabled());
            assertFalse(lego.file_RedoNoGoZone.isEnabled());
            assertFalse(lego.abordMisson.isEnabled());


            // thread automatic click enter, to ensure select right mode
            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();
        // click auto button
                        lego.auto.doClick();

                        assertTrue(lego.start_navigation.isEnabled());
                        assertTrue(lego.manual_scan.isEnabled());
                        assertTrue(lego.abordMisson.isEnabled());
                        // click manual button
                        lego.manual.doClick();
                        assertFalse(lego.start_navigation.isEnabled());
                        assertFalse(lego.manual_scan.isEnabled());
                        assertFalse(lego.abordMisson.isEnabled());
        }
        @Test
        public void testTestField() {
                LEGOGUI lego = new LEGOGUI();

                // test append function for JTextFiled
                String text = lego.text.getText();
                String append = "append";
                lego.append(append);
                assertEquals(text+"\n"+append,lego.text.getText());

                // test setText function for JTextFiled
                String setText = "setText";
                lego.setText(setText);
                assertEquals(setText,lego.text.getText());

                // test TextAreaPrintStream
                String Stream = "Stream";
                text = lego.text.getText();
                System.out.println(Stream);
                assertEquals(text+Stream,lego.text.getText().trim());
        }

        @Test
        public void testRobotCurrentCoordinateValue() {
                LEGOGUI lego = new LEGOGUI();
                lego.addNew.doClick ();
                lego.nms.mi.Confirm.doClick();

                // first test default robot coordinate x and y value
                // and the label relate display on GUI
```

```
                int posX, posY;
                posX = lego.map.getCurrentPixel().getxPos();
                posY = lego.map.getCurrentPixel().getyPos();
                assertEquals(0,posX);
                assertEquals(0,posX);
                assertEquals(0,Integer.parseInt(lego.xValue.getText().trim()));
                assertEquals(0,Integer.parseInt(lego.yValue.getText().trim()));

                // then test when robot current position has been changed
                // and the label relate display on GUI
                lego.map.setCurrentPixel(lego.map.findPixel(5,8));
                posX = lego.map.getCurrentPixel().getxPos();
                posY = lego.map.getCurrentPixel().getyPos();
                assertEquals(5,posX);
                assertEquals(8,posY);
                lego.updateCoordinateValue();
                assertEquals(5,Integer.parseInt(lego.xValue.getText().trim()));
                assertEquals(8,Integer.parseInt(lego.yValue.getText().trim()));
        }
}
```

Appendix B

Glossary and Reference

B.1 Glossary

- GUI: Graphical User Interface.
- Junit: is a unit testing framework for the Java programming language.

B.2 Reference

http://en.wikipedia.org/wiki/JUnit
http://docs.oracle.com/javase/1.4.2/docs/api