COMP1549 Coursework Report template 2018/19

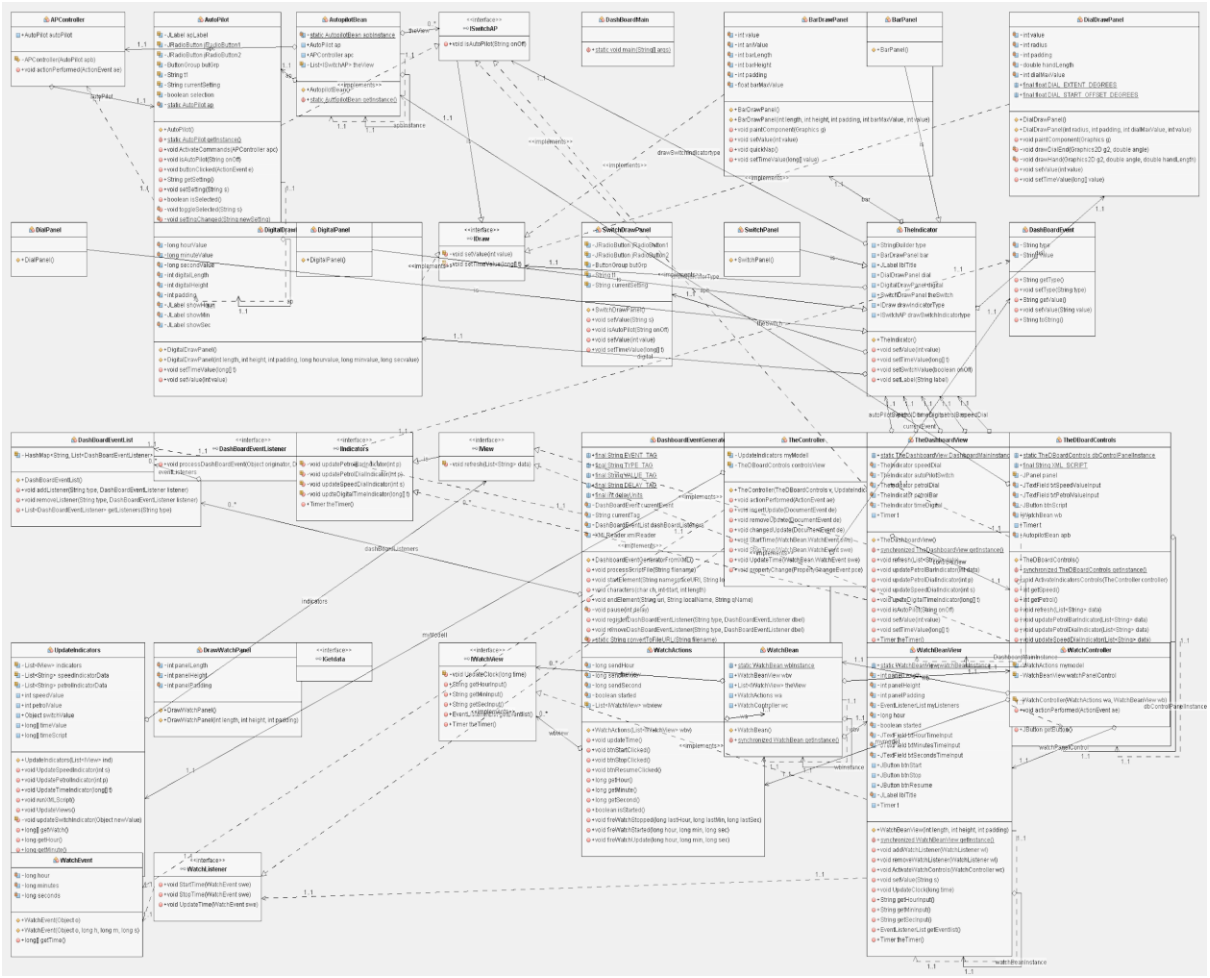
Your name	**Oscar Eko Osa**
Your login id	**000975298**
partner	**Emeka Chimezie**

Section 1 – Brief statement of levels you have completed (same for both partners)

Which type of dashboard application did you implement?	air craft
1.1 Circle the parts of the coursework you have fully completed and are fully working	1a 1b 2a 2bi 2bii 3 4a 4bi 4bii 5a 5b
1.2 Circle the parts of the coursework you have partly completed or are partly working	6a 6b
Briefly explain your answer if you circled any parts in 1.2 We tried to add threading to the program but adding threading was causing errors in our program.It made the program to malfunction so we decided to removing the code from the system.	

Section 2 – UML class diagram of your application (same for both team members)

Include a UML class diagram of your application. Try to use correct notation and make it an accurate representation of what you have implemented. Include enough detail to show the important aspects of the design but not so detailed that the important information is lost in a mass of detail.



A separated picture of the class diagram has been attached to the folder containing the main project

Section 3 – Concise list of bugs and weaknesses (same for both team members)

3.1 Bugs - List plus brief description

Bugs are errors that are found in a program code.This is a fault in a computer program that could lead to the system to malfunction. Bugs tends to gives errors, which in some cases are very disastrous. This can make the system to act on unintended ways, producing unexpected results in our system output.

DashBoardEventGeneratoFromXML.java
Method invokes System.exit(...)
of integer multiplication cast to long
Switch statement found where default case is missing

TheIndicator.java (1 bug)
Non-transient non-serializable instance field in serializable class

DigitalDrawPanel.java
Uninitialized read of field in constructor(hourValue,minuteValue, secondValue)

DialDrawPanel.java
Unchecked/unconfirmed cast

BarDrawPanel.java
Integral division result cast to double or float
Unchecked/unconfirmed cast

WatchEvent.java
Class is Serializable, but doesn't define serialVersionUID

WatchBeanView.java
Field should be package protected

3.2 Weaknesses - List plus brief description

We have problem implementing threading in the program. This caused the program to malfunction. This is one of the weakness of the program and we could not figure out how to amend the thread to work perfectly. The other weakness is that oru bean does not show an icon on the palette if we run the program.

Section 4 - Documentation of each of the levels you completed (same for both team members)

4.1 Level 1 - Brief description (up to 200 words) of the ways in which your program is more reliable, flexible and easy to use than the example code you were supplied. Please include word count.

The Program can be changed without altering the performance of the application. Our GUi is designed in such a way that the user can be able to use the time in the application which we consider very important. We were able to add the autopilot feature, which can be able to start the program automatically. Our program is based on the object-oriented programming so that the code can be reused and changed. We created interface classes such as the iView, indicators, watchlisterner to provide methods that can be shared by other classes. The method used in the interface classes can easily be traced and changed in the future.

4.2 Level 2

4.2.1 Give a UML class diagram for the inheritance hierarchy(s) you implemented. This may be an extract of your earlier UML diagram. The only details needed are the classes and the relationships between them.

4.2.2 Give a UML class diagram for the interfaces. This may be an extract of your earlier UML diagram. The only details needed are the interfaces and the classes that implement them.

4.2.3 Brief reflection (up to 300 words) on how the object-oriented features you implemented improve the design of the system in order to make it easier to maintain and enhance in the future. Please include word count.

We used inheritance in the iview class to minimise the number of code. It inherited from the Iindicators and iswitchap class.This helps to reduce the number of codes used in our program.Any part of the system can be changed without affecting the whole program. The design of our system is a modular structure, making it possible for part of the system to be updated in case of errors without a need to make a large change to our system. The system is easier to maintain since each part of the code can be reused in the system or amended. This reduce the time taken to work on the system by reusing codes. It is possible to create and modify code as new objects can be created with little or no differences to existing objects. The use of the encapsulation, polymorphism and abstraction makes our system to act in more sophisticated way and reduces the runtime errors. The use of inheritance and the interface in our program helps to make our system less complex since it can be enhanced without difficulty, in the future.
There are classes that we used interface such as the iswitchap class. The other classes can implement method declared inside the interface class which also will help to make our code organize and flexible. This can easily be enhanced too in the future

4.3 Level 3

List of classes you tested using JUnit and the number of tests for each class

Indicatorssuite class
iviewtest class
the controller class
Thedboardcontrols class
Thedashboardview class
Updateindicator class

4.4.1 Pattern 1

Name: ***Singleton***

UML class diagram showing the classes and interfaces involved in your implementation of the pattern and the relationships between them.

We included the singleton class diagram in the zip file.

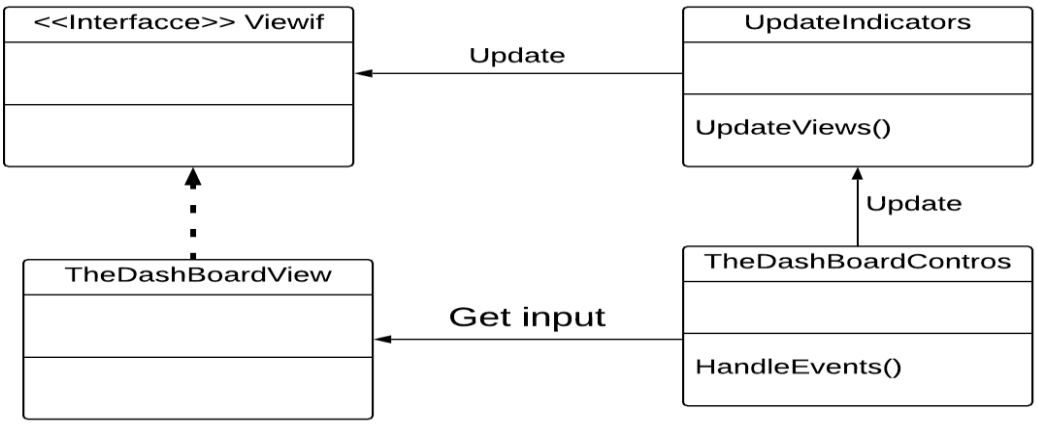
Brief reflection (up to 150 words) about how this design pattern has improved the design of the system in order to make it easier to maintain and enhance in the future. Please include word count.

The singleton pattern used in our system is used to control the creation of many objects in our program. We implemented this pattern to make sure that only one instance of the class is created in the system. This class can be accessed in any class in our program, to enable to creation of the same instance of object if needed. This improved the design of our system because the singleton class can be reused to create the same object.
We used the singleton pattern in the autopilot to create only one instance of the autopilotbean. We also make sure that it checks if any instance of the object is existing because we only need one instance of the autopilotbean to display in our system. Autopilot class also have singleton pattern to be able to get the instance object of the autopilot ap.

4.4.2 Pattern 2

Name:***MVC (model View Control)***

UML class diagram showing the classes and interfaces involved in your implementation of the pattern and the relationships between them.



Brief reflection (up to 150 words) about how this design pattern has improved the design of the system in order to make it easier to maintain and enhance in the future. Please include word count.

We used the model view control pattern in the system to be able change the application logic without affecting code that interacts with the user. The MVC enable the code to be more reliable and flexible so the code could be altered easily. Our system application logic can be decoupled. You can change the user interface code without affecting the application logic and vice versa. The implementation of the MVC pattern makes it easier for us to be able to change the classes. We can amend the model class only, the view class only or the controller class only without affecting each other. It is easy to locate the class that needs to be changed rather than having code cluster or one class that will make it hard to find every method in the system.

4.4.3 Pattern 3

Name: **Strategy Pattern**

UML class diagram showing the classes and interfaces involved in your implementation of the pattern and the relationships between them.

We included the strategy pattern diagram in the coursework zip file.

Brief reflection (up to 150 words) about how this design pattern has improved the design of the system in order to make it easier to maintain and enhance in the future. Please include word count.

We used strategy pattern in the program since some of our objects may have different behaviors at the runtime. The interface class has the update methods for the petrol,speed,digitaltimeindicators. The methods are used in the updateindicator classes but has different methods.

4.5 Level 5

4.5.1 Component 1

Classname:Autopilot

A brief (up to 50 words) description of its purpose and role.

The Autopilot is a component that we added to the system to be display the on and off button on the system. The purpose of autopilot is on and off the dashboard display.Basically, the dashboard display can be controlled with the autopilot.

Number of properties you have coded for the component

Does the BeanInfo file cause the icon to be displayed?

4.5.2 Component 2

Classname: Watchbean

A brief (up to 50 words) description of its purpose and role.

This JBean is used to show the time in the dashboard. We added the time to indicate the current time. As the user will like to know the current time while using the application(dashboard).

Number of properties you have coded for the component

Does the BeanInfo file cause the icon to be displayed?
The beaninfo file does not cause the icon to be displayed.

4.6 Level 6

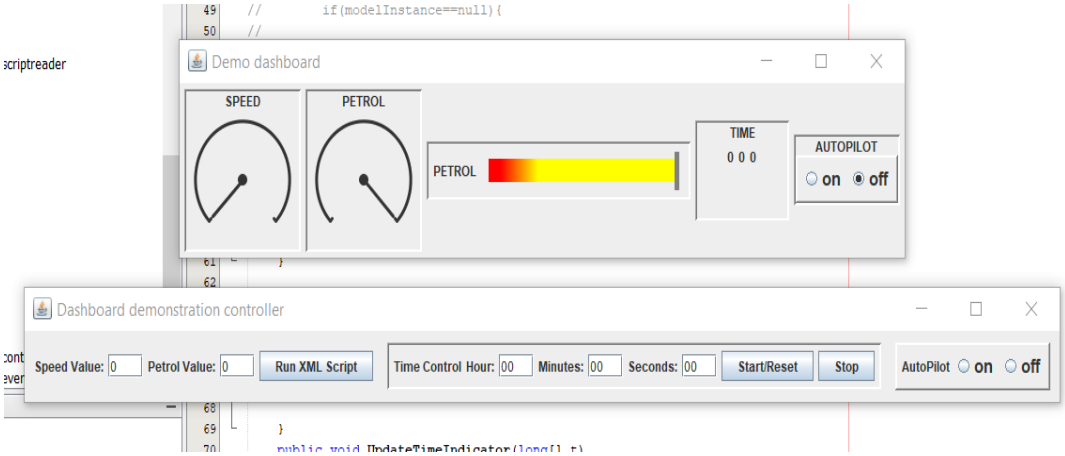
4.6.1 Stretch Feature a)
Brief description of the feature (up to 200 words)

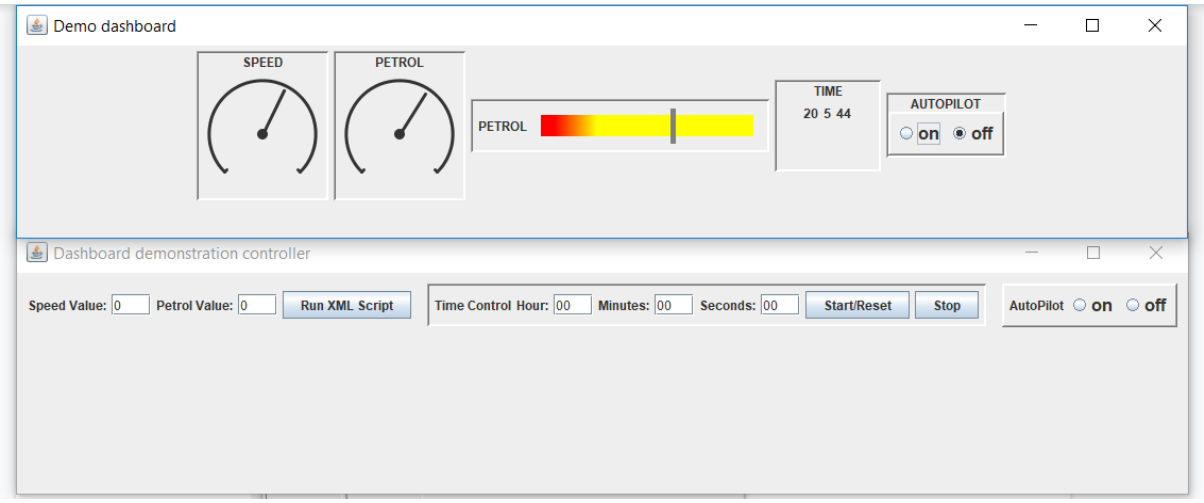
List of strengths and weaknesses of your implementation of the feature

4.6.2 Stretch Feature b)
Brief description of the feature (up to 200 words)

List of strengths and weaknesses of your implementation of the feature

Section 5 – Annotated screenshots demonstrating each of the stages that you have completed (same for both team members)





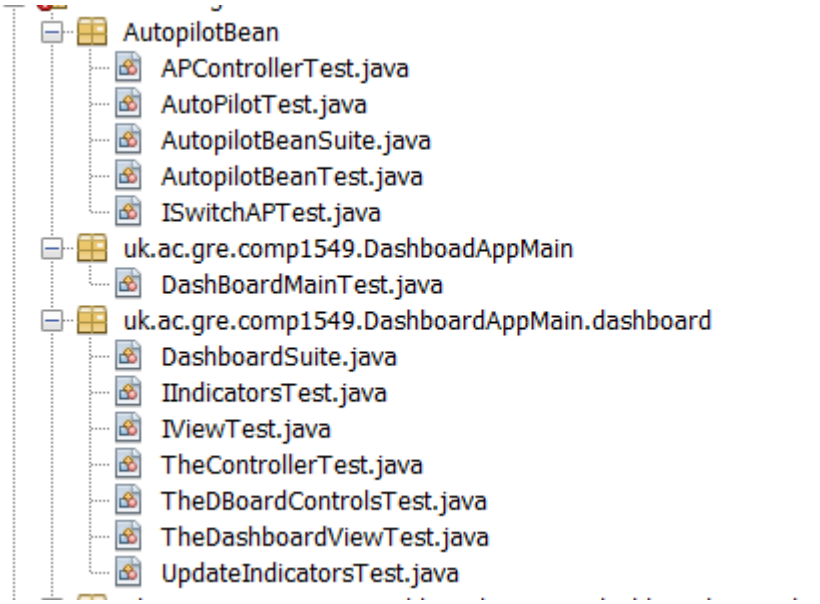
Level 2

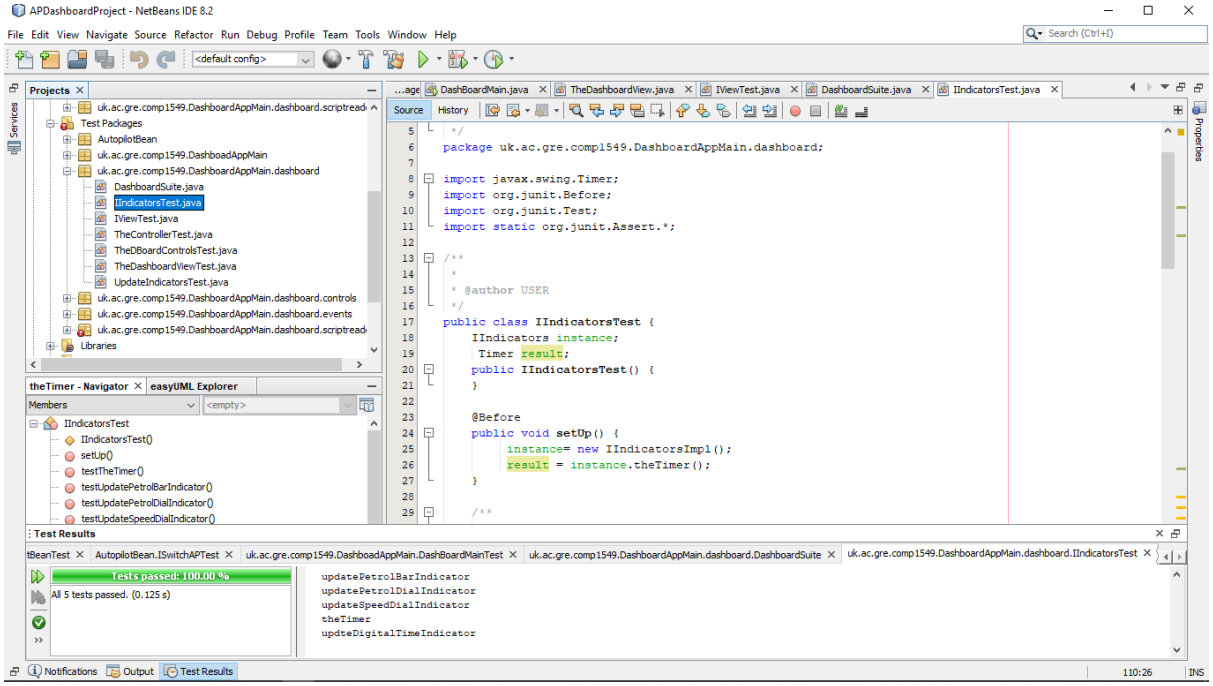
```
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.event.ActionEvent;
import javax.swing.ButtonGroup;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.border.BevelBorder;

/**
 *
 * @author USER
 */
public class AutoPilot extends JPanel implements ISwitchAP {
    JLabel apLabel = new JLabel("AutoPilot");
JRadioButton jRadioButton1 = new JRadioButton();
    JRadioButton jRadioButton2 = new JRadioButton();
    ButtonGroup butGrp = new ButtonGroup();
    String t1 = "on", t2 = "off";
    String currentSetting = "";
    boolean selection=false;
    private static AutoPilot ap=null;
   private AutoPilot()
```

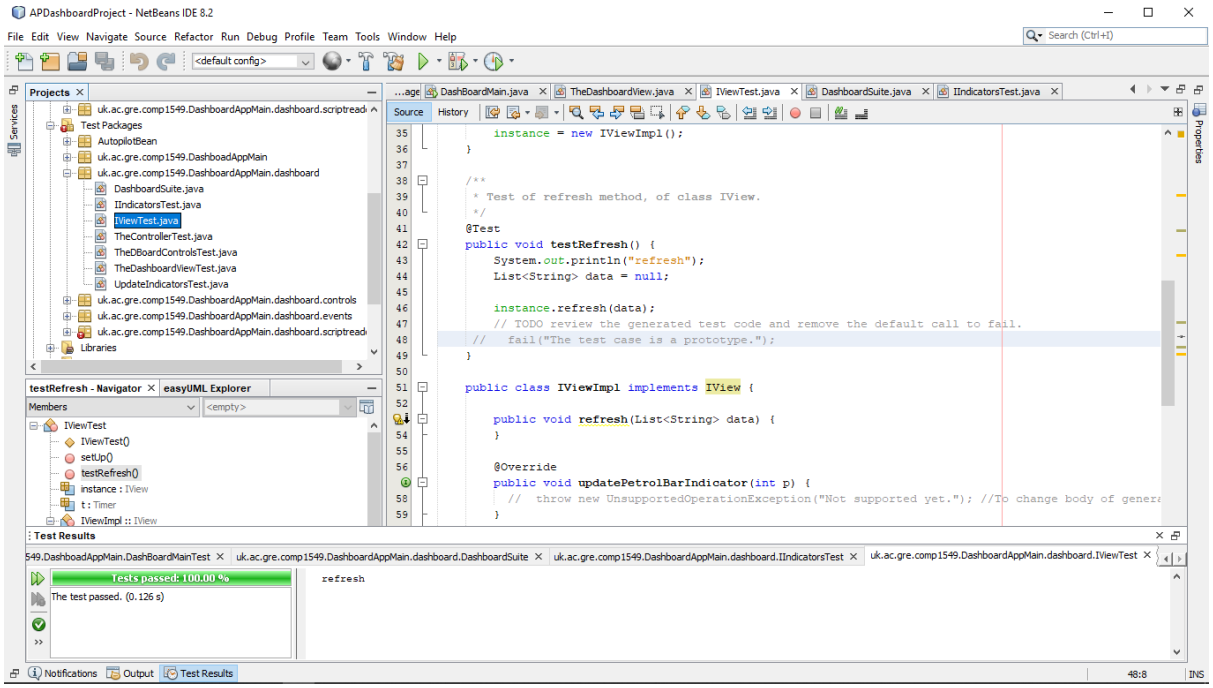
```
 */
public class DrawWatchPanel extends JPanel{
    private int panelLength; // length/width of the bar
    private int panelHeight; // height of the bar

    private int panelPadding;

    public DrawWatchPanel() {
       this(50, 30, 8);
    }

    public DrawWatchPanel(int length, int height, int padding)
    {
        setPreferredSize(new Dimension(length + (2 * padding), height + (2 * padding)));
    this.panelLength=length;
    this.panelHeight=height;
    this.panelPadding=padding;
    }
```

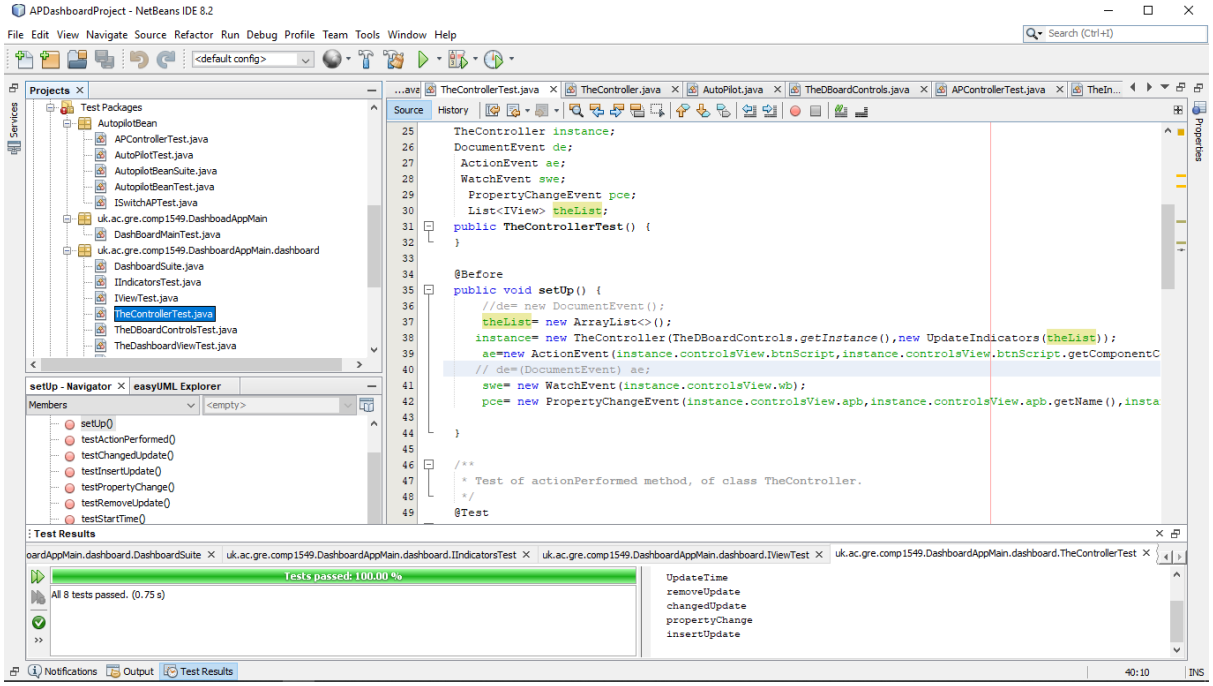
```java
import javax.swing.event.EventListenerList;
import javax.swing.*;

/**
 *
 * @author USER
 */
public interface IWatchView {
    void UpdateClock(long time);
    public String getHourInput();
    public String getMinInput();
   public  String getSecInput();
   public EventListenerList getEventlist();
   public Timer theTimer();
}
```

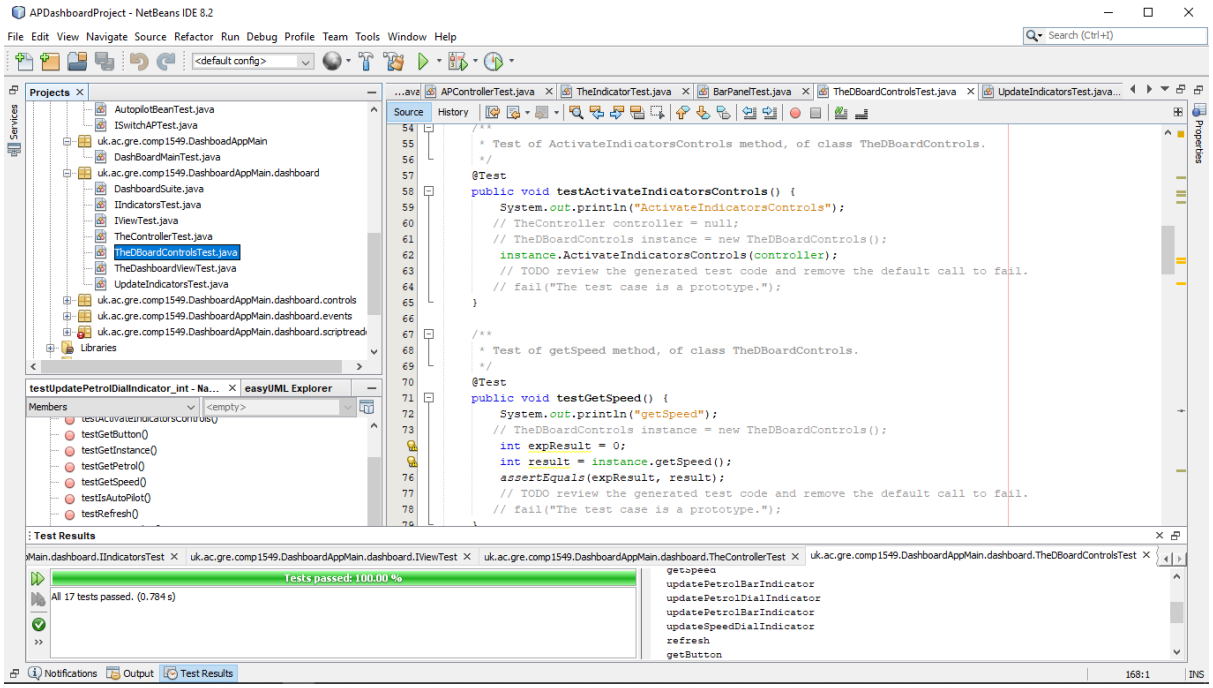
Level3

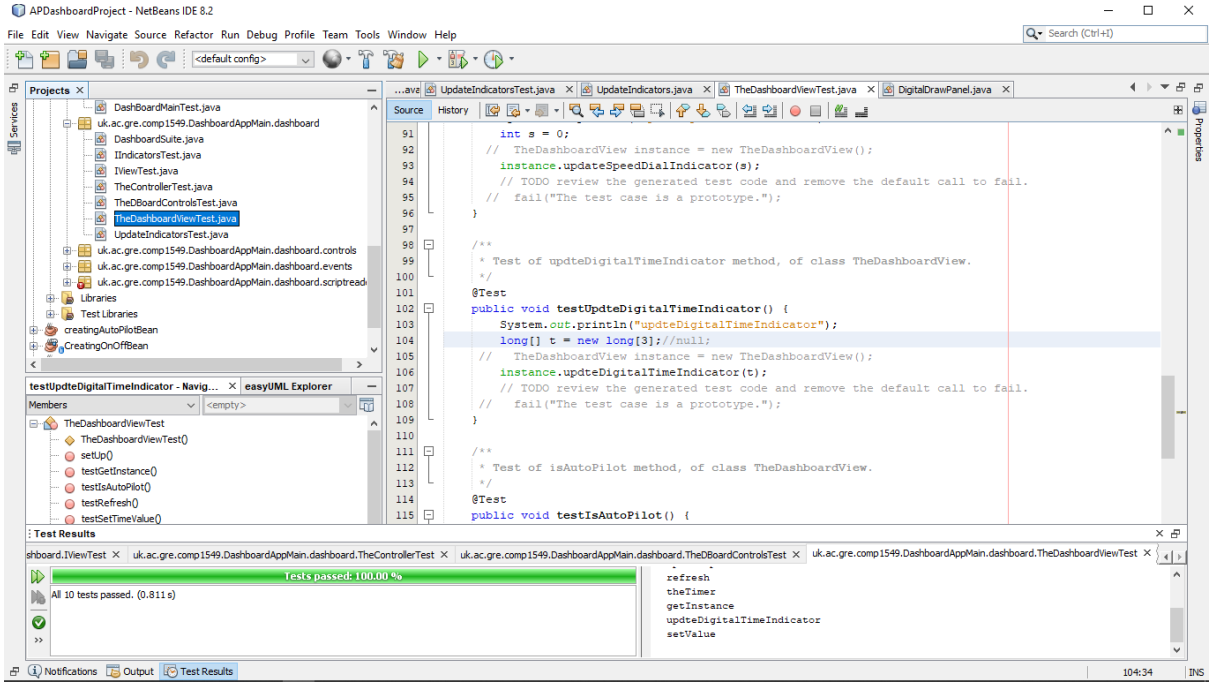


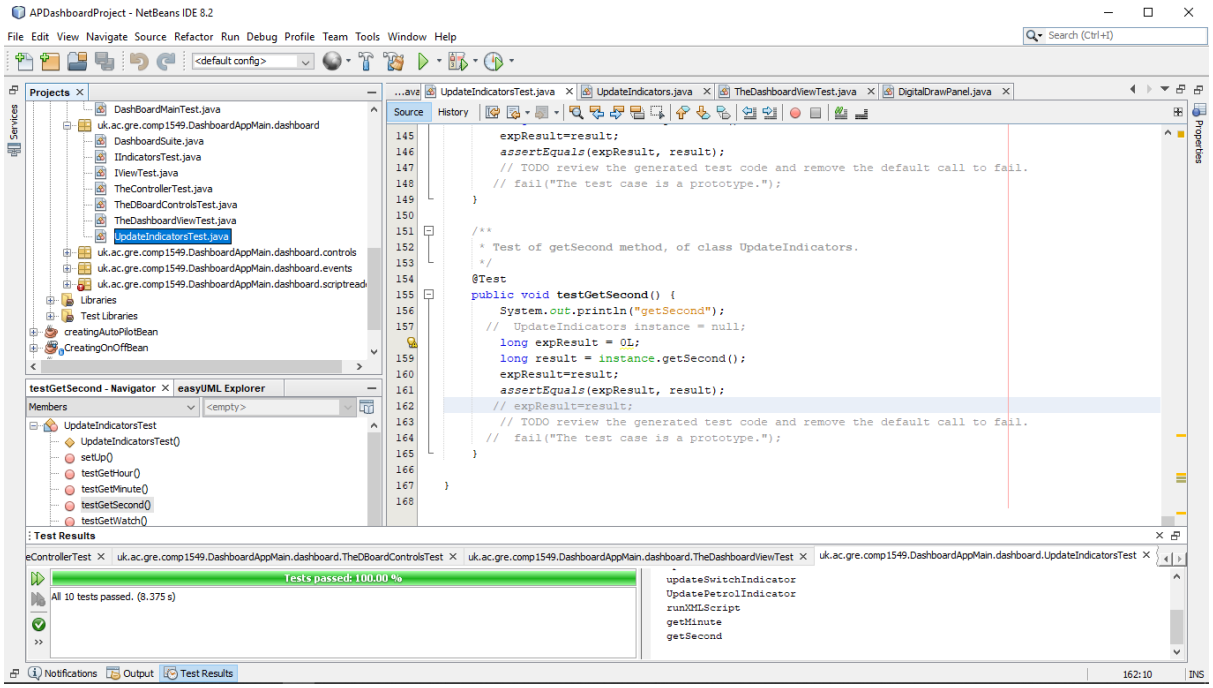

APDashboardProject - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
Test Packages
AutopilotBean
uk.ac.gre.comp1549.DashboadAppMain
uk.ac.gre.comp1549.DashboardAppMain.dashboard
DashboardSuite.java
IIndicatorsTest.java
IViewTest.java
TheControllerTest.java
TheDBoardControlsTest.java
TheDashboardViewTest.java
UpdateIndicatorsTest.java
uk.ac.gre.comp1549.DashboardAppMain.dashboard.controls
uk.ac.gre.comp1549.DashboardAppMain.dashboard.events
Libraries
DashBoardMain.java
TheDashboardView.java
IViewTest.java
DashboardSuite.java
IIndicatorsTest.java
Source
History
package uk.ac.gre.comp1549.DashboardAppMain.dashboard;
import javax.swing.Timer;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;
* @author USER
public class IIndicatorsTest {
IIndicators instance;
Timer result;
public IIndicatorsTest() {
@Before
public void setUp() {
instance= new IIndicatorsImpl();
result = instance.theTimer();
theTimer - Navigator
easyUML Explorer
Members
IIndicatorsTest
IIndicatorsTest()
setUp()
testTheTimer()
testUpdatePetrolBarIndicator()
testUpdatePetrolDialIndicator()
testUpdateSpeedDialIndicator()
Test Results
Tests passed: 100.00 %
All 5 tests passed. (0.125 s)
updatePetrolBarIndicator
updatePetrolDialIndicator
updateSpeedDialIndicator
theTimer
updteDigitalTimeIndicator
Notifications
Output
Test Results
110:26
INS

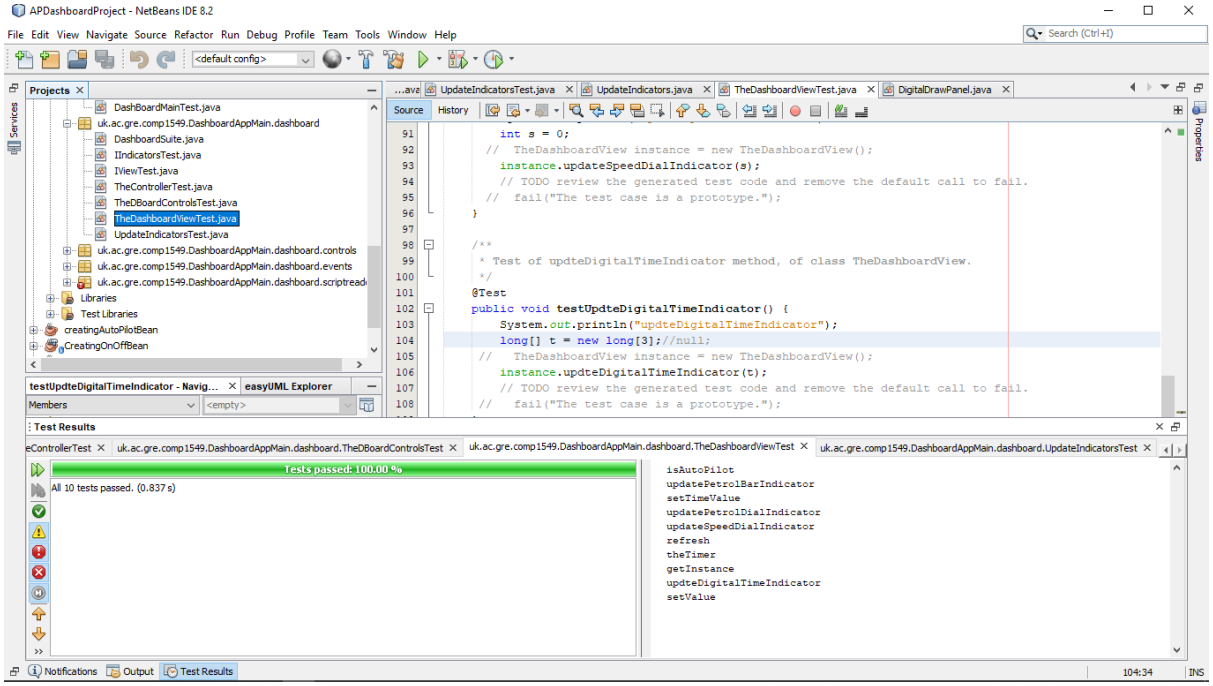

APDashboardProject - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
Test Packages
AutopilotBean
uk.ac.gre.comp1549.DashboadAppMain
uk.ac.gre.comp1549.DashboardAppMain.dashboard
DashboardSuite.java
IIndicatorsTest.java
IViewTest.java
TheControllerTest.java
TheDBoardControlsTest.java
TheDashboardViewTest.java
UpdateIndicatorsTest.java
uk.ac.gre.comp1549.DashboardAppMain.dashboard.controls
uk.ac.gre.comp1549.DashboardAppMain.dashboard.events
Libraries
DashBoardMain.java
TheDashboardView.java
IViewTest.java
DashboardSuite.java
IIndicatorsTest.java
Source
History
instance = new IViewImpl();
/**
* Test of refresh method, of class IView.
*/
@Test
public void testRefresh() {
System.out.println("refresh");
List<String> data = null;
instance.refresh(data);
// TODO review the generated test code and remove the default call to fail.
// fail("The test case is a prototype.");
public class IViewImpl implements IView {
public void refresh(List<String> data) {
@Override
public void updatePetrolBarIndicator(int p) {
// throw new UnsupportedOperationException("Not supported yet."); //To change body of gener
testRefresh - Navigator
easyUML Explorer
Members
IViewTest
IViewTest()
setUp()
testRefresh()
instance : IView
t : Timer
IViewImpl :: IView
Test Results
Tests passed: 100.00 %
The test passed. (0.126 s)
refresh
Notifications
Output
Test Results
48:8
INS

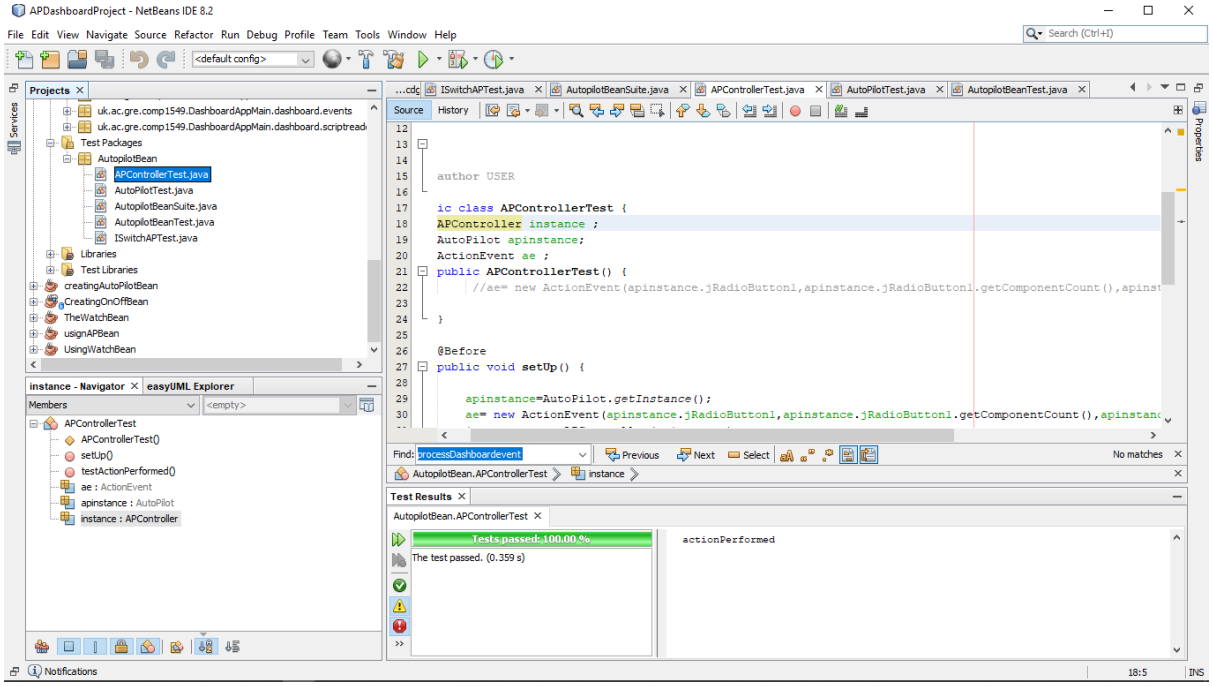

APDashboardProject - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects
Test Packages
AutopilotBean
APControllerTest.java
AutoPilotTest.java
AutopilotBeanSuite.java
AutopilotBeanTest.java
ISwitchAPTest.java
uk.ac.gre.comp1549.DashboadAppMain
DashBoardMainTest.java
uk.ac.gre.comp1549.DashboardAppMain.dashboard
DashboardSuite.java
IIndicatorsTest.java
IViewTest.java
TheControllerTest.java
TheDBoardControlsTest.java
TheDashboardViewTest.java
setUp - Navigator
easyUML Explorer
Members
setUp()
testActionPerformed()
testChangedUpdate()
testInsertUpdate()
testPropertyChange()
testRemoveUpdate()
testStartTime()
TheControllerTest.java
TheController.java
AutoPilot.java
TheDBoardControls.java
APControllerTest.java
Source
History
Test Results
Tests passed: 100.00 %
All 8 tests passed. (0.75 s)
UpdateTime
removeUpdate
changedUpdate
propertyChange
insertUpdate
Notifications
Output
Test Results
40:10
INS

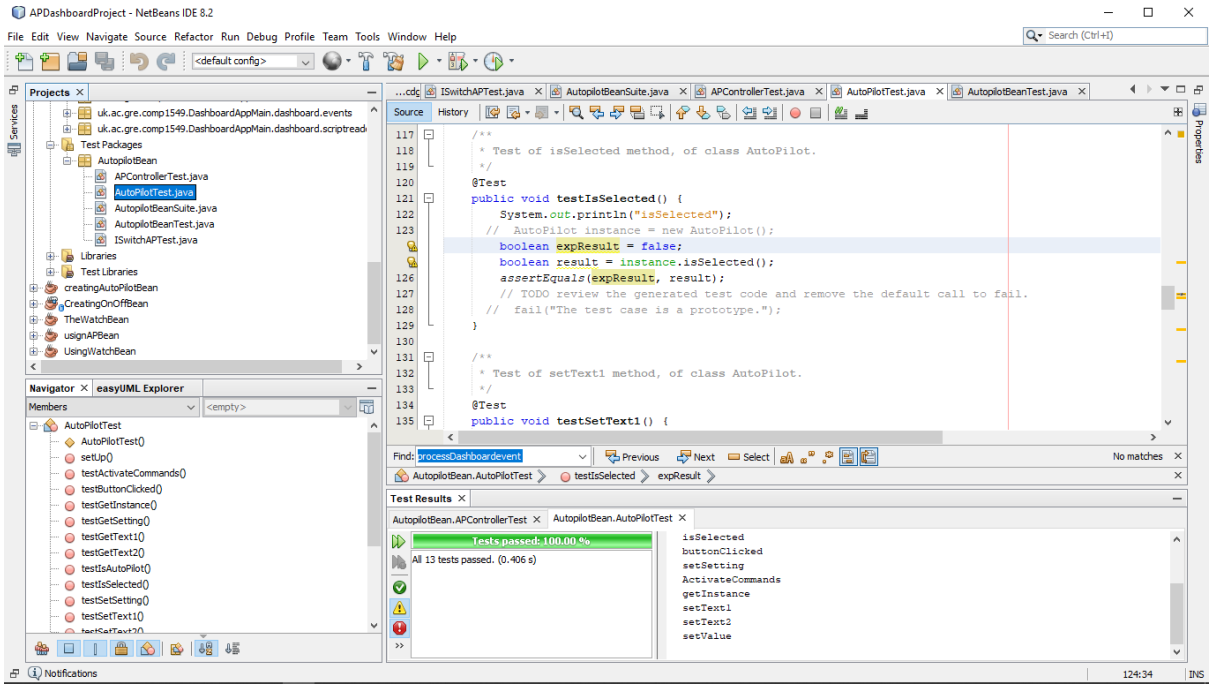

APDashboardProject - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects
AutopilotBeanTest.java
ISwitchAPTest.java
uk.ac.gre.comp1549.DashboadAppMain
DashBoardMainTest.java
uk.ac.gre.comp1549.DashboardAppMain.dashboard
DashboardSuite.java
IIndicatorsTest.java
IViewTest.java
TheControllerTest.java
TheDBoardControlsTest.java
TheDashboardViewTest.java
UpdateIndicatorsTest.java
uk.ac.gre.comp1549.DashboardAppMain.dashboard.controls
uk.ac.gre.comp1549.DashboardAppMain.dashboard.events
uk.ac.gre.comp1549.DashboardAppMain.dashboard.scriptread
Libraries
testUpdatePetrolDialIndicator_int - Na...
easyUML Explorer
Members
testGetButton()
testGetInstance()
testGetPetrol()
testGetSpeed()
testIsAutoPilot()
testRefresh()
APControllerTest.java
TheIndicatorTest.java
BarPanelTest.java
TheDBoardControlsTest.java
UpdateIndicatorsTest.java
Source
History
Test Results
Tests passed: 100.00 %
All 17 tests passed. (0.784 s)
getSpeed
updatePetrolBarIndicator
updatePetrolDialIndicator
updatePetrolBarIndicator
updateSpeedDialIndicator
refresh
getButton
Notifications
Output
Test Results
168:1
INS

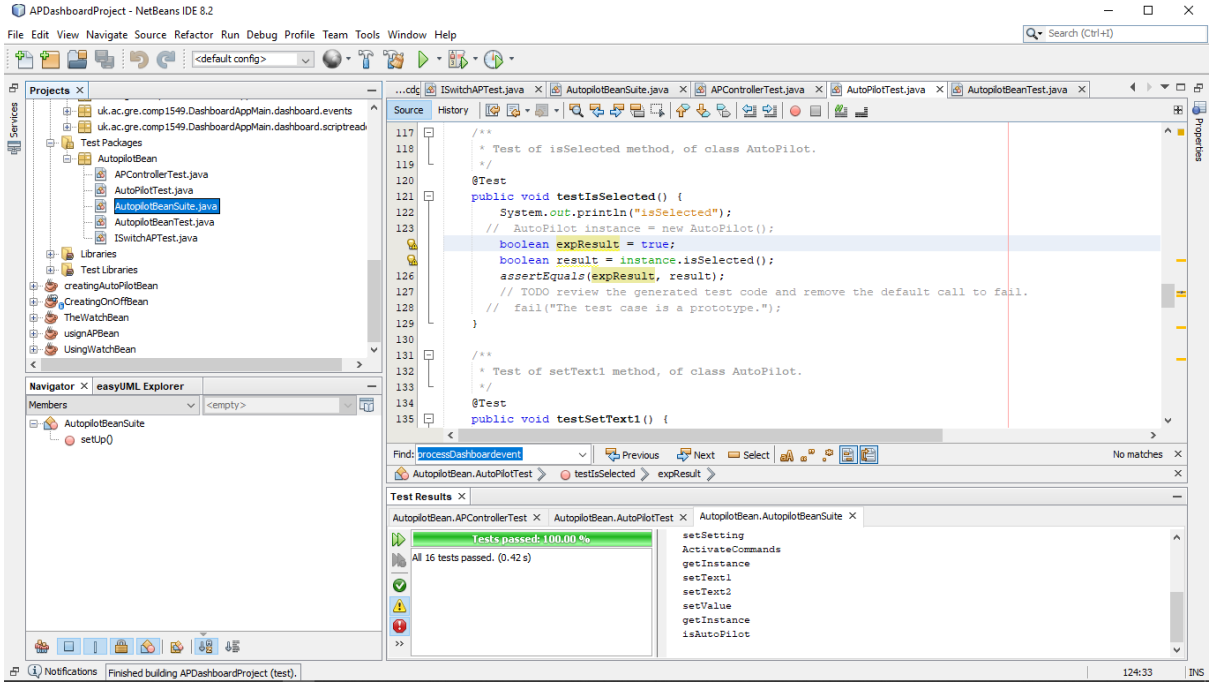

APDashboardProject - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects
DashBoardMainTest.java
uk.ac.gre.comp1549.DashboardAppMain.dashboard
DashboardSuite.java
IIndicatorsTest.java
IViewTest.java
TheControllerTest.java
TheDBoardControlsTest.java
TheDashboardViewTest.java
UpdateIndicatorsTest.java
uk.ac.gre.comp1549.DashboardAppMain.dashboard.controls
uk.ac.gre.comp1549.DashboardAppMain.dashboard.events
uk.ac.gre.comp1549.DashboardAppMain.dashboard.scriptread
Libraries
Test Libraries
creatingAutoPilotBean
CreatingOnOffBean
UpdateIndicatorsTest.java
UpdateIndicators.java
TheDashboardViewTest.java
DigitalDrawPanel.java
Source
History
testUpdteDigitalTimeIndicator - Navig...
easyUML Explorer
Members
<empty>
TheDashboardViewTest
TheDashboardViewTest()
setUp()
testGetInstance()
testIsAutoPilot()
testRefresh()
testSetTimeValue()
Test Results
Tests passed: 100.00 %
All 10 tests passed. (0.811 s)
refresh
theTimer
getInstance
updteDigitalTimeIndicator
setValue
Notifications
Output
Test Results
104:34
INS

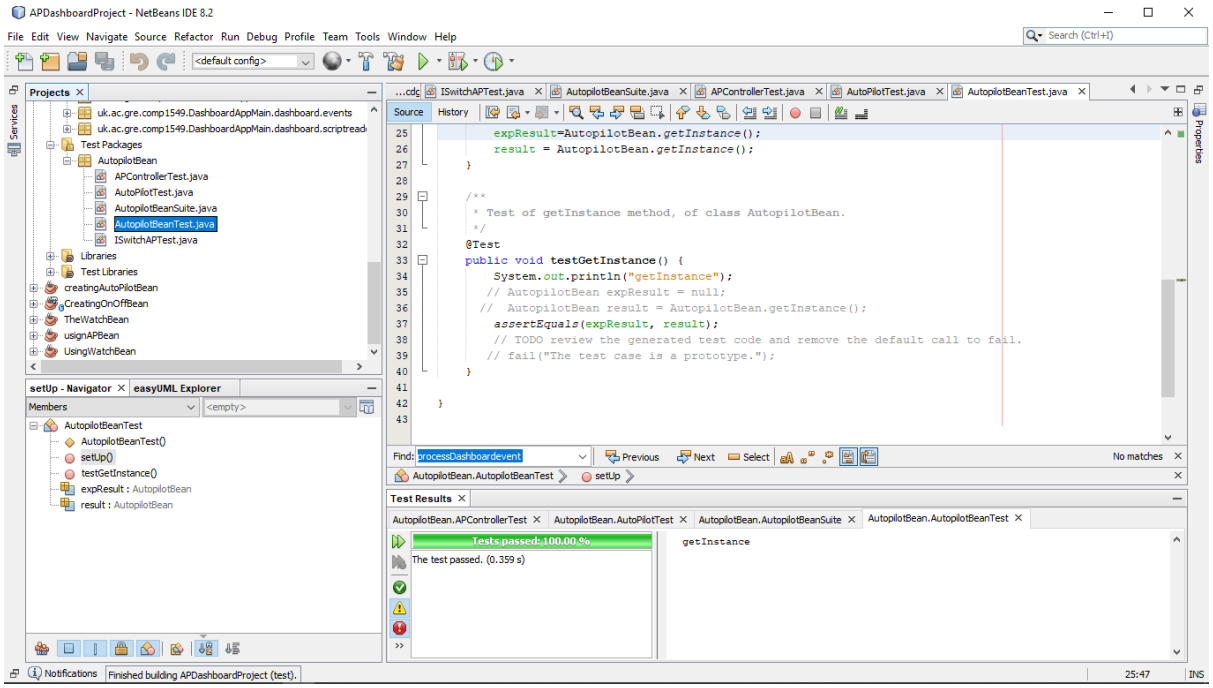

APDashboardProject - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects
DashBoardMainTest.java
uk.ac.gre.comp1549.DashboardAppMain.dashboard
DashboardSuite.java
IIndicatorsTest.java
IViewTest.java
TheControllerTest.java
TheDBoardControlsTest.java
TheDashboardViewTest.java
UpdateIndicatorsTest.java
uk.ac.gre.comp1549.DashboardAppMain.dashboard.controls
uk.ac.gre.comp1549.DashboardAppMain.dashboard.events
uk.ac.gre.comp1549.DashboardAppMain.dashboard.scriptread
Libraries
Test Libraries
creatingAutoPilotBean
CreatingOnOffBean
UpdateIndicatorsTest.java
UpdateIndicators.java
TheDashboardViewTest.java
DigitalDrawPanel.java
Source
History
testGetSecond - Navigator
easyUML Explorer
Members
<empty>
UpdateIndicatorsTest
UpdateIndicatorsTest()
setUp()
testGetHour()
testGetMinute()
testGetSecond()
testGetWatch()
Test Results
Tests passed: 100.00 %
All 10 tests passed. (8.375 s)
updateSwitchIndicator
UpdatePetrolIndicator
runXMLScript
getMinute
getSecond
Notifications
Output
Test Results
162:10
INS

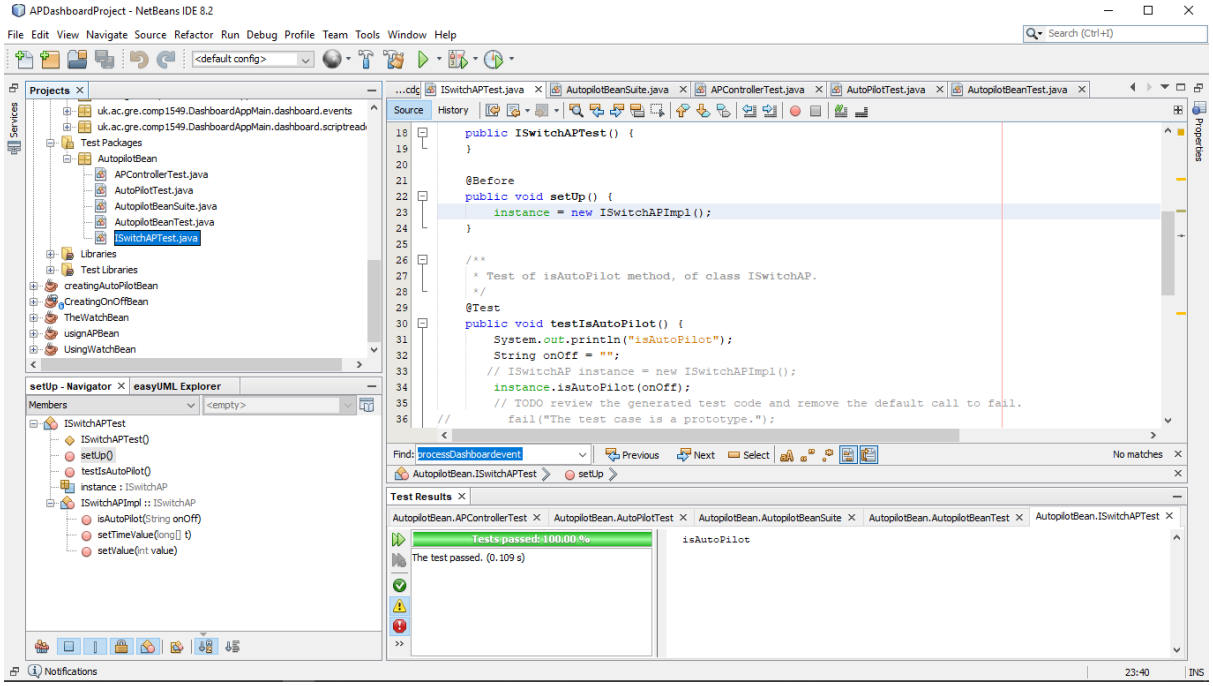

APDashboardProject - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects
DashBoardMainTest.java
uk.ac.gre.comp1549.DashboardAppMain.dashboard
DashboardSuite.java
IIndicatorsTest.java
IViewTest.java
TheControllerTest.java
TheDBoardControlsTest.java
TheDashboardViewTest.java
UpdateIndicatorsTest.java
uk.ac.gre.comp1549.DashboardAppMain.dashboard.controls
uk.ac.gre.comp1549.DashboardAppMain.dashboard.events
Libraries
Test Libraries
creatingAutoPilotBean
CreatingOnOffBean
UpdateIndicatorsTest.java
UpdateIndicators.java
TheDashboardViewTest.java
DigitalDrawPanel.java
Source
History
int s = 0;
// TheDashboardView instance = new TheDashboardView();
instance.updateSpeedDialIndicator(s);
// TODO review the generated test code and remove the default call to fail.
// fail("The test case is a prototype.");
}
/**
* Test of updteDigitalTimeIndicator method, of class TheDashboardView.
*/
@Test
public void testUpdteDigitalTimeIndicator() {
System.out.println("updteDigitalTimeIndicator");
long[] t = new long[3];//null;
// TheDashboardView instance = new TheDashboardView();
instance.updteDigitalTimeIndicator(t);
// TODO review the generated test code and remove the default call to fail.
// fail("The test case is a prototype.");
testUpdteDigitalTimeIndicator - Navig...
easyUML Explorer
Members
Test Results
Tests passed: 100.00 %
All 10 tests passed. (0.837 s)
isAutoPilot
updatePetrolBarIndicator
setTimeValue
updatePetrolDialIndicator
updateSpeedDialIndicator
refresh
theTimer
getInstance
updteDigitalTimeIndicator
setValue
Notifications
Output
Test Results
104:34
INS

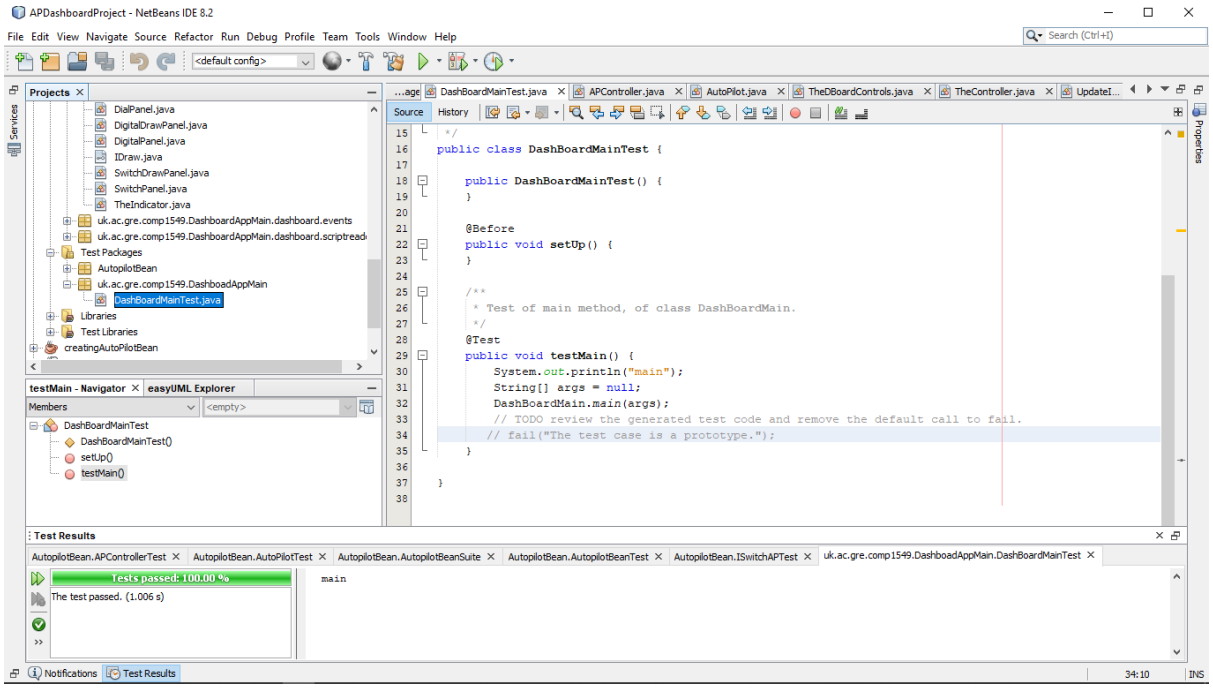

APDashboardProject - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Test Packages
AutopilotBean
APControllerTest.java
AutoPilotTest.java
AutopilotBeanSuite.java
AutopilotBeanTest.java
ISwitchAPTest.java
Libraries
Test Libraries
creatingAutoPilotBean
CreatingOnOffBean
TheWatchBean
usignAPBean
UsingWatchBean
instance - Navigator
easyUML Explorer
APControllerTest
APControllerTest()
setUp()
testActionPerformed()
ae : ActionEvent
apinstance : AutoPilot
instance : APController
author USER
ic class APControllerTest {
APController instance ;
AutoPilot apinstance;
ActionEvent ae ;
public APControllerTest() {
}
@Before
public void setUp() {
apinstance=AutoPilot.getInstance();
Test Results
Tests passed: 100.00 %
The test passed. (0.359 s)
actionPerformed


APDashboardProject - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Test Packages
AutopilotBean
APControllerTest.java
AutoPilotTest.java
AutopilotBeanSuite.java
AutopilotBeanTest.java
ISwitchAPTest.java
Libraries
Test Libraries
creatingAutoPilotBean
CreatingOnOffBean
TheWatchBean
usignAPBean
UsingWatchBean
Navigator
easyUML Explorer
AutoPilotTest
AutoPilotTest()
setUp()
testActivateCommands()
testButtonClicked()
testGetInstance()
testGetSetting()
testGetText1()
testGetText2()
testIsAutoPilot()
testIsSelected()
testSetSetting()
testSetText1()
/**
* Test of isSelected method, of class AutoPilot.
*/
@Test
public void testIsSelected() {
System.out.println("isSelected");
// AutoPilot instance = new AutoPilot();
boolean expResult = false;
boolean result = instance.isSelected();
assertEquals(expResult, result);
// TODO review the generated test code and remove the default call to fail.
// fail("The test case is a prototype.");
}
/**
* Test of setText1 method, of class AutoPilot.
*/
@Test
public void testSetText1() {
Test Results
Tests passed: 100.00 %
All 13 tests passed. (0.406 s)
isSelected
buttonClicked
setSetting
ActivateCommands
getInstance
setText1
setText2
setValue


APDashboardProject - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Test Packages
AutopilotBean
APControllerTest.java
AutoPilotTest.java
AutopilotBeanSuite.java
AutopilotBeanTest.java
ISwitchAPTest.java
Libraries
Test Libraries
creatingAutoPilotBean
CreatingOnOffBean
TheWatchBean
usignAPBean
UsingWatchBean
Navigator
easyUML Explorer
AutopilotBeanSuite
setUp()
Source
History
/**
* Test of isSelected method, of class AutoPilot.
*/
@Test
public void testIsSelected() {
System.out.println("isSelected");
// AutoPilot instance = new AutoPilot();
boolean expResult = true;
boolean result = instance.isSelected();
assertEquals(expResult, result);
// TODO review the generated test code and remove the default call to fail.
// fail("The test case is a prototype.");
}
/**
* Test of setText1 method, of class AutoPilot.
*/
@Test
public void testSetText1() {
Find: processDashboardevent
Previous
Next
Select
No matches
AutopilotBean.AutoPilotTest
testIsSelected
expResult
Test Results
Tests passed: 100.00 %
All 16 tests passed. (0.42 s)
setSetting
ActivateCommands
getInstance
setText1
setText2
setValue
getInstance
isAutoPilot
Notifications
Finished building APDashboardProject (test).
124:33
INS


APDashboardProject - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Test Packages
AutopilotBean
APControllerTest.java
AutoPilotTest.java
AutopilotBeanSuite.java
AutopilotBeanTest.java
ISwitchAPTest.java
Libraries
Test Libraries
creatingAutoPilotBean
CreatingOnOffBean
TheWatchBean
usignAPBean
UsingWatchBean
setUp - Navigator
easyUML Explorer
AutopilotBeanTest
AutopilotBeanTest()
setUp()
testGetInstance()
expResult : AutopilotBean
result : AutopilotBean
Source
History
expResult=AutopilotBean.getInstance();
result = AutopilotBean.getInstance();
}
/**
* Test of getInstance method, of class AutopilotBean.
*/
@Test
public void testGetInstance() {
System.out.println("getInstance");
// AutopilotBean expResult = null;
// AutopilotBean result = AutopilotBean.getInstance();
assertEquals(expResult, result);
// TODO review the generated test code and remove the default call to fail.
// fail("The test case is a prototype.");
}
}
Find: processDashboardevent
Previous
Next
Select
No matches
AutopilotBean.AutopilotBeanTest
setUp
Test Results
Tests passed: 100.00 %
The test passed. (0.359 s)
getInstance
Notifications
Finished building APDashboardProject (test).
25:47
INS


APDashboardProject - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
ISwitchAPTest.java
AutopilotBeanSuite.java
APControllerTest.java
AutoPilotTest.java
AutopilotBeanTest.java
Source History
public ISwitchAPTest() {
}
@Before
public void setUp() {
instance = new ISwitchAPImpl();
}
/**
* Test of isAutoPilot method, of class ISwitchAP.
*/
@Test
public void testIsAutoPilot() {
System.out.println("isAutoPilot");
String onOff = "";
// ISwitchAP instance = new ISwitchAPImpl();
instance.isAutoPilot(onOff);
// TODO review the generated test code and remove the default call to fail.
// fail("The test case is a prototype.");
Find: processDashboardevent
Previous
Next
Select
No matches
AutopilotBean.ISwitchAPTest
setUp
Test Results
Tests passed: 100.00 %
The test passed. (0.109 s)
isAutoPilot
setUp - Navigator
easyUML Explorer
Members
ISwitchAPTest
ISwitchAPTest()
setUp()
testIsAutoPilot()
instance : ISwitchAP
ISwitchAPImpl :: ISwitchAP
isAutoPilot(String onOff)
setTimeValue(long[] t)
setValue(int value)
Notifications
23:40
INS


APDashboardProject - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
DashBoardMainTest.java
APController.java
AutoPilot.java
TheDBoardControls.java
TheController.java
Source History
*/
public class DashBoardMainTest {
public DashBoardMainTest() {
}
@Before
public void setUp() {
}
/**
* Test of main method, of class DashBoardMain.
*/
@Test
public void testMain() {
System.out.println("main");
String[] args = null;
DashBoardMain.main(args);
// TODO review the generated test code and remove the default call to fail.
// fail("The test case is a prototype.");
}
}
Test Results
Tests passed: 100.00 %
The test passed. (1.006 s)
main
testMain - Navigator
easyUML Explorer
DashBoardMainTest
DashBoardMainTest()
setUp()
testMain()
Notifications
Test Results
34:10
INS

Level 4

```
 */
public class AutopilotBean extends JPanel {
    private static AutopilotBean apbInstance=null;
    public AutoPilot ap;
    public APController apc;
    List<ISwitchAP> theView;

    public AutopilotBean()
    {
   // ap= AutoPilot.
        ap= AutoPilot.getInstance();
    theView= new ArrayList<>();
    theView.add(ap);
    apc= new APController(ap);
    ap.ActivateCommands(apc);
    add(ap);

    }
    public static AutopilotBean getInstance()

    {
    if(apbInstance==null)
    {
    apbInstance= new AutopilotBean();
    }
    return apbInstance;
    }
}
```

```
public class WatchBean extends JPanel {

    public static WatchBean wbInstance=null;
   public  WatchBeanView wbv;//= WatchBeanView.getInstance();

    public List<IWatchView> theView ;//= new ArrayList<>();
    public WatchActions wa;

   public WatchController wc;
     public WatchBean()
     {
     wbv= WatchBeanView.getInstance();
     theView = new ArrayList<>();
     theView.add(wbv);
     wa= new WatchActions(theView);
     wc= new WatchController(wa, wbv);
     wbv.ActivateWatchControls(wc);
    add(wbv);
     }

     public static synchronized WatchBean getInstance()
     {
         if(wbInstance==null)
         {
         wbInstance= new WatchBean();
         }


   return wbInstance; }
}
```

MVC pattern

```java
/**
 *
 * @author USER
 */
//This is the model part of the MVC
public class UpdateIndicators {

   // public static UpdateIndicators modelInstance=null;
    List<IView> indicators;
    List<String> speedIndicatorData= new ArrayList<>();;
    List<String> petrolIndicatorData= new ArrayList<>();;
   public int speedValue;
    public int petrolValue;
    public Object switchValue;
    public long[] timeValue=new long[3];
    public long[]timeScript=new long[3];
    public UpdateIndicators(List<IView> ind)
    {
        indicators=ind;
       // speedIndicatorData= new ArrayList<>();
       // petrolIndicatorData= new ArrayList<>();
    }

//    public void RunScript(List<String> s)
//    {
//
//    }
//     public static synchronized UpdateIndicators getInstance()
```

```java
    private TheIndicator autoPilotSwitch= new SwitchPanel();
   private TheIndicator petrolDial=new DialPanel();
   private TheIndicator petrolBar= new BarPanel();
   private TheIndicator timeDigital= new DigitalPanel();
   public Timer t;
  // private AutopilotBean autoPilot = AutopilotBean.getInstance();

    /**
     * Constructor. Does maybe more work than is good for a constructor.
     */
    //This the panel with all the indicators
    public TheDashboardView() {
       // this.setVisible(true);
        // Set up the dashboard screen
        JFrame dashboard = new JFrame("Demo dashboard");
        dashboard.setDefaultCloseOperation(WindowConstants.HIDE_ON_CLOSE);
        dashboard.setLayout(new FlowLayout());

        // add the speed Dial
       // speedDial = new DialPanel();
        speedDial.setLabel("SPEED");
        dashboard.add(speedDial);

        // add the petrol Dial
       // petrolDial = new DialPanel();
        petrolDial.setLabel("PETROL");
        petrolDial.setValue(100);
        dashboard.add(petrolDial);
```

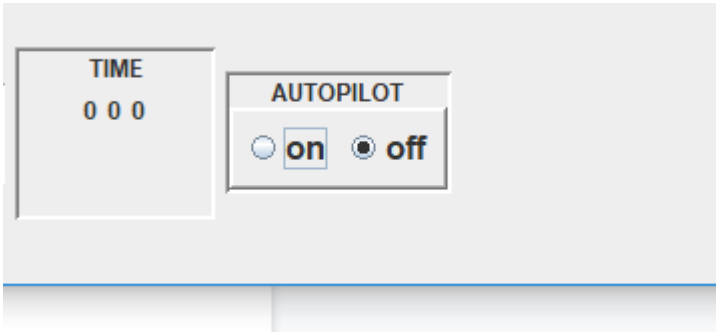
```
//This is the control part of the MVC design pattern
public class TheDBoardControls extends JFrame implements IView{

    public static TheDBoardControls dbControlPanelInstance = null;
     public static final String XML_SCRIPT = "dashboard_script.xml";
     JPanel panel = new JPanel();
     private JTextField txtSpeedValueInput= new JTextField("0", 3);

    private JTextField txtPetrolValueInput= new JTextField("0", 3);
    public JButton btnScript= new JButton("Run XML Script");
   // private JPanel timeControlPanel= new JPanel();
   // private JButton btnStartWatch = new JButton("Start Watch");
   // private JButton btnStopWatch = new JButton("Stop Watch");
   // private JButton btnResumeWatch = new JButton("Resume Watch");
   public WatchBean wb;//= WatchBean.getInstance();
    public Timer t;
   public AutopilotBean apb;

    public TheDBoardControls()
    {

        setTitle("Dashboard demonstration controller");
       setLayout(new FlowLayout());
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


       panel.add(new JLabel("Speed Value:"));
        panel.add(txtSpeedValueInput);
```

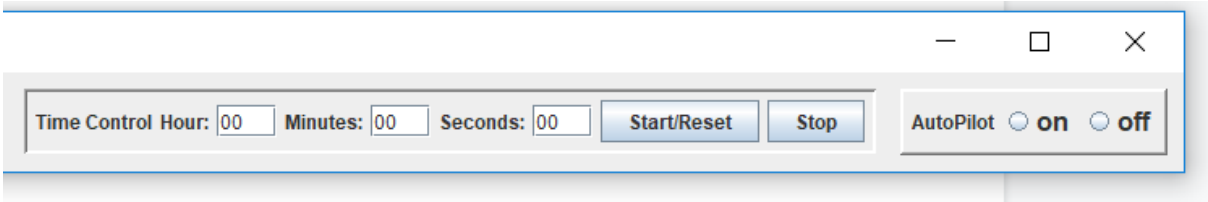
Strategy pattern

```
package uk.ac.gre.comp1549.DashboardAppMain.dashboard;

import AutopilotBean.ISwitchAP;
import java.util.List;
import javax.swing.Timer;

/**
 *
 * @author USER
 */
public interface IIndicators  {
    void updatePetrolBarIndicator(int p);
    void updatePetrolDialIndicator(int p);
    void updateSpeedDialIndicator(int s);
    void updteDigitalTimeIndicator(long[] t);
    public Timer theTimer();
}

//class updateBarIndicator implements IIndicators
//{
//    // @Override
//     public String updates() {
//     return "Bar Indicator";
```

```
    // fields that appear on the dashboard itself
//    private DialPanel speedDial= new DialPanel();
//    private DialPanel petrolDial=new DialPanel();
//    private BarPanel petrolBar= new BarPanel();

    private TheIndicator speedDial= new DialPanel();
    private TheIndicator autoPilotSwitch= new SwitchPanel();
    private TheIndicator petrolDial=new DialPanel();
    private TheIndicator petrolBar= new BarPanel();
    private TheIndicator timeDigital= new DigitalPanel();
    public Timer t;
   // private AutopilotBean autoPilot = AutopilotBean.getInstance();

    /**
     * Constructor. Does maybe more work than is good for a constructor.
     */
    //This the panel with all the indicators
    public TheDashboardView() {
       // this.setVisible(true);
        // Set up the dashboard screen
        JFrame dashboard = new JFrame("Demo dashboard");
        dashboard.setDefaultCloseOperation(WindowConstants.HIDE_ON_CLOSE);
        dashboard.setLayout(new FlowLayout());

        // add the speed Dial
```

Level 5





Section 6 - 350-500 words reflecting on pair programming (write this section individually)

Having already been involved in in group work project, I thought that this time was not going to be different from the other times, but It was.
Although I was accustomed to working in groups, I have never worked with someone else on pair programming, so the concept was new to me.
It did affect my productivity and the way I am used to work. With this I have learned that the experience on pair programming is never the same if every time you work with different people.
To be honest, my partner was not a great programmer (neither I am) but what I mean is that certain concepts that should've been learned and controlled already were not mastered. For example, polymorphism and inheritance which were 2 key concepts to implements in our code were still very difficult to understand for my partner.
Therefore, when we started working on the coursework, I had to write most of code most of the code while trying to explain what was happening to my pair programming partner. It got harder and harder as we kept going forward.
Because of this to be more efficient we decided that whenever we meet to work on our coursework I was going to do most of the code and the diagrams while he does part of the documentation which involves writing and when I was done writing the code we were going to go through the code together so that he understands how what the code was meant to do and that is how we did it.
After a while my partner was even able to develop some new features for our code such as putting numbers in the radial indicators which showed his improvement. Although we decided not to include it as we agreed that the graphics side of our coursework was not our main focus.
As well as this we had problems trying to implement the strategy pattern into our code. We knew that we needed to use it but could figure out how to wire our classes and interfaces together to make it work but we finally achieved it.
So, I said at the beginning the experience in pair programming will vary depending on the person that you are working with. I had a great experience although difficult as I saw partner developing coding skills that were new to him and at the same time, I learned the importance of patience and teaching.

Level 8

COMP1549 Logbook Upload Template 2018/19

Complete this document for each of the four logbook exercises to include in your final report.

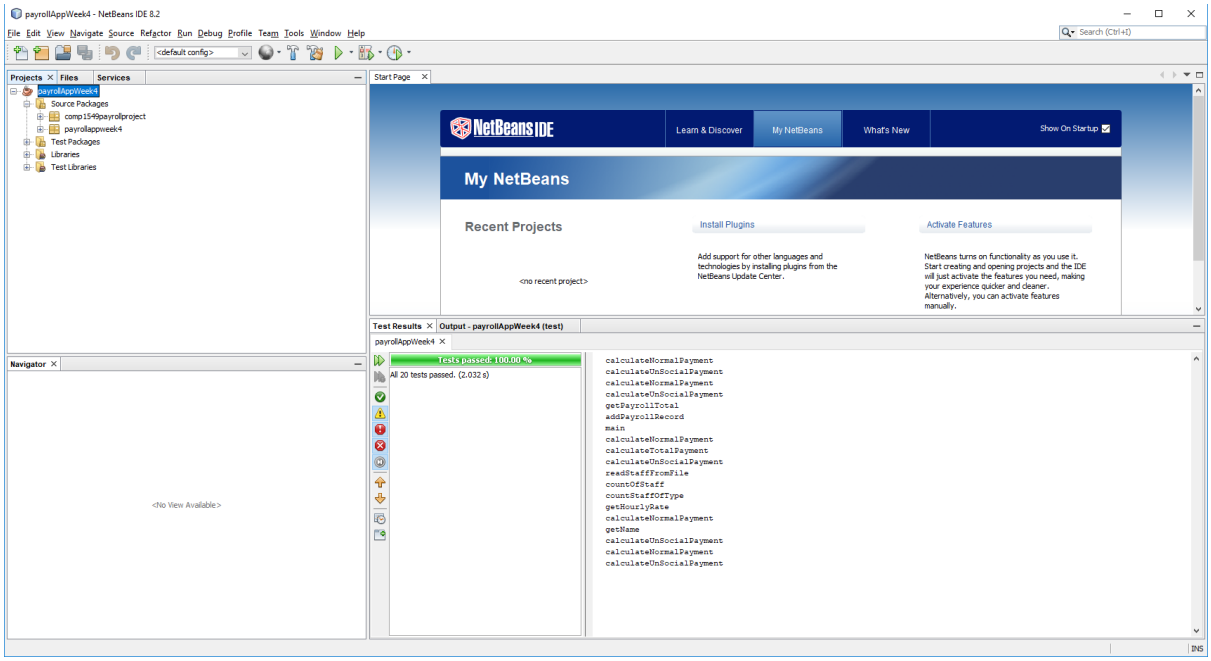
Basic Information

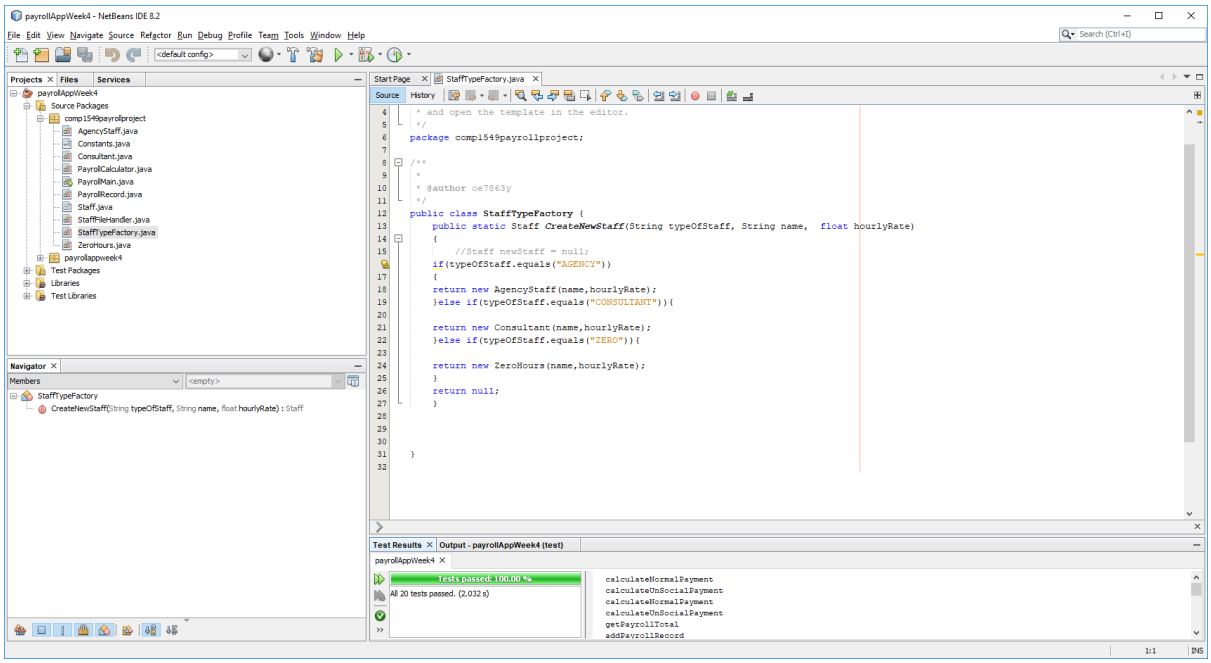
1.1 Student name	**Oscar Eko Osa**
1.2 Who did you work with? Name and/or id	**Emeka Chimezie**
1.3 Which lab topic does this document relate to?	Design Patterns
1.4 How well do you feel you have done?	• I have completed most of it though there is room for improvement •
1.5 Briefly explain your answer to question 1.4	

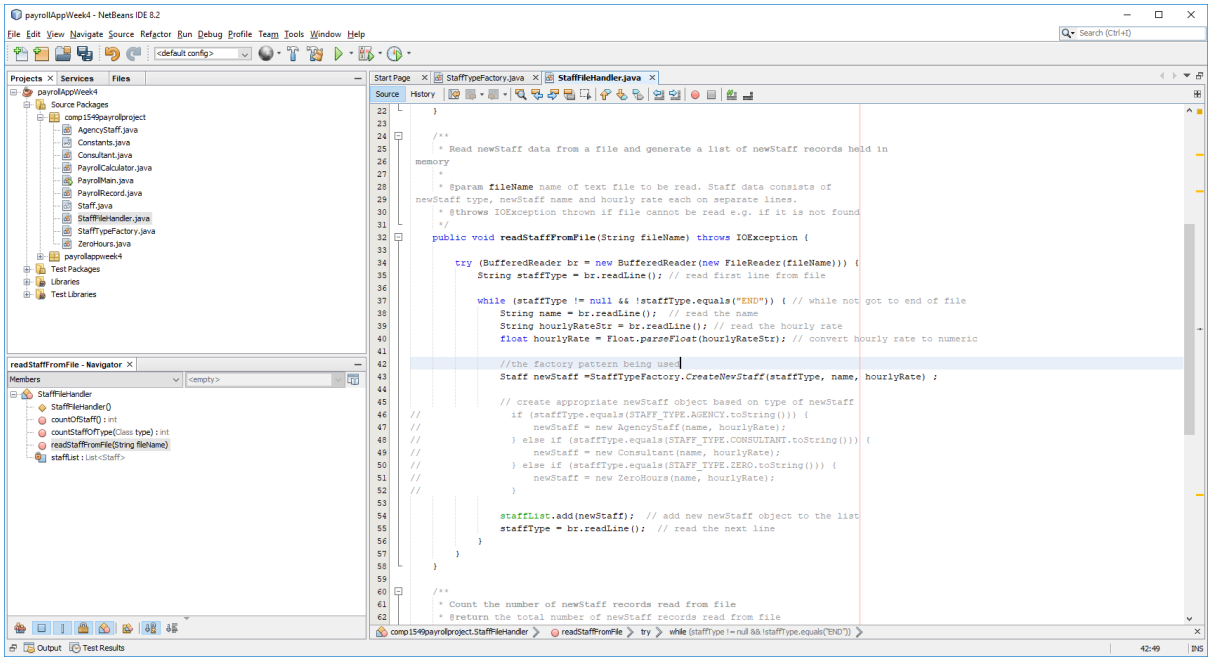
1. **Implementation**

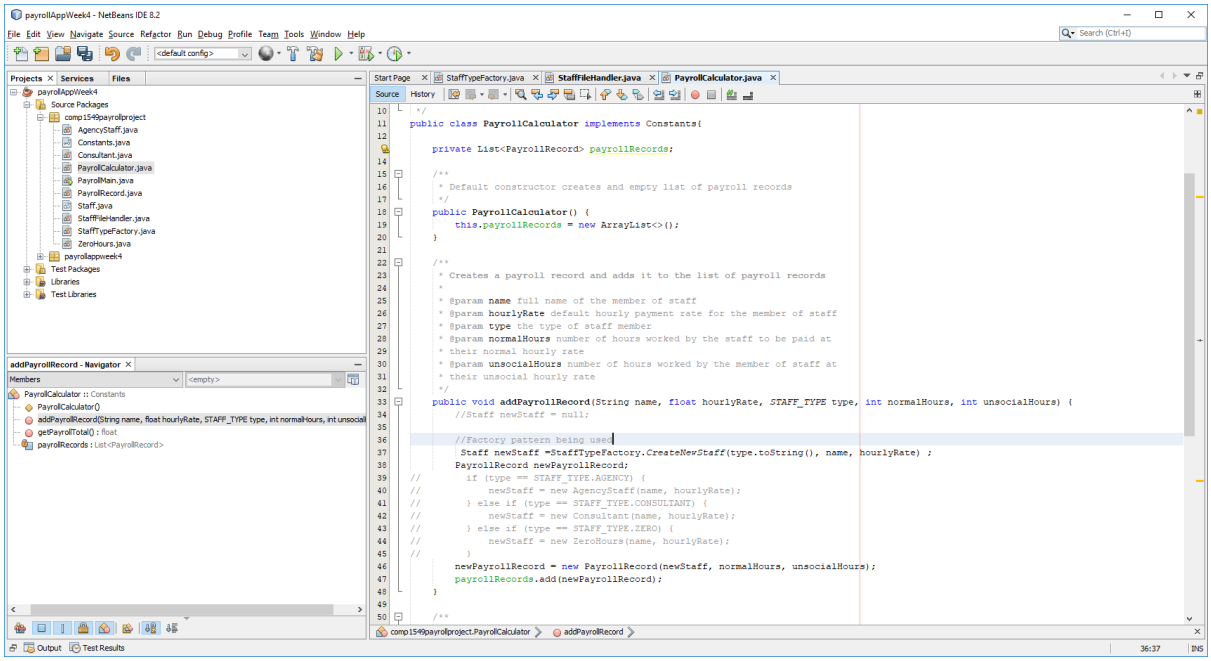
2. 2.1 Annotated screen shots demonstrating what you have achieved.

a)

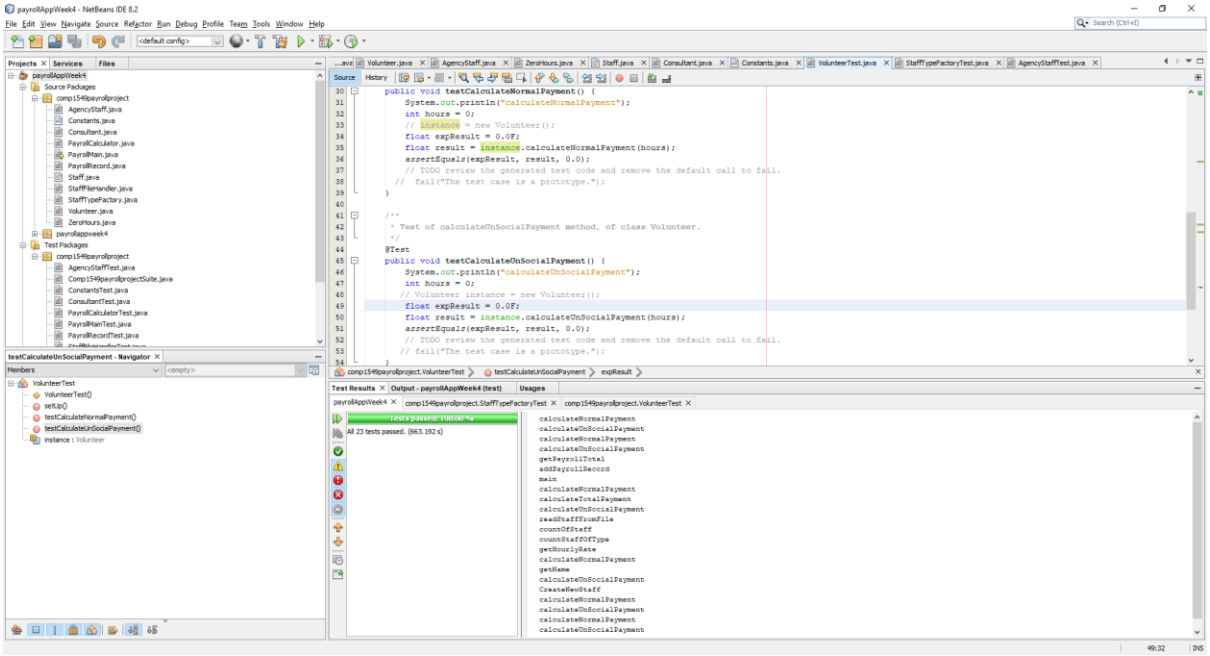




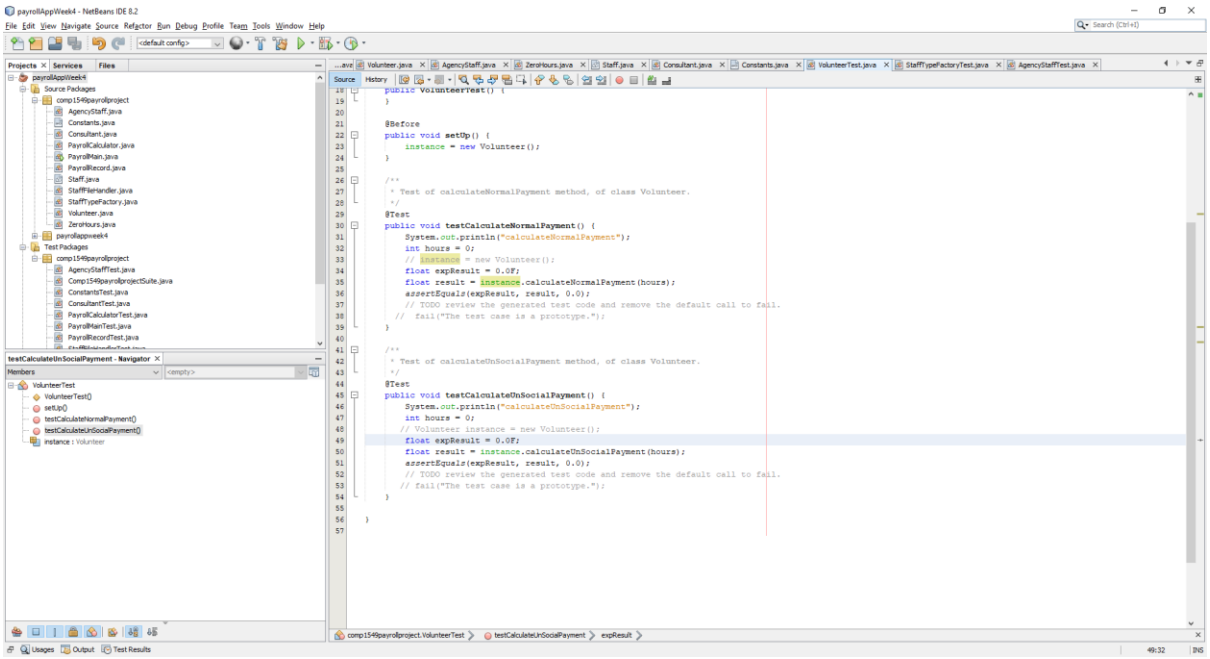

payrollAppWeek4 - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Services Files
payrollAppWeek4
Source Packages
comp1549payrollproject
AgencyStaff.java
Constants.java
Consultant.java
PayrollCalculator.java
PayrollMain.java
PayrollRecord.java
Staff.java
StaffFileHandler.java
StaffTypeFactory.java
ZeroHours.java
payrollappweek4
Test Packages
Libraries
Test Libraries
readStaffFromFile - Navigator
Members
StaffFileHandler
StaffFileHandler()
countOfStaff() : int
countStaffOfType(Class type) : int
readStaffFromFile(String fileName)
staffList : List<Staff>
Start Page
StaffTypeFactory.java
StaffFileHandler.java
Source
History
Output
Test Results
42:49
INS


payrollAppWeek4 - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Services Files
payrollAppWeek4
Source Packages
comp1549payrollproject
AgencyStaff.java
Constants.java
Consultant.java
PayrollCalculator.java
PayrollMain.java
PayrollRecord.java
Staff.java
StaffFileHandler.java
StaffTypeFactory.java
ZeroHours.java
payrollappweek4
Test Packages
Libraries
Test Libraries
addPayrollRecord - Navigator
Members
PayrollCalculator :: Constants
PayrollCalculator()
addPayrollRecord(String name, float hourlyRate, STAFF_TYPE type, int normalHours, int unsocialHours)
getPayrollTotal() : float
payrollRecords : List<PayrollRecord>
Start Page
StaffTypeFactory.java
StaffFileHandler.java
PayrollCalculator.java
Source
History
Output
Test Results
36:37
INS

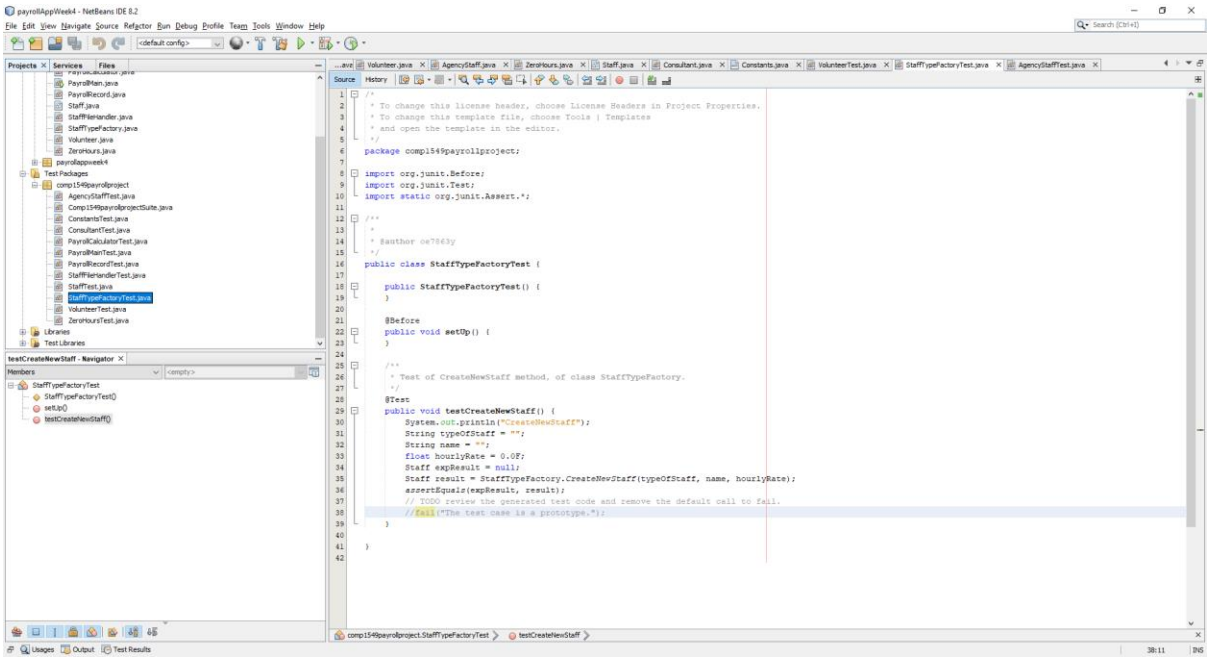
b)



Volunteer test class



Factory test class



2.2 Copy and paste **code that you wrote or amended**. Please **format** it nicely and **make it easy** for the tutor to see and read your code.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package comp1549payrollproject;

/**
 *
 * @author oe7863y
 */

import static comp1549payrollproject.Constants.*;


public class Volunteer extends Staff{

  public Volunteer()
  {
  this("Unknown Volunteer staff",0);
  }

  public Volunteer(String name, float hourlyRate)
  {
  super(name,hourlyRate);
  }
```

```
        @Override
        public float calculateNormalPayment(int hours) {
          // throw new UnsupportedOperationException("Not supported yet."); //To change
      body of generated methods, choose Tools | Templates.

          float normalHours= hours;
         normalHours =0;
          return getHourlyRate()*normalHours ;
        }

        @Override
        public float calculateUnSocialPayment(int hours) {
          // throw new UnsupportedOperationException("Not supported yet."); //To change
      body of generated methods, choose Tools | Templates.
          return getHourlyRate()*hours;
        }

}


package comp1549payrollproject;

import java.util.ArrayList;
import java.util.List;
import static comp1549payrollproject.Staff.*;

/**
 * Class which holds a list of payroll records and calculates the current total
 * payroll bill.
 */
public class PayrollCalculator implements Constants{

    private List<PayrollRecord> payrollRecords;

    /**
     * Default constructor creates and empty list of payroll records
     */
    public PayrollCalculator() {
        this.payrollRecords = new ArrayList<>();
    }

    /**
     * Creates a payroll record and adds it to the list of payroll records
     *
     * @param name full name of the member of staff
     * @param hourlyRate default hourly payment rate for the member of staff
     * @param type the type of staff member
     * @param normalHours number of hours worked by the staff to be paid at
     * their normal hourly rate
```

```
     * @param unsocialHours number of hours worked by the member of staff at
     * their unsocial hourly rate
     */
    public void addPayrollRecord(String name, float hourlyRate, STAFF_TYPE type, int
normalHours, int unsocialHours) {
        //Staff newStaff = null;

        //Factory pattern being used
         Staff newStaff =StaffTypeFactory.CreateNewStaff(type.toString(), name, hourlyRate) ;
        PayrollRecord newPayrollRecord;
//        if (type == STAFF_TYPE.AGENCY) {
//            newStaff = new AgencyStaff(name, hourlyRate);
//        } else if (type == STAFF_TYPE.CONSULTANT) {
//            newStaff = new Consultant(name, hourlyRate);
//        } else if (type == STAFF_TYPE.ZERO) {
//            newStaff = new ZeroHours(name, hourlyRate);
//        }
        newPayrollRecord = new PayrollRecord(newStaff, normalHours, unsocialHours);
        payrollRecords.add(newPayrollRecord);
    }

    /**
     * Calculate and return the total current payroll bill
     *
     * @return the current total payroll bill including both normal and unsocial
     * hours worked based on all the records in the payroll records list.
     */
    public float getPayrollTotal() {
        float total = 0.0F;
        for (PayrollRecord pr : payrollRecords) {  // loop through all the records in the list
            total += pr.calculateTotalPayment();
        }
        return total;
    }
}
```

COMP1549 Logbook Upload Template 2018/19

Complete this document for each of the four logbook exercises to include in your final report.

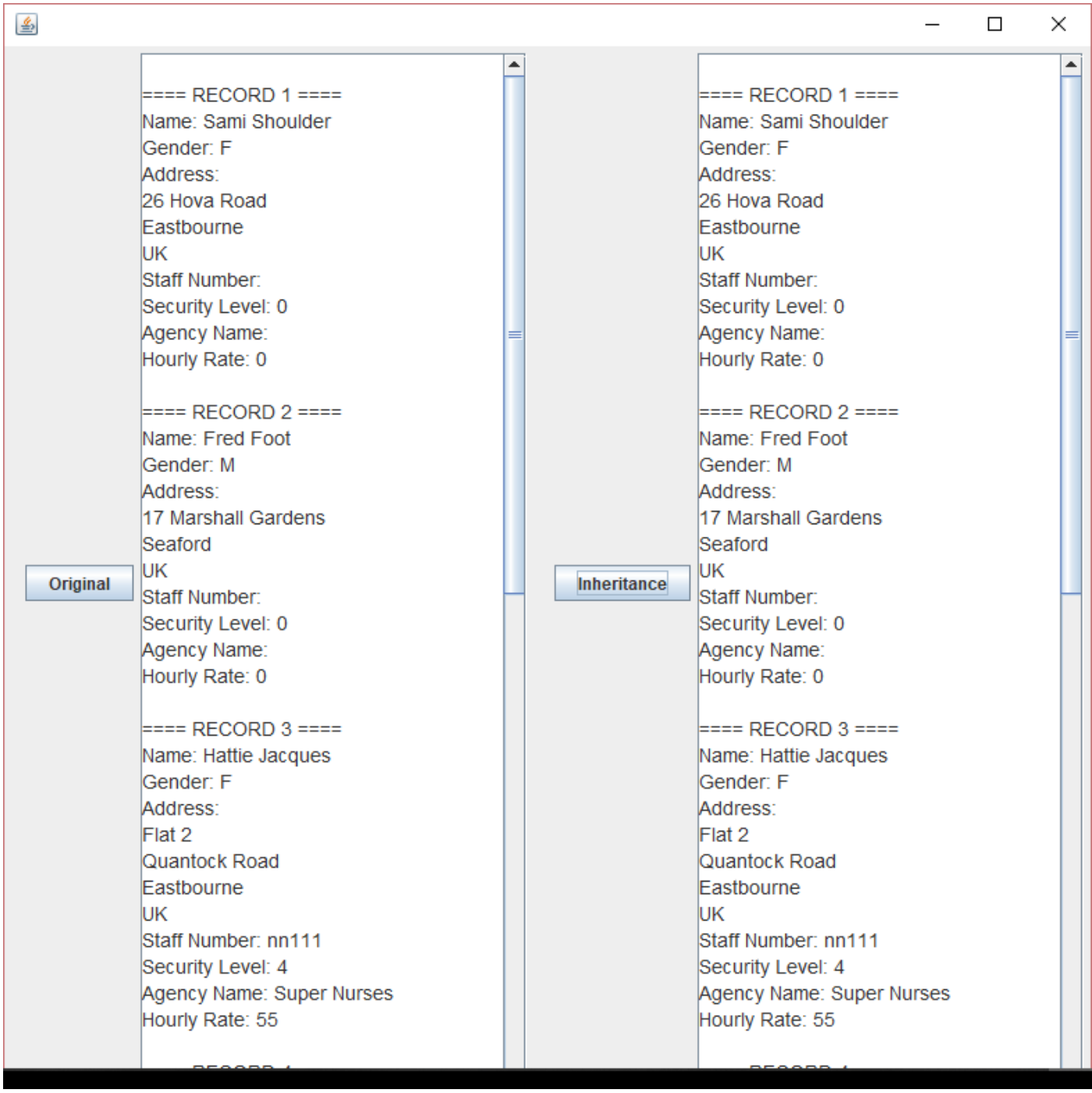
Basic Information

1.1 Student name	**Oscar Eko Osa**
1.2 Who did you work with? Name and/or id	**Emeka Chimezie**
1.3 Which lab topic does this document relate to?	1. Inheritance
1.4 How well do you feel you have done?	• I have completed the exercise and am totally satisfied with my work

1.5	Briefly explain your answer to question 1.4	

3. Implementation

2.1 Annotated screen shots demonstrating what you have achieved.



2.2 Copy and paste **code that you wrote or amended**. Please **format** it nicely and **make it easy** for the tutor to see and read your code.

```
package uk.ac.gre.comp1549.hs;

/*
 * Prototype class representing a member of staff directly employed
 * by the hospital
 */
public class DirectEmployee extends AgencyStaff{

  /* private String firstName;
    private String familyName;
    private char gender;
    private PostalAddress postalAddress;
    private String staffNumber;
    private int securityLevel;*/
    private String grade;
    private int salary;

    public DirectEmployee(String grade, int salary, String staffNumber, int securityLevel,
String firstName, String familyName, char gender, PostalAddress postalAddress) {
        super.firstName = firstName;
        super.familyName = familyName;
     super.gender = gender;
        super.postalAddress = postalAddress;
        super.staffNumber = staffNumber;
        super.securityLevel = securityLevel;
        this.grade = grade;
        this.salary = salary;
    }

    public DirectEmployee() {
        super.firstName = "";
       super.familyName = "";
        super.gender = 'U';
        super.postalAddress = new PostalAddress();
        super.staffNumber = "";
        this.grade = "";
    }

   /* public String getFirstName() {
        return firstName;
    }

    public String getFamilyName() {
        return familyName;
    }
```

```
    public char getGender() {
        return gender;
    }

    public PostalAddress getPostalAddress() {
        return postalAddress;
    }

    public String getStaffNumber() {
        return staffNumber;
    }

    public int getSecurityLevel() {
        return securityLevel;
    }*/

    public String getGrade() {
        return grade;
    }

    public int getSalary() {
        return salary;
    }

}
```

```
package uk.ac.gre.comp1549.hs;

/*
 * Prototype class representing a patient of the hospital
 */
public class Patient extends AgencyStaff{

  /*  private String firstName;
    private String familyName;
    private char gender;
    private PostalAddress postalAddress;*/
    private String patientId;
    private String doctorStaffNumber;

    public Patient(String patientId, String doctorStaffNumber, String firstName, String
familyName, char gender, PostalAddress postalAddress) {
        super.firstName = firstName;
        super.familyName = familyName;
        super.gender = gender;
        super.postalAddress = postalAddress;
        this.patientId = patientId;
        this.doctorStaffNumber = doctorStaffNumber;
    }
```

```
    public Patient() {
        super.firstName = "";
        super.familyName = "";
        super.gender = 'U';
        super.postalAddress = new PostalAddress();
        this.patientId = "";
        this.doctorStaffNumber = "";
    }

    /* public String getFirstName() {
        return firstName;
    }

    public String getFamilyName() {
        return familyName;
    }

    public char getGender() {
        return gender;
    }

    public PostalAddress getPostalAddress() {
        return postalAddress;
    }*/

    public String getPatientId() {
        return patientId;
    }

    public String getDoctorStaffNumber() {
        return doctorStaffNumber;
    }
}
```

Documents of the logbooks have been attached along with this document.