Advanced Programming: Practical Coursework

Nkem Akwari: 000978914

Trevor Kiggundu: 001001720

A report submitted in fulfilment of the requirements for the module, Advanced Programming, Computing and Information Systems Department, University of Greenwich.

Table of Contents

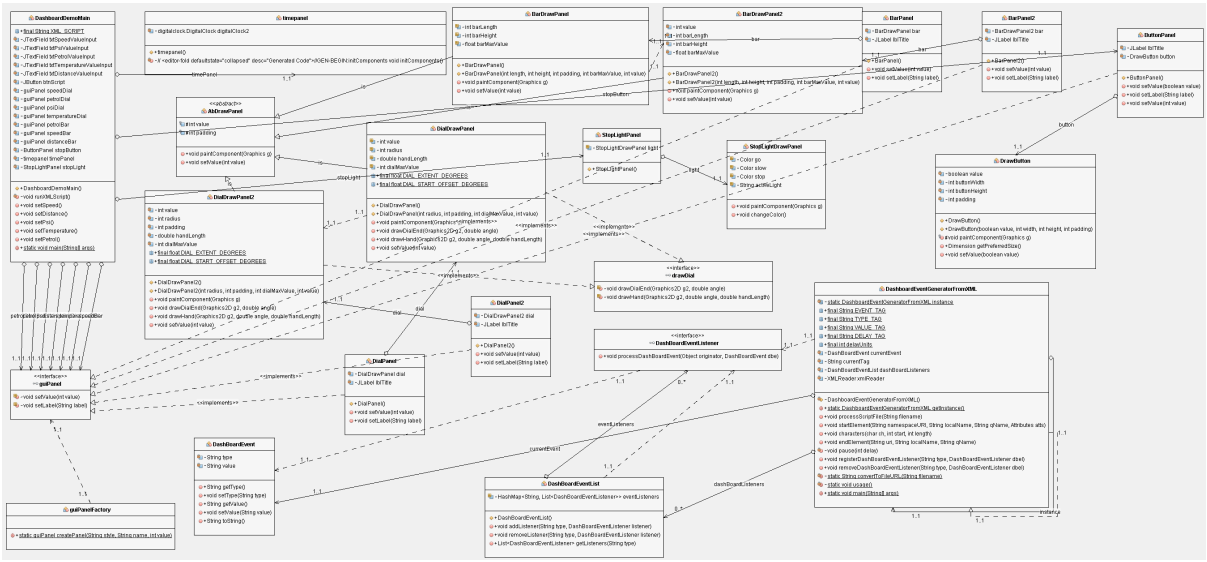
Section 1 – Brief statement of completed levels:

Which type of dashboard application did you implement?	Train
1.1 Circle the parts of the coursework you have fully completed and are fully working	1a, 1b, 2a, 2bi, 2bii, 3, 4a, 4bi, 5a, 5b
1.2 Circle the parts of the coursework you have partly completed or are partly working	4bii, 6a, 6b

Briefly explain your answer if you circled any parts in 1.2:

- 4bii: We were only able to implement 2 design patterns in our program.
- 6a, 6b: We were not able to complete the level 6 requirements.

Section 2 – UML class diagram:

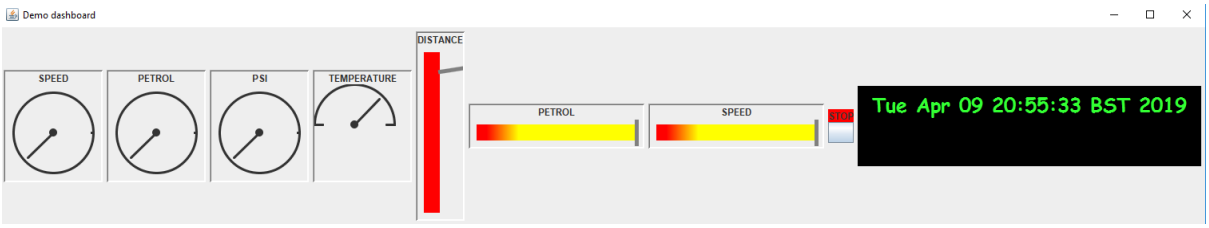


Section 2.5 – User documentation regarding opening the NetBeans Project:

Running the project:

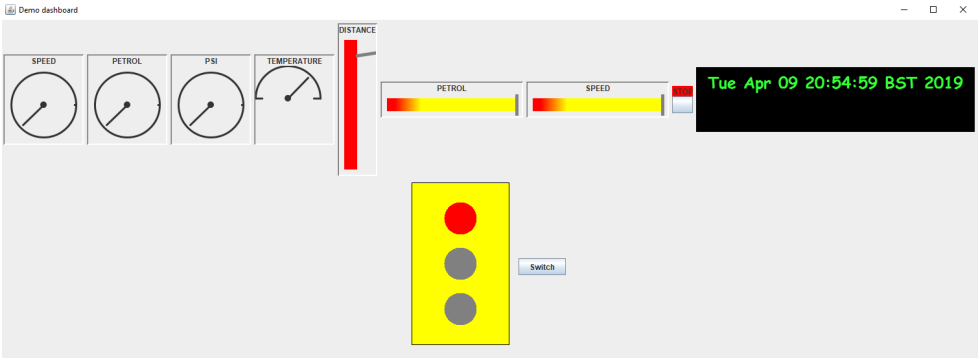
When the user presses ‘run’, the program will show most of the dashboard but not all of it:

Half-dashboard:



The user should then expand the frame to see the full dashboard with the stoplight:

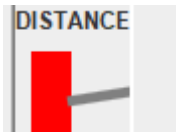
Full Dashboard:



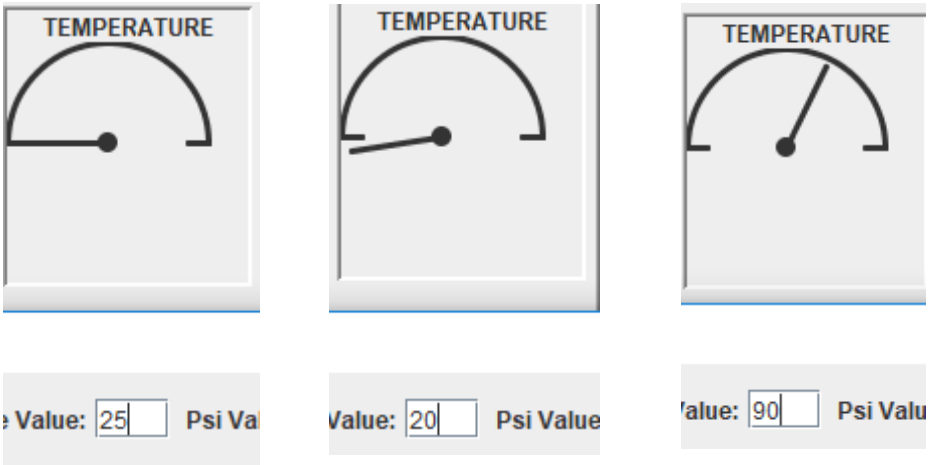
Section 3 – Concise list of bugs and weaknesses:

3.1 Bugs:

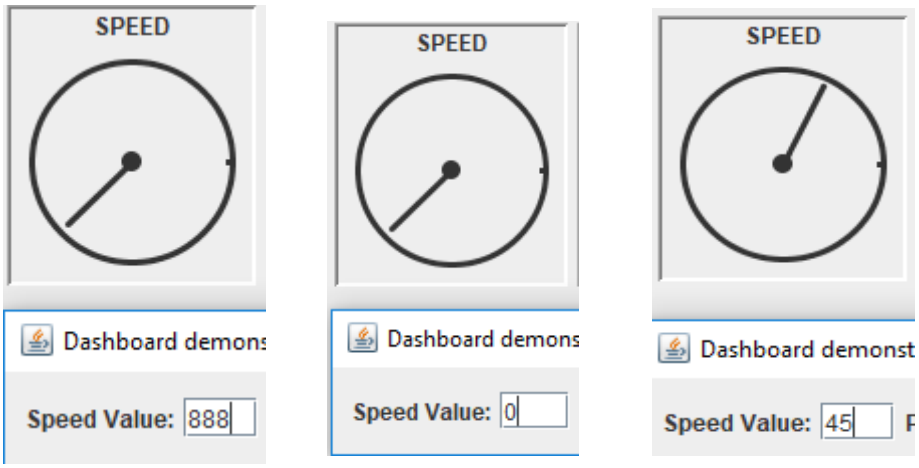
- Distance bar value indicator only works from time to time. If the program is run too many times without restarting the IDE, then it becomes offset.



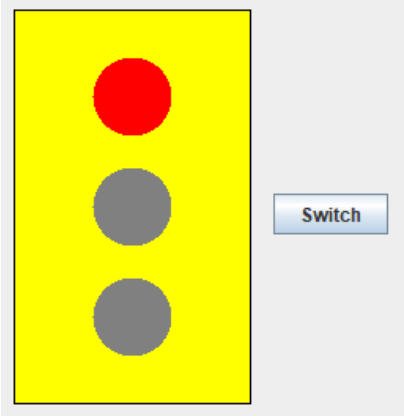
- Half-Dial: Entering a number under 25 will cause the dial to over-extend past the zero marker: However, values from 25-100 work well.

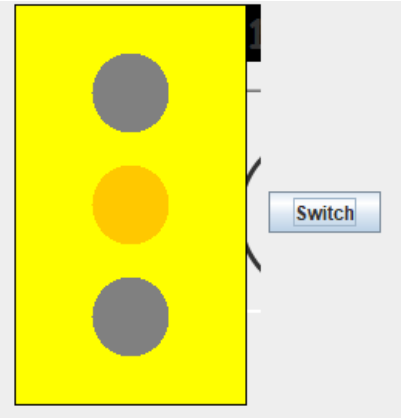


- Dial: Entering any 3-figure number causes the dials to overextend and almost return to 0, a feature which would confuse the user if they were not paying attention. However, values from 0-99 work well.



- Traffic Light: Switching the traffic light colours displays a weird black graphic behind the traffic light. This is purely a graphical error as it does not change the actual functionality of the traffic light.





- Stop Button: Stop button was working initially as a means of ending the program in addition to exiting on close but stopped working and is only there for display now/level 1 and 2 requirements.

3.2 Weaknesses:

- Design Patterns: We were only able to implement 2 design patterns into our program. Our attempt at MVC fell flat completely so we decided not to include it in the program at all to avoid errors.

- Level 6: We were not able to implement a game and/or threading to make the dials run smoother.

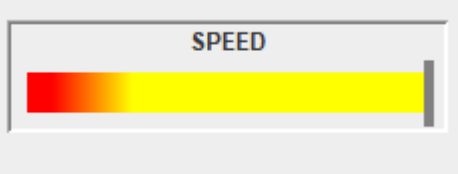
- Our GUI could have been more complex, we ended up adapting the one Zena gave us. If given more time, we would have tried to make it more realistic using our JavaFX application.

Section 4 - Documentation of each of the completed levels:

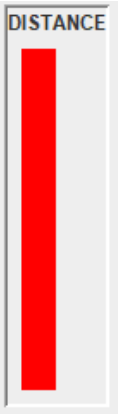
4.1 Level 1:

We were grateful that Zena supplied with example code as this gave us a place to start brainstorming ideas:

We added a speed bar so that the user could also track that in accordance to the speed dial. Both the dial and bar run with the XML script provided:

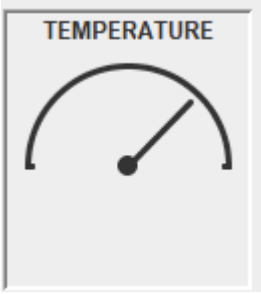


We also added a distance bar that increases in value as the program runs. This bar is also run by a user input value and runs with the XML script provided:

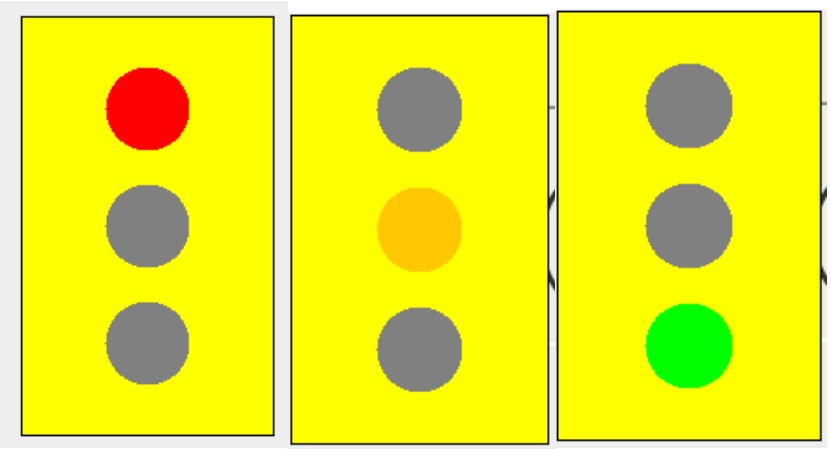


A PSI dial was added that increases/decreases as some of the other parameters change. The user is now able to view the pressure increase as the program runs:

A temperature half dial is added to show the change in temperature as the program runs:



Finally, a traffic light is added to complete the level 1 requirements. This also helps the user distinguish our train design from a plane design, as traffic light controls are not used on aircraft.



Word Count: 158

4.2 Level 2:

Object Oriented Features Implemented:

- Inheritance: We implemented inheritance into our program by allowing the ‘speed’, ‘petrol’ and ‘PSI’ dials to all inherit their characteristics from the ‘DialDrawPanel’ class. We also used the same common logic from the ‘DialDrawPanel’ class to be extracted to create a separate class called ‘DialDrawPanel2’, with its unique logic helping to meet specification requirements such as having a ‘half-dial’ on our JPanel. This child class was used to create the ‘temperature’ half dial. We also implemented this into our program when creating the vertical ‘distance’ bar shown below. The horizontal ‘Petrol’ bar provided in the sample code from Zena, and the duplicate ‘Speed’ bar we created both shared the same parameters regarding the dimensions such as length, height, padding and the maximum bar value. These were made in the ‘BarDrawPanel’ class. We created a separate class called ‘BarDrawPanel2’ where we made different dimensions for those specific parameters using the same private variables (length, barLength, barHeight, padding, barMaxValue) that would not be affected by the ‘BarDrawPanel’ class.

  BarDrawPanel:

```
public BarDrawPanel() {
    this(200, 20, 8, 100, 0);
}
```

```
Rectangle2D barx = new Rectangle2D.Double(padding, padding, barLength, barHeight);
GradientPaint redtoyellow = new GradientPaint(0 + (float) barx.getWidth() * 0.1F, 0, Color.RED, (float) barx.getWidth() * 0.3F, 0, Color.YELLOW);
g2.setPaint(redtoyellow);
g2.fill(barx);
```

```
g2.setStroke(new BasicStroke(barLength/40, BasicStroke.CAP_SQUARE, 0));
g2.setPaint(Color.GRAY);
Line2D valueIndicator = new Line2D.Double(padding + (barLength * value / barMaxValue), padding/2F, padding + (barLength * value / barMaxValue), barHeight + (padding * 1.5F));
g2.draw(valueIndicator);
```

  BarDrawPanel2:

```
public BarDrawPanel2() {
    this(20, 200, 8, 100, 0);
}
```

```
Rectangle2D barx = new Rectangle2D.Double(padding, padding, barLength, barHeight);
GradientPaint redtored = new GradientPaint(0 + (float) barx.getWidth() * 0.1F, 0, Color.RED, (float) barx.getWidth() * 0.3F, 0, Color.RED);
g2.setPaint(redtored);
g2.fill(barx);
```

```
g2.setStroke(new BasicStroke(barHeight/40, BasicStroke.CAP_SQUARE, 0));
  g2.setPaint(Color.GRAY);
  Line2D valueIndicator = new Line2D.Double( padding + (barHeight * value / barMaxValue), padding/2F, padding + (barLength * value / barMaxValue), barLength + (padding * 1.5F));
  g2.draw(valueIndicator);
```

- Interfaces:
    1. guiPanel: We created the ‘guiPanel’ interface in an attempt to enhance the flexibility of the program in regard to creating the bars shown on the panel. Many of the parameters were already inherited as described in the bullet point above, so we aimed to make it less complicated to create a type of display (ex; bar) by separating them into styles. With the creation of the JPanel also came the creation of a class called ‘guiPanelFactory’, that references the ‘guiPanel’. If say a style if equal to our ‘bar1’ style, then that specific style is identified as that object. This was created in relation to the ‘BarDrawPanel’ and ‘BarDrawPanel2’ classes.

       guiPanel:

```
public interface guiPanel {
      void setValue(int value);
     void setLabel(String label);
}
```

       guiPanelFactory:

```
public class guiPanelFactory {
    public static guiPanel createPanel(String style, String name, int value) {
        guiPanel uiPanel = null;
        if (name == null) {
            return null;
        } else if (style == "barl") {

            uiPanel = new BarPanel();
            uiPanel.setLabel(name);
            uiPanel.setValue(value);
        }
        else if (style == "bar2") {

            uiPanel = new BarPanel2();
            uiPanel.setLabel(name);
            uiPanel.setValue(value);
        }
         else if (style == "diall") {

            uiPanel = new DialPanel();
            uiPanel.setLabel(name);
            uiPanel.setValue(value);
        }
         else if (style == "dial2") {

             uiPanel = new DialPanel2();
            uiPanel.setLabel(name);
            uiPanel.setValue(value);
        }
return uiPanel;

}
}
```

2. drawDial: We created this method for the 'DrawDialPanel' and 'DrawDialPanel2' classes to replicate what the 'guiPanel' was created to do for the 'BarDrawPanel' and 'BarDrawPanel2' classes.

drawDial interface:

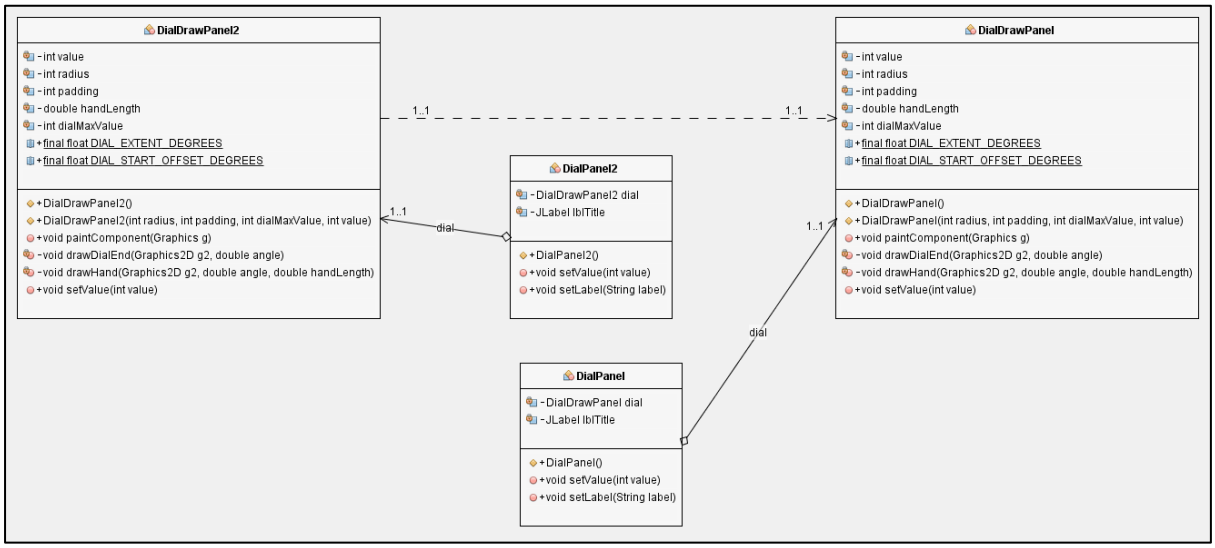
```
public interface drawDial {

    void drawDialEnd(Graphics2D g2, double angle);

/**
     * Draw the hand on the dial to indicate the current value
     * @param g2 - graphics object used to draw on the JPanel
     * @param  angle - the angle on the dial at which the hand is to point
     */
    void drawHand(Graphics2D g2, double angle, double handLength);
}
```

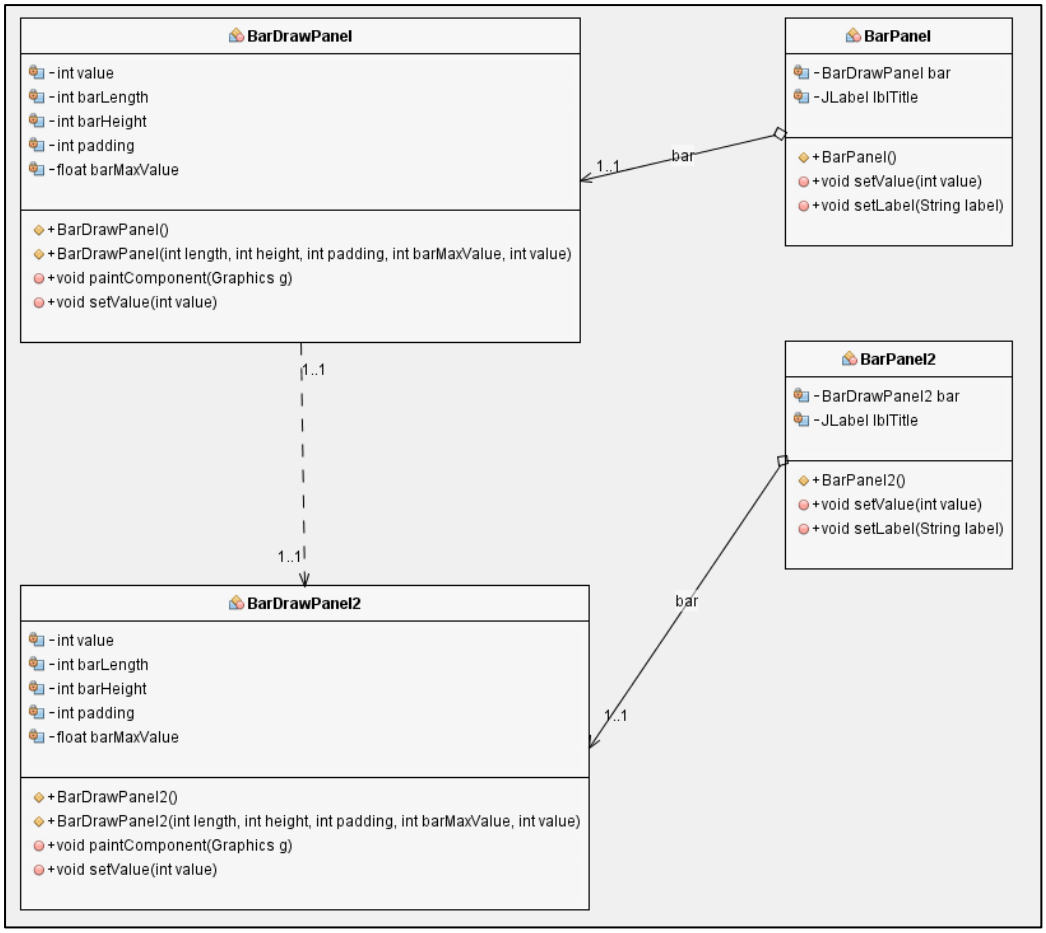
‘DrawDialPanel’ and ‘DrawDialPanel2’ implementation:

```
public class DialDrawPanel extends AbDrawPanel implements drawDial{

    int value; // current value - where the hand will point

    int radius; // radius of dial
    double handLength; // length of indicator hand
    int dialMaxValue;  // dial runs from 0 to dialMaxValue
```

```
public class DialDrawPanel2 extends AbDrawPanel implements drawDial{

    private int value; // current value - where the hand will point

    private int radius; // radius of dial
    private int padding; // padding outside the dial
    private double handLength; // length of indicator hand
    private int dialMaxValue;  // dial runs from 0 to dialMaxValue
```

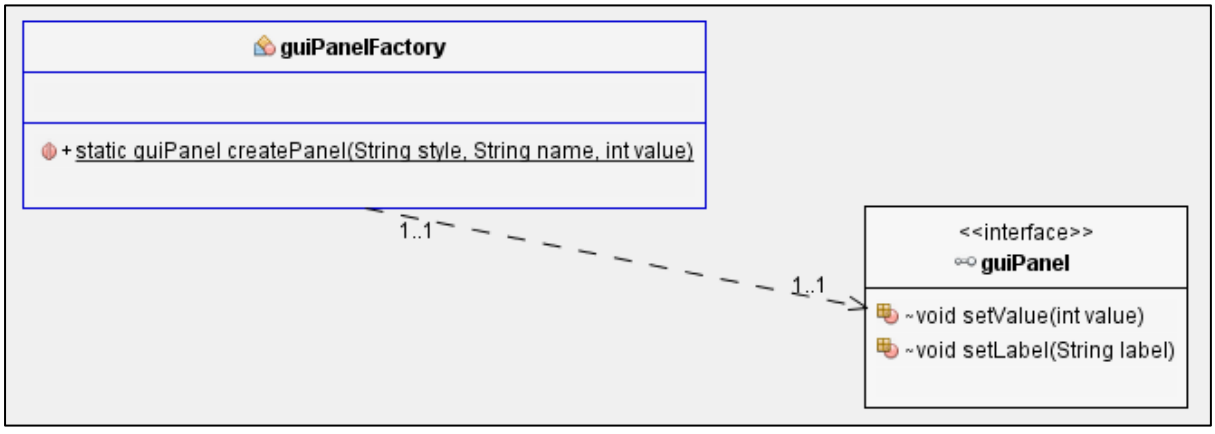
4.2.1 Give a UML class diagram for the inheritance hierarchy(s) you implemented:



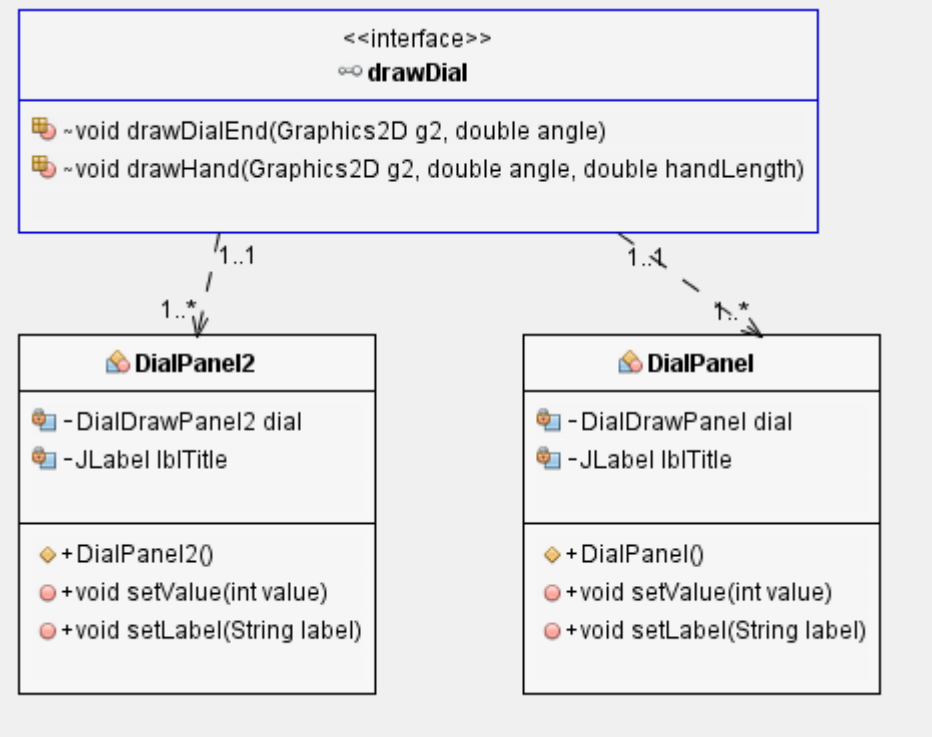


4.2.2 Give a UML class diagram for the interfaces:

guiPanel:



drawDial:



The object-oriented features we added were interfaces and inheritance. The interfaces we added decreased the number of instances of the 'drawDial' and 'guiPanel' classes, making the program less messy and easier to understand if it were to be continued by another individual. Inheritance also makes the program less messy, as it reduces the amount of duplicate code in the program, something we had a lot of as a lot of the parameters for the bars/dials were the same. This caused a fair bit of confusion sometimes as a lot of our classes were also similarly named (ex: 'BarDrawPanel' and 'BarDrawPanel2'), so it helped eliminate earlier error that we were making because of that.

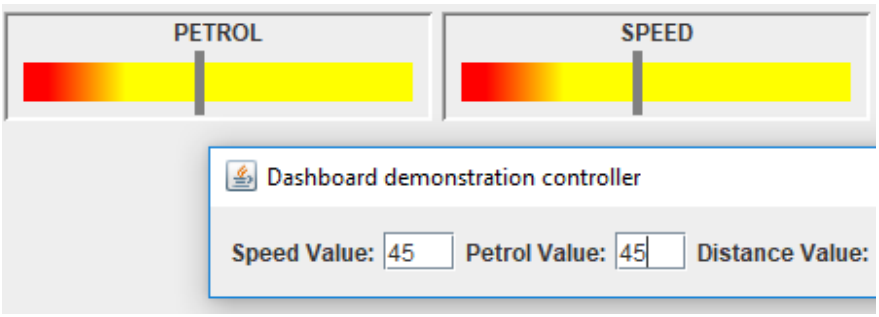
Word Count: 113

4.3 Level 3:

White Box and JUnit Testing:

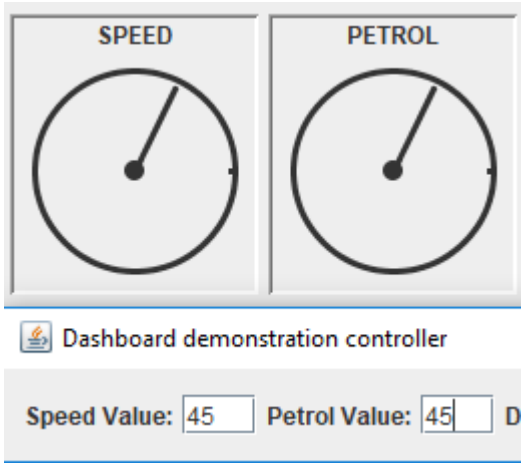
Class(es) Tested:	Test:	Expected Result	Actual Result	Did it work?
DashboardDemoMain BarPanel BarDrawPanel	Display 45 on Speed and Petrol bars	Successful display of both numbers	Successful display of both numbers	Yes

Screenshot:



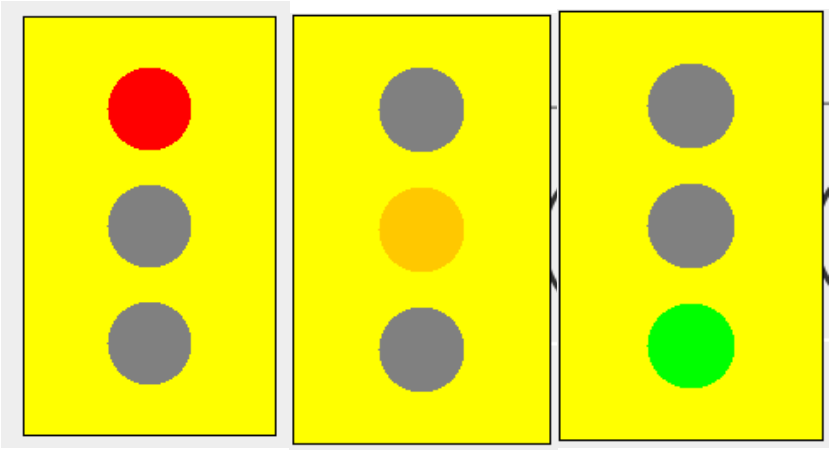
Class(es) Tested:	Test:	Expected Result	Actual Result	Did it work?
DashboardDemoMain DialPanel DialDrawPanel	Display 45 on Speed and Petrol dials	Successful display of both numbers	Successful display of both numbers	Yes

Screenshot:



Class(es) Tested:	Test:	Expected Result	Actual Result	Did it work?
DashboardDemoMain StopLightPnael StopLighDrawtPnael	Change color of lights on stop light	Successful change of colors	Successful change of colors	Yes

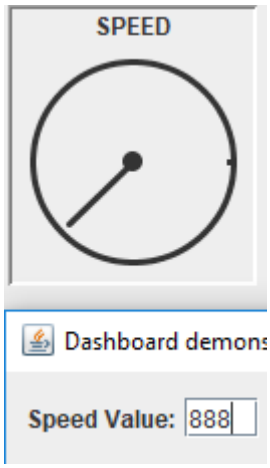
Screenshot:

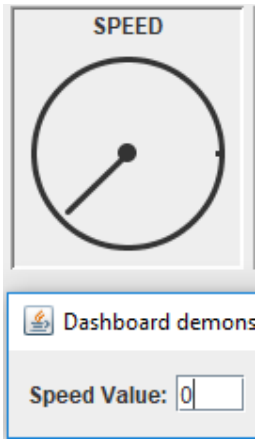


Class(es) Tested:	Test:	Expected Result	Actual Result	Did it work?
DashboardDemoMain DialPanel DialDrawPanel	Check for proper dial function	Inserting realistic/ unrealistic values will not overextend the dials	Overextended dials	No

Screenshot:

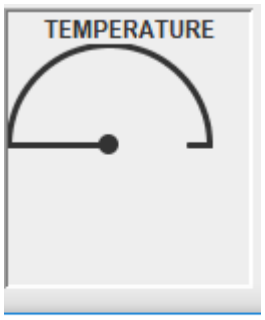
Dials:

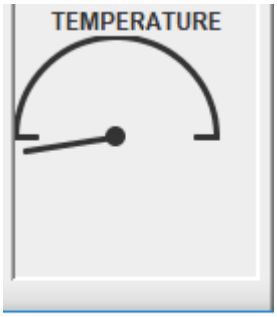




Class(es) Tested:	Test:	Expected Result	Actual Result	Did it work?
DashboardDemoMain DialPanel2 DialDrawPanel2	Check for proper dial function	Inserting realistic/ unrealistic values will not overextend the dials	Overextended dials	No

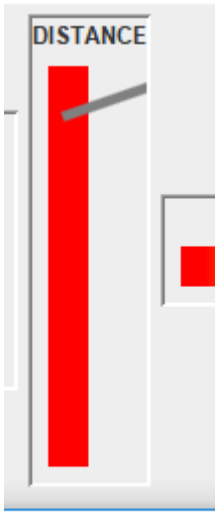
Screenshots:





Class(es) Tested:	Test:	Expected Result	Actual Result	Did it work?
DashboardDemoMain BarPanel2 BarDrawPanel2	Display 45 on distance bar	Successful display of both numbers	Bar moving sideways and not completing the expected result.	Partially

Screenshot:



Class(es) Tested:	Test:	Expected Result	Actual Result	Did it work?
DashboardDemoMain BarPanel	setLabel setValue	setLabel and setValue methods of the BarPanel class working correctly.	setLabel and setValue methods of the BarPanel class working correctly.	Yes

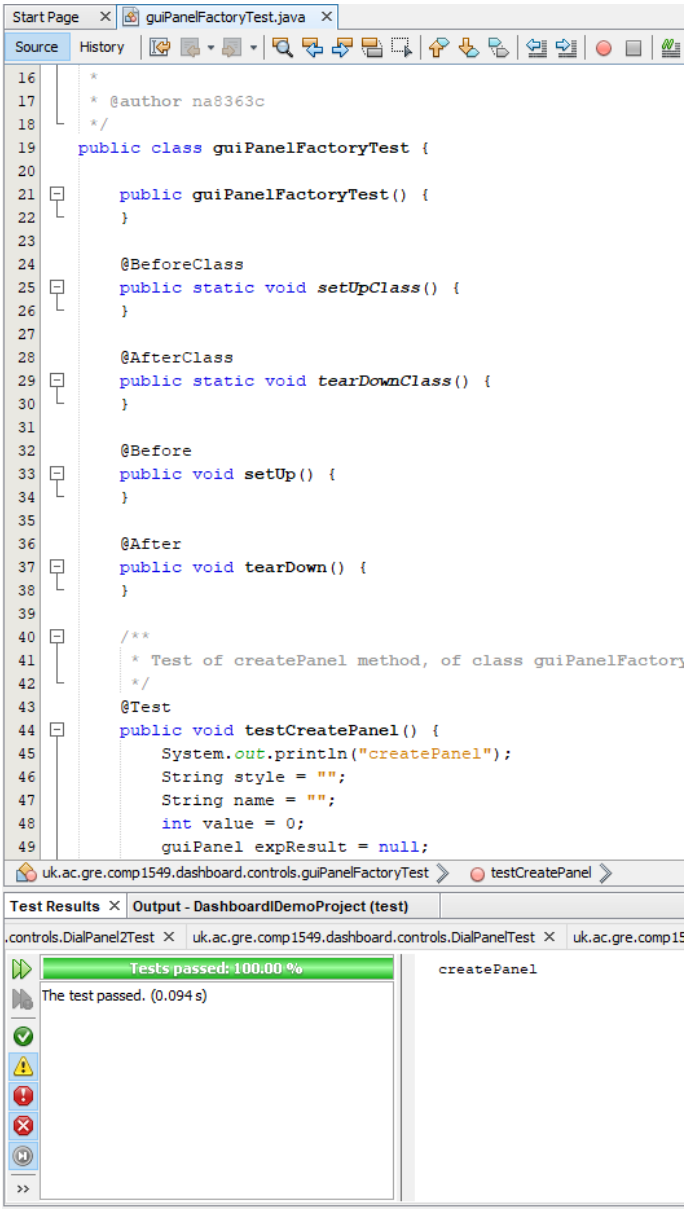
Screenshot:

```
public class BarPanelTest {

    public BarPanelTest() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }

    @Before
    public void setUp() {
    }

    @After
    public void tearDown() {
    }

    /**
     * Test of setValue method, of class BarPanel.
     */
    @Test
    public void testSetValue() {
        System.out.println("setValue");
        int value = 0;
        BarPanel instance = new BarPanel();
        instance.setValue(value);
        // TODO review the generated test code and remove the def
    }

    /**
```

```
Results × | Output - DashboardIDemoProject (test)
.gre.comp1549.dashboard.controls.AbDrawPanelTest × | uk.ac.gre.comp1549.dashboard.controls.BarDrawPa
Tests passed: 100.00 %
oth tests passed. (0.187 s)
setLabel
setValue
```

Class(es) Tested:	Test:	Expected Result	Actual Result	Did it work?
DashboardDemoMain DialPanel2	setLabel setValue	setLabel and setValue methods of the DialPanel2 class working correctly.	setLabel and setValue methods of the DialPanel2 class working correctly.	Yes

Class(es) Tested:	Test:	Expected Result	Actual Result	Did it work?
DashboardDemoMain guiPanelFactory	createPanel	createPanel methods of the guiPanelFactory class are working correctly.	createPanel methods of the guiPanelFactory class are working correctly.	Yes



Class(es) Tested:	Test:	Expected Result	Actual Result	Did it work?
DashboardDemoMain guiPanelTest	setLabel setValue	setLabel and setValue methods of the guiPanelTest class working correctly.	setLabel and setValue methods of the guiPanelTest class working correctly.	Yes

```
public class guiPanelTest {

    public guiPanelTest() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }

    @Before
    public void setUp() {
    }

    @After
    public void tearDown() {
    }

    /**
     * Test of setValue method, of class guiPanel.
     */
    @Test
    public void testSetValue() {
        System.out.println("setValue");
        int value = 0;
        guiPanel instance = new guiPanelImpl();
        instance.setValue(value);
        // TODO review the generated test code and remove the default call to fail.
       // fail("The test case is a prototype.");
    }
```

```
uk.ac.gre.comp1549.dashboard.controls.guiPanelTest > testSetLabel >
Results × Output
:.gre.comp1549.dashboard.controls.guiPanelTest ×
Tests passed: 100.00 %
Both tests passed. (0.125 s)
setLabel
setValue
```

Class(es) Tested:	Test:	Expected Result	Actual Result	Did it work?
DashboardDemoMain drawDialTest	drawHand drawDialHand	drawHand and drawDialHand methods of the drawDialTest class working correctly.	drawHand and drawDialHand methods of the drawDialTest class working correctly.	Yes

```
/**
 *
 * @author na8363c
 */
public class drawDialTest {

    public drawDialTest() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }

    @Before
    public void setUp() {
    }

    @After
    public void tearDown() {
    }

    /**
     * Test of drawDialEnd method, of class drawDial.
     */
    @Test
    public void testDrawDialEnd() {
        System.out.println("drawDialEnd");
        Graphics2D g2 = null;
        double angle = 0.0;
        drawDial instance = new drawDialImpl();
```

uk.ac.gre.comp1549.dashboard.controls.drawDialTest > testDrawHand >

Results | Output

.gre.comp1549.dashboard.controls.guiPanelTest | uk.ac.gre.comp1549.dashboard.controls.DrawButtonTest | uk.ac.gre.com

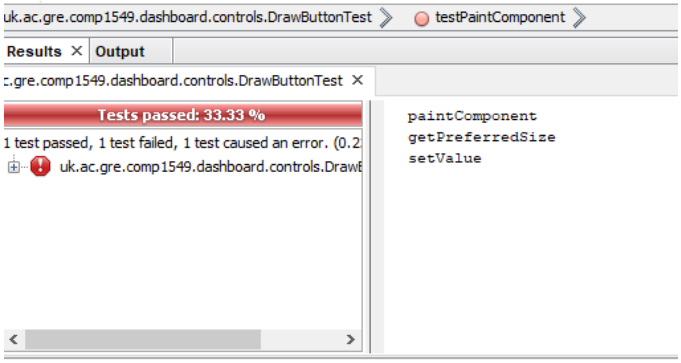
Tests passed: 100.00 %

Both tests passed. (0.094 s)

```
drawHand
drawDialEnd
```

Class(es) Tested:	Test:	Expected Result	Actual Result	Did it work?
DashboardDemoMain drawButtonTest	paintComponent getPrefferedSize setValue	paintComponent, getPrefferedSize and setValue methods of the drawButtonTest class working correctly.	paintComponent, getPrefferedSize and setValue methods of the drawButtonTest class working correctly.	No/Partially

```
import java.awt.Dimension;
import java.awt.Graphics;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;

/**
 *
 * @author na8363c
 */
public class DrawButtonTest {

    public DrawButtonTest() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }

    @Before
    public void setUp() {
    }

    @After
    public void tearDown() {
    }
```

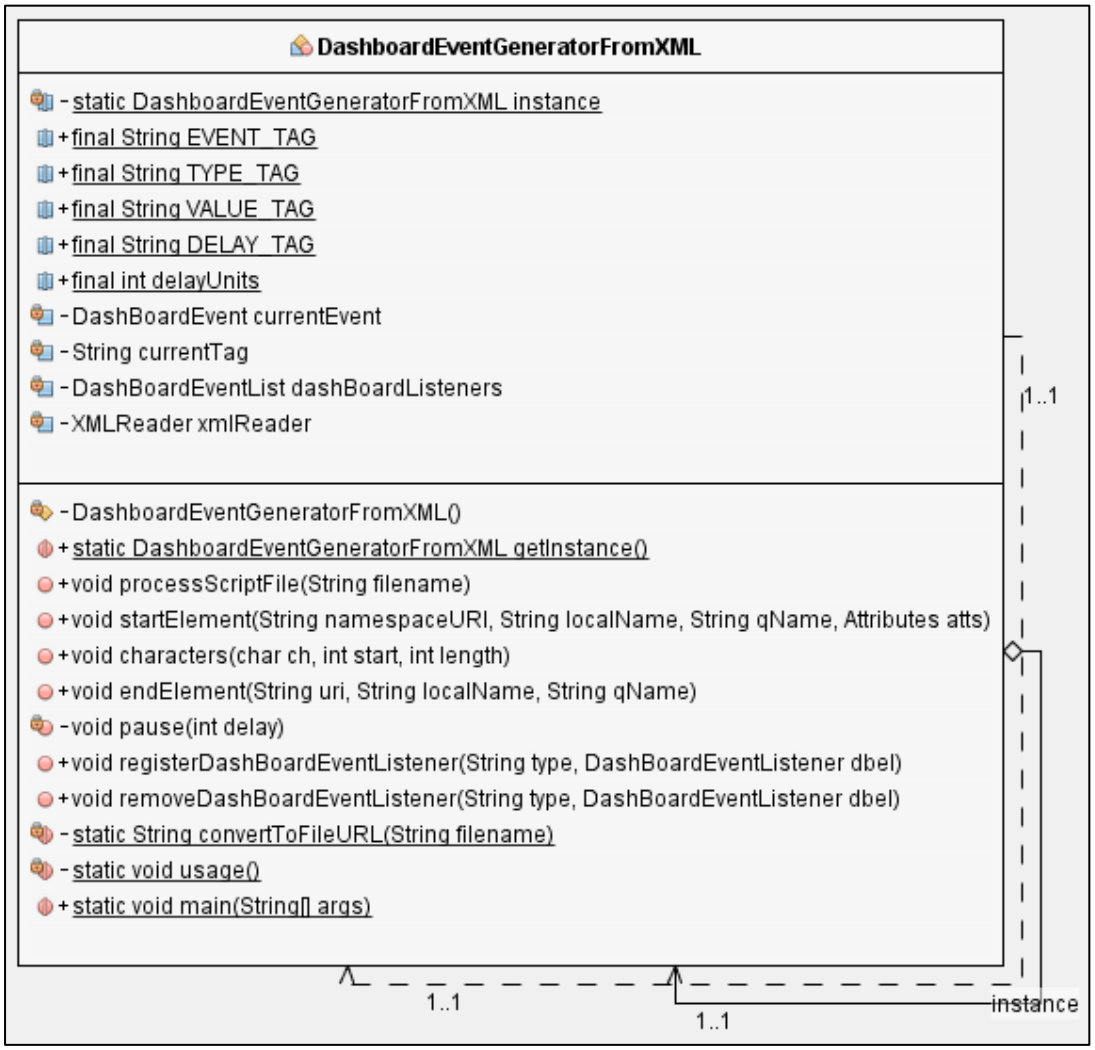


4.4 Level 4:

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.



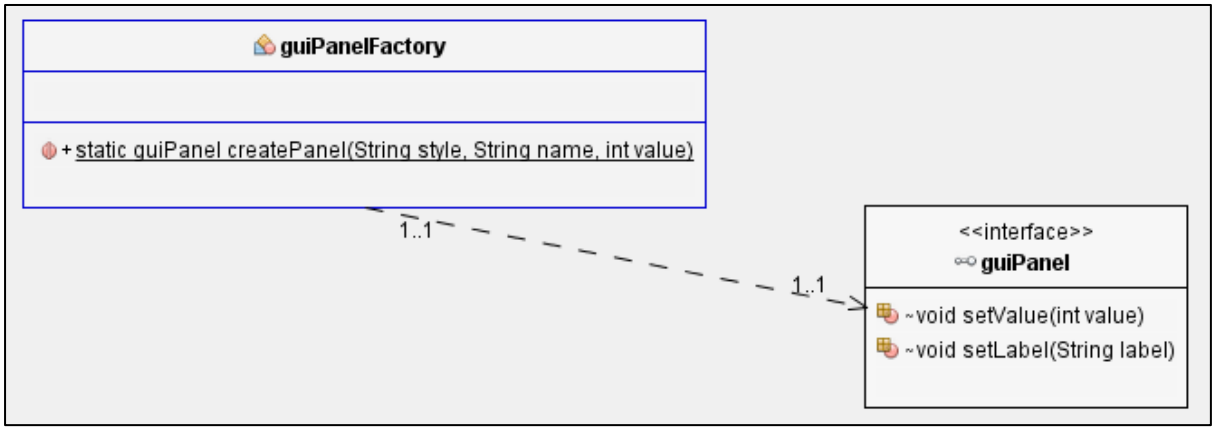
The singleton design pattern we implemented made sure that there was only one instance of the 'DashboardEventGeneratorFromXML' class as shown in the diagram above. This improved the design and flexibility of the program, as any future changes to this class regarding instantiation are controlled by the class alone, allowing for easier instance control as it only needs to be initialised once. In extreme cases, one can also say that it aids in better memory usage.

Word Count: 75

4.4.2 Pattern 2

Name: *Factory*

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



We created an interface named ‘guiPanel’ which is referenced and implemented by the ‘guiPanelFactory’ java class. This design pattern improves the system because it aids in the creation of the dials/bars created in our GUI. It separated the draw panels we were given and created into ‘styles’ so that if a certain style is chosen, it can easily be identified within the JFrame/JPanel. The factory was used to get the objects of the concrete classes, (‘BarPanel’, ‘BarPanel2’, ‘DialPanel’, ‘DialPanel2’ to pass on information about the type of display (dial or panel) that was to be created. This aids in making the program object oriented, and decreases on the number of instances where bad programming practice such as ‘spaghetti coding’, where the code becomes too messy to understand as the program becomes filled with more content.

Word Count: 137

4.5 Level 5:

4.5.1 Component 1

Classname(s): DigitalClock.java, ClockThread.java

A brief description of its purpose and role:

Gets the current date and time and displays it on the dashboard.

Number of properties you have coded for the component (4):

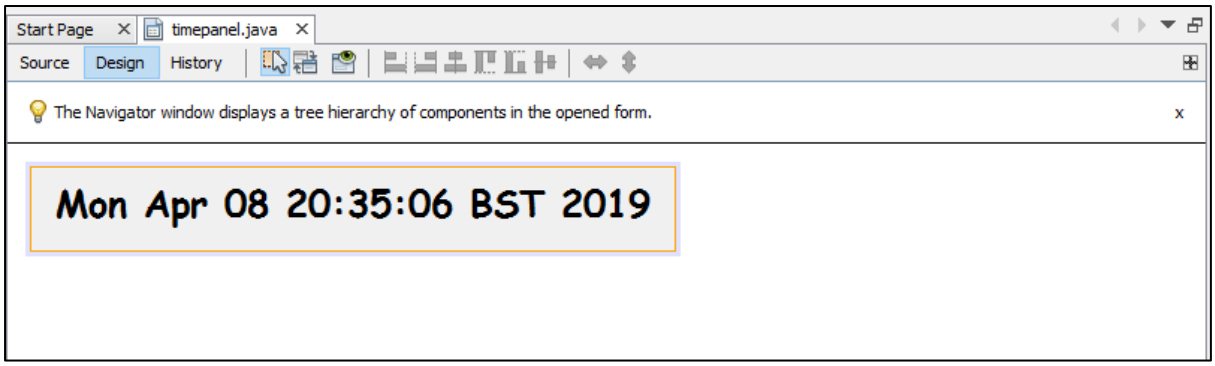
- Background Colour
- Foreground Colour
- Font
- Size

Does the BeanInfo file cause the icon to be displayed?

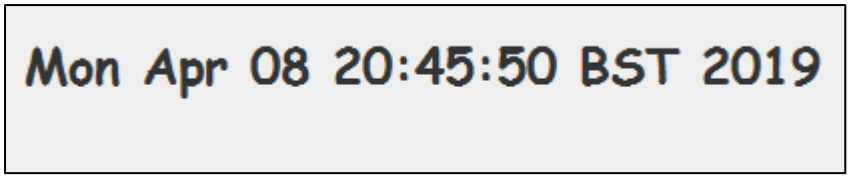
- Yes

Screenshots:

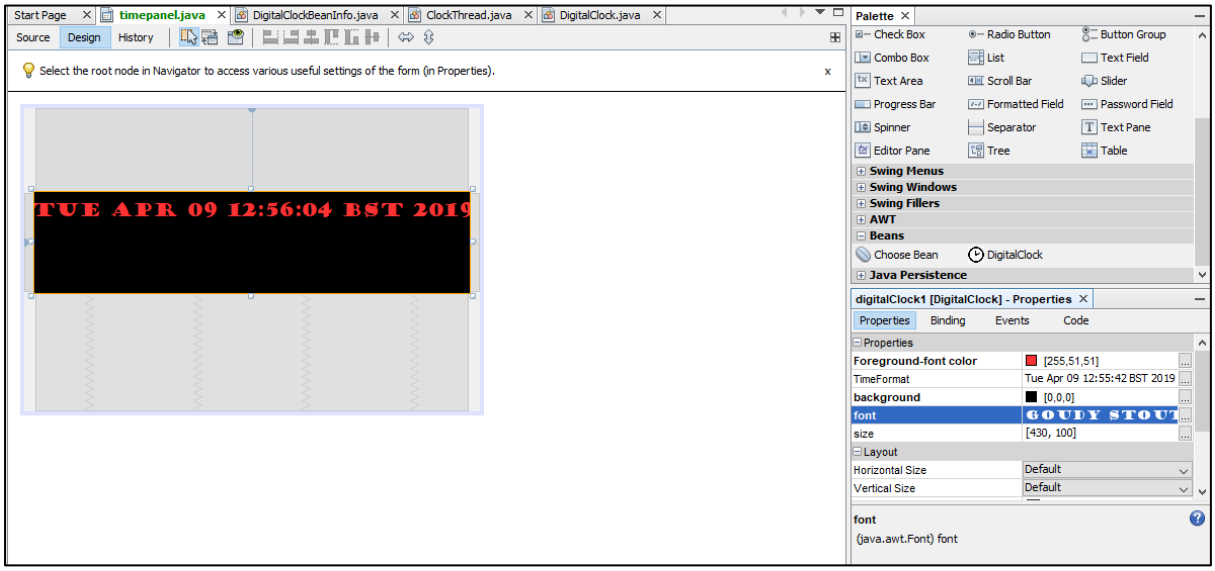
In design palette:



On Dashboard:



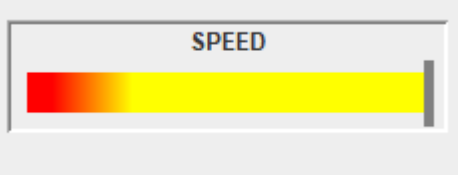
Examples of property changes in colour, font and size:



Section 5 – Annotated screenshots demonstrating each of the stages completed:

- Level 1: Level 1 requirements asked for a vertical bar indicator and at least three other display indicators of at least two different types, with one of the options being an extra vertical bar and dial, as well as one traffic light and one half-dial. The script was to also be able to change the values on the displays, something that we demonstrated in full working fashion during the demos. Shown below are screenshots of our completed level 1 requirements:

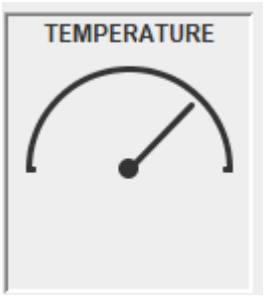
Extra bar:



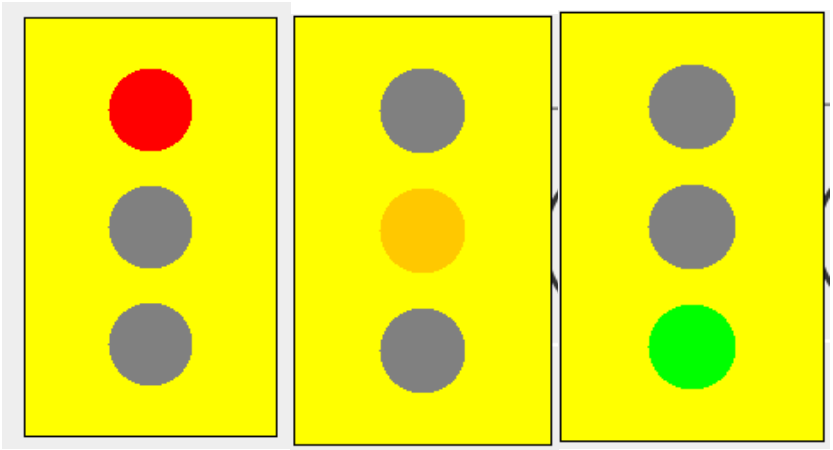
Extra vertical bar:

Extra dial:

New half-dial:

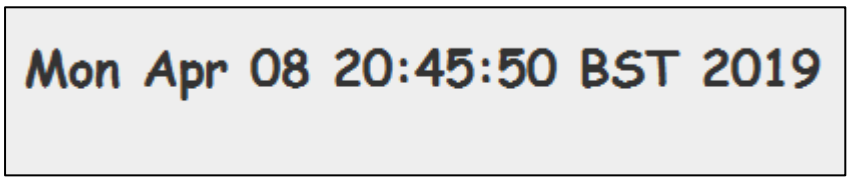


Traffic Light:



- Level 2: Level 2 requirements asked for an additional display indicator of a different type as well as good use of object-oriented features, that are described in the 4.2 header above. Shown below is the additional display feature that we added, a digital clock, as well as proof of the use of interfaces and inheritance:

  Additional display indicator:



Object Oriented Features:

Inheritance:

BarDrawPanel:

```
public BarDrawPanel() {
    this(200, 20, 8, 100, 0);
}
```

```
Rectangle2D barx = new Rectangle2D.Double(padding, padding, barLength, barHeight);
GradientPaint redtoyellow = new GradientPaint(0 + (float) barx.getWidth() * 0.1F, 0, Color.RED, (float) barx.getWidth() * 0.3F, 0, Color.YELLOW);
g2.setPaint(redtoyellow);
g2.fill(barx);
```

```
g2.setStroke(new BasicStroke(barLength/40, BasicStroke.CAP_SQUARE, 0));
g2.setPaint(Color.GRAY);
Line2D valueIndicator = new Line2D.Double(padding + (barLength * value / barMaxValue), padding/2F, padding + (barLength * value / barMaxValue), barHeight + (padding * 1.5F));
g2.draw(valueIndicator);
```

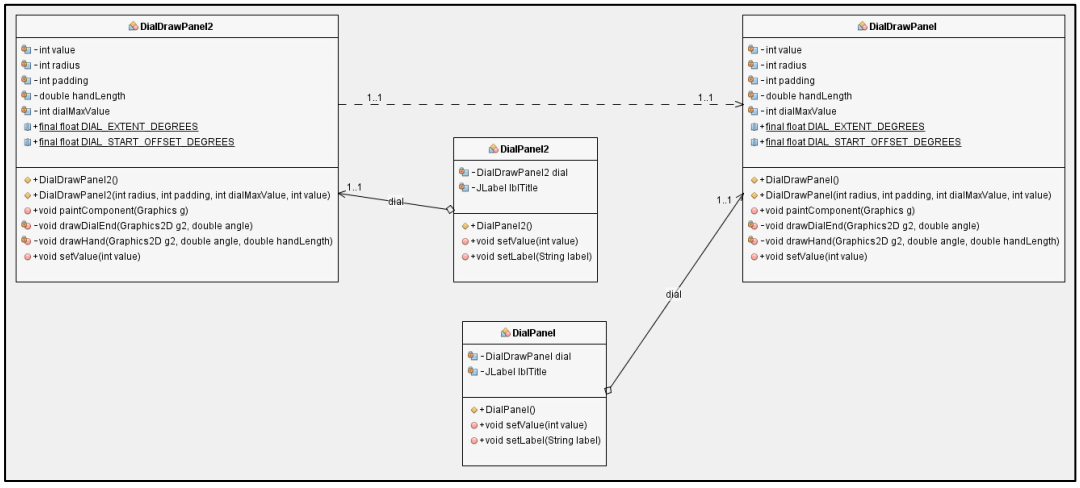
BarDrawPanel2:

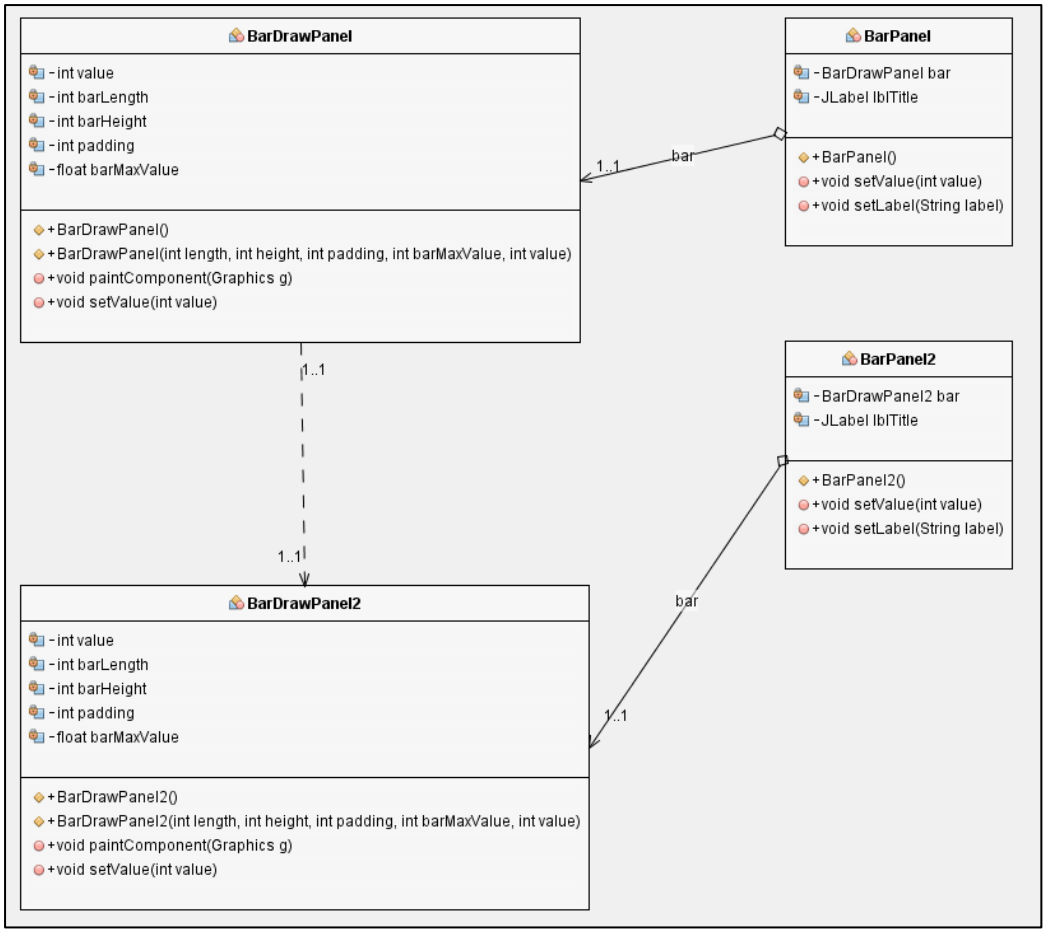
```
public BarDrawPanel2() {
    this(20, 200, 8, 100, 0);
}
```

```
Rectangle2D barx = new Rectangle2D.Double(padding, padding, barLength, barHeight);
GradientPaint redtored = new GradientPaint(0 + (float) barx.getWidth() * 0.1F, 0, Color.RED, (float) barx.getWidth() * 0.3F, 0, Color.RED);
g2.setPaint(redtored);
g2.fill(barx);
```

```
g2.setStroke(new BasicStroke(barHeight/40, BasicStroke.CAP_SQUARE, 0));
  g2.setPaint(Color.GRAY);
  Line2D valueIndicator = new Line2D.Double( padding + (barHeight * value / barMaxValue), padding/2F, padding + (barLength * value / barMaxValue), barLength + (padding * 1.5F));
  g2.draw(valueIndicator);
```

Class Diagrams:





Interfaces:

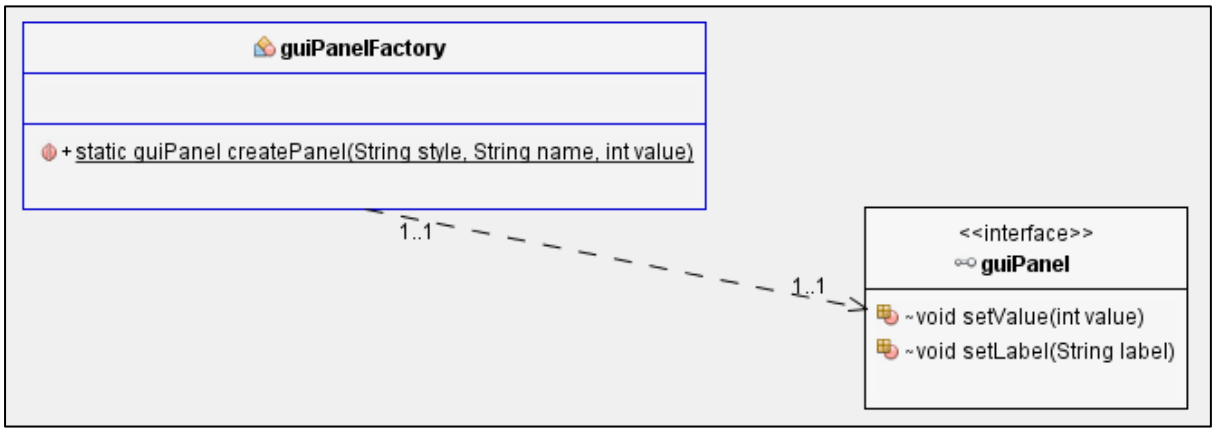
guiPanel:

```
public interface guiPanel {
      void setValue(int value);
     void setLabel(String label);
}
```

guiPanelFactory:

```
public class guiPanelFactory {
    public static guiPanel createPanel(String style, String name, int value) {
        guiPanel uiPanel = null;
        if (name == null) {
            return null;
        } else if (style == "bar1") {

            uiPanel = new BarPanel();
            uiPanel.setLabel(name);
            uiPanel.setValue(value);
        }
        else if (style == "bar2") {

            uiPanel = new BarPanel2();
            uiPanel.setLabel(name);
            uiPanel.setValue(value);
        }
         else if (style == "dial1") {

            uiPanel = new DialPanel();
            uiPanel.setLabel(name);
            uiPanel.setValue(value);
        }
         else if (style == "dial2") {

             uiPanel = new DialPanel2();
            uiPanel.setLabel(name);
            uiPanel.setValue(value);
        }
return uiPanel;

}
}
```

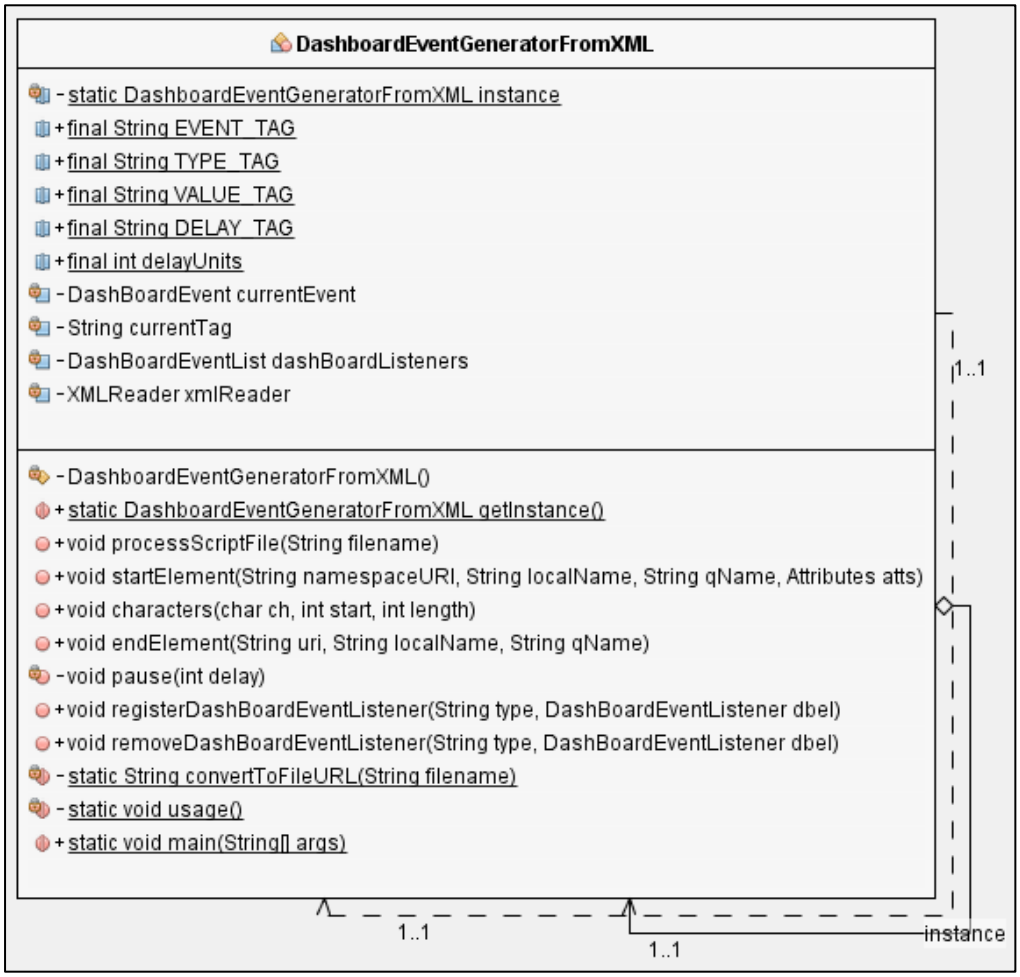
Class Diagram:



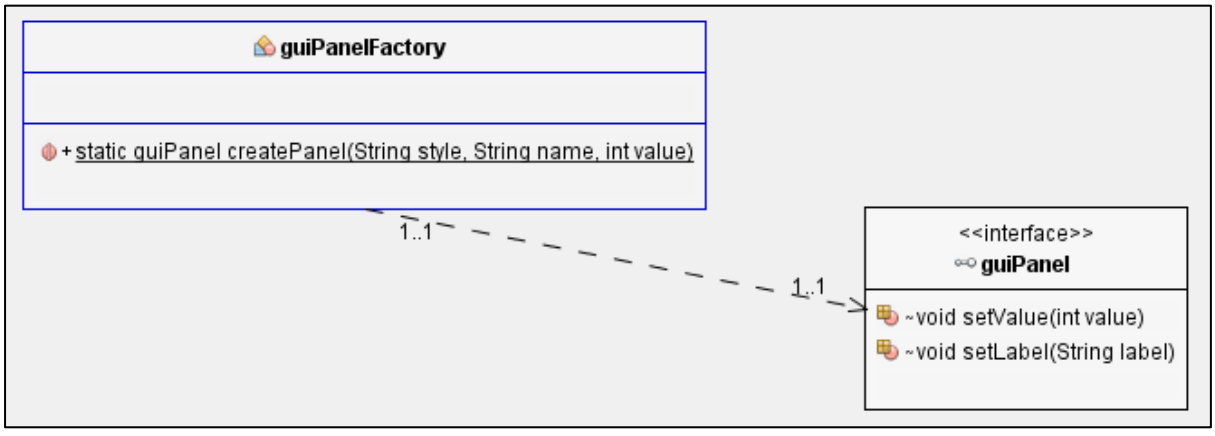
- Level 3: Level 3 requirements asked us to make good use of unit testing by carrying out JUnit testing. Shown below are screenshots showing proof of our JUnit Testing:

- Level 4: Level 4 requirements asked for implementation of acceptable design patterns in an attempt to improve the design of the system: Shown below are screenshots showing proof of our use of design patterns:

  Singleton Pattern Class Diagram:



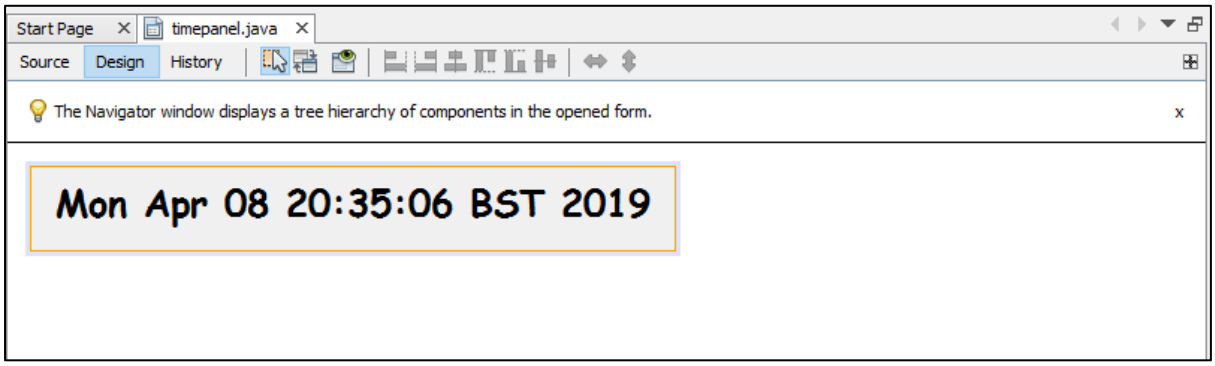
  Factory Pattern Class Diagram:



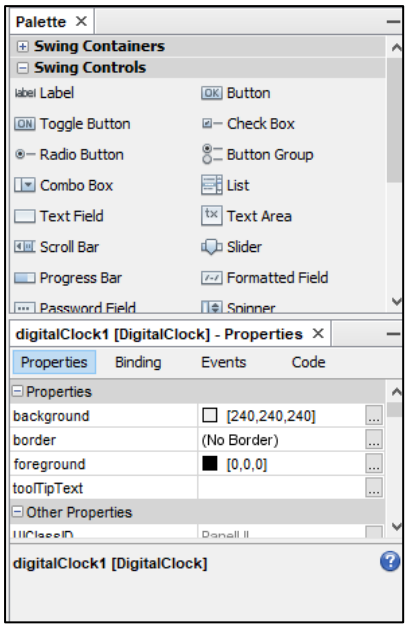
- Level 5: Level 5 requirements asked for the creation of a JavaBean as related to the standard covered on the course. Shown below are screenshots showing proof of our creation of an acceptable bean using a digital clock:

On Dashboard:

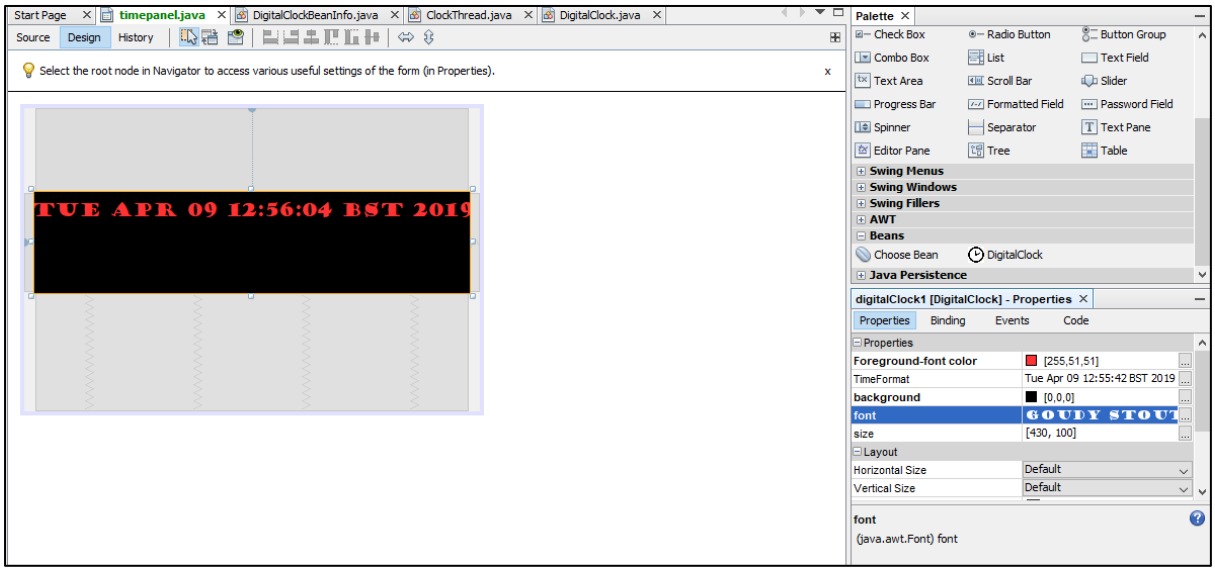
Digital clock on NetBeans component palette/ ability to be dragged and dropped onto a JFrame:



BeanInfo for the palette:

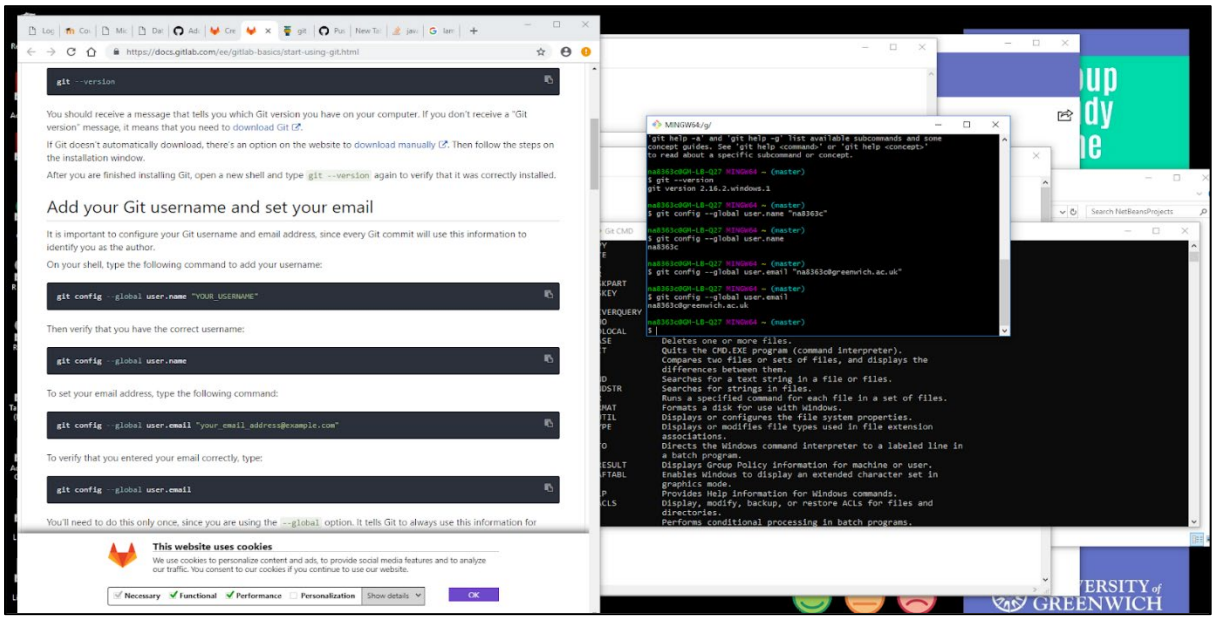


Bean Flexibility:

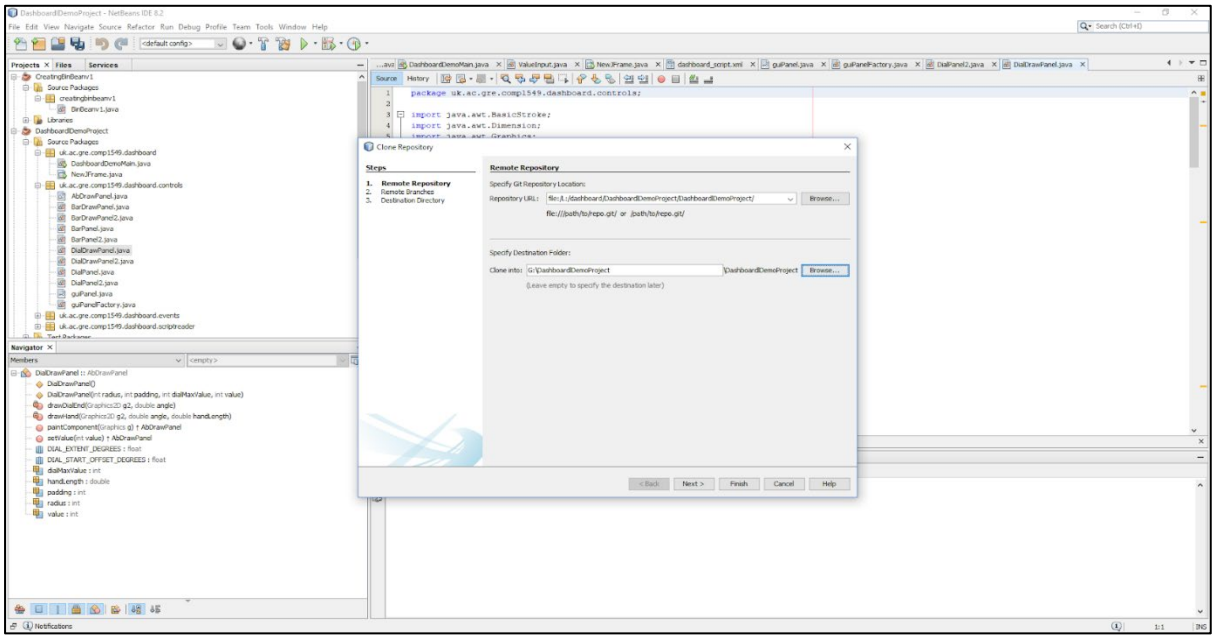


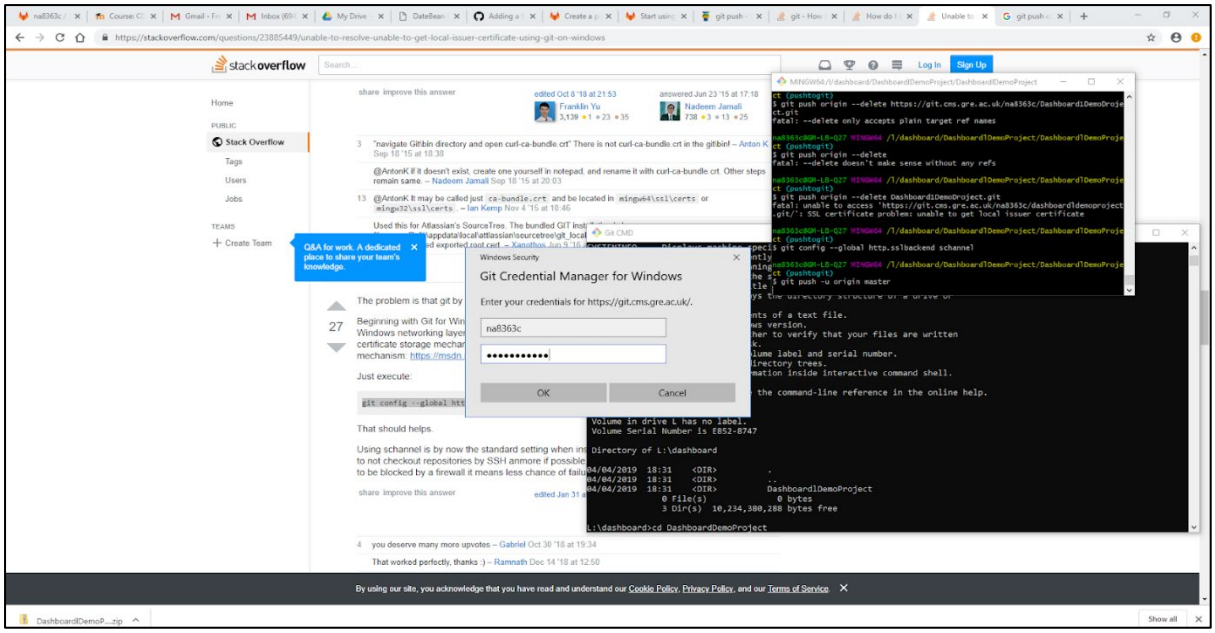
Section 6- Git use for version control:

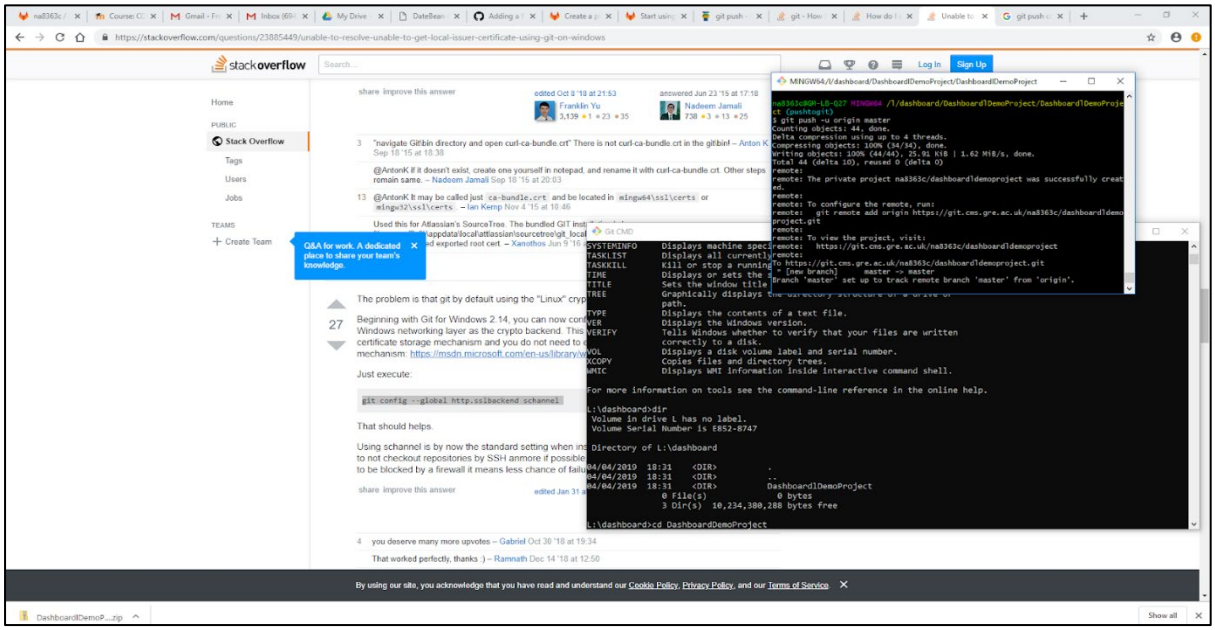
Creation of the git account:



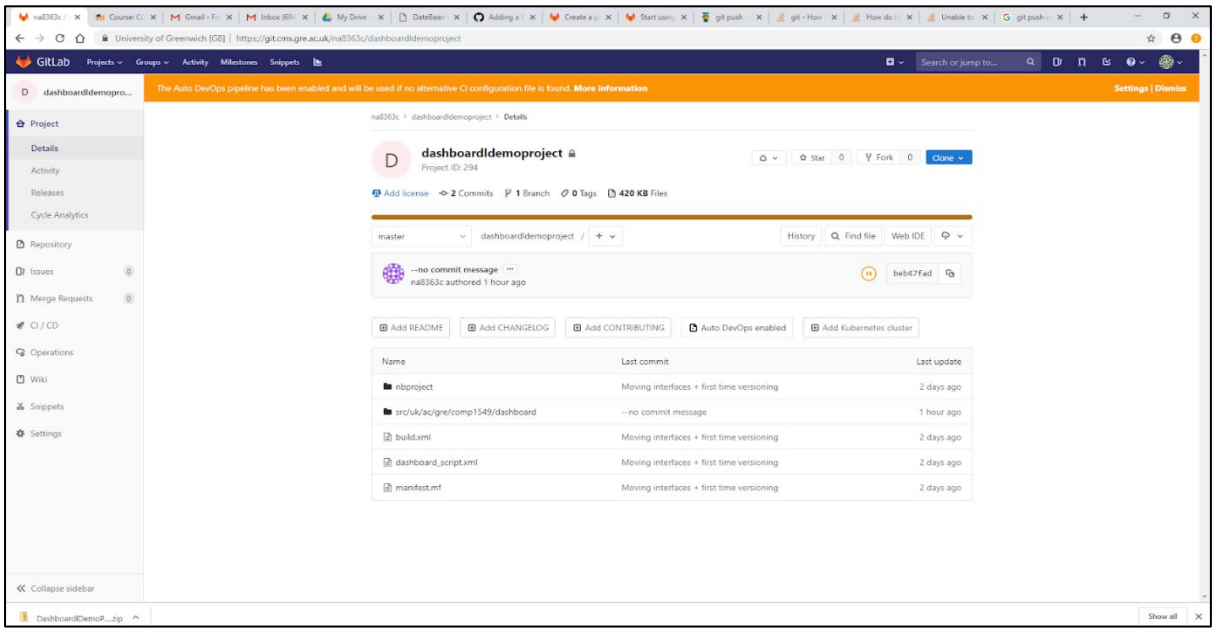
Specifying the Remote Git Repository:

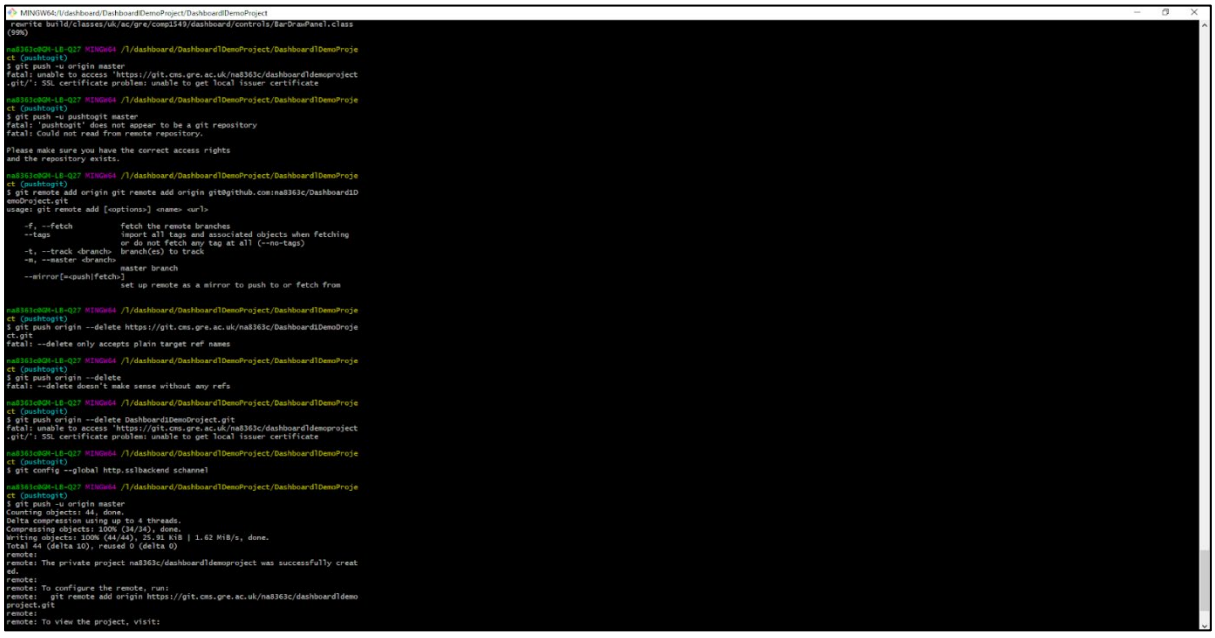



stackoverflow
Windows Security
Git Credential Manager for Windows
Enter your credentials for https://git.cms.gre.ac.uk/.
na8363c
OK
Cancel


stackoverflow
git config --global http.sslbackend schannel

Git upload:


GitLab
Projects
Groups
Activity
Milestones
Snippets
The Auto DevOps pipeline has been enabled and will be used if no alternative CI configuration file is found. More information
Settings | Dismiss
Project
Details
Activity
Releases
Cycle Analytics
Repository
Issues
Merge Requests
CI / CD
Operations
Wiki
Snippets
Settings
Collapse sidebar
dashboardldemoproject
Project ID: 294
Star
Fork
Clone
Add license
2 Commits
1 Branch
0 Tags
420 KB Files
master
History
Find file
Web IDE
--no commit message
na8363c authored 1 hour ago
beb47fad
Add README
Add CHANGELOG
Add CONTRIBUTING
Auto DevOps enabled
Add Kubernetes cluster
Name
Last commit
Last update
nbproject
Moving interfaces + first time versioning
2 days ago
src/uk/ac/gre/comp1549/dashboard
--no commit message
1 hour ago
build.xml
Moving interfaces + first time versioning
2 days ago
dashboard_script.xml
Moving interfaces + first time versioning
2 days ago
manifest.mf
Moving interfaces + first time versioning
2 days ago
Show all


rewrite build/classes/uk/ac/gre/comp1549/dashboard/controls/BarDrawPanel.class
(99%)
$ git push -u origin master
fatal: unable to access 'https://git.cms.gre.ac.uk/na8363c/dashboard1demoproject
.git/': SSL certificate problem: unable to get local issuer certificate
$ git push -u pushtogit master
fatal: 'pushtogit' does not appear to be a git repository
fatal: Could not read from remote repository.
Please make sure you have the correct access rights
and the repository exists.
$ git remote add origin git@github.com:na8363c/Dashboard1D
emoProject.git
usage: git remote add [<options>] <name> <url>
-f, --fetch fetch the remote branches
--tags import all tags and associated objects when fetching
or do not fetch any tag at all (--no-tags)
-t, --track <branch> branch(es) to track
-m, --master <branch>
master branch
--mirror[=<push|fetch>]
set up remote as a mirror to push to or fetch from
$ git push origin --delete https://git.cms.gre.ac.uk/na8363c/Dashboard1DemoProje
ct.git
fatal: --delete only accepts plain target ref names
$ git push origin --delete
fatal: --delete doesn't make sense without any refs
$ git push origin --delete Dashboard1DemoProject.git
fatal: unable to access 'https://git.cms.gre.ac.uk/na8363c/dashboard1demoproject
.git/': SSL certificate problem: unable to get local issuer certificate
$ git config --global http.sslbackend schannel
$ git push -u origin master
Counting objects: 44, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (34/34), done.
Writing objects: 100% (44/44), 25.91 KiB | 1.62 MiB/s, done.
Total 44 (delta 10), reused 0 (delta 0)
remote:
remote: The private project na8363c/dashboard1demoproject was successfully creat
ed.
remote:
remote: To configure the remote, run:
remote: git remote add origin https://git.cms.gre.ac.uk/na8363c/dashboard1demo
project.git
remote:
remote: To view the project, visit:

Section 7 - Paired Programming Reflection:

The concept of paired programming is one that I have that I have experienced for the second time this term, and it is one that that I have enjoyed thoroughly. It gave a good balance between working in larger groups and working alone, both having their own pros and cons. I felt that it greatly improved my productivity, as I always felt the pressure not to let my partner down, especially since we both had other modules to work on. We were able to bounce ideas off each other, and I felt that I benefited the most from this as I learnt a lot from completing the logbooks together. The fact that we were both responsible for each other's grade motivated us to complete the tasks that we had to complete in due time. As mentioned before, we did have other modules to work on, so time management was a bit of an issue sometimes especially since my partner and I do live a fair bit away from each other. However, as other deadlines passed, we were able to focus our best efforts on this particular module. We did have a lot of errors/mistakes while completing this coursework. We spent entirely too much time brainstorming and trying to create a separate program that did not follow Zena's provided demo, even venturing into the possibility of making a JavaFX application. It was eventually easier to adapt Zena's program. This did, however, hinder our productivity, as all that wasted time could have been used to produce a better program, and we will take account of that during our next project. We also encountered small errors in programming, but these were easily fixable once we looked over our code again. Even with these fixes, as with anything, there is always room for improvement. Better planning and research about the module before time started would have definitely resulted in gaining the knowledge I did earlier rather than later in the term; I plan to do this rigorously for my third-year classes. I am glad that I was able to experience the Advanced Programming module and look forward to the opportunities it unlocks for me in the future.

Section 8- Logbooks:

Logbook 1 – Inheritance

Basic Information

1.1 Student name	**Nkem Akwari**
1.2 Who did you work with? Name and/or id	**Trevor Kiggundu**
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	**I feel as I have completed the work to the specification detailed in the lab document and this is reflected with how my code looks and the end result functionality in terms of using inheritance to streamline a system and cut down on code.**

1. **Implementation**

2.1 Annotated screenshots demonstrating what you have achieved.

Figure 1- basic hospital program running without amending, original button outputs those records, inheritance does nothing:

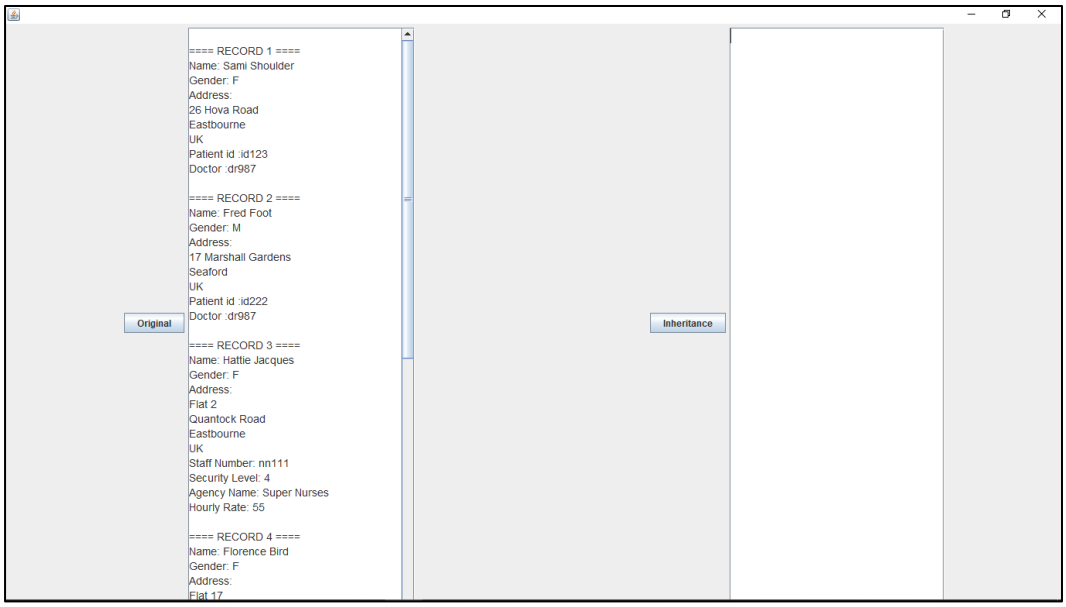
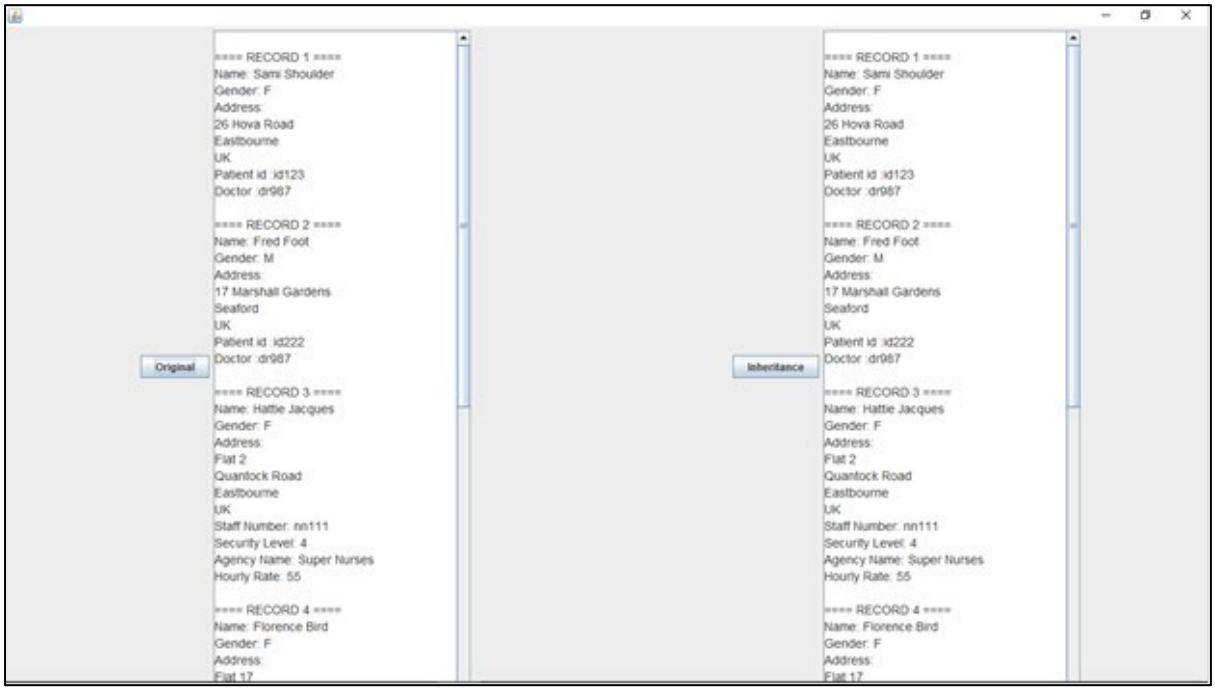


Figure 2 - Added correct functionality to handleinheritancebutton(). Output is the same as original:



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.

```
//Creation of person class to hold basic fields
public class Person {

    protected String firstName;
    protected String familyName;
    protected char gender;
    protected PostalAddress postalAddress;

    public Person(String firstName, String familyName, char gender, PostalAddress postalAddress) {
        this.firstName = firstName;
        this.familyName = familyName;
        this.gender = gender;
        this.postalAddress = postalAddress;
```

```
    }
     public Person() {
        this.firstName = "";
        this.familyName = "";
        this.gender = 'U';
        this.postalAddress = new PostalAddress();
    }

    public String getFirstName() {
        return firstName;
    }

    public String getFamilyName() {
        return familyName;
    }

    public char getGender() {
        return gender;
    }

    public PostalAddress getPostalAddress() {
        return postalAddress;
    }
}
```

//changed aspects of AgencyStaff to support inheritance of person

```
public class AgencyStaff extends Person {

    private String staffNumber;
    private int securityLevel;
    private String agencyName;
    private int hourlyRate;
```

```
    public AgencyStaff(String agencyName, int hourlyRate, String staffNumber, int
securityLevel, String firstName, String familyName, char gender, PostalAddress
postalAddress) {
        super(firstName, familyName, gender, postalAddress);
        this.staffNumber = staffNumber;
        this.securityLevel = securityLevel;
        this.agencyName = agencyName;
        this.hourlyRate = hourlyRate;
    }

    public AgencyStaff() {
        this.firstName = super.firstName ;
        this.familyName = super.familyName;
        this.gender = super.gender;
        this.postalAddress = super.postalAddress;
        this.staffNumber = "";
        this.agencyName = "";
    }

    public String getStaffNumber() {
        return staffNumber;
    }

    public int getSecurityLevel() {
        return securityLevel;
    }

    public String getAgencyName() {
        return agencyName;
    }

    public int getHourlyRate() {
        return hourlyRate;
    }
```

```
    }
        //changed aspects of directemployee to support inheritance of person
    public class DirectEmployee extends Person {

        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, familyName, gender, postalAddress);
            this.staffNumber = staffNumber;
            this.securityLevel = securityLevel;
            this.grade = grade;
            this.salary = salary;
        }

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

        public String getStaffNumber() {
            return staffNumber;
        }

        public int getSecurityLevel() {
            return securityLevel;
```

```
    }

    public String getGrade() {
        return grade;
    }

    public int getSalary() {
        return salary;
    }

}
```

//Changed aspects of Patient to support inheritance of person

```
public class Patient extends Person{

    private String patientId;
    private String doctorStaffNumber;

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

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

```
        public String getPatientId() {
            return patientId;
        }

        public String getDoctorStaffNumber() {
            return doctorStaffNumber;
        }
    }
    //created code for handleinheritancebutton() that shows how inheritance cuts down
code
    private void handleInheritanceButton() {
            // set up the data
            List<Object> listOfRecords = new ArrayList<>();
            listOfRecords.add(patient1);
            listOfRecords.add(patient2);
            listOfRecords.add(agencystaff1);
            listOfRecords.add(agencystaff2);
            listOfRecords.add(directEmployee1);
            listOfRecords.add(directEmployee2);

            txtInheritance.setText(""); // clear the output

            // Loop through the data displaying the details
            int count = 0;
            for (Object o : listOfRecords) {
               count++;
            /**starts with the superclass, which all current objects are a part of.
             *Through polymorphism, this means each entry that goes through this method
             *will display the instance variables that it shares with person first
             */
               if (o instanceof Person) {
                  Person pe = (Person) o;
                  txtInheritance.append(String.format("\n==== RECORD %d ====\n",
count));
```

```
                txtInheritance.append("Name: " + pe.getFirstName() + " " +
pe.getFamilyName() + "\n");
                txtInheritance.append("Gender: " + pe.getGender() + "\n");
                txtInheritance.append("Address:\n" +
pe.getPostalAddress().getDisplayableAddress() + "\n");
            }
            /*
                *With the rest, it will check if it fits into any of the
                *subclasses during each if loop, and if so, adds the rest of the
                *relevant instance variables to its record before going to the
                *next arraylistentry and doing the same until the arraylist has
                *been iterated through.
             */
            if (o instanceof AgencyStaff) {
                AgencyStaff as = (AgencyStaff) o;

                txtInheritance.append("Staff Number: " + as.getStaffNumber() + "\n");
                txtInheritance.append("Security Level: " + as.getSecurityLevel() + "\n");
                txtInheritance.append("Agency Name: " + as.getAgencyName() + "\n");
                txtInheritance.append("Hourly Rate: " + as.getHourlyRate() + "\n");
            } else if (o instanceof DirectEmployee) {
                DirectEmployee de = (DirectEmployee) o;
                txtInheritance.append("Staff Number: " + de.getStaffNumber() + "\n");
                txtInheritance.append("Security Level: " + de.getSecurityLevel() + "\n");
                txtInheritance.append("Grade :" + de.getGrade() + "\n");
                txtInheritance.append(String.format("Salary: £%,d\n", de.getSalary()));

            } else if (o instanceof Patient) {
                Patient pa = (Patient) o;
                txtInheritance.append("Patient id :" + pa.getPatientId() + "\n");
                txtInheritance.append("Doctor :" + pa.getDoctorStaffNumber() + "\n");
            }
        }
```

```
    // Loop through the data displaying the details
}
```

Logbook 2 – JUnit testing

Basic Information

1.1 Student name	**Trevor Kiggundu**
1.2 Who did you work with? Name and/or id	**Nkem Akwari**
1.3 Which lab topic does this document relate to?	JUnit
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	**I feel as I have completed the work to the specification detailed in the lab document. This is justified by the tests and explanations being at the satisfactory level needed for completion.**

1. **Implementation**

2.1 Annotated screenshots demonstrating what you have achieved.

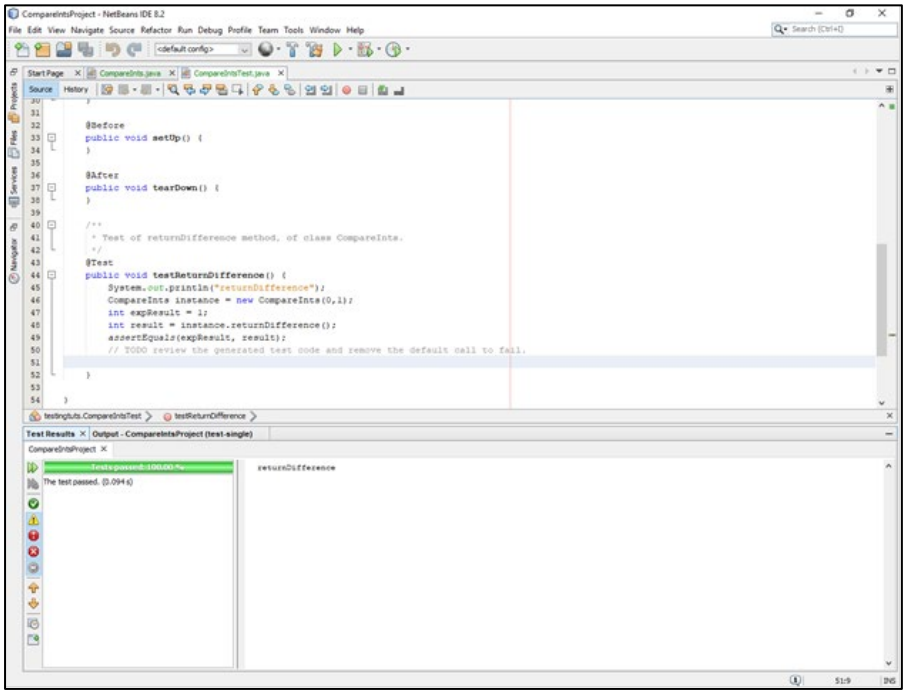
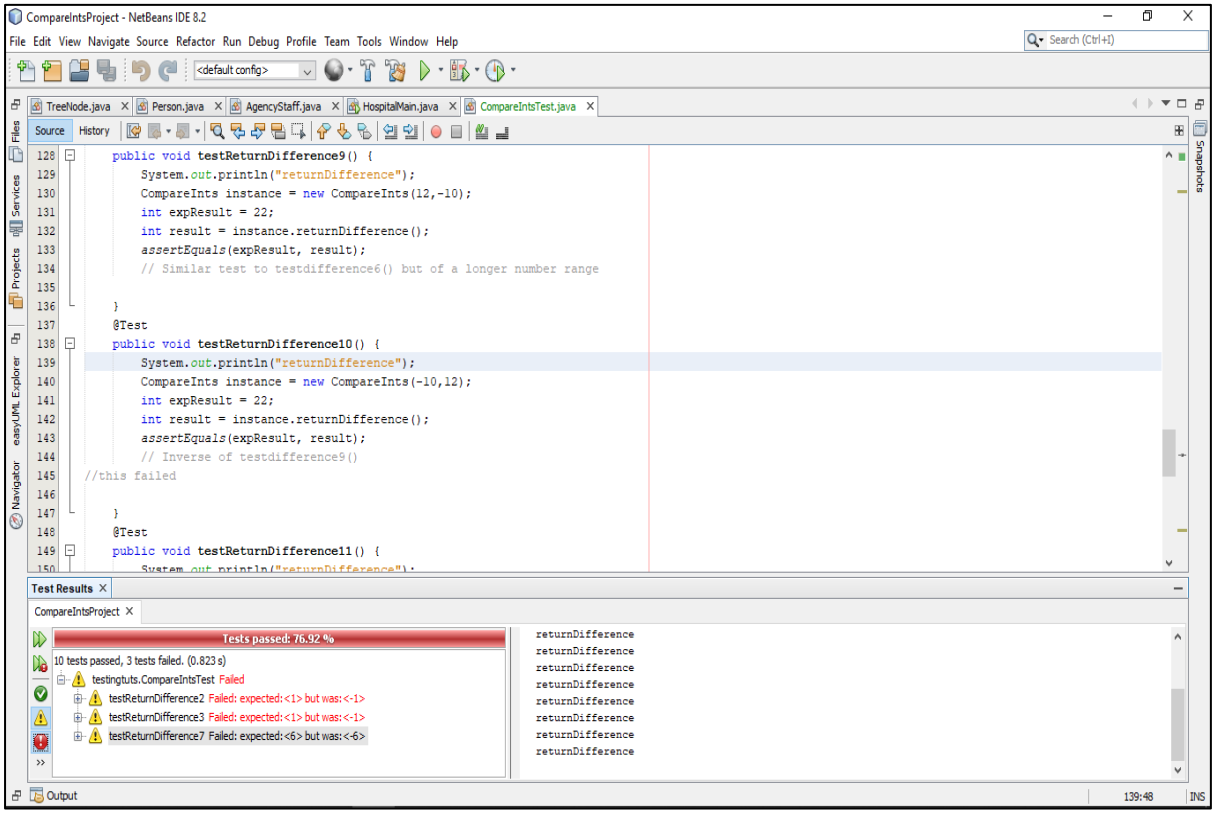


Figure 2 -Creation of the first testreturndifference(), testing, three of the 14 tests failed:



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.

//Test package for compareints

```
public class CompareIntsTest {

    public CompareIntsTest() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
```

```
public static void tearDownClass() {
}

@Before
public void setUp() {
}

@After
public void tearDown() {
}

/**
 * Test of returnDifference method, of class CompareInts.
 */
@Test
public void testReturnDifference() {
    System.out.println("returnDifference");
    CompareInts instance = new CompareInts(0,1);
    int expResult = 1;
    int result = instance.returnDifference();
    assertEquals(expResult, result);
    // basic example test given by document

}
   @Test
public void testReturnDifference2() {
    System.out.println("returnDifference");
    CompareInts instance = new CompareInts(-1,0);
    int expResult = 1;
    int result = instance.returnDifference();
    assertEquals(expResult, result);
    //Test used to test if a negative as "one" can work with "two" being 0
  //This failed.
```

```
    }
     @Test
    public void testReturnDifference3() {
        System.out.println("returnDifference");
        CompareInts instance = new CompareInts(0,-1);
        int expResult = 1;
        int result = instance.returnDifference();
        assertEquals(expResult, result);
        //Inverse of testreturndifference2(), negative in second place, postive in first
        //this failed

    }
     @Test
    public void testReturnDifference4() {
        System.out.println("returnDifference");
        CompareInts instance = new CompareInts(10,-1);
        int expResult = 11;
        int result = instance.returnDifference();
        assertEquals(expResult, result);
        //Testing if a positive as the first number can be taken away from a negative
which is the second number

    }
    @Test
    public void testReturnDifference5() {
        System.out.println("returnDifference");
        CompareInts instance = new CompareInts(2,2);
        int expResult = 0;
        int result = instance.returnDifference();
        assertEquals(expResult, result);
        //Testing if two of the same number can work with this class

    }
    @Test
```

```
    public void testReturnDifference6() {
        System.out.println("returnDifference");
        CompareInts instance = new CompareInts(5,-5);
        int expResult = 10;
        int result = instance.returnDifference();
        assertEquals(expResult, result);
        // Testing if a positve value with the same digit value can be used with a negative
of the same digit number

    }

    @Test
    public void testReturnDifference7() {
        System.out.println("returnDifference");
        CompareInts instance = new CompareInts(-7,-1);
        int expResult = 6;
        int result = instance.returnDifference();
        assertEquals(expResult, result);
        //Another test testing if a negative as the first number will work correctly with a
positive as the second
   //this failed

    }
    @Test
    public void testReturnDifference8() {
        System.out.println("returnDifference");
        CompareInts instance = new CompareInts(0,4);
        int expResult = 4;
        int result = instance.returnDifference();
        assertEquals(expResult, result);
        // Testing if zero as first number works with a positive value as second

    }
    @Test
```

```
public void testReturnDifference9() {
    System.out.println("returnDifference");
    CompareInts instance = new CompareInts(12,-10);
    int expResult = 22;
    int result = instance.returnDifference();
    assertEquals(expResult, result);
    // Similar test to testdifference6() but of a longer number range

}
@Test
public void testReturnDifference10() {
    System.out.println("returnDifference");
    CompareInts instance = new CompareInts(-10,12);
    int expResult = 22;
    int result = instance.returnDifference();
    assertEquals(expResult, result);
    // Inverse of testdifference9()

}
@Test
public void testReturnDifference11() {
    System.out.println("returnDifference");
    CompareInts instance = new CompareInts(-1,-1);
    int expResult = 0;
    int result = instance.returnDifference();
    assertEquals(expResult, result);
    //Test to see if two negative numbers work

}
@Test
public void testReturnDifference12() {
    System.out.println("returnDifference");
    CompareInts instance = new CompareInts(2,6);
    int expResult = 4;
```

```
        int result = instance.returnDifference();
        assertEquals(expResult, result);
//basic test to see if these postive numbers work
    }
    @Test
    public void testReturnDifference13() {
        System.out.println("returnDifference");
        CompareInts instance = new CompareInts(6,2);
        int expResult = 4;
        int result = instance.returnDifference();
        assertEquals(expResult, result);
        // inverse positions from testreturndifference12()

    }
```

Logbook 3 – Design Patterns

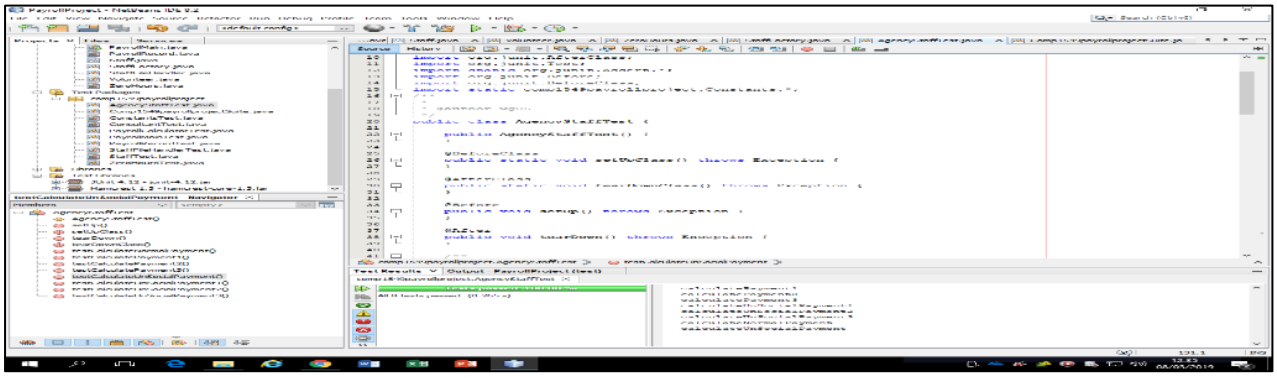
Basic Information

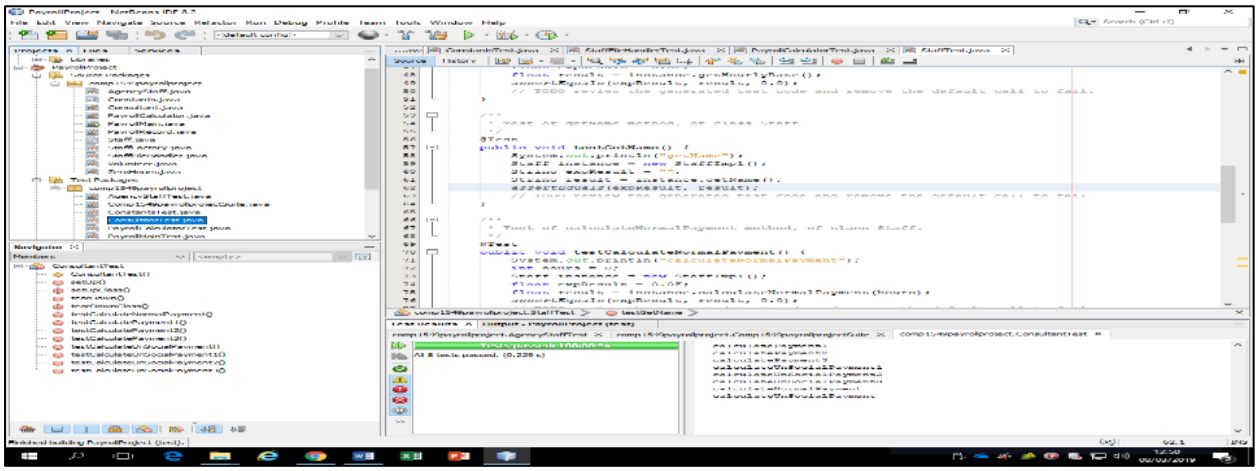
1.1 Student name	**Nkem Akwari**
1.2 Who did you work with? Name and/or id	**Trevor**
1.3 Which lab topic does this document relate to?	Design Patterns
1.4 How well do you feel you have done?	• I have completed the exercise and am totally satisfied with my work
1.5 Briefly explain your answer to question 1.4	**I feel as I have completed the work to the specification detailed in the lab document and this is reflected with how my code looks and the end result's functionality in terms of applying the proper design patterns to the sample code, and adequately testing if this works correctly through JUnit**

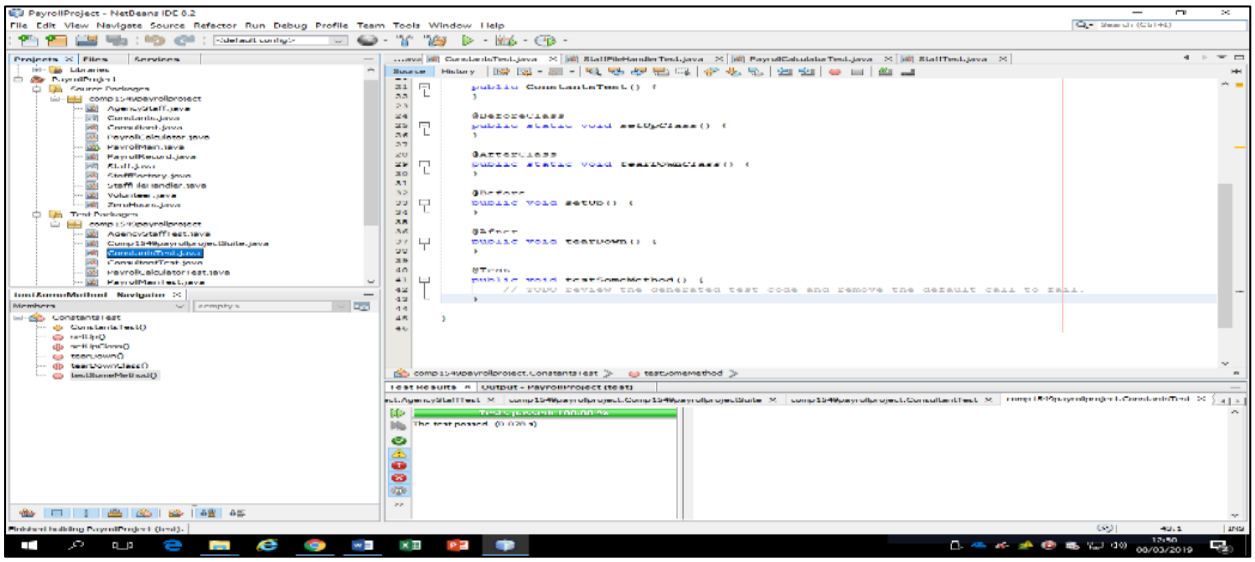
2. **Implementation**

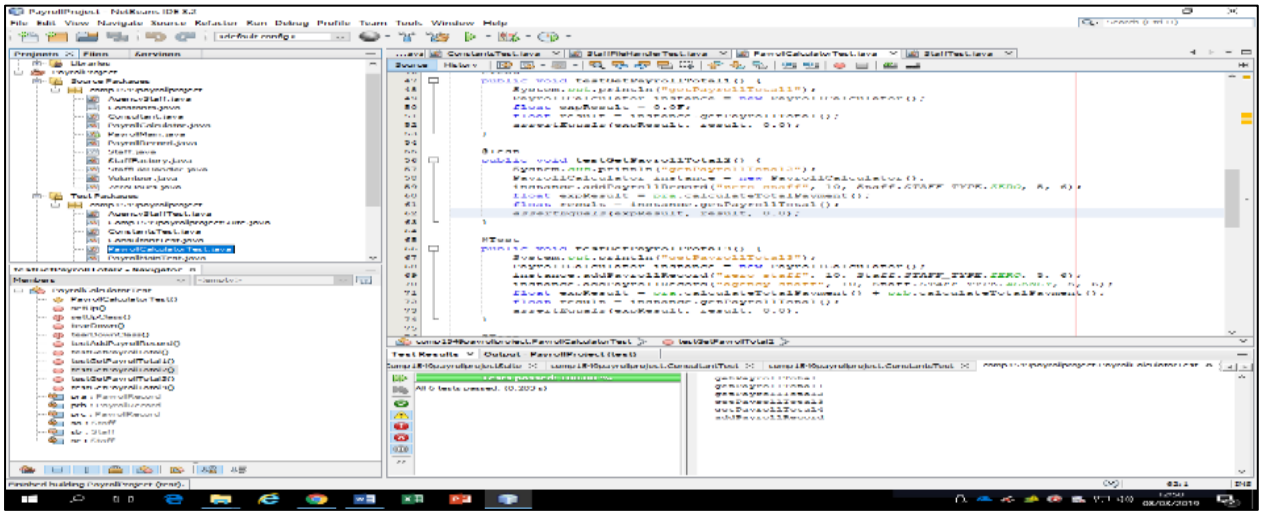
2.1 Annotated screen shots demonstrating what you have achieved.

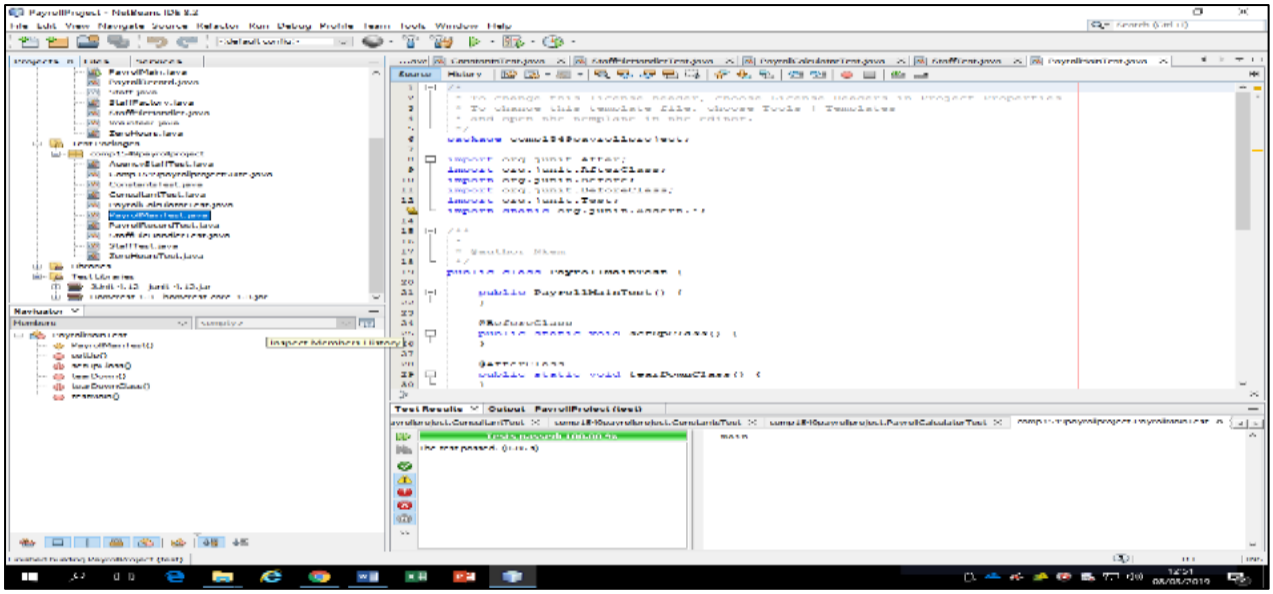
<u>Testing the files before adding the Volunteer and stafffactory classes- all passed</u>

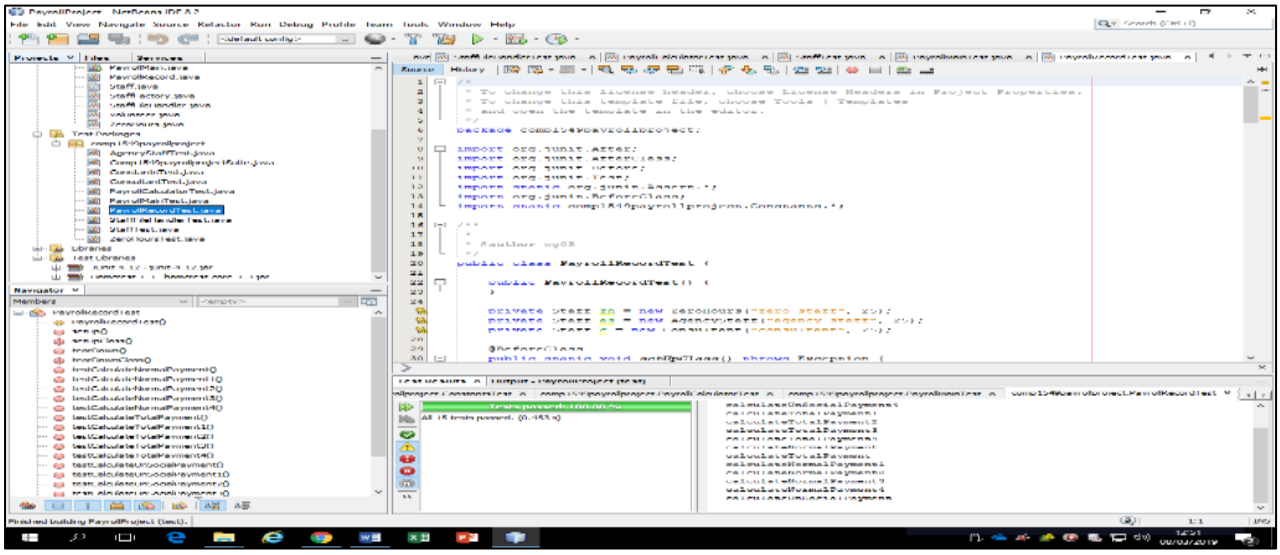


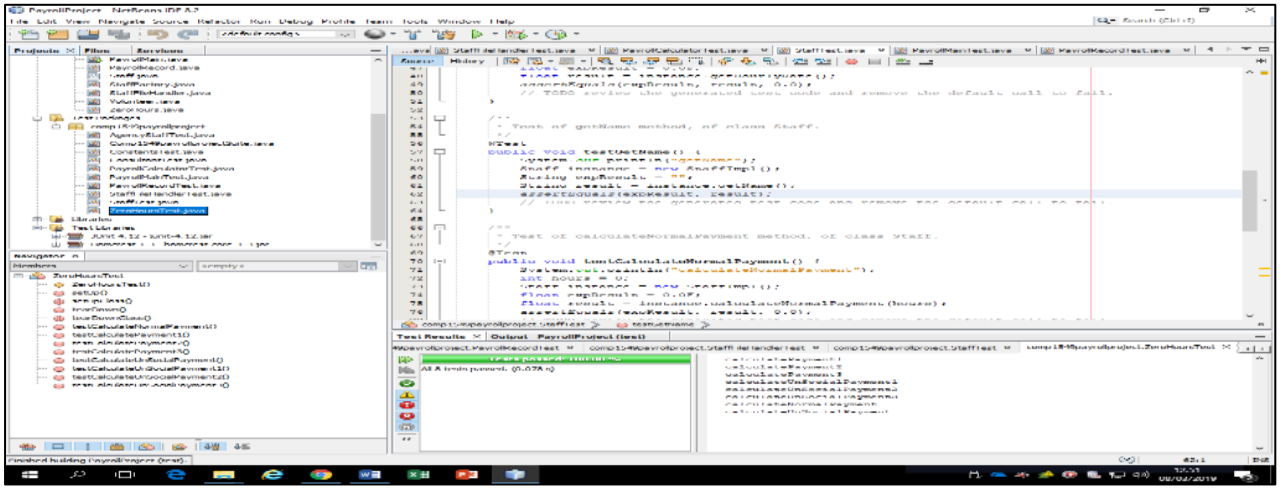


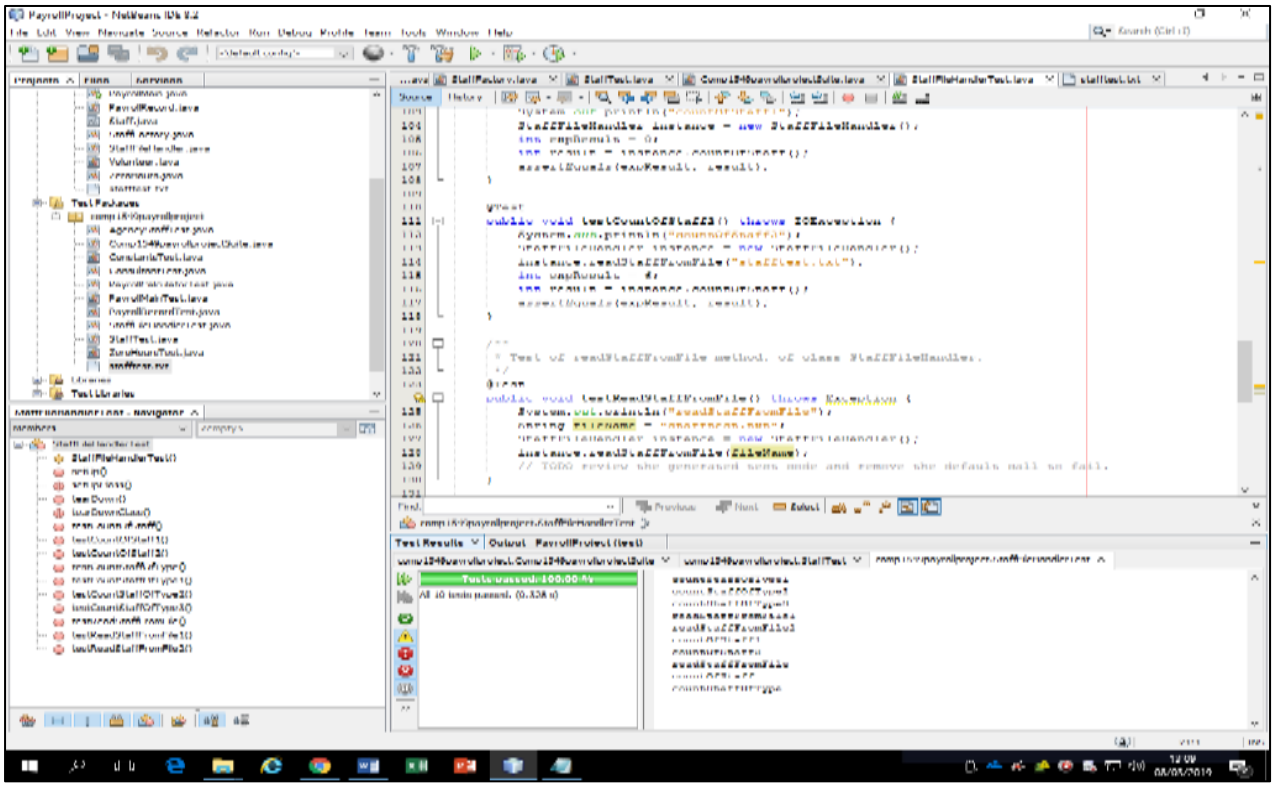


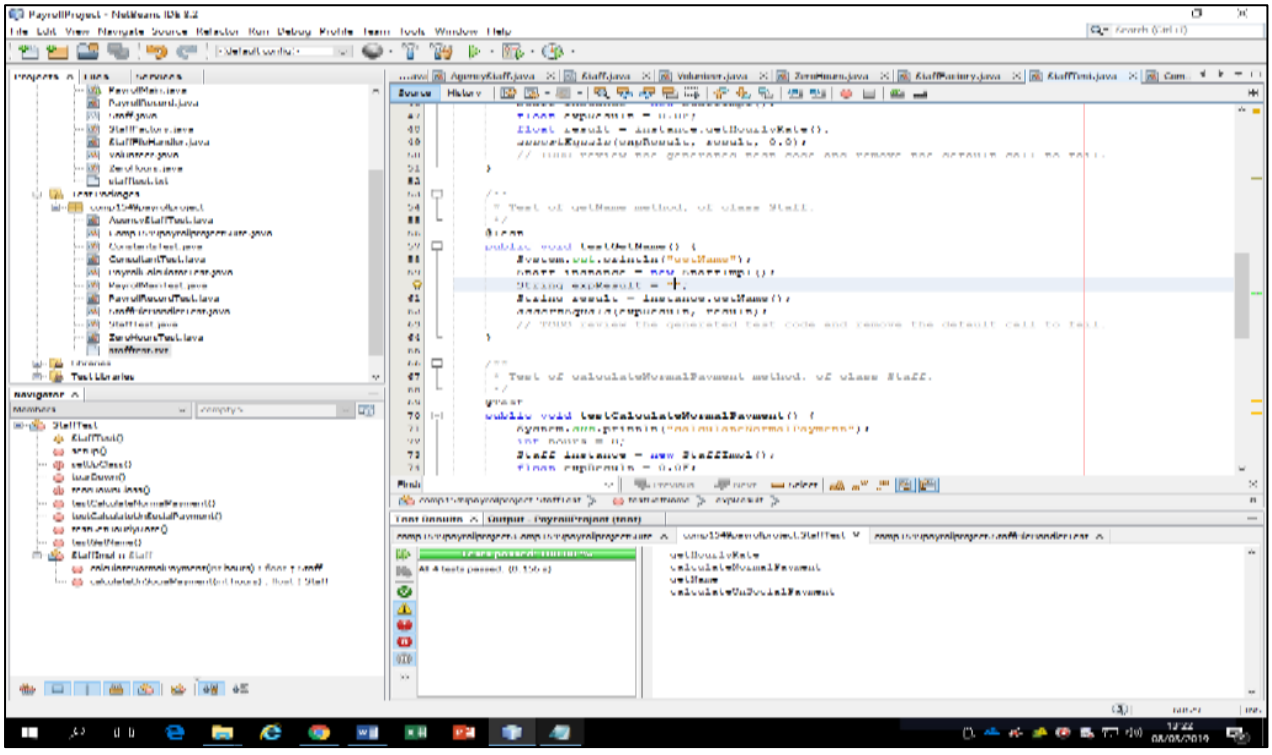


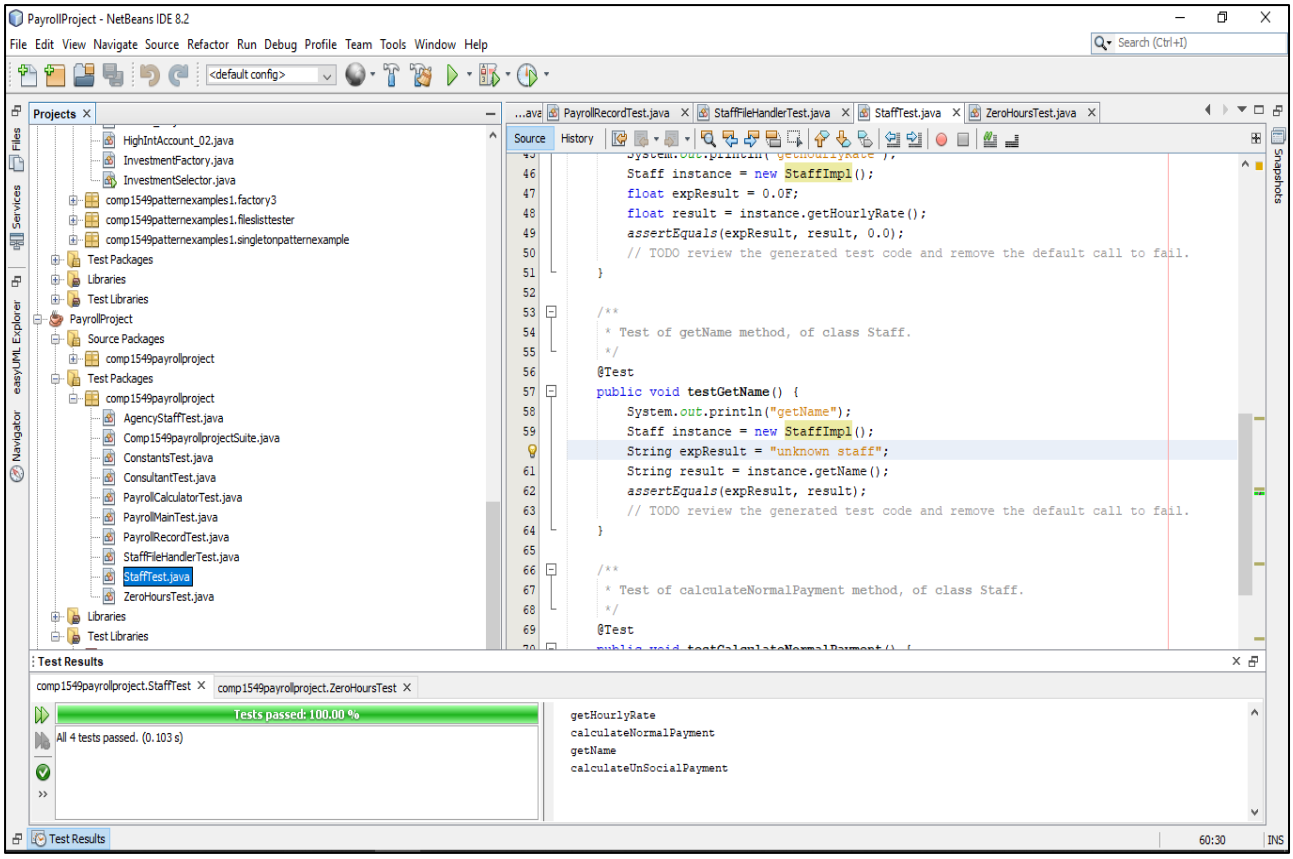


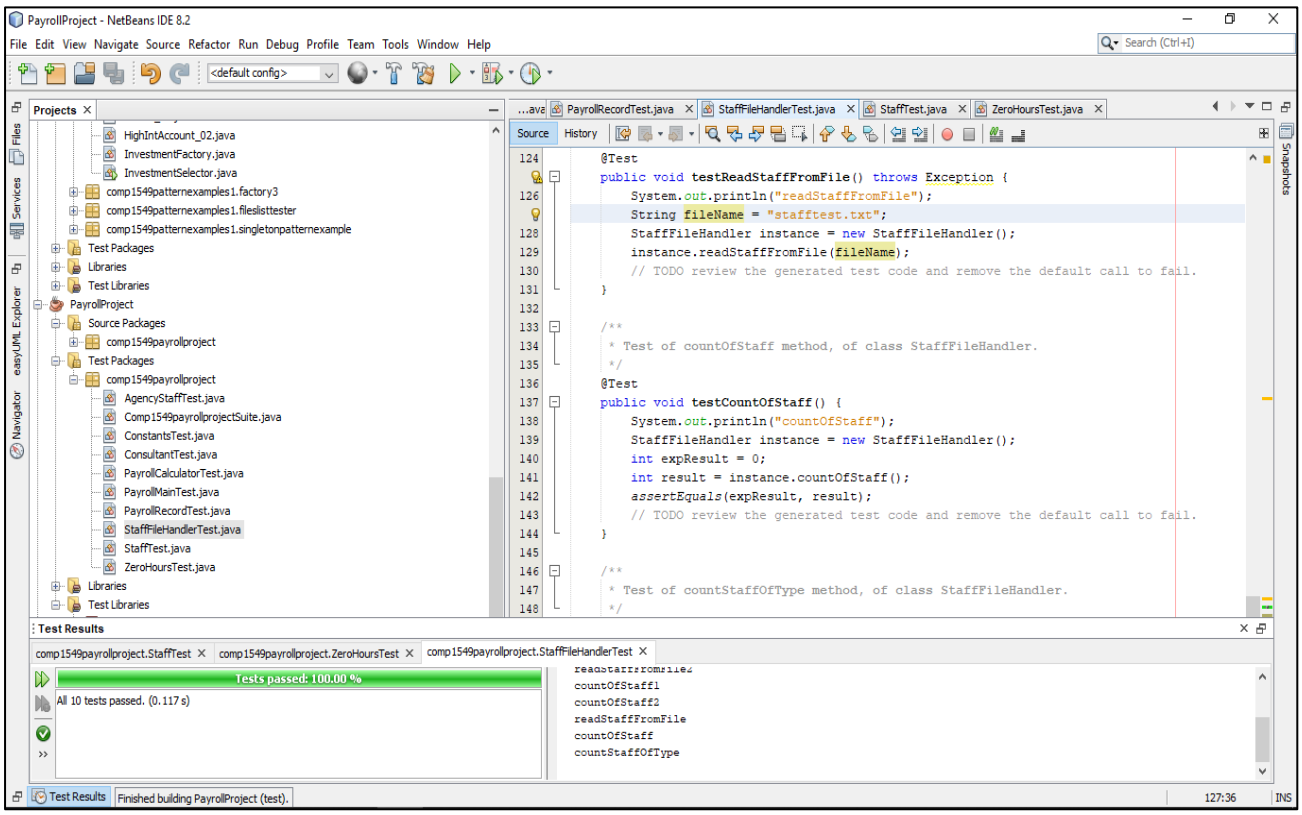


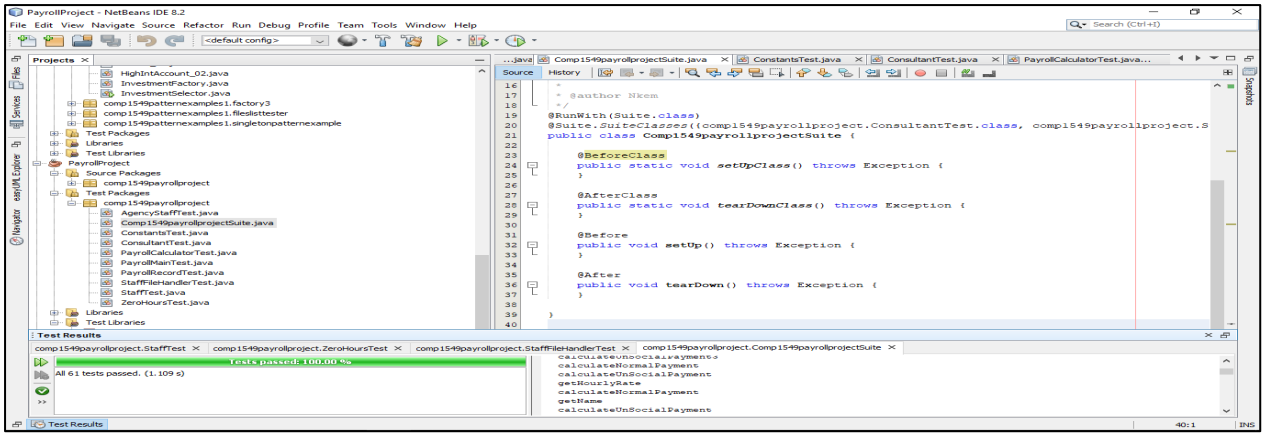




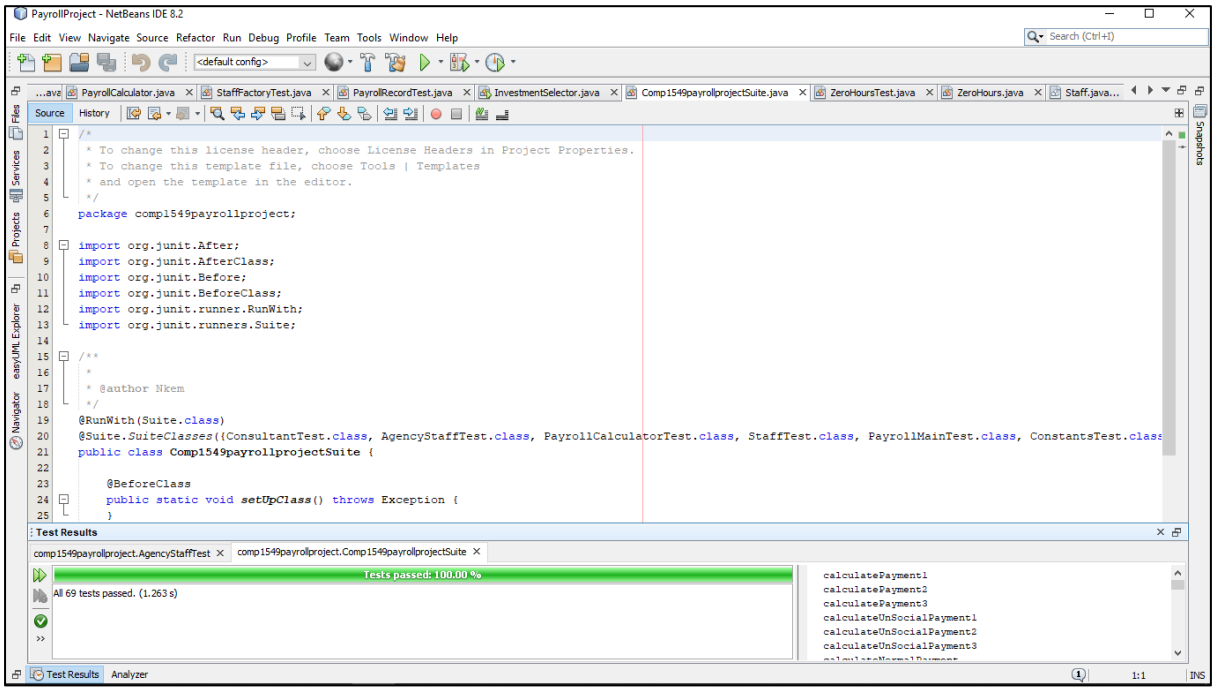



PayrollProject - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
HighIntAccount_02.java
InvestmentFactory.java
InvestmentSelector.java
comp1549patternexamples1.factory3
comp1549patternexamples1.filesisttester
comp1549patternexamples1.singletonpatternexample
Test Packages
Libraries
Test Libraries
PayrollProject
Source Packages
comp1549payrollproject
Test Packages
comp1549payrollproject
AgencyStaffTest.java
Comp1549payrollprojectSuite.java
ConstantsTest.java
ConsultantTest.java
PayrollCalculatorTest.java
PayrollMainTest.java
PayrollRecordTest.java
StaffFileHandlerTest.java
StaffTest.java
ZeroHoursTest.java
Libraries
Test Libraries
PayrollRecordTest.java
StaffFileHandlerTest.java
StaffTest.java
ZeroHoursTest.java
Source
History
Staff instance = new StaffImpl();
float expResult = 0.0F;
float result = instance.getHourlyRate();
assertEquals(expResult, result, 0.0);
// TODO review the generated test code and remove the default call to fail.
/**
* Test of getName method, of class Staff.
*/
@Test
public void testGetName() {
System.out.println("getName");
Staff instance = new StaffImpl();
String expResult = "unknown staff";
String result = instance.getName();
assertEquals(expResult, result);
// TODO review the generated test code and remove the default call to fail.
/**
* Test of calculateNormalPayment method, of class Staff.
*/
@Test
Test Results
comp1549payrollproject.StaffTest
comp1549payrollproject.ZeroHoursTest
Tests passed: 100.00 %
All 4 tests passed. (0.103 s)
getHourlyRate
calculateNormalPayment
getName
calculateUnSocialPayment
Test Results
60:30
INS


PayrollProject - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
HighIntAccount_02.java
InvestmentFactory.java
InvestmentSelector.java
comp1549patternexamples1.factory3
comp1549patternexamples1.filesisttester
comp1549patternexamples1.singletonpatternexample
Test Packages
Libraries
Test Libraries
PayrollProject
Source Packages
comp1549payrollproject
Test Packages
comp1549payrollproject
AgencyStaffTest.java
Comp1549payrollprojectSuite.java
ConstantsTest.java
ConsultantTest.java
PayrollCalculatorTest.java
PayrollMainTest.java
PayrollRecordTest.java
StaffFileHandlerTest.java
StaffTest.java
ZeroHoursTest.java
Libraries
Test Libraries
PayrollRecordTest.java
StaffFileHandlerTest.java
StaffTest.java
ZeroHoursTest.java
Source
History
@Test
public void testReadStaffFromFile() throws Exception {
System.out.println("readStaffFromFile");
String fileName = "stafftest.txt";
StaffFileHandler instance = new StaffFileHandler();
instance.readStaffFromFile(fileName);
// TODO review the generated test code and remove the default call to fail.
/**
* Test of countOfStaff method, of class StaffFileHandler.
*/
@Test
public void testCountOfStaff() {
System.out.println("countOfStaff");
StaffFileHandler instance = new StaffFileHandler();
int expResult = 0;
int result = instance.countOfStaff();
assertEquals(expResult, result);
// TODO review the generated test code and remove the default call to fail.
/**
* Test of countStaffOfType method, of class StaffFileHandler.
*/
Test Results
comp1549payrollproject.StaffTest
comp1549payrollproject.ZeroHoursTest
comp1549payrollproject.StaffFileHandlerTest
Tests passed: 100.00 %
All 10 tests passed. (0.117 s)
countOfStaff1
countOfStaff2
readStaffFromFile
countOfStaff
countStaffOfType
Test Results
Finished building PayrollProject (test).
127:36
INS



Testing the files after adding the Volunteer and stafffactory classes- all passed



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.

From ZeroHoursTest

@Test

```
    public void testCalculatePayment4() {
        System.out.println("calculatePayment3");
        int hours = 5;
        Staff instance = StaffFactory.createStaff("ZERO", "Rose", 12);
        float expResult = 60.0F;
        float result = instance.calculateNormalPayment(hours);
        assertEquals(expResult, result, 0.0);
        //test if factory still produces a correct result

    }
```

From StaffFactoryTest

```
    @Test
    //test for null/blank constructor
    public void testCreateStaff() {
        System.out.println("createStaff");
        String type = "";
        String name = "";
        float hourly_rate = 0.0F;
        Staff expResult = null;
        Staff result = StaffFactory.createStaff(type, name, hourly_rate);
        assertEquals(expResult, result);
    }
    @Test
```

```
    //testing if a method using a factory pattern created object works properly
    public void testStaffMethod() {
        System.out.println("createStaff");
        String type = "ZERO";
        String name = "Hal";
        float hourly_rate = 10.0F;
        Staff exp = new ZeroHours("Hal", 10);
        float expResult = exp.calculateNormalPayment(3);
        Staff res = StaffFactory.createStaff(type, name, hourly_rate);
        float result = res.calculateNormalPayment(3);
        assertEquals(expResult, result, 0.0);
    }
```

From VolunteerTest

```
/**
     * Test of calculateNormalPayment method, of class Volunteer.
     */
    @Test
    public void testCalculateNormalPayment() {
        System.out.println("calculateNormalPayment");
        int hours = 0;
        Volunteer instance = new Volunteer();
        float expResult = 0.0F;
        float result = instance.calculateNormalPayment(hours);
        assertEquals(expResult, result, 0.0);
```

```
    // TODO review the generated test code and remove the default call to fail.
}
@Test
public void testCalculatePayment2() {
    System.out.println("calculatePayment2");
    int hours = 5;
    Volunteer instance = new Volunteer("Johnny", 10);
    float expResult = 0.0F;
    float result = instance.calculateNormalPayment(hours);
    assertEquals(expResult, result, 0.0);

}


/**
 * Test of calculateUnSocialPayment method, of class Volunteer.
 */
 @Test
public void testCalculateUnSocialPayment() {
    System.out.println("calculateUnSocialPayment");
    int hours = 0;
    Volunteer instance = new Volunteer();
    float expResult = 0.0F;
```

```
        float result = instance.calculateUnSocialPayment(hours);
        assertEquals(expResult, result, 0.0);

    }
    @Test
    public void testCalculateUnSocialPayment2() {
        System.out.println("calculateUnSocialPayment");
        int hours = 4;
        Volunteer instance = new Volunteer("Jade", 8);
        float expResult = 48.0F;
        float result = instance.calculateUnSocialPayment(hours);
        assertEquals(expResult, result, 0.0);

    }
```

Volunteer Class Code

```
//creation of volunteer class that is a subclass of staff
public class Volunteer extends Staff {

    //default constructor for volunteer
    public Volunteer() {
        this("unknown volunteers", 0);

    }
```

```
/*Basic constructor using a name parameter for the volunteer, this variable is
   from staff, and a float hourlyrate for the hourlyrate for volunteer

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

   //volunteers do not have normal payment, so this should always return zero
   @Override
   public float calculateNormalPayment(int hours) {
      return 0.0F;
   }

   /*this multiplies the hourly rate with the number of hours(this parameter)
   and the unsocial_rate_multiplier inherited from Constants
   which is then returned
   */
   @Override
   public float calculateUnSocialPayment(int hours) {
            return getHourlyRate() * hours * UNSOCIAL_RATE_MULTIPLIER;

   }
```

```
}
```

Failed PayrollCalculator class' add payroll method

```
public void addPayrollRecord(String name, float hourlyRate, STAFF_TYPE type, int normalHours, int unsocialHours) {

        //modified the code to instead call the factory class in order to create

        //a new instance of a staff subclass, which is also added to the payroll record,

        Staff newStaff = StaffFactory.createStaff(type.toString() , name, hourlyRate);


        PayrollRecord newPayrollRecord;


        newPayrollRecord = new PayrollRecord(newStaff, normalHours, unsocialHours);

        payrollRecords.add(newPayrollRecord);

    }
```

Due to some errors with testing blank strings with the PayrollCalculator, I had to add a nullpointerargumentexception via a try-catch in order to allow for null values to be recognised and not return an error with the test

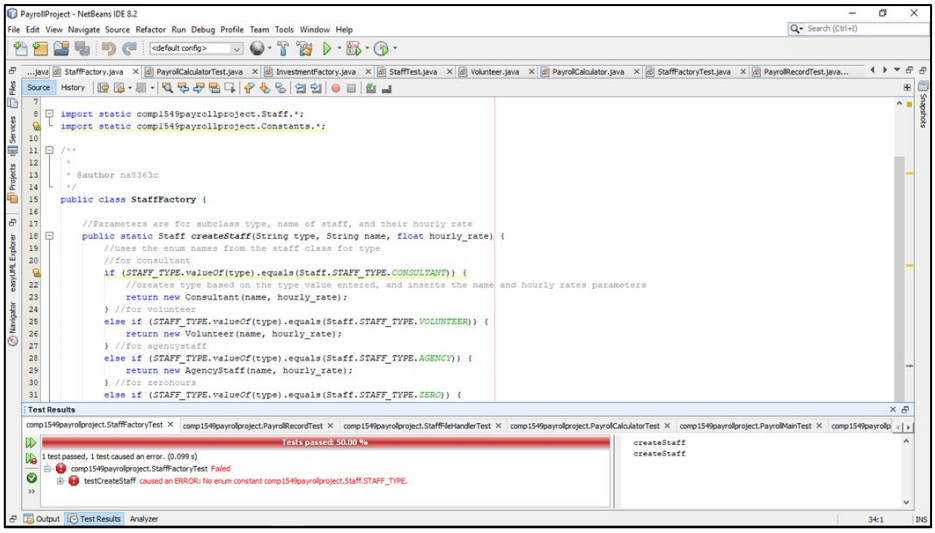
StaffFactory-original class

```
public class PayrollCalculator {

    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) {

       //modified the code to instead call the factory class in order to create
       //a new instance of a staff subclass, which is also added to the payroll record,
            Staff newStaff;
                 PayrollRecord newPayrollRecord;
      //try catch designed to handle if the type entered is null
   try
       {
        newStaff = StaffFactory.createStaff(type.toString() , name, hourlyRate);
        newPayrollRecord = new PayrollRecord(newStaff, normalHours,
unsocialHours);
       payrollRecords.add(newPayrollRecord);
       }
       //notes a nullpointerexception if there is a blank or null parameter in
       catch(NullPointerException ex)
```

```
            {
                System.out.println("NullPointerexception has occured");
            }
        }

        /**
         * 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;
        }
```

Similar to the Payroll calculator, an IllegalArgumentException happened with a null “type” string for the constructor for the Factory class, leading to issues with testing the factory class. To rectify this, we added a try-catch for the creation of the staff subclasses via factory pattern, as well as refactoring the structure of the class to support this cha



StaffFactory class- refactored

```
public class StaffFactory {

    //Parameters are for subclass type, name of staff, and their hourly rate
    public static Staff createStaff(String type, String name, float hourly_rate) {
//declares null abstract object that will use polymorphism alongside the factory pattern
        Staff newStaff = null;
        //uses the enum names from the staff class for type
        //for consultant
//try-catch set up
            try{
        if (STAFF_TYPE.valueOf(type).equals(Staff.STAFF_TYPE.CONSULTANT))
{
            //initialises the newStaff class as one the enum values entered through the
“type” string parameter, and inserts the name and hourly rates parameters
            newStaff= new Consultant(name, hourly_rate);
        } //for volunteer
        else if
(STAFF_TYPE.valueOf(type).equals(Staff.STAFF_TYPE.VOLUNTEER)) {
             newStaff=  new Volunteer(name, hourly_rate);
        } //for agencystaff
```

```
            else if (STAFF_TYPE.valueOf(type).equals(Staff.STAFF_TYPE.AGENCY)) {
                newStaff=  new AgencyStaff(name, hourly_rate);
            } //for zerohours
            else if (STAFF_TYPE.valueOf(type).equals(Staff.STAFF_TYPE.ZERO)) {
                newStaff=  new ZeroHours(name, hourly_rate);
            } /*if nothing valid is inserted as a parameter, then it returns null through the new
catch created to note if the illegalargument exception happens*/
                    }
                    catch(IllegalArgumentException e){

                    }
        //return newStaff
            return newStaff;
        }
```

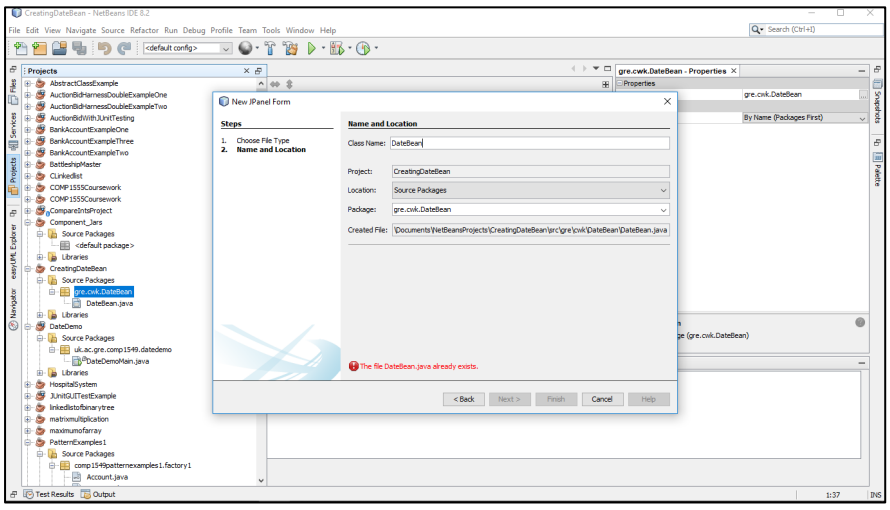
Logbook 4 – Creating Software Components

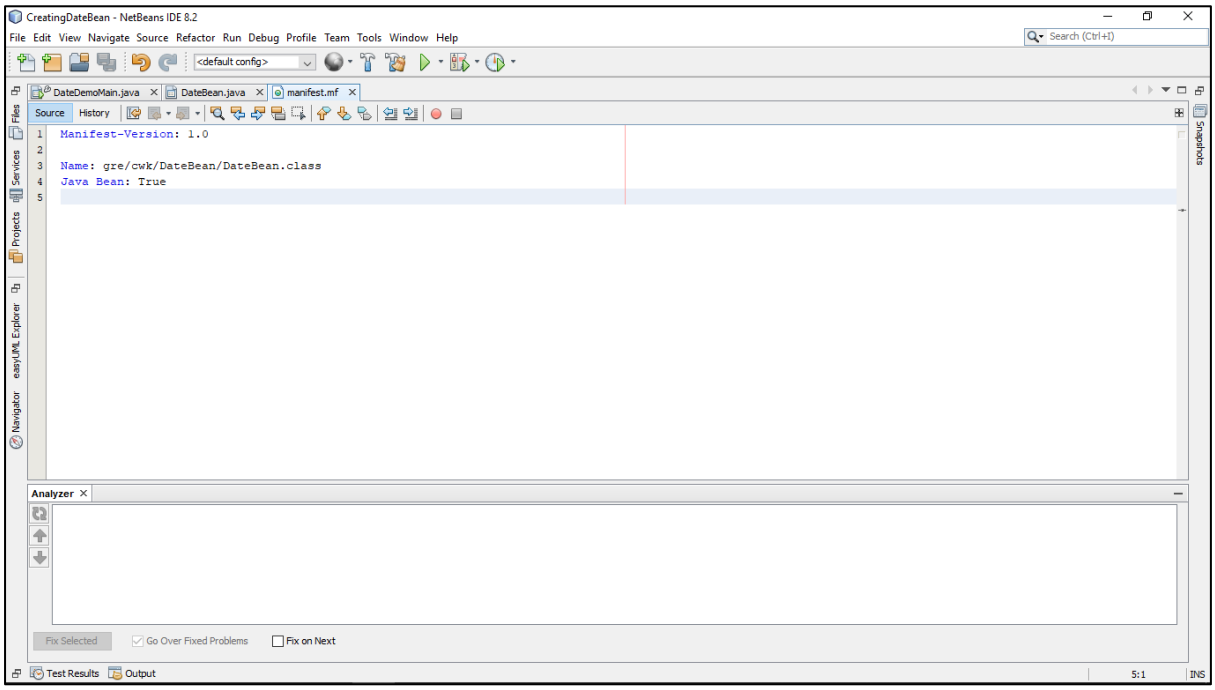
Basic Information

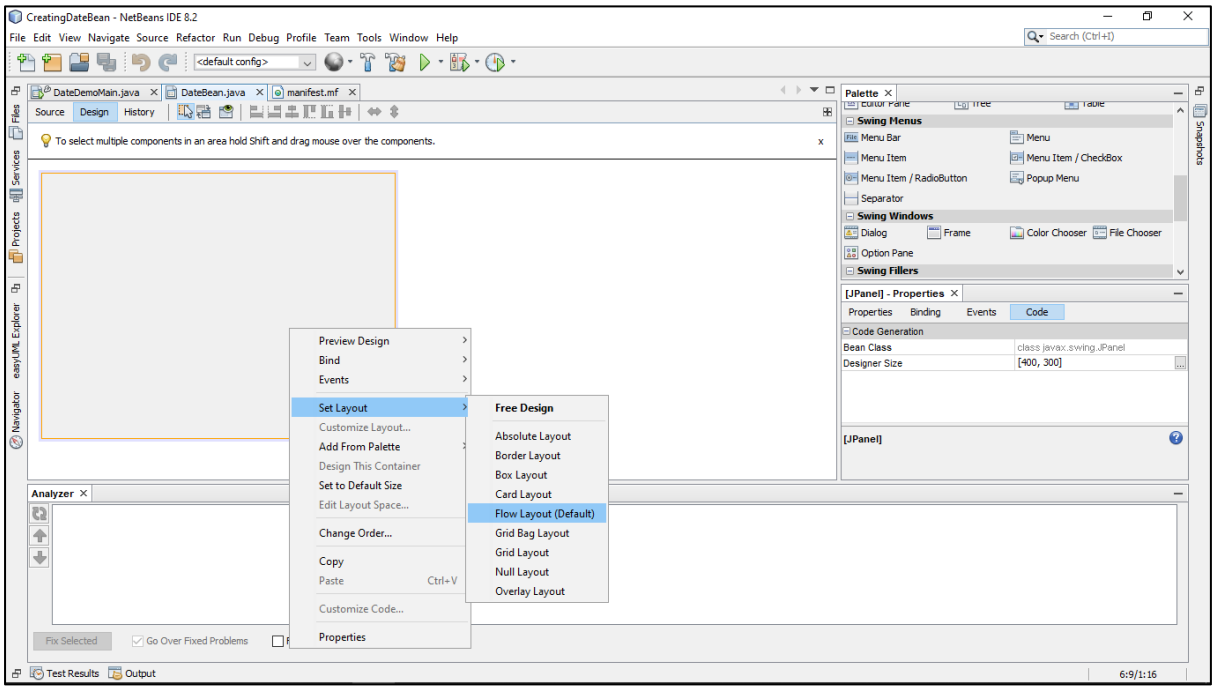
1.1 Student name	**Nkem Akwari**
1.2 Who did you work with? Name and/or id	**Trevor Kiggundu**
1.3 Which lab topic does this document relate to?	Creating software components
1.4 How well do you feel you have done?	• I have completed the exercise and am totally satisfied with my work
1.5 Briefly explain your answer to question 1.4	**While I have completed the work, I feel like this was the most challenging logbook exercise to do and highlighted a lot of errors I have made that I need to make sure I do not repeat, as well as new methods of coding. The optional tasks highlighted this due to the fact that we weren't knowledgeable with what was needed and had to look up references and ended up taking up more time that initially thought due to our attempts at trying to figure out what we were doing incorrectly with things such as events.**

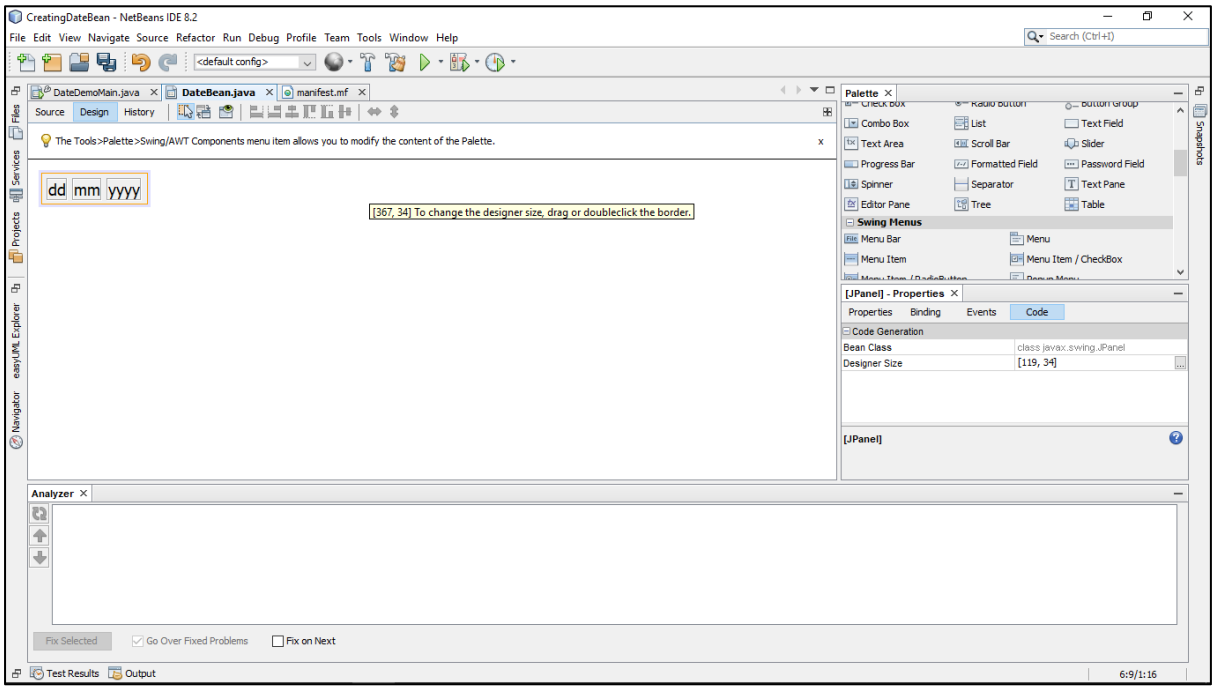
1. Implementation

2.1 Annotated screen shots demonstrating what you have achieved.




CreatingDateBean - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
DateDemoMain.java
DateBean.java
manifest.mf
Source
History
Manifest-Version: 1.0
Name: gre/cwk/DateBean/DateBean.class
Java Bean: True
Analyzer
Fix Selected
Go Over Fixed Problems
Fix on Next
Test Results
Output
5:1
INS


CreatingDateBean - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
DateDemoMain.java
DateBean.java
manifest.mf
Source
Design
History
To select multiple components in an area hold Shift and drag mouse over the components.
Palette
Swing Menus
Menu Bar
Menu
Menu Item
Menu Item / CheckBox
Menu Item / RadioButton
Popup Menu
Separator
Swing Windows
Dialog
Frame
Color Chooser
File Chooser
Option Pane
Swing Fillers
[JPanel] - Properties
Properties
Binding
Events
Code
Code Generation
Bean Class
class javax.swing.JPanel
Designer Size
[400, 300]
[JPanel]
Preview Design
Bind
Events
Set Layout
Customize Layout...
Add From Palette
Design This Container
Set to Default Size
Edit Layout Space...
Change Order...
Copy
Paste
Ctrl+V
Customize Code...
Properties
Free Design
Absolute Layout
Border Layout
Box Layout
Card Layout
Flow Layout (Default)
Grid Bag Layout
Grid Layout
Null Layout
Overlay Layout
Analyzer
Fix Selected
Go Over Fixed Problems
Test Results
Output
6:9/1:16


CreatingDateBean - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
DateDemoMain.java
DateBean.java
manifest.mf
Source Design History
The Tools>Palette>Swing/AWT Components menu item allows you to modify the content of the Palette.
dd mm yyyy
[367, 34] To change the designer size, drag or doubleclick the border.
Palette
Combo Box
List
Text Field
Text Area
Scroll Bar
Slider
Progress Bar
Formatted Field
Password Field
Spinner
Separator
Text Pane
Editor Pane
Tree
Table
Swing Menus
Menu Bar
Menu
Menu Item
Menu Item / CheckBox
[JPanel] - Properties
Properties Binding Events Code
Code Generation
Bean Class
class javax.swing.JPanel
Designer Size
[119, 34]
[JPanel]
Analyzer
Fix Selected
Go Over Fixed Problems
Fix on Next
Test Results
Output
6:9/1:16

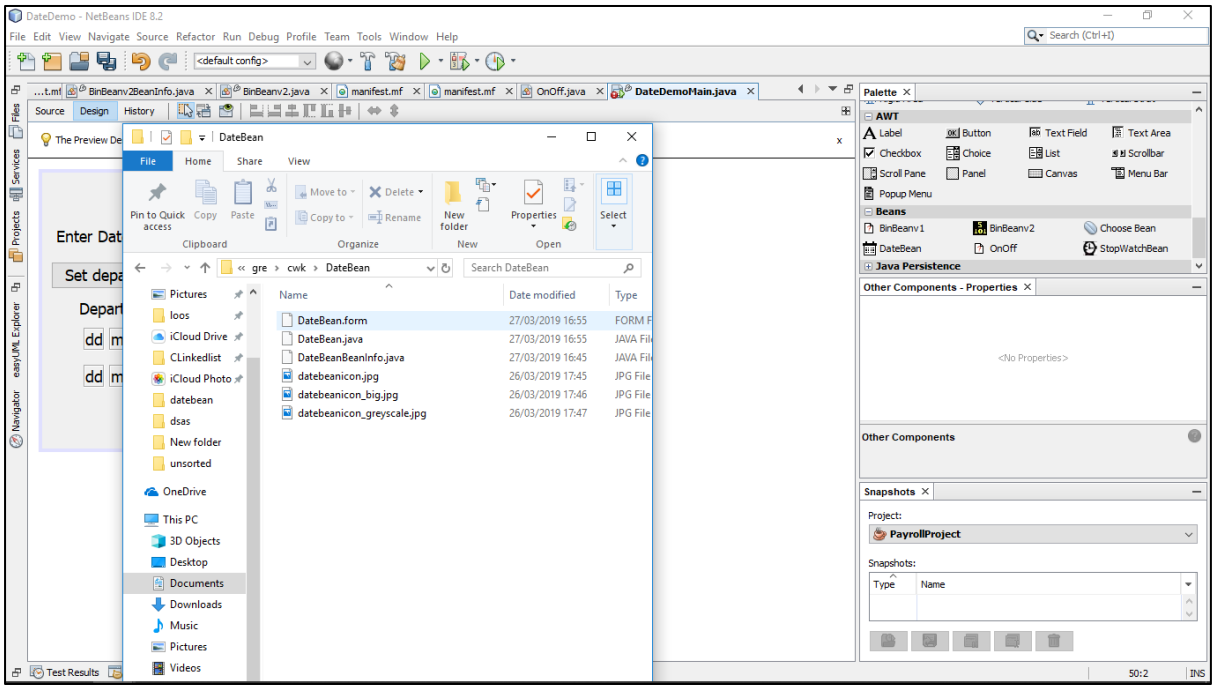

DateDemo - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
DateDemoMain.java
Source Design History
Enter Dat
Set depa
Depart
dd m
dd m
DateBean
File Home Share View
Pin to Quick access
Copy
Paste
Move to
Copy to
Delete
Rename
New folder
Properties
Select
Clipboard
Organize
New
Open
gre › cwk › DateBean
Search DateBean
Pictures
loos
iCloud Drive
CLinkedlist
iCloud Photo
datebean
dsas
New folder
unsorted
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Name
Date modified
Type
DateBean.form 27/03/2019 16:55 FORM F
DateBean.java 27/03/2019 16:55 JAVA File
DateBeanBeanInfo.java 27/03/2019 16:45 JAVA File
datebeanicon.jpg 26/03/2019 17:45 JPG File
datebeanicon_big.jpg 26/03/2019 17:46 JPG File
datebeanicon_greyscale.jpg 26/03/2019 17:47 JPG File
Palette
AWT
Label
Button
Text Field
Text Area
Checkbox
Choice
List
Scrollbar
Scroll Pane
Panel
Canvas
Menu Bar
Popup Menu
Beans
BinBeanv1
BinBeanv2
Choose Bean
DateBean
OnOff
StopWatchBean
Java Persistence
Other Components - Properties
<No Properties>
Other Components
Snapshots
Project:
PayrollProject
Snapshots:
Type
Name
Test Results
50:2
INS

Figure 5 Adding icons for the bean:

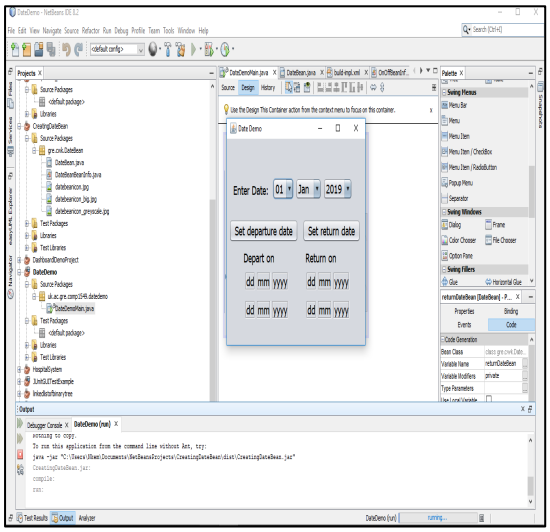


Figure 6: Evidence of the code running in full

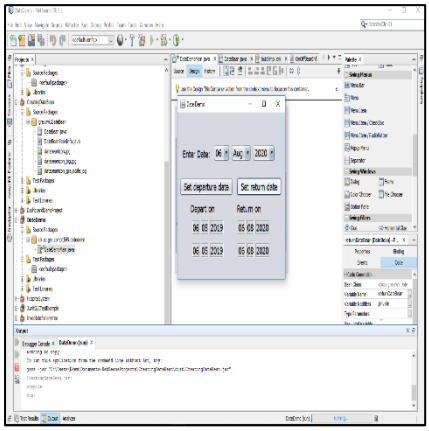


Figure 7: Evidence of the code running in full (2):

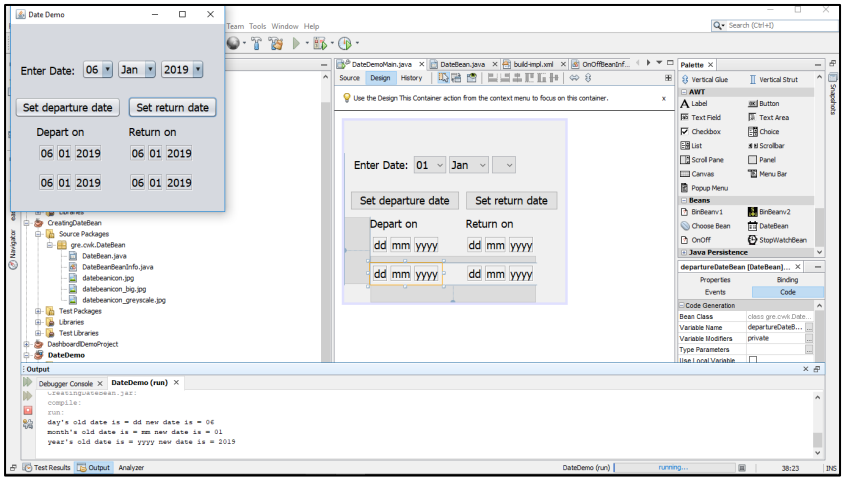
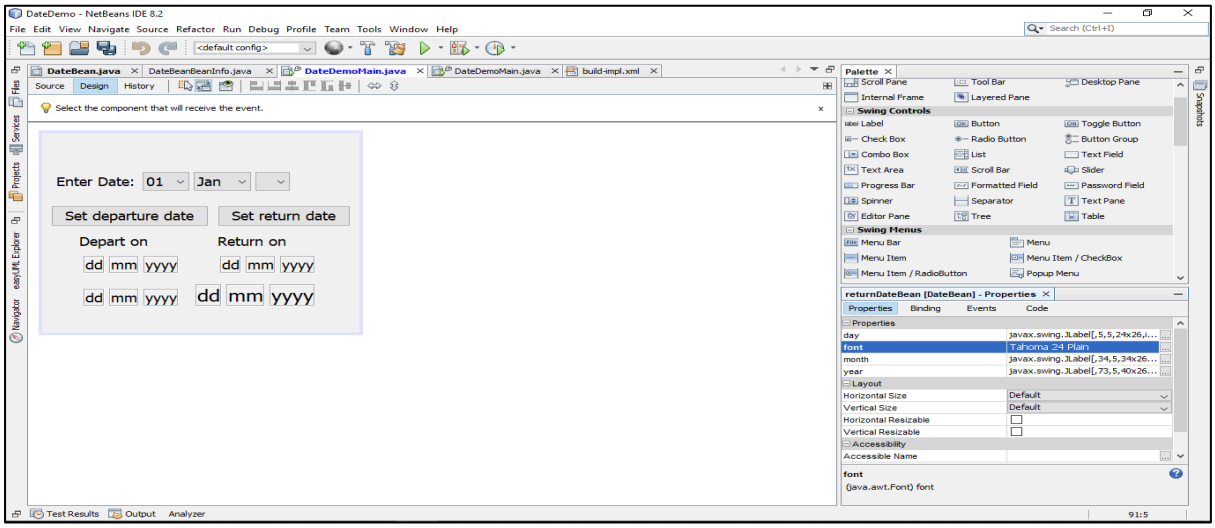


Figure 8: Evidence of the code running 2. Note how the output shows the old date and the new date for the returnDateBean due to the propertyChangeEvent:



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.

//Manifest code

Manifest-Version: 1.0

Name: gre/cwk/DateBean/DateBean.class

Java Bean: True

//BeanInfo Code

public class DateBeanBeanInfo extends SimpleBeanInfo {

// Get the properties to expose. Here we have decided to expose the "date","month" and year properties defined

// in the DateBean class itself and the "font" property inherited from JLabel

public PropertyDescriptor[] getPropertyDescriptors() {

```
        try {
            PropertyDescriptor d = new PropertyDescriptor("day", DateBean.class,
"getDay", "setDay");
            PropertyDescriptor m = new PropertyDescriptor("month", DateBean.class,
"getMonth", "setMonth");
            PropertyDescriptor y = new PropertyDescriptor("year", DateBean.class,
"getYear", "setYear");

            PropertyDescriptor f = new PropertyDescriptor("font", DateBean.class,
"getFont", "setFont");

            PropertyDescriptor[] pds = new PropertyDescriptor[]{
               d, m, y, f};

            return pds;

        } catch (IntrospectionException ex) {

            ex.printStackTrace();

            return null;

        }

    }
```

```
    // Get the image to use as an icon.  Note that the image files need to be included
    // in the bean's jar file.  One way is to put them in the same folder as the .java files
    // Another way is to create a folder (e.g. called icons) and in NetBeans
    // right-click the project and choose Properties->Sources to add that folder
    // to the Source Packages folders
        public Image getIcon(int iconKind) {

            switch (iconKind) {
                case BeanInfo.ICON_COLOR_16x16:
                    return loadImage("datebeanicon.jpg");
                case BeanInfo.ICON_COLOR_32x32:
                    return loadImage("datebeanicon_big.jpg");
                case BeanInfo.ICON_MONO_16x16:
                    return loadImage("datebeanicon_greyscale.jpg");
                case BeanInfo.ICON_MONO_32x32:
                    return loadImage("datebeanicon_greyscale.jpg");
            }
            return null;
        }

    }
```

//Modified DateBeanDemo Code

```
        private void btnSetDepartureDateActionPerformed(java.awt.event.ActionEvent evt)
{
```

```
        // Get the departure date input as ddmmmyyyy
        String strDateInput = "" + cmbDay.getSelectedItem() +
cmbMonth.getSelectedItem() + cmbYear.getSelectedItem();

        // Parse the departure date input to create a LocalDate object
        DateTimeFormatter dtf = DateTimeFormatter.ofPattern("ddMMMyyyy");
        departureDate = LocalDate.parse(strDateInput, dtf);
        if (returnDate != null && returnDate.isBefore(departureDate)) {
           JOptionPane.showMessageDialog(this, "can't depart after returning", "Error",
JOptionPane.ERROR_MESSAGE);
           return;
        }

        // Convert departure date to format for display
        String strDD = String.format("%02d", departureDate.getDayOfMonth());
        String strMM = String.format("%02d", departureDate.getMonthValue());
        String strYYYY = "" + departureDate.getYear();


        // Display the departure date - this code will change when the display field has
        // been converted to a JavaBean component
        lblDD.setText(strDD);
        lblMM.setText(strMM);
        lblYYYY.setText(strYYYY);
```

```
            departureDateBean.setDay(strDD);
            departureDateBean.setMonth(strMM);
            departureDateBean.setYear(strYYYY);
        }

        /**
         * Get the date entered in the set of combo boxes, parse it, check that it
         * is not before the departure date and display it.
         *
         * @param evt event object
         */
        private void btnSetReturnDateActionPerformed(java.awt.event.ActionEvent evt)
{
            // Get the return date input as ddmmmyyyy
            String strDateInput = "" + cmbDay.getSelectedItem() +
cmbMonth.getSelectedItem() + cmbYear.getSelectedItem();

            // Parse the return date input to create a LocalDate object
            DateTimeFormatter dtf = DateTimeFormatter.ofPattern("ddMMMyyyy");
            returnDate = LocalDate.parse(strDateInput, dtf);
            if (departureDate != null && departureDate.isAfter(returnDate)) {
                JOptionPane.showMessageDialog(this, "can't return before departing",
"Error", JOptionPane.ERROR_MESSAGE);
                return;
```

```
        }

        // Convert return date to format for display
        String strDD = String.format("%02d", returnDate.getDayOfMonth());
        String strMM = String.format("%02d", returnDate.getMonthValue());
        String strYYYY = "" + returnDate.getYear();

        // Display the return date - this code will change when the display field has
        // been converted to a JavaBean component
        lblDD1.setText(strDD);
        lblMM1.setText(strMM);
        lblYYYY1.setText(strYYYY);
        //add an event listener in order to create the console output
        departureDateBean.addPropertyChangeListener((PropertyChangeEvent evt1) ->
{
            System.out.println("old date is = " + evt1.getOldValue() + " new date is = " +
evt1.getNewValue());
        });

        returnDateBean.setDay(strDD);
        returnDateBean.setMonth(strMM);
        returnDateBean.setYear(strYYYY);

    }
```

//DateBean Code

```
public class DateBean extends javax.swing.JPanel {
    //property change support in order to support the event listener in Date Demo
    public PropertyChangeSupport pcs = new PropertyChangeSupport(this);
    /**
     * Creates new form DateBean
     */
    public DateBean() {
        initComponents();

        //creates a new propertychange listener in order to check if an event happens
        //once the event happens, it returns both the new and old value for the item which
triggered the event

    }
    //add listner to property changes in Date Demo so pcs causes the elements to
properly react
    @Override
        public void addPropertyChangeListener(PropertyChangeListener listener) {
         pcs.addPropertyChangeListener(listener);
     }

    //setter/getter for day label
public String getDay(){
    //returns current value as a string
return lblDD.getText();
}

public void setDay(String dd){
        //fire property change for day
    //turns a string into the display for the current label
firePropertyChange("day", getDay(),dd );
lblDD.setText(dd);
}
public String getMonth(){
```

```
return lblMM.getText();
}

public void setMonth(String mm){
        //fire property change for month
    firePropertyChange("month", getMonth(),mm );
lblMM.setText(mm);
}
public String getYear(){

return lblYYYY.getText();

}

public void setYear(String yyyy){
    //fire property change for year
firePropertyChange("year", getYear(),yyyy );
lblYYYY.setText(yyyy);

}
@Override
public void setFont(Font font){
    //if statement to get around the nullpointer exception created when changing
    //the font with no text in
    if(lblDD != null && lblMM != null && lblYYYY != null ){
        //sets all values to the currently inputted font
        lblDD.setFont(font);
        lblYYYY.setFont(font);
    lblMM.setFont(font);
   // this.setFont(font);
```

```
    }
}

    @Override
    //returns super in order to get overall font setting
    public Font getFont(){
return super.getFont();
}

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        lblDD = new javax.swing.JLabel();
        lblMM = new javax.swing.JLabel();
        lblYYYY = new javax.swing.JLabel();

        setFont(new java.awt.Font("Tahoma", 0, 18)); // NOI18N
        setName("DateBeanContainer"); // NOI18N

        lblDD.setFont(new java.awt.Font("Tahoma", 0, 18)); // NOI18N
        lblDD.setText("dd");
        lblDD.setBorder(javax.swing.BorderFactory.createEtchedBorder());
        add(lblDD);

        lblMM.setFont(new java.awt.Font("Tahoma", 0, 18)); // NOI18N
        lblMM.setText("mm");
        lblMM.setBorder(javax.swing.BorderFactory.createEtchedBorder());
        add(lblMM);
```

```
        lblYYYY.setFont(new java.awt.Font("Tahoma", 0, 18)); // NOI18N
        lblYYYY.setText("yyyy");
        lblYYYY.setBorder(javax.swing.BorderFactory.createEtchedBorder());
        add(lblYYYY);
    }// </editor-fold>


    // Variables declaration - do not modify
    private javax.swing.JLabel lblDD;
    private javax.swing.JLabel lblMM;
    private javax.swing.JLabel lblYYYY;
    // End of variables declaration
}
```

References:

- Anon (n.d.) Software Design Patterns, *GeeksforGeeks*, [online] Available at: https://www.geeksforgeeks.org/software-design-patterns/ (Accessed April 9, 2019).

- *Java Swing StopLight GUI* (2016) YouTube video, added by KelliKOnline[Online]. Available at https://www.youtube.com/watch?v=BUpatHRLarI (Accessed April 9, 2019).

- J*ava Swings Tutorials - 24 - Creating Digital Clock in Java Swings* (2015) YouTube video, added by Infinity[Online]. Available at https://www.youtube.com/watch?v=e3PnuTUjmQs (Accessed April 9, 2019).

- Tutorialspoint.com (n.d.) Design Patterns Singleton Pattern, *www.tutorialspoint.com*, [online] Available at: https://www.tutorialspoint.com/design_pattern/singleton_pattern.htm (Accessed April 9, 2019).

- Tutorialspoint.com (n.d.) Java Interfaces, *www.tutorialspoint.com*, [online] Available at: https://www.tutorialspoint.com/java/java_interfaces.htm (Accessed April 9, 2019).