**Lab: Lab 2**

**Registration number: 1703055**

|  |
| --- |
| **Problem statement**  The purpose of this lab was to practice the creation of Graphical User Interfaces (GUIs) with the use of Java Swing libraries but to also build upon the idea of inheritance with objects where by the methods and variable definitions are inherited in *rectangle* from *square*. |

|  |
| --- |
| **Program description(s)**  *Exercise 1:* For the first exercise, I simply wrote the *SquareSimp* application using the code pre-supplied. This supplies a main class for initialising the frame and two other classes:  *FilledFrame* class which extends the JFrame to create a frame which contains all the appropriate buttons added to a panel which is placed into the NORTH of the frame by utilising the borderLayout jswing layout type. In this class the squarePanel is initialised and placed in the frame  *Then the squarePanel* class extends the JPanel that is within the *FilledFrame* in order to draw the square component  Overall this results in a window being displayed as shown below with a green square to the central panel and buttons to the north. However the buttons do nothing as they do not have a button listener in order to handle the action of clicking the button.  *Exercise 2:* The program must be reactive to user interactions with the “Say Hi!” button. For this exercise I wrote an anonymous *ActionListener* in order to deal with button presses of the “Say Hi!” button. For this the message button has an anonymous *ActionListener()* implemented. This is anonymous in order to pass the JPane parameter directly to the *actionPerformed()* constructor in the implemented listener in order for a show MessageDialog to appear.  *Exercise 3:* The program must handle actions on the other 3 buttons for the resizing of the square. In order to do this I created an ActionListener class called butListener which takes in two arguments; first an integer value for the size that I want the square to be changed to and second argument is a reference to the FilledFrame class in order for the class to be able to access the frame the square is in. within the butListener class is a constructor for the butListeners attributes to be outlines and an *actionPerformed()* method which will be called when the button is pressed. This method sets the new intended size of the square and then repaints the frame for it to be rendered. Finally the *addActionListener* which is added to each of the buttons individually, calls to the actionPerformed() constructor passing the two attributes needed – the desired new size and frame of the square. |

|  |
| --- |
| **Test description**  ***Exercise 1:***  Test 1: The program should display a green square to the central panel and buttons to the north. Buttons should not have any actions.  Result 1: **Pass** -The program works as intended as shown below. Green square is present and buttons do not have any actions. All elements are in the correct locations as required.  Test 2: No further testing necessary, for this exercise.  Result 2: n/a  ***Exercise 2:***  Test 1: The program must be reactive to user interactions with the “Say Hi!” button.  Result 1: **Pass** – As seen below the pop up appears as required and will disappear with the user clicking ‘ok’ to dismiss.  ***Exercise 3:***  Test 1: Does the program resize with each of the different button presses, for different sizes?  Result 1: **Pass –** see below to screen shots which show that the square is resized for each of the button presses. |
| **Known bugs**  No known bugs |
| **Possible improvements**  I could add a button which changes the colour of the square. Alternatively, for added functionality, a menu or other button to change shape would be an improvement to this program. |
| **Comments**  Some students may have had problems with gitHub but I thought using it for this module is really helpful for accessibility of my code at both at home or university. I think we should defiantly be taught about version control in our modules. |
| **Extra credit**  No extra credit made |
| **References**  **CE203 Part 2 (Week 3 to 4) slides.**  In particular slide 40 onwards on Border Layout and slides 55 onwards on Action Listeners.  **Java API on ActionListener interface.**  Found here <https://docs.oracle.com/javase/7/docs/api/java/awt/event/ActionListener.html> |