Name: Lum Chee Xiang Michael (Lan Zhixiang)
Student ID: 111102750
Assignment 4

Module: Introduction to Java and Object-oriented Programming
Module number: CO1109-01

University of London International Programmes
BSc in Computing and Information Systems: Introduction to Java and Object-oriented programming
Module number: CO1109-01
Assignment 4

Name: Lum Chee Xiang Michael (Lan Zhixiang)
Student ID: 111102750
Module: Introduction to Java and Object-oriented Programming
Module number: CO1109-01
Assignment 4

Source codes (Original program) (“Show message and confirm dialog box”, 2007)

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

public class ShowMessageDialog{
  JButton button;
  public static void main(String[] args){
  ShowMessageDialog md = new ShowMessageDialog();
  }

  public ShowMessageDialog(){
  JFrame frame = new JFrame("Message Dialog Box");
  button = new JButton("Show simple message dialog box");
  button.addActionListener(new MyAction());
  JPanel panel = new JPanel();
  panel.add(button);
  button = new JButton("Show \"Ok/Cancel\" message
dialog box");
  button.addActionListener(new MyAction());
  panel.add(button);
  button = new JButton("Show \"Yes/No/Cancel\" dialog box");
  button.addActionListener(new MyAction());
  panel.add(button);
  frame.add(panel);
  frame.setSize(400, 400);
  frame.setVisible(true);
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }

  public class MyAction implements ActionListener{
  public void actionPerformed(ActionEvent ae){
  String str = ae.getActionCommand();
  if(str.equals("Show simple message dialog box")){
  JOptionPane.showMessageDialog(null, "This is the simple message
dialog box.", "Roseindia.net", 1);
  }
  else if(str.equals("Show \"Ok/Cancel\" message dialog box")){
  if(JOptionPane.showConfirmDialog(null, "This is the \"Ok/Cancel\"
message dialog box.", "Roseindia.net", JOptionPane.OK_CANCEL_OPTION) == 0)
  JOptionPane.showMessageDialog(null, "You clicked on \"Ok\" button",
 "Roseindia.net", 1);
  else
  JOptionPane.showMessageDialog(null, "You clicked on \"Cancel\" button",
 "Roseindia.net", 1);
  }
  else if(str.equals("Show \"Yes/No/Cancel\" dialog box")){
  JOptionPane.showConfirmDialog(null, "This is the \"Yes/No/Cancel\"
 message dialog box.");
  }
  }}}
```

Name: Lum Chee Xiang Michael (Lan Zhixiang) Module: Introduction to Java and Object-oriented Programming
Student ID: 111102750 Module number: CO1109-01
Assignment 4

Source code (Candidate's written program)

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

public class guisandbuttons
{
   JButton button; //this section defines the text that will appear on the main program's buttons
   String buttontext1 = "Explanation: How this program works";
   String buttontext2 = "About this program";
   String buttontext3 = "Easter Eggs";
   String buttontext4 = "Quit program";
   String buttontextyes = "yes";

   public static void main (String[] args)
   {
      guisandbuttons app = new guisandbuttons();  //this section creates an instance of the guiandbuttons method used for initializing the entire program
      app.initialize();
   }


   public void startingWindow()
   {
      JOptionPane.showMessageDialog(null, "Welcome to my experimental Swing GUI for Coursework 4", "University of London: Java Coursework 4", 1); //this is a simple Alert window that is displayed before the main program window is drawn onscreen

   }

    public void mainApp()
   {
      JFrame frame = new JFrame("University of London: Java Coursework 4"); //this line sets a title for the program's window

      button = new JButton(buttontext1);
      button.addActionListener(new captureClicks());
      JPanel panel = new JPanel(new GridLayout(5,3,2,2)); //GridLayout is used to lock the buttons in place so that they remain in their fixed positions even after the window is resized
      panel.add(button);
      button = new JButton(buttontext2);
      button.addActionListener(new captureClicks());
      panel.add(button);
      button = new JButton(buttontext3);
      button.addActionListener(new captureClicks());
      panel.add(button);
      button = new JButton(buttontext4);
      button.addActionListener(new captureClicks());
      panel.add(button);
      frame.add(panel);
      frame.setSize(500, 200);
```

```
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public class captureClicks implements ActionListener
    {

        int selection = 0;
        int selection1 = 0;
        String[] choices = {"I'm out of here!", "No thanks", "Huh?"}; //this array is used to hold the values for various text Strings which will be used to replace the standard button text found on dialogue boxes
        String[] joke = {"Nobody wants to play with me..."};
        String[] pissed = {"Sorry... T.T"};

        public void actionPerformed(ActionEvent ae)
        {
            String[] explanation = new String[5]; //this array houses the content that is to be displayed upon selection of the Program Explanation button
            explanation[0] = "I will now explain how this program works.";
            explanation[1] = "The goal is to create a program that a user can use to navigate back and forth, \nmuch like how a typical installer prompt or a basic step-by-step FAQ window \nwould work.";
            explanation[2] = "String arrays are used to contain the text to be displayed in each window";
            explanation[3] = "Finally, the act of moving back and forth is mapped to 'YES' and 'NO' buttons, which have been renamed to reflect their functions";
            explanation[4] = "When the last entry in the array is reached, the window closes once the user selects 'Next'.";

            String[] navigate1 = {"Next"};
            String[] navigate2 = {"Back", "Next"};
            String buttonword = ae.getActionCommand();
            if(buttonword.equals(buttontext1)) //this section handles the 'back-and-forth' display of text based on the user's selection on the corresponding Alert windows that are generated
            {
                selection1=JOptionPane.showOptionDialog(null, explanation[0], "Thank you for viewing", JOptionPane.YES_NO_OPTION, JOptionPane.INFORMATION_MESSAGE, null, navigate2, navigate2[1]);
                {
                    int i = 0;
                    while (i <= 4)
                    {
                        if (selection1 == JOptionPane.NO_OPTION)
                        {
                            i++;
                            if(i<=4)
                            selection1=JOptionPane.showOptionDialog(null, explanation[i], "Thank you for viewing", JOptionPane.YES_NO_OPTION, JOptionPane.INFORMATION_MESSAGE, null, navigate2, navigate2[1]);
                            else
                            return;
                        }
                        else if (selection1 == JOptionPane.YES_OPTION)
                        {
                            i--;
                            if(i>0)
                            selection1=JOptionPane.showOptionDialog(null, explanation[i], "Thank you for viewing", JOptionPane.YES_NO_OPTION, JOptionPane.INFORMATION_MESSAGE, null, navigate2, navigate2[1]);
```

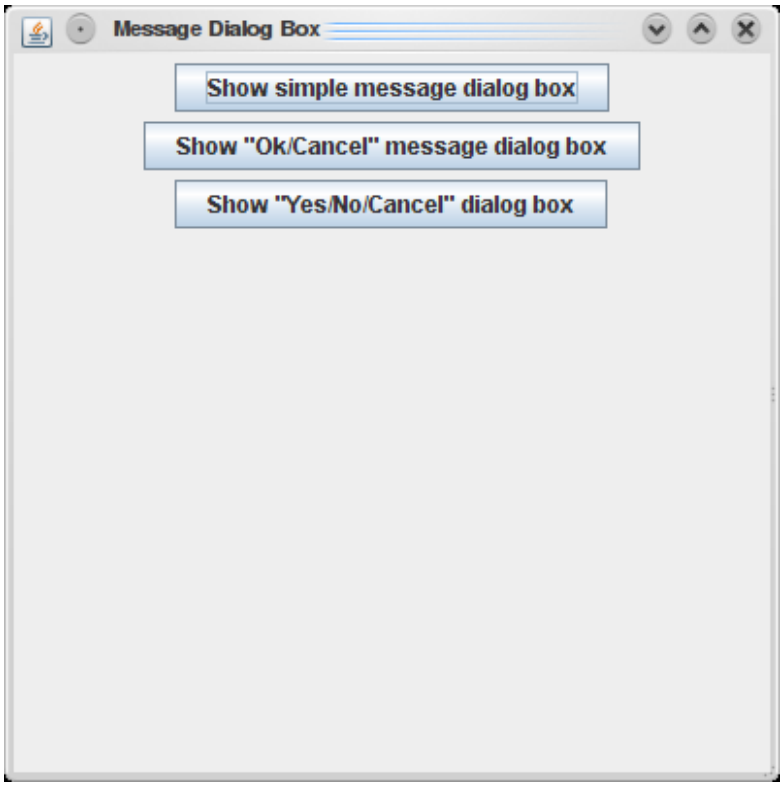
```
                    else
                        return;
                }
            }
        }
    }
    else if (buttonword.equalsIgnoreCase(buttontext2))
    {

        JFrame frame = new JFrame("About this Program"); //this section involves the passing of values to the window that will be displayed on screen when the 'About this Program' button is clicked

        JPanel panel = new JPanel(new GridLayout(4,3,2,2));
        JLabel text1 = new JLabel("<html>This program makes use of code from the following sources: <br /></html>");
        ImageIcon icon = new ImageIcon("banner.png");
        JLabel banner = new JLabel(icon);
        JLabel text2 = new JLabel("<html><br />Written with Java SDK v1.7.0 and compilled with Netbeans 7.0.1 for <br />Linux x64 distributions</html>");
        JLabel text3 = new JLabel("This project binary and its source is licensed under the GPLv2");
        panel.add(text1);
        panel.add(banner);
        panel.add(text2);
        panel.add(text3);
        frame.add(panel);
        frame.setSize(500, 200);
        frame.setVisible(true);

    }

    else if (buttonword.equalsIgnoreCase(buttontext3))
    {
        JOptionPane.showOptionDialog(null, "Sorry, no easter eggs in this program.", "Thank you for viewing", JOptionPane.YES_OPTION, JOptionPane.ERROR_MESSAGE, null, joke, joke[0]);
    }

    else if (buttonword.equals(buttontext4)) //this section handles the 'Quit' function on the program; upon clicking it, a confirmation dialogue with multiple choices will be presented to the user
    {
        selection = JOptionPane.showOptionDialog(null, "Are you sure you want to quit?", "Exit program", JOptionPane.YES_NO_CANCEL_OPTION,
  JOptionPane.QUESTION_MESSAGE, null, choices, choices[2]);
        {
            if (selection == JOptionPane.YES_OPTION)
            {
                JOptionPane.showMessageDialog(null, "The program will now terminate.", "Thank you for viewing", 1);
                System.exit(0); //if this option is selected, the program shuts down immediately
            }

            else if (selection == JOptionPane.NO_OPTION)
            {
                JOptionPane.showMessageDialog(null, "The program will not terminate.", "Thank you for staying", 1);
            }
```

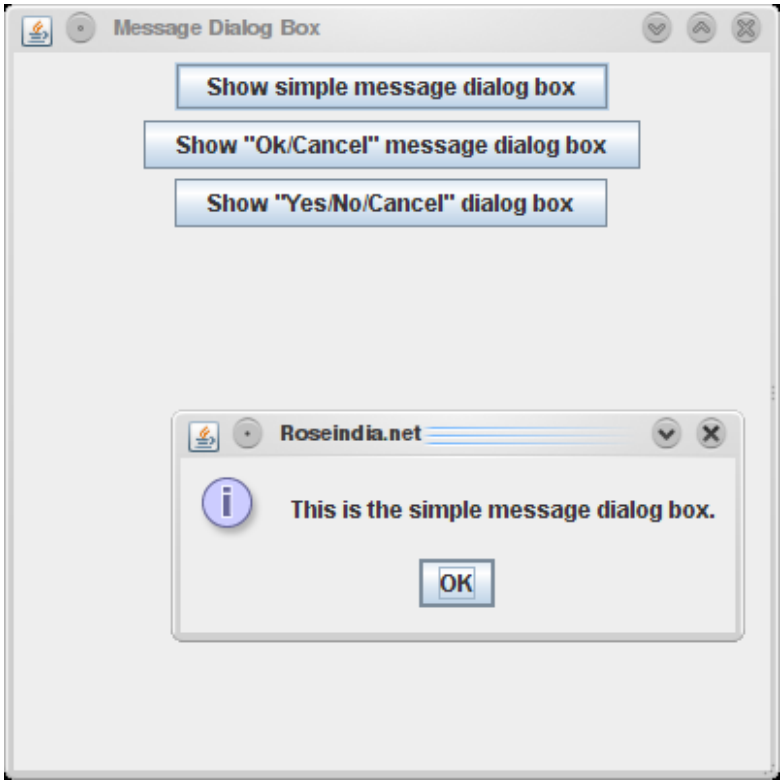
```
                else if (selection == JOptionPane.CANCEL_OPTION)
                {
                   JOptionPane.showOptionDialog(null, "Make up your mind already!", "Sheesh...",
JOptionPane.YES_OPTION,
   JOptionPane.WARNING_MESSAGE, null, pissed, pissed[0]);
                }

             }
          }
       }
   }
   public void initialize()
   {
      startingWindow();
      mainApp();
   }

}
```

Name: Lum Chee Xiang Michael (Lan Zhixiang)
Student ID: 111102750
Assignment 4

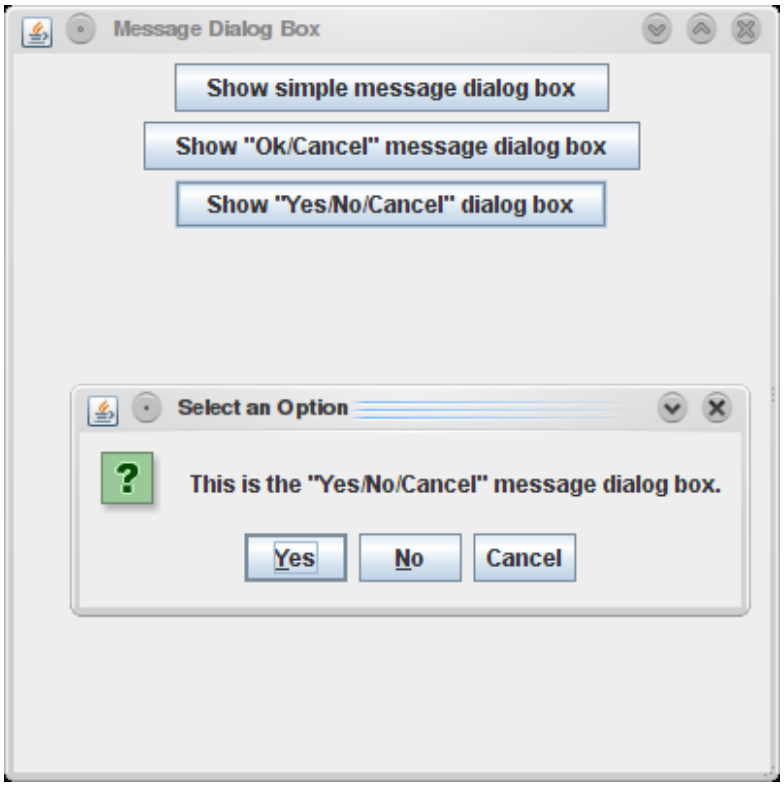
Module: Introduction to Java and Object-oriented Programming
Module number: CO1109-01

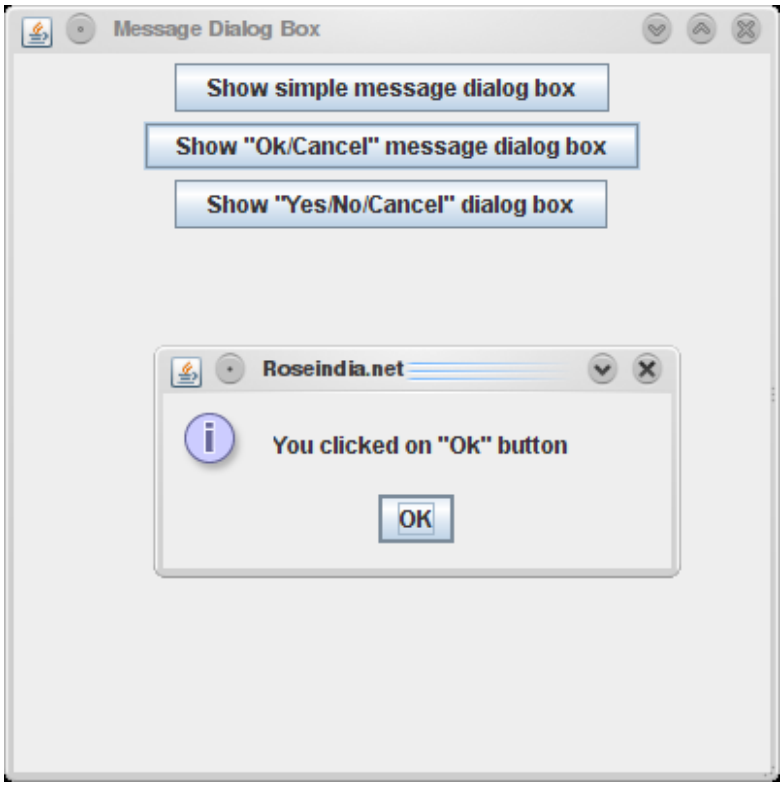
Screen dumps (Original program)

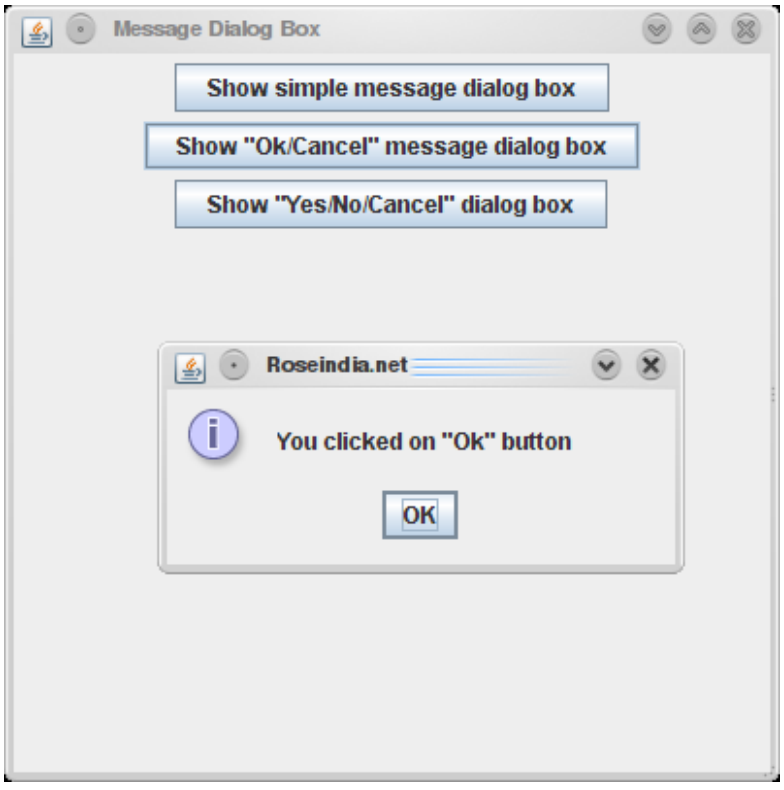





Message Dialog Box
Show simple message dialog box
Show "Ok/Cancel" message dialog box
Show "Yes/No/Cancel" dialog box
Roseindia.net
This is the "Ok/Cancel" message dialog box.
OK
Cancel


Message Dialog Box
Show simple message dialog box
Show "Ok/Cancel" message dialog box
Show "Yes/No/Cancel" dialog box
Select an Option
This is the "Yes/No/Cancel" message dialog box.
Yes
No
Cancel

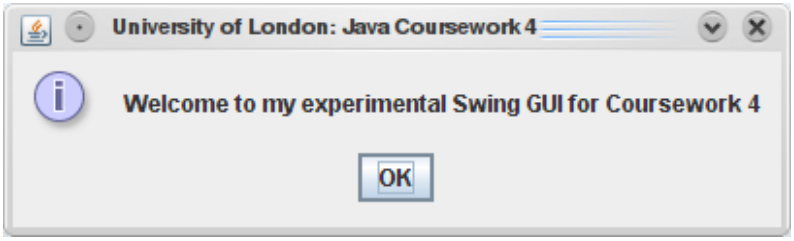

Message Dialog Box
Show simple message dialog box
Show "Ok/Cancel" message dialog box
Show "Yes/No/Cancel" dialog box
Roseindia.net
You clicked on "Ok" button
OK

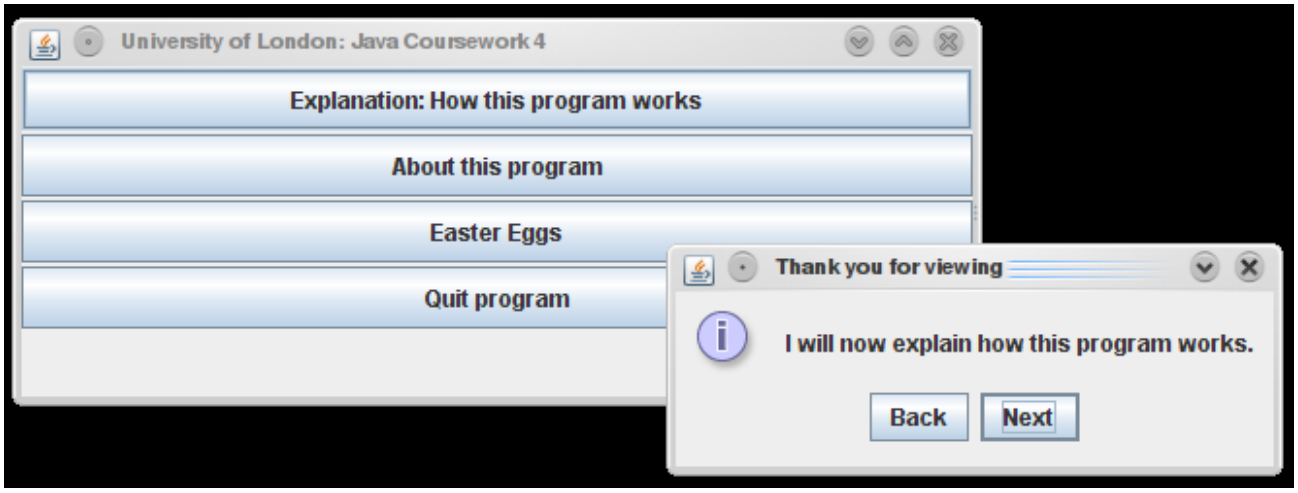

Message Dialog Box
Show simple message dialog box
Show "Ok/Cancel" message dialog box
Show "Yes/No/Cancel" dialog box
Roseindia.net
You clicked on "Ok" button
OK

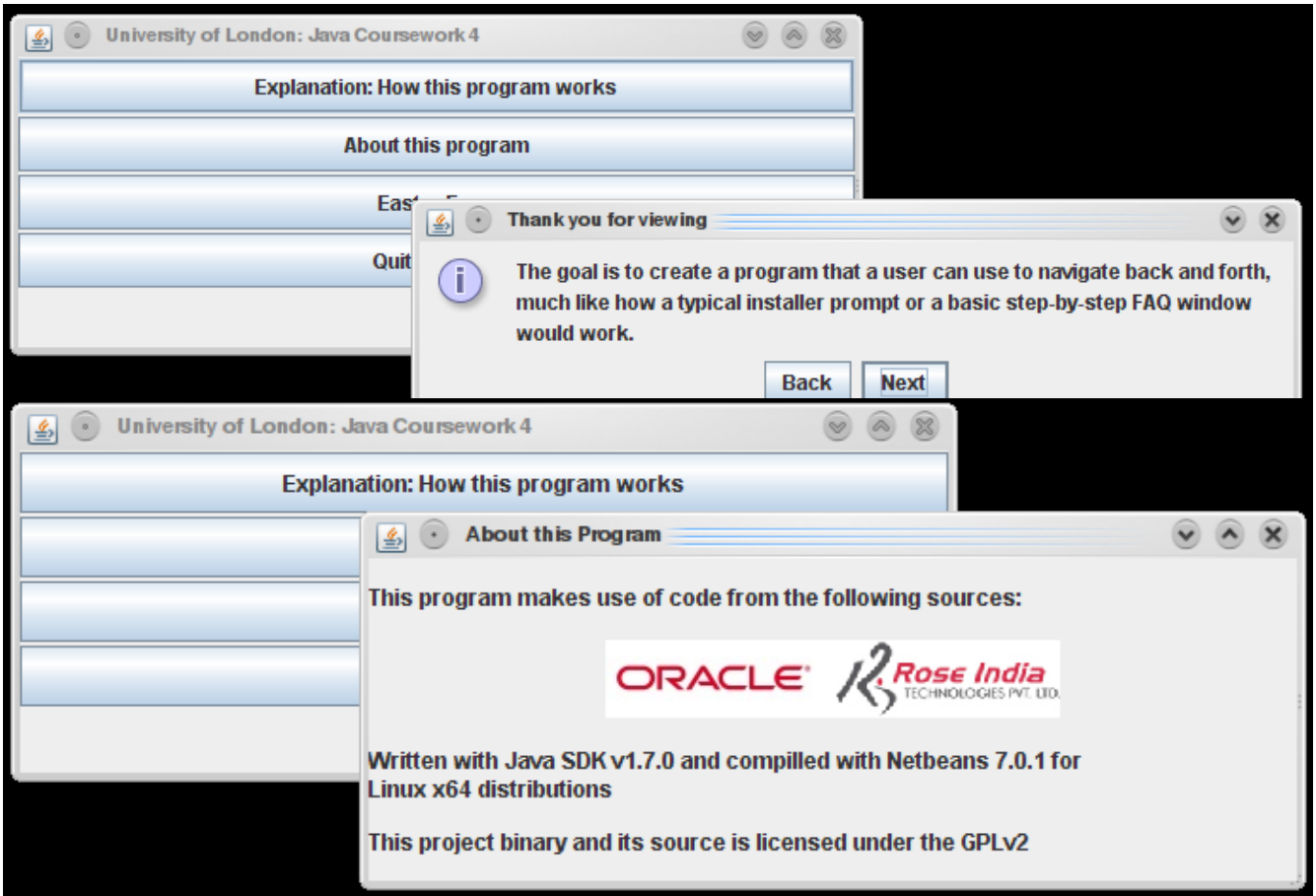
Name: Lum Chee Xiang Michael (Lan Zhixiang)
Student ID: 111102750
Assignment 4

Module: Introduction to Java and Object-oriented Programming
Module number: CO1109-01

Screen dump (Candidate's program)

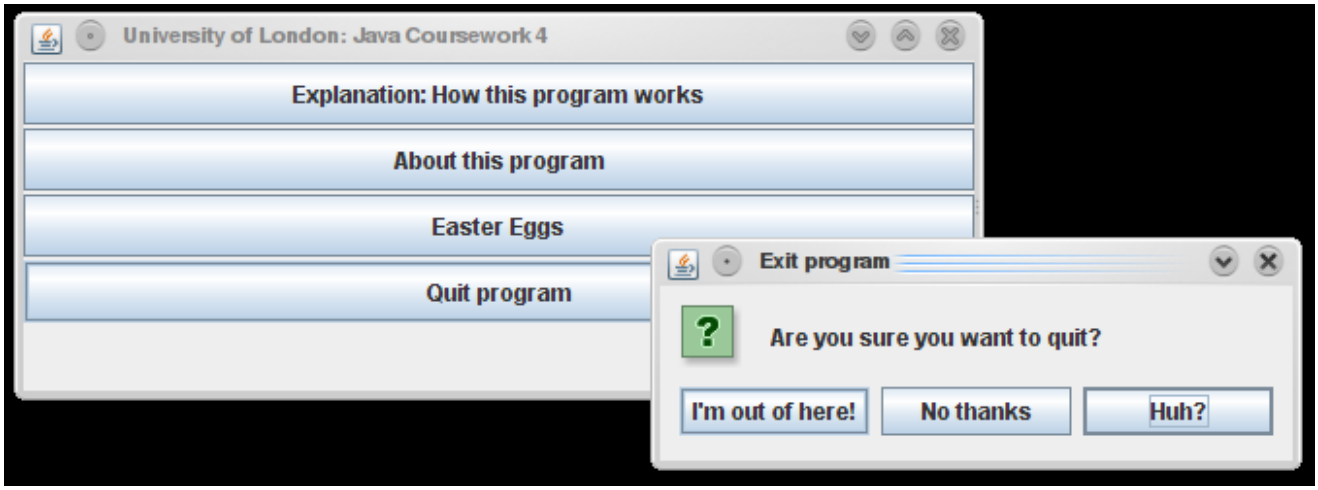


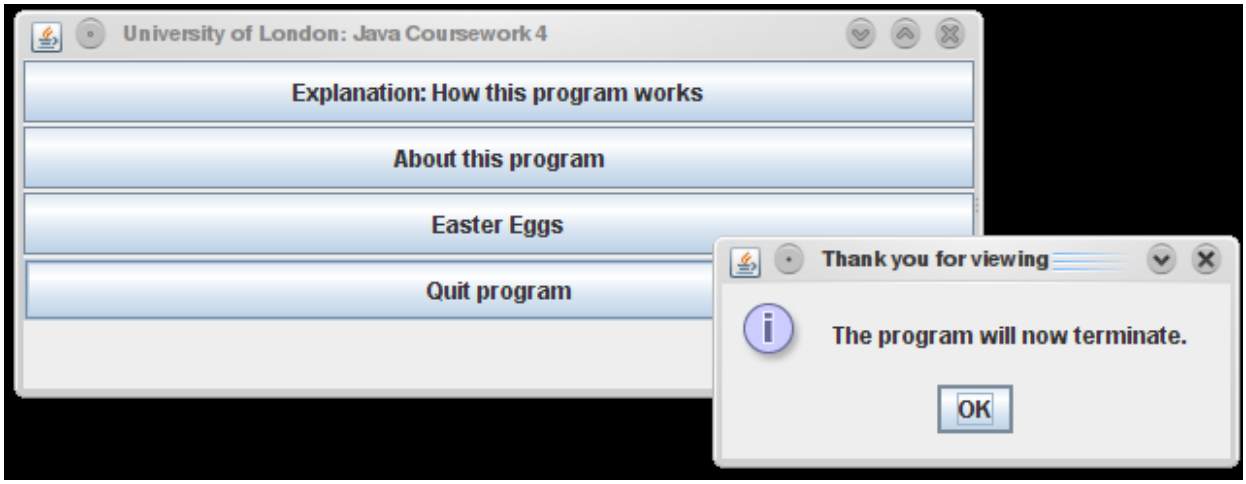




Name: Lum Chee Xiang Michael (Lan Zhixiang)
Student ID: 111102750
Assignment 4

Module: Introduction to Java and Object-oriented Programming
Module number: CO1109-01





Name: Lum Chee Xiang Michael (Lan Zhixiang) Module: Introduction to Java and Object-oriented Programming
Student ID: 111102750 Module number: CO1109-01
Assignment 4

Program Description

Motivation:

In this particular assignment, the candidate's desired learning outcome was to successfully create a primitive but workable graphical user interface consisting of a variety of windows, alert dialogues, error dialogues and confirmation dialogues which are triggered by the selection of various clickable buttons on a parent program. In addition, the candidate also aimed to gain an understanding on how images can be inserted into said windows, and whether it was possible to modify the standard text on such dialogue windows in order to provide an end-user with a customized experience that fits the parent program.

The motivation for embarking on such a task is extremely basic: as virtually all computer users in today's day and age have near zero experience with computing on a command line interface, a person who wants to pursue a career in computing or programming needs to be able to write a proper graphical front-end for any application, utility or tool that he or she is tasked to program. Furthermore, in most cases, the average end-user is not going to be concerned whether the backend code is well written or not, as they tend to judge the quality of the program based on its graphical appearance (with Linux being the exception to the rule, where high-quality code delivered over a command line interface is valued over subpar programs with graphical interfaces). We can see real-world examples of this in any application for any operating system; for example, AMD's Catalyst display drivers are widely regarded to be subpar but their driver utility has a good-looking graphical interface.

Based on this understanding, the candidate has come to the conclusion that the first step to pursuing a career in systems development and programming is heavily tied to one's ability to write a proper graphical interface. Therefore, the candidate had decided to start off by attempting to learn how a simple graphical interface is written

Description: Original program

The original program, as taken from the RoseIndia website, consists of a simple window that utilizes Java's Swing functionality to render simple application windows which most users have come to expect from virtually any application available for a computer today.

Upon execution, the original program will open a simple window that contains three buttons: the first will display a simple alert dialogue window, while the other two will open confirmation dialogues with varying button choices. Selecting each of the different buttons in the aforementioned confirmation dialogue boxes will trigger different alert windows, as demonstrated in the screen dumps above. Closer inspection of the source code reveals that this is accomplished via the following means:

- the Alert and Confirmation dialogues are provided by the
- for the confirmation dialogues, the button choices can be controlled to a certain degree by utilizing JOptionPane.YES_NO_OPTION or JOptionPane.YES_NO_CANCEL_OPTION to specify whether there should be two or three buttons in the confirmation dialogue.
    - In addition, the candidate has found out that the Yes, No and Cancel options are but arbitrary buttons that can be mapped to a variety of functions, this can be seen when the 'Cancel' button on the sample program is capable of opening up its own dialogue window instead of closing the program.

Lastly, the program appears to be designed to run until the user explicitly terminates it by clicking on the 'X' button (or its equivalent) on the window's border, as there is no 'Quit' button to be found in the program window.

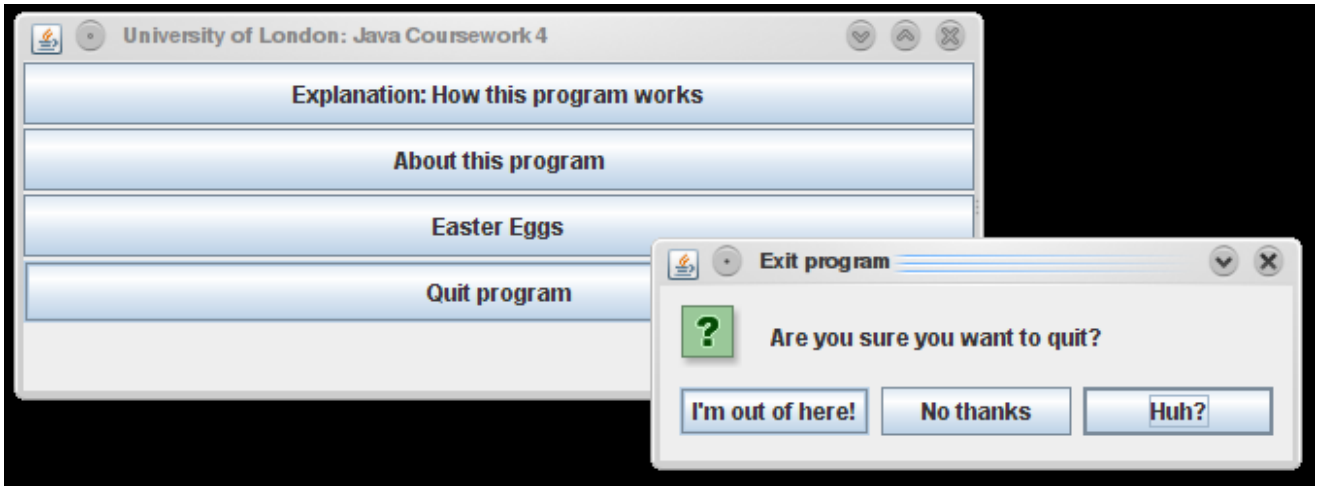
Name: Lum Chee Xiang Michael (Lan Zhixiang)
Student ID: 111102750
Assignment 4

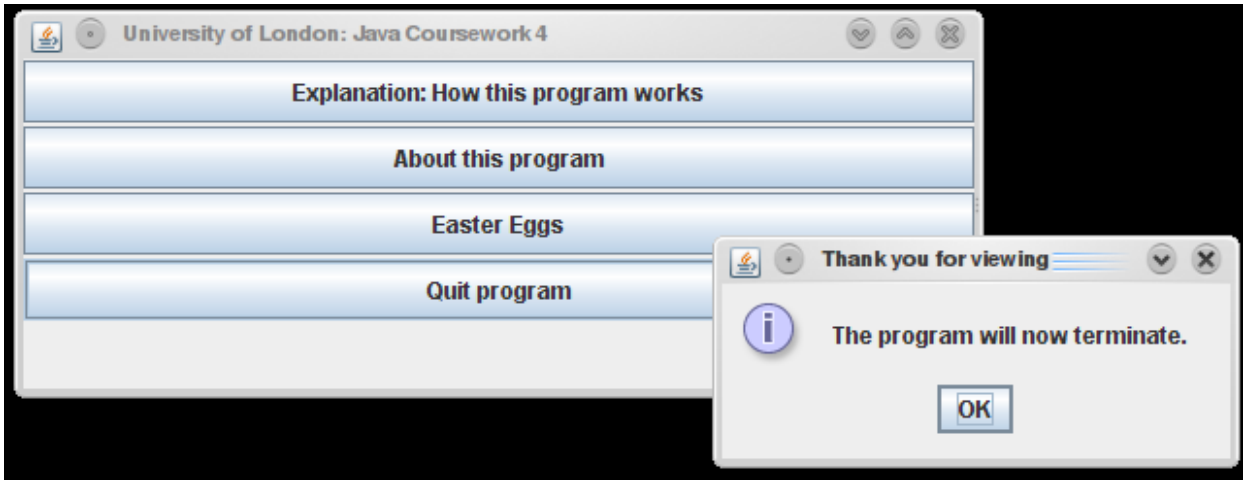
Module: Introduction to Java and Object-oriented Programming
Module number: CO1109-01

Description: Candidate's program

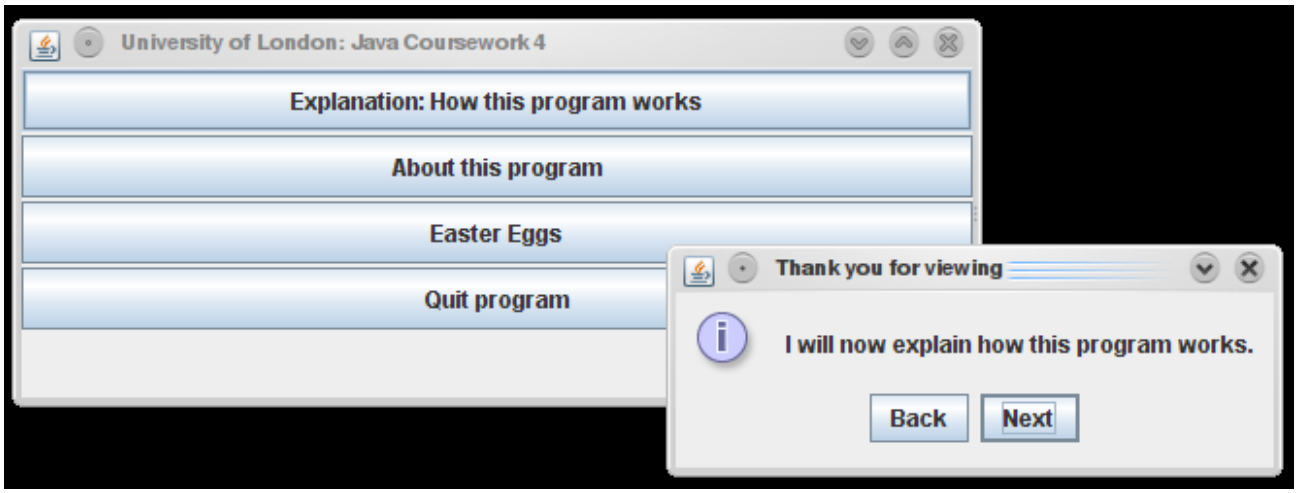
In terms of functionality, this candidate's program is largely identical to the original in that it performs the same tasks (i.e, opening various dialogue windows upon selection of specific buttons in the main program). However, several differences and edits have been made to make the program look more 'useful' instead of merely opening windows for the sake of illustrating Java's Swing GUI capabilities.

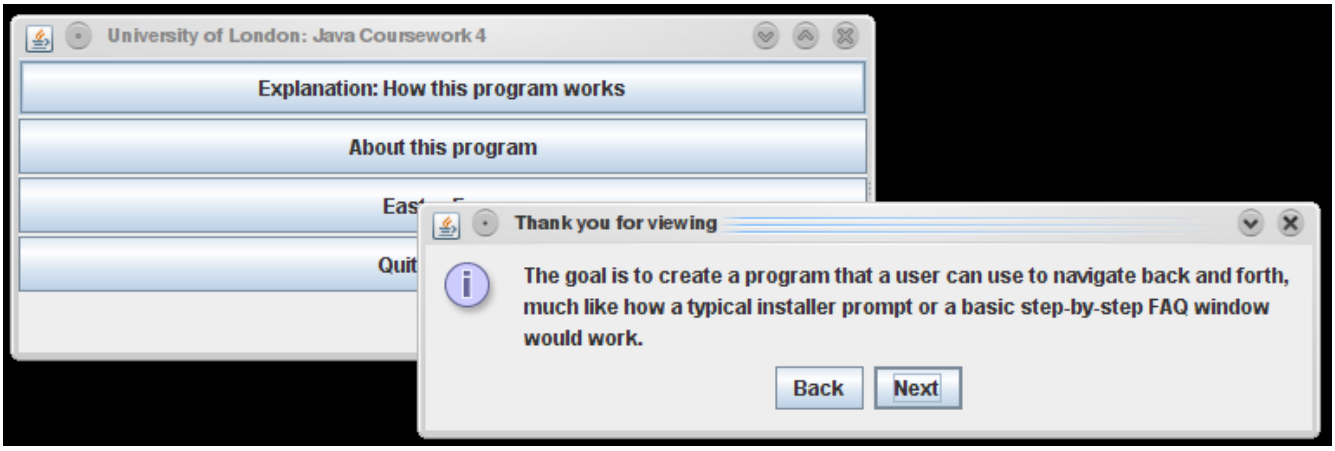
The first change the candidate has done to the original program is to include a 'Quit' button in the parent window that will bring up a confirmation prompt which will alert the user if he or she really wants to terminate the program. This is done by binding the 'Quit' button to its own confirmation window. In addition, each of the buttons in the resulting confirmation dialogue are further bound to their own alert dialogues; this is to provide the user with visual feedback and confirmation of their selected options.





Another addition made to the original program was the candidate's attempt to add some form of sequentiality to the dialogue windows based on the user's selection, much like how an installer window and the old dialogue window-based FAQs work. Upon clicking the 'Explanation: How this program works' button, Java will open a confirmation window that prompts the user to click 'Back' or 'Next', which are common button choices found in installer prompts. Selecting 'Next' will bring the user to the next part of the program, while selecting 'Back' will take the user to the previous message; upon reaching the last window in the sequence, clicking on 'Next' will simply cause the window to close so that the user has access to the parent program again; this is also true for the 'Back' button if the user navigates all the way back to the first window in the sequence.





In addition, the candidate attempted to make the program a little more real world-like by integrating images into the dialogue windows. This can be seen when the user selects the 'About this program' button, where a new window is opened in which details about the tools used to compile the program are listed, along with the sources responsible for providing original source for the candidate to modify for use in this assignment. In this window, the aforementioned sources are represented by small logos, much like how most 'About' windows in professional software often have their own custom icons and images integrated into them to complete its overall 'look and feel'.

Lastly, the 'Easter Eggs' button in the program does nothing more than to open an additional window informing the user that there are no easter eggs present in the program. However, it also allows the candidate to experiment with the various types of dialogue windows that are present in Swing's default offerings; specifically, the dialogue type used in the Easter Eggs button is that of the Error dialogue, while the rest of the program makes use of Confirmation and Alert dialogues.

How the candidate has managed to write the program that is capable of performing the aforementioned functions can be described below:

- The main program window that houses the four buttons for the user to select is done with JFrame and JPanel. This is similar, if not identical to how the original tutorial program published by RoseIndia implements the buttons within the windows.
- For the Program Explanation button, the act of moving back and forth between various windows displaying sequentially-linked messages is done via a String array to contain the relevant messages, and a while() loop to allow the user to navigate between the various messages.
  - In order to ensure that the number of windows generated for the Program Explanation function does not exceed the number of messages present in the String array, the while() loop has to keep track of the messages that are being called by the user upon each successive click. Upon reaching a certain value, clicking 'Next' or 'Back' will simply close the window due to the array length being reached; this is to prevent any exceptions relating to ArrayOutOfBounds from occurring.
- The act of embedding images in the window that is generated upon selection of the 'About this program' button is accomplished by making use of ImageIcon to generate a custom icon for the window; this custom icon is subsequently integrated into the window by adding it into a JPanel object. In addition, primitive text formatting in the window is accomplished by embedding HTML tags into the code, with an example being the use of the
 tag to force line breaks in the window.
- In order to ensure that each button in the confirmation window that is generated upon selection of the 'Quit' option in the main program, the candidate has bound a separate JOptionPane event to each button. Of particular note is the button that the user is supposed to select ('Get me out of here!') if he or she intends to terminate the application, in which a condition System.exit() is called if the user selects the aforementioned option. This is the line that is responsible for killing the program; without it, the only other way to shut the program down is to click on the 'X' button (or its equivalent) on the main program window itself.

<u>Changes made / differences between original program and candidate's program</u>

- Inclusion of Quit button for shutting down the program via a series of graphical prompts instead of using the 'X' window button to do so.
- Inclusion of a sequential link between various Alert windows accessible via a 'Back' or 'Next' button in Program Explanation
- Inclusion of images within windows as opposed to simple text-only windows in original programmes
- Addition of GridLayout() to lock all buttons in main window in their fixed position and prevent relocation when window is resized
- Added an opening dialogue message that serves as a splash screen of sorts to the program upon execution.
- Inclusion of custom button text in candidate's program
- Use of showOptionDialog() in candidate's program to generate dialogue windows instead of showMessageDialog()/showConfirmDialog() etc etc.

Name: Lum Chee Xiang Michael (Lan Zhixiang) Module: Introduction to Java and Object-oriented Programming
Student ID: 111102750 Module number: CO1109-01
Assignment 4

Problems faced

The first problem faced by the candidate in attempting to re-purpose the original program was caused by the fact that the candidate has had no prior experience in working with any form of GUI programming. This meant that the original source code that the candidate had acquired for use in this assignment initially made no sense at all to the candidate at all, and that the candidate was essentially 'flying blind' in his attempt to create a working program out of the acquired source code.

This problem was solved (somewhat) by closely analysing the code and the corresponding writeup provided by the website which published the source code for the program, as well as performing basic experiments with simple Swing dialogue boxes before attempting to move on to the more complicated implantation demanded by the source code.

Another problem faced by the candidate in the early stages of programming this application was to find a way to integrate a loop into the GUI so as to ensure that the user will be able to cycle between the messages that are being displayed in the alert dialogue boxes as seamlessly as one would do so on a typical installer prompt. This led to an unsuccessful first attempt as the candidate foolishly thought that a for() loop would be capable of serving that need; this resulted in a variety of errors and bugs showing up when the program was compiled and run; such as having the button return either nothing, or an ArrayOutOfBoundsException.

Even after replacing the for() loop with a while() loop, the problems did not stop: while the program was able to compile and run successfully, the looping functionality was still badly broken; in spite of the candidate's best efforts to bind some form of primitive loop control into the GUI's buttons with JoptionPane.NO_OPTION and JoptionPane.YES_OPTION, the program would still end up throwing out ArrayOutOfBounds exceptions after registering a certain number of clicks.

In the end, this problem was eventually solved after seeking help from an acquaintance who had prior knowledge in Java programming and was thus able to point the candidate in the right direction in the debugging process; it turned out that the candidate was actually attempting to pass a variable (int selection) that has already been used for the 'Quit' button in the code block that handled the program's looping functionality. Indeed, after the candidate created a new variable (int selection1), the errors suddenly vanished and the looping feature started to work properly.

However, not all the changes that the candidate wanted to implement on the program were capable of working properly; at least one such feature had to be scrapped at the last minute due to time constraints and the lack of understanding in proper GUI programming. This dropped feature was supposed to grant users the ability to exit the loop cleanly by simply clicking on the 'X' button at the top right corner (left corner for OS X) of the Swing GUI window, without affecting overall program functionality. As it stands, the candidate's flawed implementation of the Swing GUI means that a user who terminates the current program's looping content prematurely via the 'X' button on the window will be greeted with a frozen program that will not respond to any form of user interaction, and the only way to exit the program will be to forcibly terminate the process by destroying the window, or killing the Java Virtual Machine process in the background. However, it is the candidate's hope that, with further training in GUI programming, he will soon be able to integrate such standard features into his future assignments properly so as to gain a better understanding and a higher proficiency in the basics of object-oriented programming.

Name: Lum Chee Xiang Michael (Lan Zhixiang)
Student ID: 111102750
Module: Introduction to Java and Object-oriented Programming
Module number: CO1109-01
Assignment 4

References (APA style)

"Show message and confirm dialog box" (2007) *Rose India Technologies Pvt. Ltd.* Retrieved Mar 23, 2012, from http://www.roseindia.net/java/example/java/swing/ShowMessageDialog.shtml