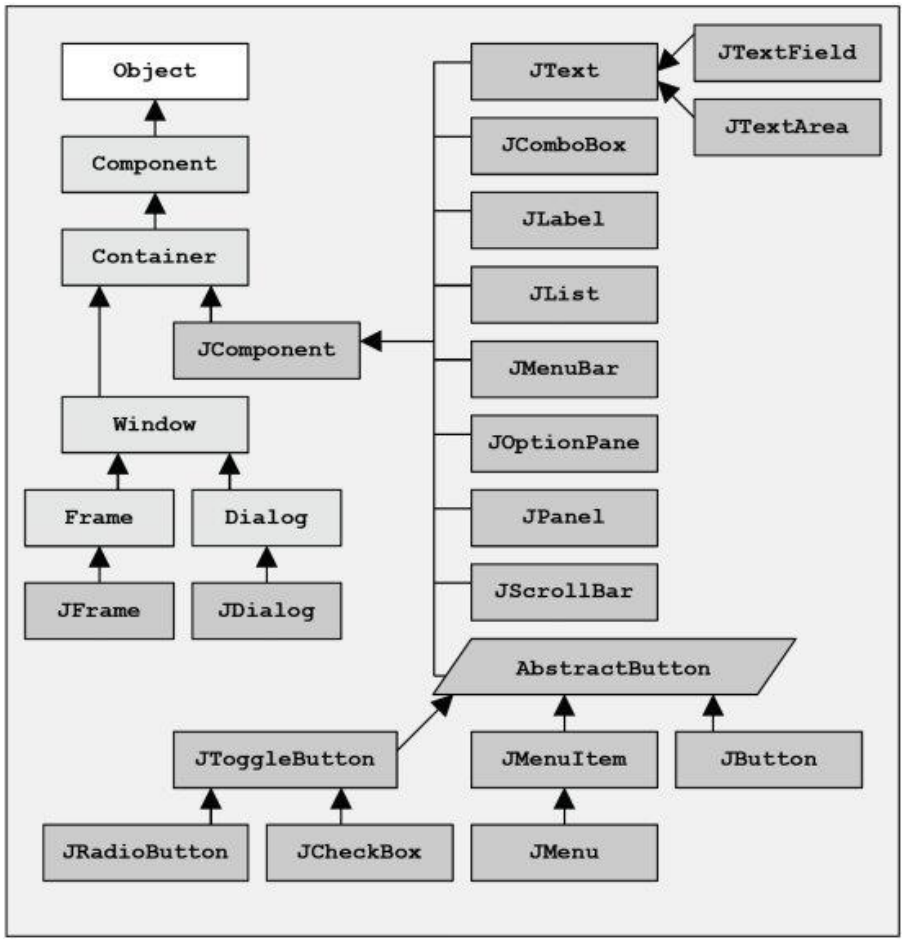
EECS 1720
LAB 5 :: Working with UI Elements and Basic ActionEvents

Lab Resources:

Java API: https://docs.oracle.com/javase/8/docs/api/
Java AWT/Swing/LayoutManagers: https://www.javatpoint.com/java-awt
Java Swing: https://docs.oracle.com/javase/tutorial/uiswing/components/index.html

Lab Files: http://www.eecs.yorku.ca/course_archive/2022-23/W/1720/labs/lab5/lab5.zip

** Note, Swing is a lightweight version of AWT, and generally easier to work with. The Swing package of classes has the following hierarchy (which echo's AWT):



Here, Swing uses JComponents instead of Components. The above diagram shows Swing classes in relation to AWT components (Swing classes begin with "J"). The above links

provide access to tutorials and example usage of each of these subclasses (to be used in this lab). Swing provides a platform independent version of AWT.

*** There are many sample code snippets in the above links for creating and incorporating each of these GUI elements into your code for each exercise. You should also review Lectures 12-15 (and code samples from weeks 7-8) on eclass.*

STEP 1: Importing an Archived Project

In this step, you will import an archived project into your Eclipse workspace, (the files for each lab exercise are embedded). Each question refers to a separate java file. The instructions for what is required for each question is included in the document below (STEP 2 onward).

a. You can download the lab project file from the link above (see Resources)
   This file is a *zip file*, also known as an *archive file*. Click on the link below to open the URL in your browser. In the dialog that opens, choose **Save**, not **Open**. This file is typically saved into your *home* or *downloads* folder.

b. Open Eclipse and import this archive
   IMPORTANT: this process is the same as the process you will use in lab tests, so please make sure you follow it, so that you do not have submission difficulties during the lab tests.

   DO NOT DOUBLE CLICK ON THE FILE TO OPEN. By importing this project, you will add it into your *EECS1720* workspace. Do this by:

   i. Open Eclipse (**Applications → Programming → Eclipse**) & set/choose your workspace
   ii. Close "Welcome" tab, and navigate to **File → Import → General → Existing Projects into Workspace.** Hit "next"
   iii. Choose the archive file (zip file you downloaded in (a)). After selecting, hit **Finish**, and the file will open up as a project in your Project Explorer.
   iv. We are now ready to proceed with the Lab Exercises.

STEP 2: Lab Exercises

The exercises in this lab are pitched at creating some basic layouts for a GUI applications

Exercise 01
(BoxLayout/FlowLayout, ImageIcon, JLabel, JText and Font):

Goal: The goal for this exercise is to create a Java AWT/Swing application that constructs the following Graphical User Interface (GUI) shown in Figure 1 (or as similar as possible to this). The relevant controls are alluded to above, though to get the correct colouring of text and resizing of elements, you will need to look into the set methods relating to each of the nodes you create.

Here is a visual to help guide you in the construction and layout of elements.

Note: the window will have a look and feel depending on which operating system you are running Eclipse from (e.g. the look and feel will be different on linux (remotelab) vs windows vs mac). The content inside the window (i.e. the scene), is what you need to ensure is created to match as closely as possible to these figures.

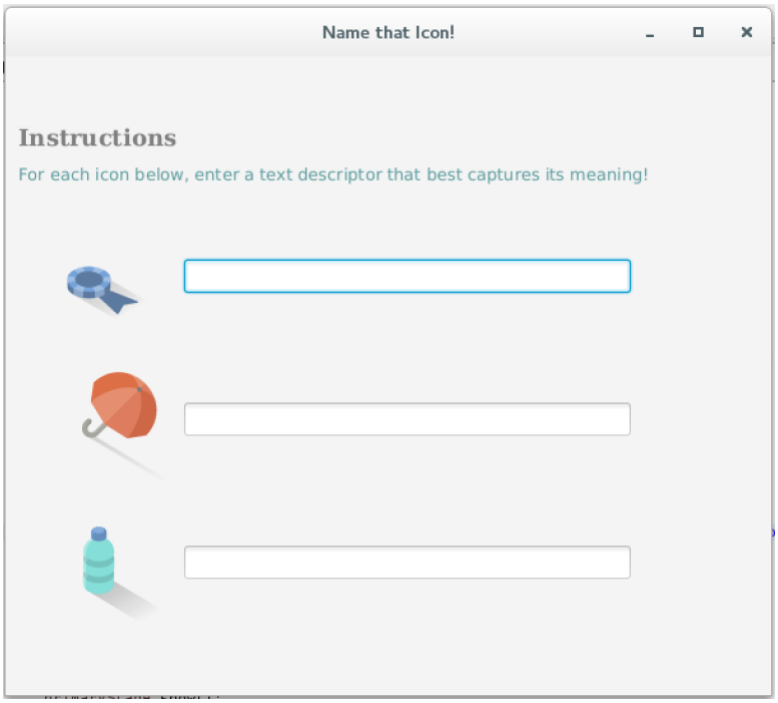


Figure 1.

The images used for the three icons are located in the project folder (in the lab05.icons package). These images are physically stored in the `src/icons/` subdirectory (within the project directory).

As a further guideline for Exercise 01:

- The "Instructions" title uses a gray colour (you can choose)
- The text explanation uses some form of blue colour (you can choose)
- The text fields and icons are indented with respect to the text instructional elements
- See how close you can get to the above (it is ok if you cannot get exact)

Exercise 02 (GridLayout, ImageIcon, and JButtons):

Goal: The goal for this exercise is to create a GUI AWT/Swing application that constructs the following Graphical User Interface (GUI) shown in Figure 2. The relevant controls are alluded to above, though to get the correct colouring of text and resizing of elements, you will need to look into the set methods relating to each of the nodes you create.

Here is a visual to help guide you in the construction and layout of elements:

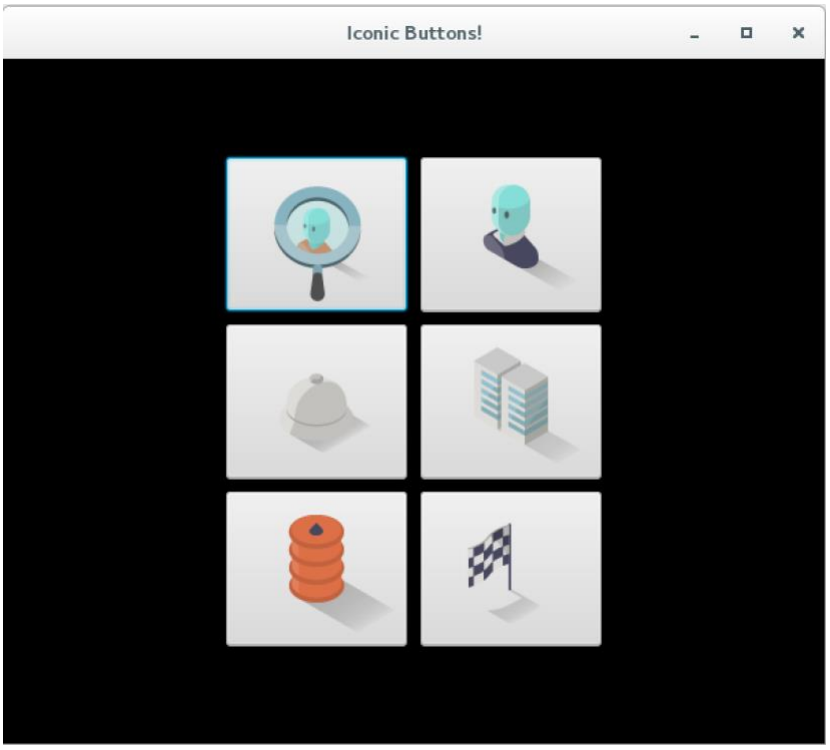


Figure 2.

Note that the images used for these icons are also located in the project folder (in the icons package).

Exercise 03 (JRadioButton, ImageIcon and JToggleButton):

Goal: The goal for this exercise is to create a GUI AWT/Swing application that constructs the following Graphical User Interface (GUI) shown in Figure 3. In this application, two sets of JToggleButtons (radio buttons) are generated. The idea is that there are two sets of chart types to select from. It is expected that one item from each group will always be selected. When you create your application, ensure that the items selected in Figure 3 are the items selected by default when your application runs. Interacting with the application, a user can then switch which radio buttons are selected of course.

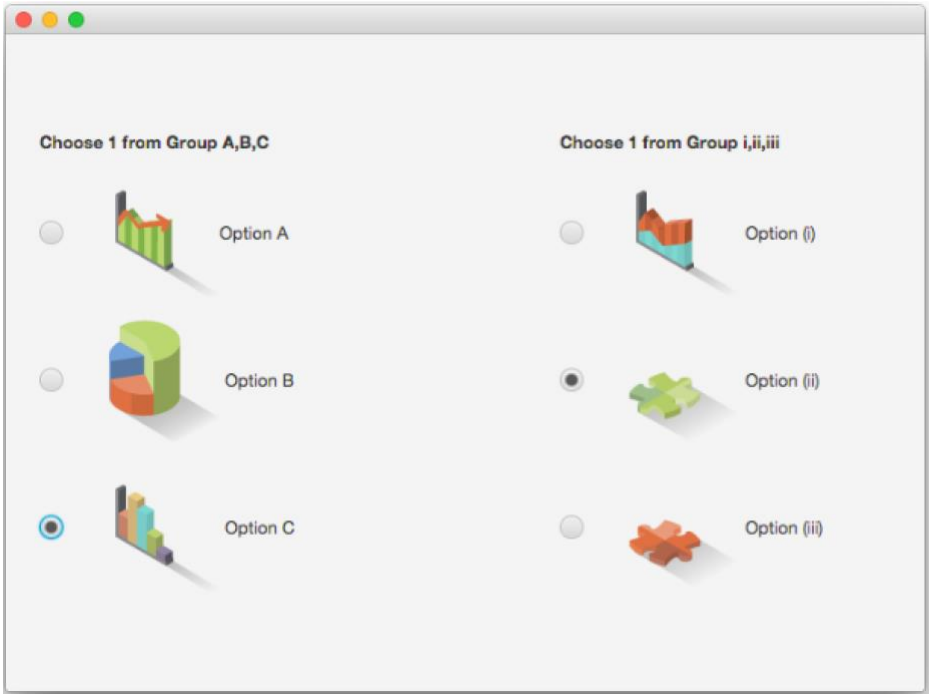


Figure 3.

Please see the lecture notes for more information and the sample code from the lectures for more reference examples for creating radio button groups.

Exercise 04
(Handling a basic ActionEvent on a JButton; Modifying a JLabel)

Goal: The goal for this exercise is to create a simple Java AWT/Swing application that constructs the Graphical User Interface (GUI) shown in Figure 1 (or as similar as possible to this). The relevant controls are JButton and JLabel. In your application, your main class should inherit from JFrame, and should implement the ActionListener interface.

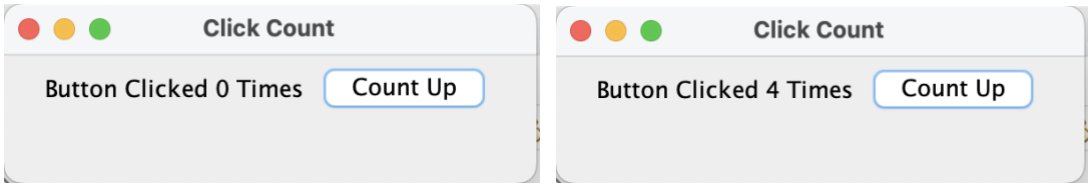


Figure 1. (look and feel slightly different on linux/windows)

Construct a handler (i.e. let your main class implement the actionPerformed method (see lectures), such that the following occurs:

- Your class constructor should initialize some class variables to track/allow access to the GUI objects that need to be accessed or changed
- You may use a simple FlowLayout for the purpose of this lab
- You may resize (or initialize the size of the JFrame so that the GUI controls are all visible
- When the button is clicked:
    - A class variable representing the number of clicks that have occurred should be incremented (use an int initialized to 0)
    - The JLabel object to the left of the button should update to reflect the number of clicks that have happened so far (after two clicks the image to the right in figure 1 shows the updated state of the GUI)
    - A message should be printed to the console. For example after 5 clicks you should see the following messages in the console:

```
Counting Up!  [count = 1]
Counting Up!  [count = 2]
Counting Up!  [count = 3]
Counting Up!  [count = 4]
Counting Up!  [count = 5]
```

Exercise 05
(Handling ActionEvents on JButton's and JRadioButtons; Modifying a JLabel)

Goal: The goal for this exercise is to build upon Exercise01.java, in order to create a Counter that displays a value that can count either up (increment) or count down (decrement), depending on the value of a radio button selected within the GUI.

Two JLabels should be used in order to display the current value of a counter (an integer that can increment and decrement), as well as a label to display how many total button clicks have occurred. The counter may be positive/negative, while the number of clicks will be positive only).
The current value of the counter can also be set directly via interaction with a third text field and a button called "Reset Counter", which when clicked will set the counter directly to the value currently typed into the JTextField. The text field will be used to set a value to use in resetting the counter, but the counter should only be reset when the user clicks the Reset Counter button.

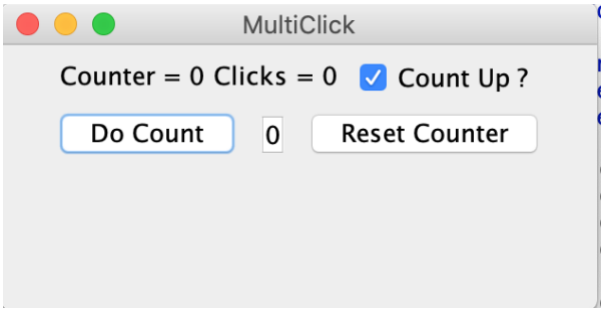


Figure 2.

For this exercise, there is no specific messages that need to be sent to the console (although you may consider printing messages so that you know which buttons and controls are being interacted with.

Note: There are two GUI controls (the buttons) that need to be registered with an event handler. If you take the approach in Exercise01, then you can only write 1 event handler (and so within it will need to figure out which button was clicked. You can use **getSource()** or **getActionCommand()** to determine this (you invoke these via the ActionEvent parameter in the **actionPerformed(ActionEvent e)** method).

If you prefer to write two separate handlers, then you will need to solve this question by creating two inner classes (see lecture slides). In this case the inner class should implement the ActionListener interface, and override the **actionPerformed(ActionEvent e)** method.

STEP 3: Submission (Deadline: Tuesday 21st March, 11:59pm, 2023)

You will formally submit the java files associated with all Exercises 01-05. SUBMIT ALL OF THE *.java FILES (**NO ZIP FILES OR CLASS FILES!!**).

use the web-submit function to submit
Web-submit can be found at the link: https://webapp.eecs.yorku.ca/submit/