CSC 160 WINTER 16 LAB 4-2 EVENTS AND EVENT LISTENERS

PROFESSOR GODFREY C. MUGANDA
NORTH CENTRAL COLLEGE

1. User Interface Components

A *user interface component* is something that you put on the GUI of a program that allows the user to interact with the program. Examples of user interface components are buttons, text fields, labels, scrollbars, checkboxes, radiobuttons, menus, menu items, and the like.

User interface components allow the user to enter data into the program (like text fields), display information to the user (like labels), and allow the user to control the program by making it do things.

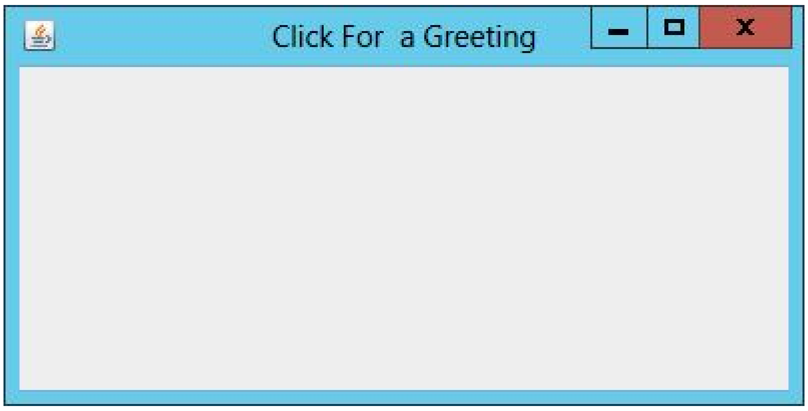
Start a project, and enter code like the following (use your own name) to create and show a frame with no user interface components.

```
package mugandacsc150lab4_2;

import javax.swing.JFrame;

public class MugandaCSC150Lab4_2
{
    public static void main(String[] args)
    {
        JFrame frame = new JFrame("Click For  a Greeting");
        frame.setSize(400, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

When you run the program, you should see something like this:

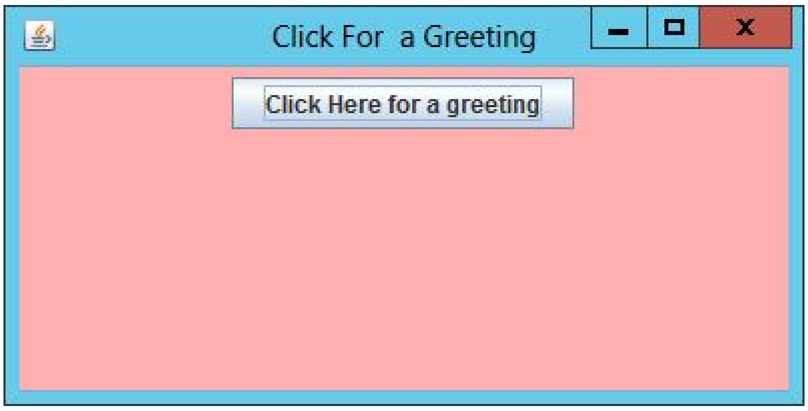


1.1. **Lab Task 1.** Add code to change the color of the frame to pink, or yellow, or whatever other color you like, and also set a `FlowLayout` manager in the frame. Refer to previous labs if you do not remember how to do this.

1.2. **Lab Task 2.** Define a button in by using the statement

```
JButton greetingButton = new JButton("Click Here a Greeting");
```

and add this button to the frame. When you run the program, it should look like this:



2. Events

An *event* is something that occurs at a component that the program needs to respond to. There are different classes in Java that describe events.

The `ActionEvent` class describes events that occur at a button when the user clicks the button. An `ActionEvent` also occurs at a text field when a user enters data into the text field and presses the ENTER key.

An `ActionEvent` object has two important methods:

(1) `Object getSource()`
This method returns a reference to the object at which the action event occurred. This may be a button or a text field.
(2) `String getActionCommand()`
This method returns a string that describes the object that is the source of the event. If the event source is a button, the action command string will

(by default) be the text on the button. If the event source is a text field, the action command string will be the text in the text field.

3. Interfaces

An *interface* is a collection of one or more placeholders for methods that will later be implemented by classes. A method place holder is called an *abstract method.* An abstract method is like a method definition, but without the body of the method.

There is an interface in Java, called `ActionListener`, that holds a placeholder for a method to be called to respond to an `ActionEvent`. This interface is defined in the Java libraries, but it looks like this

```
interface ActionListener
{
  abstract void actionPerformed(ActionEvent evt);
}
```

When we click on *greetingButton*, we want the button to call the method

```
  void actionPerformed(ActionEvent)
  {
      JOptionPane.showMessageDialog(null, "Hello, World of GUI!");
  }
```

The way to do this, is to write a class that implements the `ActionListener` method interface and overrides the place holder (the abstract method) with the method shown above

```
class GreetingListener implements ActionListener
{
    @Override
    public void actionPerformed(ActionEvent e)
    {
       JOptionPane.showMessageDialog(null, "Click Here for a greeting");
    }
}
```

Add this class to your file, making sure that you do not next it inside the class you already have. It is safe to put it after or before the other class, as long as it comes after all the import statements.

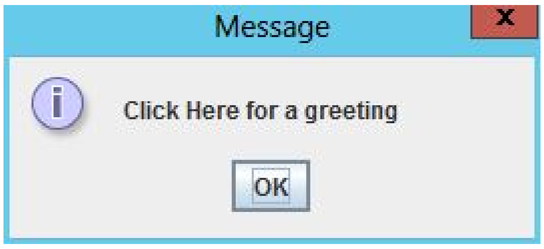
Now, you need to create a object of the class `GreetingListener`, by putting the following line of code inside the main method. The safest place to put it is after all the other code in `main`.

```
 GreetingListener listener = new GreetingListener();
```

Finally, you need to add this listener to the greeting button:

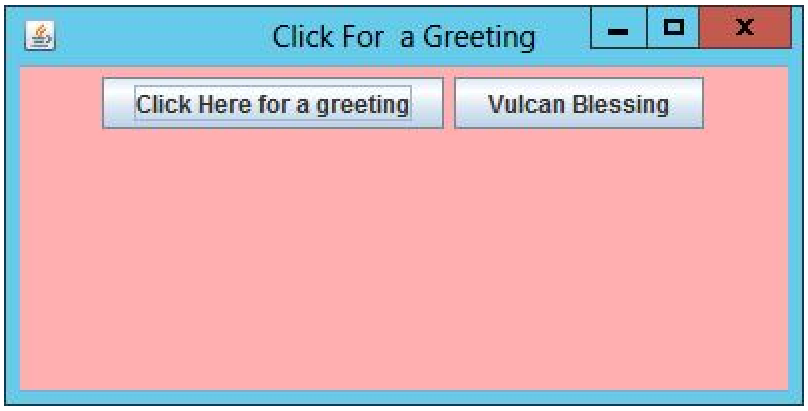
```
greetingButton.addActionListener(listener);
```

Now run the program and click on the button. You should see a message dialog like this:

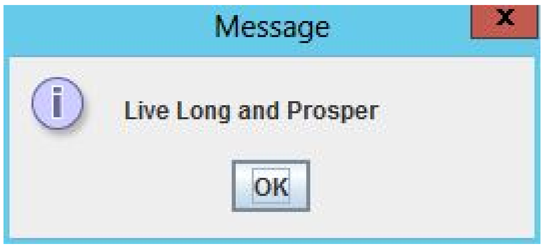


Click Here for a greeting

3.1. **Lab Task 3.** Modify the program so that the frame displays two buttons: the original button, and another button with the text `Vulcan Blessing`:



When the second button is clicked, it should display a message dialog with the standard Vulcan blessing as shown below:



Live Long and Prosper

To do this, determine whether you need another listener class or if you can modify the one you already have. Make sure you use `addActionListener` to add a listener to the second button.

Due Date is Friday of Week 4.