Lab 10 Name:_____ Checked:_____

Objectives:

Learn about listeners and events, to create interactive graphical user interfaces.

Files:

http://www.csc.villanova.edu/~map/1051/Chap04/SmilingFace.java
http://www.csc.villanova.edu/~map/1051/Chap04/SmilingFacePanel.java

http://www.csc.villanova.edu/~map/1051/Chap04/PushCounter.java
http://www.csc.villanova.edu/~map/1051/Chap04/PushCounterPanel.java

A. Adding a button to **`SmilingFacePanel.java`**:

Before you begin, download and compile SmilingFace.java and SmilingFacePanel.java; run SmilingFace.java and observe the panel it creates.

1. Import the class **`JButton`** from the package **`javax.swing`**, i.e., add the import statement:

```
import javax.swing.JButton;
```

2. Add an instance variable for the button – let's call it **`clicker`**

```
private JButton clicker;
```

3. In the SmilingFacePanel constructor, instantiate a `JButton` object and assign it to `clicker`; then add the button to the panel.

```
clicker = new JButton("click here");

add (clicker);
```

`clicker` represents a button that may be associated with an action. For the moment it does nothing when clicked, but compile your program and run it anyway. You should see the same picture, but with a button on it. When you click on the button, nothing happens. Next, we will fix this - make something happen...

B. Experiment with **`PushCounterPanel`**:

Download and compile `PushCounter.java` and `PushCounterPanel.java`; run `PushCounter` and observe the behavior of the button.

The `PushCounterPanel.java` uses a JButton object. Just like our program, above, it is instantiated and added to the panel in the constructor.

```
//********************************************************************
//  PushCounterPanel.java       Authors: Lewis/Loftus
//  Demonstrates a graphical user interface and an event listener.
//********************************************************************
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class PushCounterPanel extends JPanel
{
   private int count;
   private JButton push;
   private JLabel label;

   //-----------------------------------------------------------------
   //  Constructor: Sets up the GUI.
   //-----------------------------------------------------------------
   public PushCounterPanel ()
   {
      count = 0;

      push = new JButton ("Push Me!");
      push.addActionListener (new ButtonListener());

      label = new JLabel ("Pushes: " + count);

      add (push);
      add (label);

      setPreferredSize (new Dimension(300, 40));
      setBackground (Color.cyan);
   }

   //*****************************************************************
   //  Represents a listener for button push (action) events.
   //*****************************************************************
   private class ButtonListener implements ActionListener
   {
      //--------------------------------------------------------------
      //  Updates the counter and label when the button is pushed.
      //--------------------------------------------------------------
      public void actionPerformed (ActionEvent event)
      {
         count++;
         label.setText("Pushes: " + count);
      }
   }
}
```

Associating an action with a button

The action(s) associated with clicking on a button are handled through something called an `ActionListener`. Observe that the `PushCounterPanel` class contains, inside it, the definition of another class: the `ButtonListener` class (the class that handles the event that a button gets pushed in the `PushCounterPanel` class). Because it is defined inside the`PushCounterPanel.java` class, `ButtonListener` class is called an *inner class*.

The `ButtonListener` class only has one method, `actionPerformed()` that specifies what has to happen (in this case a counter is incremented and the text in a label is replaced with the new value of the counter).

Before proceeding, answer/do the following:

1. What is the name of the variable representing the button in PushCounterPanel class?_____

2. What is the name of the variable representing the button in the SmilingFacePanel class (as modified in part A above)?_____

3. Circle the line of code that associates the action with the button in PushCounterPanel class. Copy it out here:

   _____

4. Modify `actionPerformed`() (method in the inner class) so that when the button is clicked, the background color changes to a different color.

C. Implement an action for the button in `SmilingFacePanel`

1. Import the package **`java.awt.event.*`**

2. We will create a listener for `clicker` named `ClickerListener` as an inner class in `SmilingFacePanel`. The action will be to cause the background to change to **black** when clicker is pressed. It should not do anything else. We will do this as follows:

   - Add instance variables for red, green, and blue to be used in the background color. These are integers:

```
private int red, green, blue;
```

- Add the following code to your `SmilingFacePanel` class.

```
private class ClickerListener implements ActionListener
{
   public void actionPerformed (ActionEvent event)

   {
      red = 0; green = 0; blue = 0;
      setBackground (new Color(red, green, blue));
   }

}
```

- The clicker button should be controlled by the ClickerListener – we specify this in the constructor, as follows:

```
clicker.addActionListener(new ClickerListener());
```

3. Now we will add another button that, rather than causing the background to turn black with a single click, it will increase the amount of red in the background, gradually.

a. Follow the steps you used to implement the `clicker` button to add another `JButton` called `redClicker` with the text "More red".
b. In the constructor, assign the values: `red=0; green=0; blue=0;`
c. Still in the constructor, create a new color for the background, using `red, green, blue:`

```
setBackground (new Color(red, green, blue));

// this is the color black when red=green=blue=0
```

d. Add a listener for `redClicker` named `RedListener` (copy/paste the code for ClickerListener and adapt it). In the `actionPerformed()` method of this listener:
   - `red +=20;`
   - `setBackground (new Color(red, green, blue));`

   What happens when you keep clicking the "More red" button repeatedly?

   _____

   (Fix the `actionPerformed()` method to prevent this from happening.)

4. Once you get the red button to work, repeat the above steps to create buttons to increment the amount of blue and green.

D. Implement a button to roll a die in your `HappinessFacePanel`

Using the class HappinessPanel from the previous lab, add a button that causes one of the dice to be rolled.