Lab 5: Swingin' with Callbacks

Due: 9/26/2012 – midnight

As explained in class, interfaces are fundamental to systems that use *callbacks*. Remember, these are objects that we set up to 'call back' later, when the time is right.

In java's GUI package `javax.swing`, we build interfaces out of windows and buttons. To couple our custom code with these standard GUI elements, we must implement the interfaces expected by them. When set up correctly, our code is the thing that is 'called back' when the button is pressed or other event occurs.

For this lab, we will be implementing the interfaces related to buttons and mouse events.

1 Your job

I've provided java code in a pdf, `SwingLab.pdf`, associated with Lab 5. When run, this code sets up a panel with 3 buttons labeled *Red*, *Yellow*, and *Blue*. This program will compile and run without any modifications, but the buttons do nothing.

Your job is to make these buttons change the panel color to the color named on the button.

You're also going to report via `System.out` when the mouse enters and leaves a button's area:

```
The mouse has entered the yellow button area.
The mouse has exited the yellow button area.
```

2 How to do it

You will need to provide implementations of the `java.awt` interfaces `ActionListener` and `MouseListener`. Here's an example:

```
class RedButtonListener implements ActionListener
{
    public void actionPerformed(ActionEvent event)
    {
        panel.setBackground(Color.RED);
    }
}
```

```
ActionListener redListener = new RedButtonListener();
bRed.addActionListener(redListener);
```

Notice how this class is referring to `panel`. This assumes you are including this class as an *inner class* directly in the `main` method. **Why?**

Notice also how this class is referring to `Color.RED`. You can refer to all the colors needed for this assignment in a similar way.

Here are the interfaces you will need to implement. Note: the `import` statements at the top of the provided code make these interface definitions available to you. Also note: a single class can implement multiple interfaces.

```
public interface ActionListener
{
    public void actionPerformed(ActionEvent event);
}

public interface MouseListener
{
    public void mouseEntered(MouseEvent event);
    public void mouseExited(MouseEvent event);
    public void mousePressed(MouseEvent event);
    public void mouseReleased(MouseEvent event);
    public void mouseClicked(MouseEvent event);
}
```

To implement `MouseListener`, you have to implement all five of those methods. To ignore a mouse event, provide an "empty" implementation of the corresponding interface method. **All of your implemented methods will probably just ignore the argument that is passed in**.

3 Bonus

There are two possibilities of bonus points for this lab. Some of you would really benefit from some extra points.

1. Implement the requirements using only one inner class. (You can use more inner classes if needed to "do something cool.")
2. Play around, come up with something cool and tell me about it.

4 Turn In

Put all java files in a directory named `YourName_1110_Lab5`, zip up that directory and submit the zip file via blackboard by the due date.