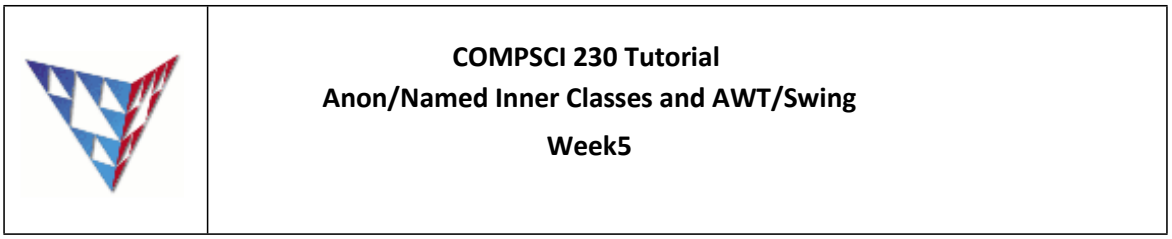


COMPSCI 230 Tutorial

Anon/Named Inner Classes and AWT/Swing

Week5

Q1: Add another inner class to class Test01. The new class should implement Printable and should have a method to print out the String "Triangle".

```
public class Test01{
public interface Printable{
      public void print();
     }
     public class PrintRectangle implements Printable{
      public void print(){
      System.out.println("Rectangle");
    }
}
```

```
//Add your class here
public class PrintTriangle implements Printable{
      public void print(){

      System.out.println("Triangle");

      }
  }
```

Q2: Read the following code and answer the questions below.

```
public class AnonymousInnerClassExercise{
        public AnonymousInnerClassExercise() {
            Frame frame = new Frame();
            frame.addWindowListener(new WindowAdapter() {
                public void windowClosing(WindowEvent e) {
                System.exit(0);
                }
            });
        }
  }
```

A) Identify the superclass of the anonymous inner class given in the code above.

```
WindowAdapter
```

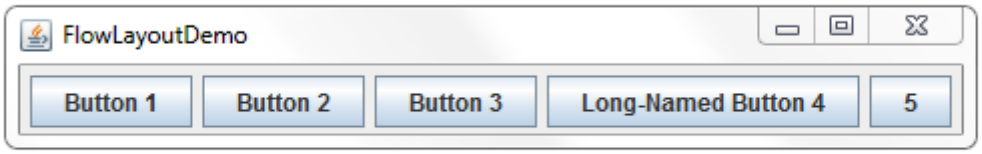
B) Rewrite the code using a named inner class.

```
public class AnonymousInnerClassExercise1 {
      private class WindowCloser extends WindowAdapter{
            public void windowClosing(WindowEvent e) {
                  System.exit(0);
            }
      }
      public AnonymousInnerClassExercise1() {
            Frame frame = new Frame();
            frame.addWindowListener( new WindowCloser() );
      }
}
```

C) Rewrite the code using another top-level class

```
class WindowCloser extends WindowAdapter{
      public void windowClosing(WindowEvent e) {
            System.exit(0);
      }
}
public class AnonymousInnerClassExercise2 {
      public AnonymousInnerClassExercise2() {
            Frame frame = new Frame();
            frame.addWindowListener( new WindowCloser() );
      }
}
```

Q3: To generate the FlowLayoutDemo, you have to use JButton, JFrame in your code, what have been missed at import ?



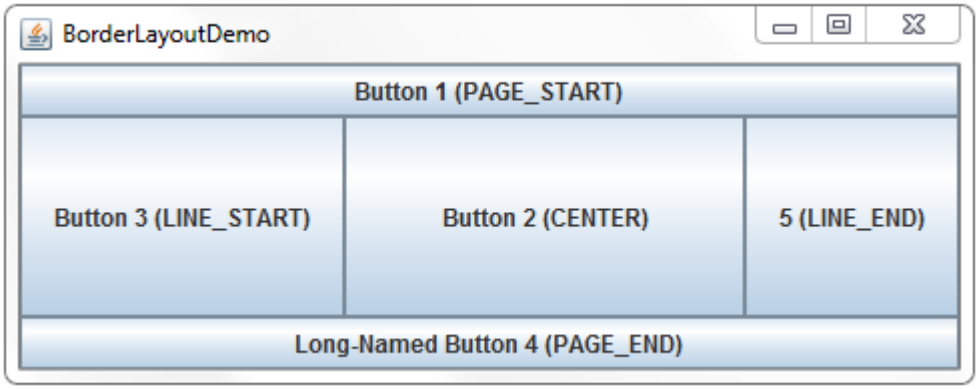

```
import java.awt.Container;
import java.awt.FlowLayout;

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

Q4: we have to add the five buttons into our layout, please complete the following method

```
public static void addComponentsToPane(Container pane) {
            pane.setLayout(new FlowLayout());
            //complete the code in here
            pane.add(new JButton("Button 1"));
            pane.add(new JButton("Button 2"));
            pane.add(new JButton("Button 3"));
            pane.add(new JButton("Long-Named Button 4"));
            pane.add(new JButton("5"));
      }
```

Q5: Look at the BorderLayoutDemo.java , please complete addComponentsToPane method, and let the button be allocated at different position.




```
public static void addComponentsToPane(Container pane) {
      JButton button = new JButton("Button 1 (PAGE_START)");
      pane.add(button, BorderLayout.PAGE_START);
      pane.add(button, BorderLayout.PAGE_START);
      button = new JButton("Button 2 (CENTER)");
      button.setPreferredSize(new Dimension(200, 100));
      pane.add(button, BorderLayout.CENTER);
      button = new JButton("Button 3 (LINE_START)");
      pane.add(button, BorderLayout.LINE_START);
      button = new JButton("Long-Named Button 4 (PAGE_END)");
      pane.add(button, BorderLayout.PAGE_END);
      button = new JButton("5 (LINE_END)");
      pane.add(button, BorderLayout.LINE_END);
  }
```

Open the GradientsOutlines.java source file. You should see the following window:

Currently, if you click inside the large grey area, a red circle with a black outline will appear. The buttons at the top do nothing at the present stage.

Q6: Add an ActionListener to each of the buttons: strokeButton, c1Button, and c2Button.

```
<button_var_here>.addActionListener(new ActionListener(){
@Override
public void actionPerformed( ActionEvent arg0){

}

});

This must be done for all 3 buttons.
```

Q7: For all 3 buttons we must implement JColorChooser in order to set the 3 Color variables. How do we show a JColorChooser dialog window when a button is pressed?

```
JColorChooser.showDialog(drawPanel,"Pick Colour 1", c1);
```

Change the last colour selecter param to be the respective Color var for each button. Put the above code inside the actionPerformed() method generated for the ActionListener

Q8: What is the return type from JColorChooser?

```
The JColorChooser returns a Color
```

Q9: What is returned by JColorChooser if we press the 'Cancel' button inside the popup window? Can we fix it?

```
It returns null.

Color tempColor = JColorChooser.showDialog(drawPanel,
                                    "Pick Colour 1", c1);
                          if (tempColor != null) {
                                c1 = tempColor;
                          }
```

Q10: How does the current paint() draw and outline the circle? HINT: http://docs.oracle.com/javase/tutorial/2d/geometry/strokeandfill.html

```
The inside of the circle is filled first by
g2.setPaint(c1);
g2.fill(circleShape);

Then, g2 is set to the stroke colour and the circle shape
is drawn for the outline.
g2.setColor(sColor);
g2.draw(circleShape);
```

Q11: We want to be able to fill the circle so that it uses both Color variables (c1 and c2) and creates a gradient. What is the class we need to create a gradient and how can we create one for circleShape? Make sure to change the paint used by g2 to fill the circleShape!

```
GradientPaint.
GradientPaint gp = new GradientPaint(circleShape.getBounds().x,
circleShape.getBounds().y, c1,
circleShape.getBounds().x+ circleShape.getBounds().width,
circleShape.getBounds().y + circleShape.getBounds().height, c2);

This will make a diagonal gradient from the top-left to bottom
right of the circle. Feel free to change the points used to
interpolate the colours.
```

Q12: How can we set the stroke type of the outline?

```
g2.setStroke();
```

Q13: Try to create a new style of stroke inside the setUpStrokes() method. Change strokes[2] in the array to be your new style. Don't forget to change the setStroke() call inside paint() to use stroke[2]!

JDOCS: http://docs.oracle.com/javase/8/docs/api/java/awt/BasicStroke.html

```
strokes[2] = new BasicStroke(5.0f, BasicStroke.CAP_BUTT,
                             BasicStroke.JOIN_MITER, 10.0f,
pattern1, 0.0f);

Free answer. Just so long as you have a visible outline.
```