Java Swing tips for CSc155

Adding Buttons

You can create buttons and add it to north section of **JFrame** that has **BorderLayout** as follows:

```
import javax.swing.JPanel;
import javax.swing.JButton;
…
JPanel topPanel = new JPanel();
this.add(topPanel,BorderLayout.NORTH);
JButton myButton = new JButton ("Button Label");
topPanel.add(myButton);
```

In CSc-133 we were using Codename One (CN1) **UI** framework instead of **Swing** to build our GUI. In Swing, **JFrame**, **JPanel,** and **JButton** correspond to **Form**, **Container,** and **Button** in CN1.

Creating Commands and Attaching them to Buttons

You can create a command class (e.g., **CustomCommand**) by extending from **AbstractAction. AbstractAction** corresponds to **Command** in CN1. However, the code that goes into the command class is the same in both CN1 and Java.

You can create and attach the command objects as follows:

```
CustomCommand myCommand = new CustomCommand();
myButton.setAction(myCommand);
```

setAction() corresponds to **setCommand()** in CN1.

Handling Mouse Wheel Events

Mouse wheel events are generated when wheel is rotated in a GUI component. We can handle these events by implementing **MouseWheelListener** interface:

```
public interface MouseWheelListener {
   public void mouseWheelMoved (MouseWheelEvent e);
}
```

You can make your **JFrame** a "self-listener" by making **JFrame** implement this interface and add itself as a listener for mouse wheel events generated on itself as follows:

```
this.addMouseWheelListener(this);
```

In **mouseWheelMoved()** method, you can determine the amount of wheel movement by calling **getWheelRotation()** on the **MouseWheelEvent** object which is passed as a parameter to the method. An example is shown on the next page.

Capturing Keystrokes

There are two ways of doing this. One is to use Key-Action maps. A simpler way is the following:

- Have your **Jframe** implement **KeyListener**
- This requires you to implement code for **keyPressed()**, **keyReleased()**, **keyClicked()**, and **keyTyped()**. You really only need **keyPressed()**, the others can be empty.
- **keyPressed()** takes a parameter of type **KeyEvent**. You can use that object to call **getKeyCode()** and find out what key was pressed.
- You'll need to add a couple of imports.

An example is shown on the next page:

```
import javax.swing.*;
import java.awt.BorderLayout;
import java.awt.event.*;
. . .
public class myGame extends JFrame implements GLEventListener, KeyListener, MouseWheelListener
{       . . .

   @Override
   public void mouseWheelMoved(MouseWheelEvent e) {
         // do something when the mouse wheel is moved.
         // the amount and direction of mouse wheel movement can be retrieved from e.getWheelRotation();
   }

   @Override
   public void keyPressed(KeyEvent e) {
      switch (e.getKeyCode()) {
         case KeyEvent.VK_1:
            // do something when the "1" key is pressed
            break;
         case KeyEvent.VK_2:
            // do something when the "2" key is pressed
            break;
      }
   }

   @Override
   public void keyTyped(KeyEvent e) { }   // must be present, but may be empty

   @Override
   public void keyReleased(KeyEvent e) { }   // must be present, but may be empty

}
```