CECS 277 – Lecture 16 – GUI Applications

Swing Components and the Action Listener – You can add functionality to your application by adding buttons, textboxes and other components. The ActionListener is an event handler for when these elements are clicked on or modified. Each of the components has several methods associated with them; check the Java API for details.

If you need to update a drawing panel after a modification was made to it in the actionPerformed() method then you can call the repaint() method. Due to glitches, it is often recommended that awt drawings and swing components not be mixed, there are another older set of very similar components in the awt library that can be used instead.

Button – Buttons are created and tied to the actionListener. They are initialized with text, and/or an image. When they are clicked on, the button sends the event to the actionPerformed method.

Label – A label is a non-interactive block of text. Labels are used give titles to sections of your window. Labels can use plain text, images, and html formatting.

TextField – a text field is a one line text area that allows the user to type in it. The value is returned as a string. It is initialized with a default string value, and the number of characters wide it will be displayed as. A text field may be associated with the actionListener and the event occurs when Enter is pressed.

CheckBox – Allows the user to select from multiple options. Although each of the checkboxes can be tied to the actionlistener, it is usually the case that a submit button is used to create the event that triggers the checkbox selection, then in the actionPerformed method, you can check the state of each of the checkboxes.

RadioButton – Radio buttons are similar to check boxes, except that they are combined as a group and only one can be selected at a time.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class App extends JFrame implements ActionListener {
     JPanel panel = new JPanel();
     JTextField tb = new JTextField("Color and Shape", 15);
     JLabel cLabel = new JLabel("Colors:");
     JCheckBox red = new JCheckBox("Red");
     JCheckBox blue = new JCheckBox("Blue");
     JLabel sLabel = new JLabel("Shapes:");
     ButtonGroup shapes = new ButtonGroup();
     JRadioButton circle = new JRadioButton("Circle");
     JRadioButton square = new JRadioButton("Square");
     JButton button = new JButton("Click");
     //initialize other components

     public App() {
          super("Window"); //title of the app
          setSize(225, 200); //window size
          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          this.getContentPane().add(panel);
          panel.setLayout(new FlowLayout());
          panel.add(tb);
          panel.add(cLabel);
          panel.add(red);
          panel.add(blue);
          panel.add(sLabel);
          shapes.add(circle);
          shapes.add(square);
          panel.add(circle);
          panel.add(square);
          panel.add(button);
          //add other components to panel
          button.addActionListener(this);
          tb.addActionListener(this);
          setVisible(true); //display window
     }
     public void actionPerformed(ActionEvent e){
          if(e.getSource() == button){
               if(red.isSelected()&&blue.isSelected()){
                    if(circle.isSelected()){
                         tb.setText(“Purple Circle”);
                    }else if(square.isSelected()){
                         tb.setText(“Purple Square”);
                    }
               }else if(red.isSelected()){
                    if(circle.isSelected()){
                         tb.setText(“Red Circle”);
                    }else if(square.isSelected()){
                         tb.setText(“Red Square”);
                    }
               }else if(blue.isSelected()){
                    if(circle.isSelected()){
                         tb.setText(“Blue Circle”);
                    }else if(square.isSelected()){
                         tb.setText(“Blue Square”);
                    }
               }
          }else if(e.getSource() == tb){
               tb.setText(“Yes?”);
          }
     }
}
```

Mouse Event Handlers – Allows the user to use the mouse as input to the program. Event Handlers allow your program to react whenever the user moves or clicks the mouse. All five of these methods must be created if any of them are to be used.

mouseClicked() – called when the mouse is clicked (pressed and released).
mousePressed() – called when the mouse button is pressed down but not released.
mouseReleased() – called when the mouse button has been released after being pressed.
mouseEntered() – called when the mouse pointer enters a specific area.
mouseExited() **–** called when the mouse pointer leaves a specific area.

```
import java.awt.event.*;
public class MApp extends JFrame implements MouseListener {
     public MApp(){
          addMouseListener(this);
     }
     public void mouseClicked (MouseEvent evnt) {
          int xpos = evnt.getX();
          int ypos = evnt.getY();
          ...
     }
     public void mousePressed (MouseEvent evnt) {}
     public void mouseReleased (MouseEvent evnt) {}
     public void mouseEntered (MouseEvent evnt) {}
     public void mouseExited (MouseEvent evnt) {}
}
```

MouseMotionListener – allows you to track where the mouse is and performs events when the pointer moves over certain locations. Also allows you to perform click and drag actions. Both of these methods must be created if either of them are to be used.

mouseMoved() – called whenever the mouse is moved.
mouseDragged() – called whenever the mouse is moved while the button is pressed.

```
import java.awt.event.*;
public class MApp extends JFrame implements MouseMotionListener{
     public MApp(){
          addMouseMotionListener(this);
     }
     public void mouseMoved(MouseEvent evnt){
          int xpos = evnt.getX();
          int ypos = evnt.getY();
          ...
     }
     public void mouseDragged(MouseEvent evnt){}
}
```