Swing Components

• JButton:
  Constructor takes String parameter which is text on button
  Generates ActionEvent when button clicked
  Registration of Listener – addActionListener in constructor
  Events: can call getActionCommand or getSource() on event
  Changes: can setText to be different, can disable button

• JLabel:
  Constructor takes String which is text written
  No events generated
  Changes: can setText to be different
  Can hold images instead of just text.

• JTextField:
  Constructor takes int which is preferred number of characters,
   String, String and int, or nothing
  Generates an ActionEvent when return is pressed
  Changes: can setText to be different
Swing Components


The basic JPanel is just a blank rectangle. There are at least two different ways to make use of a JPanel:

1. add other components to the panel
2. draw something in the panel.

JPanels can be used as drawing surfaces:
1) define a class that is a subclass of JPanel and
2) write a paintComponent method in that class to draw the desired content in the panel. Defining this method is overriding the method of the same name in the superclass, so the first line must call super.paintComponent();
Class Exercise

• Create a GUI that has a single square JButton in the exact center of the window. When clicked by user, the background color of this JPanel turns red.

After class, see CenterButton.java

• Go over homework 7 – converting the TestPal program to a GUI program.
Mouse Events

- Mouse events require 2 different interface implementations, depending on what you want to do

- The simplest mouse events are defined in the MouseListener interface:
  
  ```java
  public void mousePressed(MouseEvent evt);
  public void mouseReleased(MouseEvent evt);
  public void mouseClicked(MouseEvent evt);
  public void mouseEntered(MouseEvent evt);
  public void mouseExited(MouseEvent evt);
  ```

- The second type of mouse listener is the MouseMotionListener interface:
  
  ```java
  public void mouseDragged(MouseEvent evt);
  public void mouseMoved(MouseEvent evt);
  ```
Mouse Events

• The ordinary way to register a mouse listener is to write X.addMouseListener(Y) where Y is the listener and X is the component that will generate the mouse events.

• Possible modifier keys include: the Shift key, the Control key, the Alt key (called the Option key on the Mac), and the Meta key (called the Command or Apple key on Mac).
  – evt.isShiftDown(), evt.isControlDown(), evt.isAltDown(), and evt.isMetaDown() can be called to test whether the modifier keys are pressed at the same time the mouse is clicked.

• A JPanel can be registered to listen for MouseEvent:
  – addMouseListener(this); // in the constructor
4 Steps of Event Handling

Four Steps of ActionEvent Handling
1. import java.awt.event.*
2. implement ActionListener
3. define the event-handling methods
4. addActionListener to components

Four Steps of Mouse Event Handling
1. import java.awt.event.*
2. implements MouseListener
3. define the event-handling methods
4. addMouseListener to components
Rule of Drawing

Drawing rule: (often violated) all drawing shall henceforth and forever be done in a paintComponent() method.

In real-life programming of drawing applications, many people (including the author of our book) violate this rule by obtaining the Graphics content (if class extends JPanel) like so:

```java
Graphics g = getGraphics(); // Graphics context for drawing directly out of any method in the class, not just the paintComponent method.
```

If you do use the above command to access a Graphics component, you need to use `g.dispose()` in the method before it ends.
What if I don’t want to use ALL the MouseListener methods?

```java
private static class RepaintOnClick implements MouseListener {

    public void mousePressed(MouseEvent evt) {
        Component source = (Component)evt.getSource();
        source.repaint();
    }

    public void mouseClicked(MouseEvent evt) { }
    public void mouseReleased(MouseEvent evt) { }
    public void mouseEntered(MouseEvent evt) { }
    public void mouseExited(MouseEvent evt) { }
}
```

You either need to put in empty method bodies for each event or...
private static class RepaintOnClick extends MouseAdapter
    implements MouseListener
{
    public void mousePressed(MouseEvent evt) {
        Component source = (Component)evt.getSource();
        source.repaint();
    }
}

Use the MouseAdapter class, which fills the mouse methods in for you behind the scenes. Note: MouseAdapter is a class, not an interface. Also note: anonymous inner classes don’t have to be of interface type. They can also be a class type.
Using try/catch blocks instead of throwing exceptions

Any variables declared in a try block are only available inside the try block.

Variables declared outside a try block may generate errors because the compiler won’t know if they ever get a value.
Methods in the Graphics class:

- Most often used are those to draw basic shapes like rectangles, ovals, lines. Can specify the shape as filled or not filled.