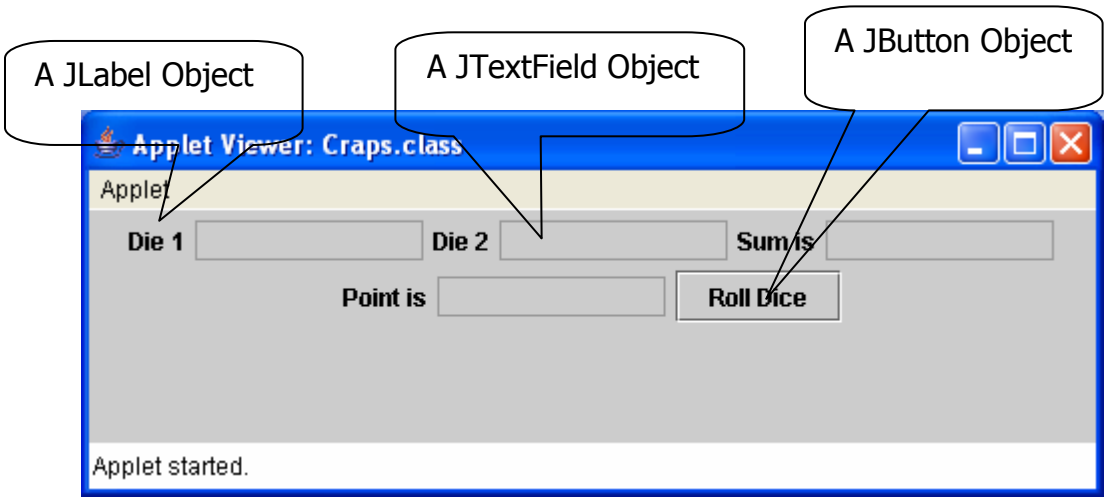
NOTES: Working with Graphical User Interfaces



A GUI in Java has at least three kinds of objects:

- components – an object that defines a screen element to display information or let the user interact with a program.
  Examples: push buttons, text fields, labels, scroll bars, and menus

- events – an object that represents an action
  Example: pressing a mouse button or typing a key on the keyboard

- listeners – objects that "wait" for an event and then respond in some way. The program must carefully establish the relationships among the listener, the event it listens for, and the component that will generate the event.

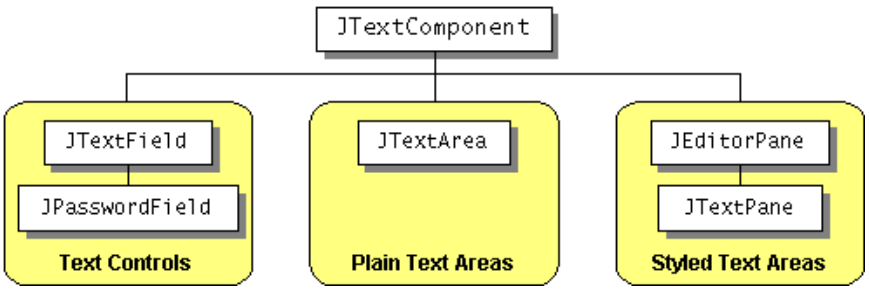
Class JFrame

java.lang.Object
  └ java.awt.Component
    └ java.awt.Container
      └ java.awt.Window
        └ java.awt.Frame
          └ **javax.swing.JFrame**

Class JTextArea

java.lang.Object
  └ java.awt.Component
    └ java.awt.Container
      └ javax.swing.JComponent
        └
javax.swing.text.JTextComponent

The following figure shows the `JTextComponent` hierarchy.



Key Terms:

Container: A GUI interface that can hold other components.
ContentPane: The part of a top-level container where components are added.
Event Handling: A mechanism to process events
Text Field: (JTextField) An area where the user can type a single line of information. Commonly used in GUI programs to accept input. A JTextField object generates an action event when the user presses the enter key.
Frame: (JFrame) A container component that is generally used for stand-alone GUI-based applications. A frame is displayed as a separate window with its own title bar.
Panel: (JPanel) Used to organize other components. It cannot be displayed on its own.
TextArea: (JTextArea) A `JTextArea` is a multi-line area that displays plain text.
Label: A component that displays a line of text. The JLabel constructor takes a String parameter.

Push Button: (JButton) A component that lets the user start an action by clicking the button with the mouse. The JButton constructor takes a String parameter that spells out the label on the button.
Icon: (ImageIcon) Used to display images. An Icon is an object of any class that implements interface Icon. Image Icons must be added to Jlabels or Jbuttons. Images must be .jpg, .gif or .png files
Icon bird = new ImageIcon ("bird.gif");
JButton button = new JButton("Bird Button",bird);

CUSTOMIZING FRAME WINDOWS

- Use the JFrame Class a customize the user interface
- The JFrame class contains the basic functions that support features available in any frame window such as minimizing the window, moving the window and resizing the window.
- Typically, an instance of a JFrame class is not created because a JFrame object is not capable of doing anything meaningful. Instead, a subclass of the JFrame class is define.
    - public class JFrameSubClass extends JFrame {

```
public JFrameSubClass {
        setTitle ("JFrame SubClass");
        setSize (300,200);
        setLocation (150,250);
        setResizable (false);
        setDefaultCloseOperation (EXIT_ON_CLOSE);
                //If this functionality is not added, the window will close but the
                //program does not terminate.
}
```

GUI components must be added to the container in which they are displayed.

```
Container cp = getContentPane();        //retrieves the content pane from the JApplet class or JFrame
JButton button = new JButton("Skip");
cp.add (button);
```

Tutorial: http://java.sun.com/docs/books/tutorial/uiswing/learn/index.html

To create a Java program that uses a GUI, we must:

- define and set up the necessary components
- create listener objects and establish the relationship between the listeners and the components that generate the events and
- define what happens as a result of user interactions
- Use e.getSource() to reuse a listener for several objects as it returns the object that caused the event to occur.

Layout Managers:

The *layout manager* for a container is an object that controls the placement of the GUI objects.

- FlowLayout – Default Layout for JApplet and JPanel. Places components sequentially from top-to bottom, left-to-right in the order they were added.
  Example:
```
FlowLayout layout = new FlowLayout();
Container cp = getContentPane();
layout.setAlignment(FlowLayout.CENTER);
Cp.setLayout (layout);
```
- BorderLayout – Default for the content panes of JFrames. Arranges components into five areas: North South, East, West and Center.
```
BorderLayout layout = new BorderLayout(10,10);
c.add(button,BorderLayout.SOUTH);
```
- GridLayout - Arranges the components into rows and columns.
```
GridLayout grid = new GridLayout(2,3,5,5);
```

HANDLING EVENTS:

- An event source generates events
- For each type of event, you must register a listener
- An action listener is associated to an action event source by calling the event source's *addActionListener* method.
- Class Heading:

```
import java.awt.event;
public class JButtonClass extends JFrame implements ActionListener
{       JButton okButton = new JButton ("OK");
        JButton cancelButton =new JButton("Cancel");
        ...
        okButton.addActionListener (this); //the frame serves as the action event listener
  public void actionPerformed (ActionEvent event)
  {  JButton clickedButton = (JButton) event.getSource();
     String buttonText = clickedButton.getText();
     setTitle ("You clicked "+buttonText);
  }
```

WORKING WITH IMAGES:

** *The ImageIcon (javax.swing) supports several image formats.*

EXAMPLE:

```
Icon pic = new ImageIcon ("myPic.jpg");
JLabel label = new JLabel ("My Picture Label", myPic,SwingConstants.LEFT);
   OR  JLabel label = new JLabel (); //Using the default constructor
       Label.setIcon (pic);
       Label.setHorizontalTextPosition (SwingConstants.CENTER);
Label.setToolTipText ("Label for my picture");
Container.add (label);
```

Panels:

Panels are created with class JPanel – a subclass of JComponent. Class JComponent inherits from class java.awt.Container, so every Jpanel is a Container. Thus JPanels may have components, including other panels, added to them.

```
private static void createAndShowGUI()
  {
     JFrame.setDefaultLookAndFeelDecorated(true); // specify the window display

     // create and set up the window
     JFrame frame = new JFrame("Phonebook"); // create the frame and set the title
     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // exit when the frame closes

     Phonebook methods = new Phonebook();
     frame.setJMenuBar(methods.createMenuBar());
     frame.setContentPane(methods.createContentPane());

     frame.setLocationRelativeTo(null); // positions the window centered

     // display window
     frame.pack(); // size the frame so that the contents fit
     frame.setResizable(false); // make it so it can be resized
     frame.setVisible(true); // show the frame
  }
```

JMENUS:

One possible sequence of steps to create and add menus is this:

1. Create a JMenuBar and attach it to a frame
2. Create a JMenu object
3. Create JMenuItem objects and add them to the JMenu
4. Attach the JMenu object to the JMenuBar

** A menu item is the event source of menu selection, so register an action listener to every menu item you add to the menu.

```
JMenu  fileMenu = new JMenu ("File");
JMenuItem item = new JMenuItem ("New");
item.addActionListener (this);
fileMenu.add(item);
fileMenu.addSeparator ();  //add a horizontal line between menu items

//Attach to a JMenuBar
JMenuBar menuBar = new JMenuBar();
setJMenuBar (menuBar);  //attach it to the frame
menuBar.add(fileMenu);



public void actionPerformed (ActionEvent event)
{  String menuName;
   menuName = event.getActionCommand();
   if (menuName.equals("Quit"))
         System.exit(0);     //used to exit the program
    else….
```

Example program #1:

```
import java.awt.*;                  //graphics
import javax.swing.*;               //applets
import java.awt.event.*;            //mouse events

public class PushCounter extends JApplet
{ private int APPLET_WIDTH = 300, APPLET_HEIGHT = 50;
   private int pushes;
   private JLabel label;
   private JButton push;

//  Sets up the GUI
public void init()
{  pushes = 0;
    push = new JButton ("Push Me!");
    push.addActionListener (new ButtonListener());
    label = new JLabel ("Pushes: " +Integer.toString(pushes));

    Container cp = getContentPane();       //retrieves the content pane from the JApplet class
    cp.setBackground (Color.cyan);
   cp.setLayout(new FlowLayout());     //components are to be placed top to bottom and left to right per row
   cp.add(push);
   cp.add(label);

    setSize (APPLET_WIDTH, APPLET_HEIGHT);
}

//Represents a listener for button push (action) events.
private class ButtonListener implements ActionListener
  {
    //Updates the counter when the button is pushed
    public void actionPerformed (ActionEvent event)
          {  pushes ++;
             label.setText ("Pushes: "+Integer.toString(pushes));
             repaint();
           }
  }//end of ButtonListener
}//end of PushCounter
```

Example: Adding scroll bars

```
textArea = new JTextArea ();
textArea.setColumns(22);
textArea.setRows(8);
textArea.setEditable(false);
JScrollPane scrollText = new JScrollPane (textArea);
scrollText.setSize(200,135);
scrollText.setBorder (BorderFactor.createLineBorder(Color.red));
contentPane.add(scrollText);
```