Putting components in a JFrame

import java.awt.*;
import javax.swing.*;

/** Demonstrate placement of components in a JFrame. Use BorderLayout.
 It places five components in the five possible areas:
 (1) a JButton in the east,
 (2) a JLabel in the west,
 (3) a JLabel in the south,
 (4) a JTextField in the north, and
 (5) a JLabel in the center.
 */

public class ComponentExample extends JFrame {

    public ComponentExample(String t) {
        super(t);
        Container cp = getContentPane();
        cp.add( new JButton("Click me"), BorderLayout.EAST);
        cp.add( new JLabel("label 2"), BorderLayout.WEST);
        cp.add( new JLabel("I got up today"), BorderLayout.SOUTH);
        cp.add( new JTextField("type here", 22), BorderLayout.NORTH);
        cp.add( new JButton("Click here"), BorderLayout.CENTER);
        pack();
    }

    public static void main(String[] args) {
        ComponentExample pack();
    }
}

Basic Components

Component Container
Button, Canvas, Choice
TextArea, Label, List, ScrollBar
TextComponent
TextField, TextArea

Note the use of subclasses to provide structure and efficiency. For example, there are two kinds of JToggleButton, so that class has two subclasses.

What components can go in a JFrame

Packages that contain classes that deal with GUIs:
java.awt: Old package.  javax.swing: New package.

Javax.swing has a better way of listening to buttons, text fields, etc. Its components are more flexible.

JFrame's content pane

Layout manager: An instance controls the placement of components.

JFrame layout manager default: BorderLayout.

BorderLayout layout manager: Can place 5 components:
Container cp= getContentPane();
Button jb= new JButton("Click here");
Label jl= new JLabel("label 2");
p.add(jb, BorderLayout.EAST);
p.add(jl, BorderLayout.WEST);
pack(); setVisible(true);

Components that can contain other components

Component: Something that can be placed in a GUI window. They are instances of certain classes, e.g.
Button, Button: Clickable button
JLabel, Label: Line of text
JTextField, TextField: Field into which the user can type:
JTextArea, TextArea: Many-row field into which user can type
JPanel, Panel: Used for graphics; to contain other components
JCheckBox: Checkable box with a title
JComboBox: Menu of items, one of which can be checked
JRadioButton: Same functionality as JCheckBox
Container: Can contain other components
JFrame, Frame: Can contain other components

java.awt is the old GUI package.
javax.swing is the new GUI package.

When they wanted to use an old name, they put J in front of it.
(e.g. Frame and JFrame)

When constructing javax.swing, the attempt was made to rely on the old package as much as possible.

So, JFrame is a subclass of Frame.

But they couldn’t do this with JPanel.
import java.awt.*; import javax.swing.*;

/** Instance has labels in east/west, JPanel with four buttons in center. */
public class PanelDemo extends JFrame {
    JPanel p = new JPanel();
    /** Constructor: a frame with title "Panel demo", labels in east/west, blank label in south, JPanel of 4 buttons in the center */
    public PanelDemo() {
        super("Panel demo");
        p.add(new JButton("0")); p.add(new JButton("1");
        p.add(new JButton("2")); p.add(new JButton("3");
        Container cp = getContentPane();
        cp.add(new JLabel("east"), BorderLayout.EAST);
        cp.add(new JLabel("west"), BorderLayout.WEST);
        cp.add(new JLabel("    "), BorderLayout.SOUTH);
        cp.add(p, BorderLayout.CENTER);
        pack(); show();
    }
}

/** Demo class Box. Comment on constructor says how frame is laid out. */
public class BoxDemo extends JFrame {
    Box b = new Box(BoxLayout.X_AXIS);
    b.add(new JButton("0"));     b.add(new JButton("1"));
    b.add(new JButton("2"));     b.add(new JButton("3"));
    Container cp = getContentPane();
    cp.add(new JLabel("east"), BorderLayout.EAST);
    cp.add(new JLabel("west"), BorderLayout.WEST);
    cp.add(new JLabel("    "), BorderLayout.SOUTH);
    cp.add(b, BorderLayout.CENTER);
    pack(); show();
}

public class BoxDemo2 extends JFrame {
    /** Constructor: frame with title t and 3 columns with n, n+1, and n+2 buttons. */
    public BoxDemo2(String t, int n) {
        super(t);
        // Create Box b1 with n buttons.
        Box b1 = new Box(BoxLayout.Y_AXIS);
        for (int i = 0; i != n; i = i+1)
            b1.add(new JButton("1 " + i));
        // Create Box b2 with n+1 buttons.
        Box b2 = new Box(BoxLayout.Y_AXIS);
        b2.add(b1);
        // Create Box b3 with n+2 buttons.
        Box b3 = new Box(BoxLayout.Y_AXIS);
        b3.add(b2);
        // Create horizontal box b containing b1, b2, b3
        Box b = new Box(BoxLayout.X_AXIS);
        b.add(b1);
        b.add(b2);
        b.add(b3);
        Container cp = getContentPane();
        cp.add(b, BorderLayout.CENTER);
        pack(); show();
    }
}

Simulate BoxLayout Manager in a JFrame

To simulate using a BoxLayout manager for a JFrame, create a Box and place it as the sole component of the JFrame:

JFrame jf = new JFrame("title");
Box b = new Box(BoxLayout.X_AXIS);
Add components to b:
  jf.add(b, BorderLayout.CENTER);

Interested in learning more about GUIs?
1. Start developing a GUI by changing an already existing one. There are a lot of details, and it is hard to get all the details right when one starts from scratch and has little idea about the Java GUI package.
2. The easiest way to learn about GUIs is to listen the ProgramLive lectures in Chapter 17. That chapter shows you code for everything, and you can also download the code from the CD and compile and use it yourself.
3. We have shown you how to place components in a GUI. We haven’t yet shown you how to “listen” to things like button clicks in a GUI. That comes later.