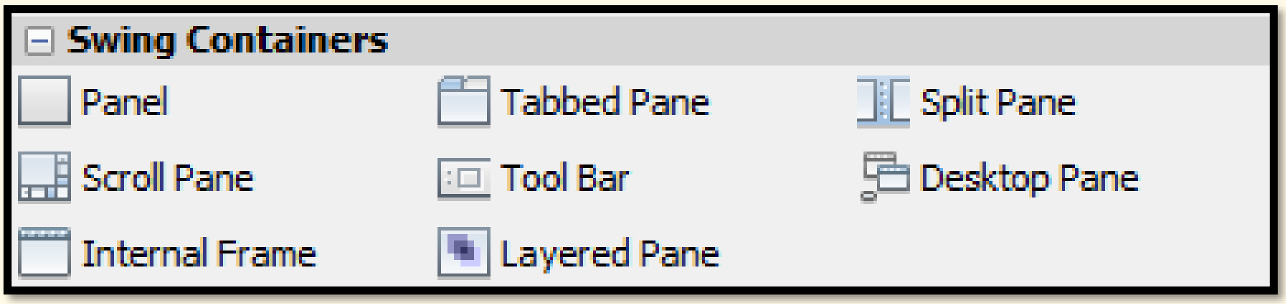
10th Edition

Event-Driven GUIs

RUNSHAW IT & COMPUTING

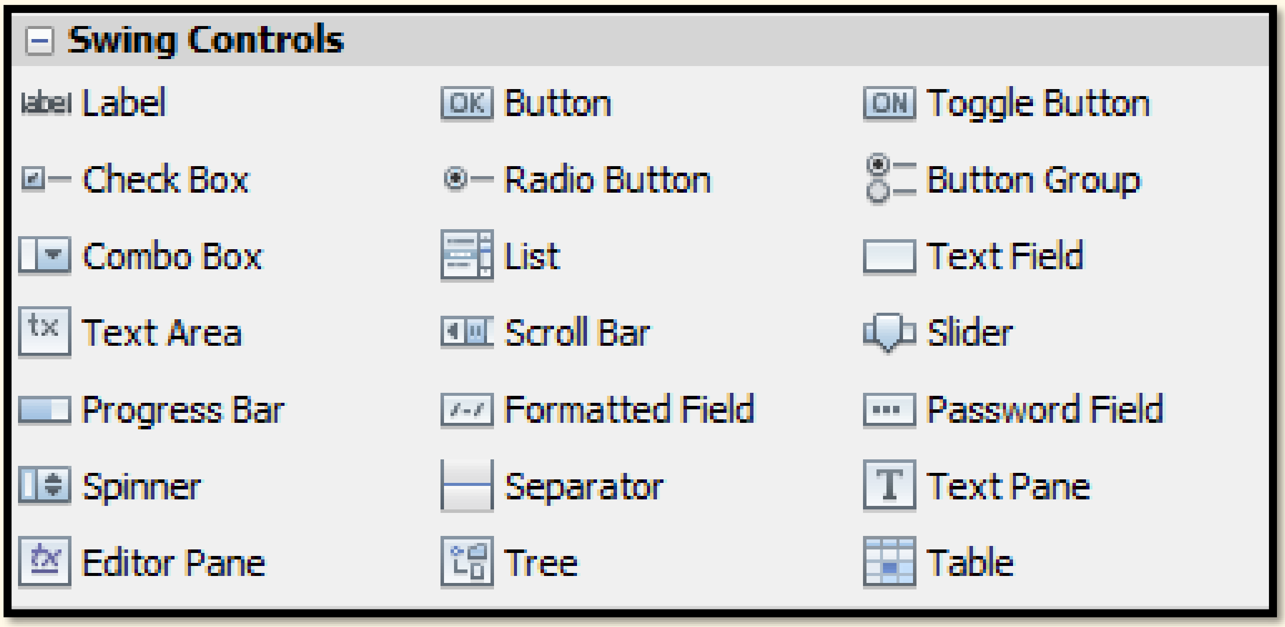
1 Component List

**Screenshot from NetBeans IDE. Investigate other components too!*



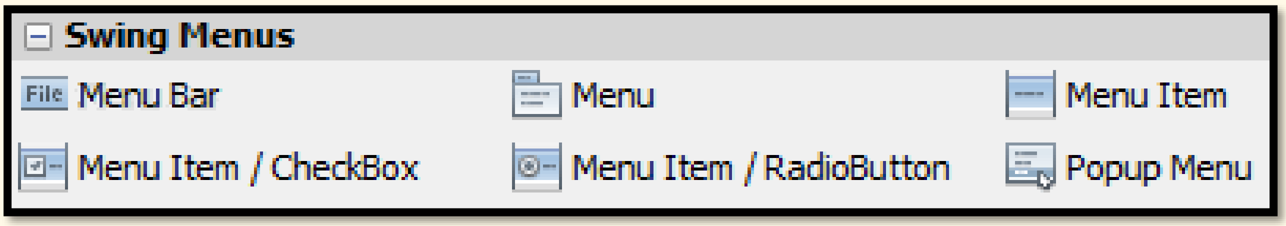
⊟ **Swing Containers**

Panel	Tabbed Pane	Split Pane
Scroll Pane	Tool Bar	Desktop Pane
Internal Frame	Layered Pane	



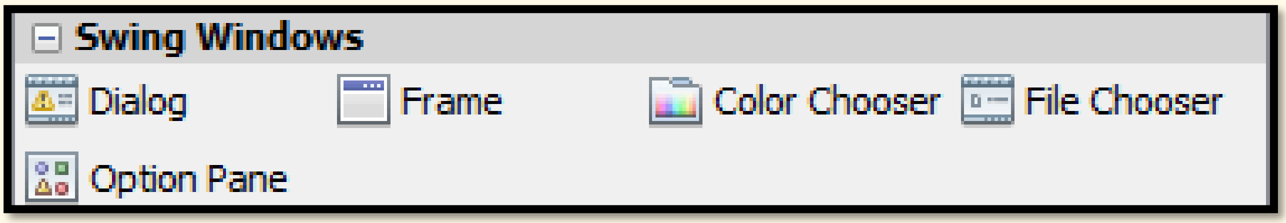
⊟ **Swing Controls**

Label	Button	Toggle Button
Check Box	Radio Button	Button Group
Combo Box	List	Text Field
Text Area	Scroll Bar	Slider
Progress Bar	Formatted Field	Password Field
Spinner	Separator	Text Pane
Editor Pane	Tree	Table



⊟ **Swing Menus**

Menu Bar	Menu	Menu Item
Menu Item / CheckBox	Menu Item / RadioButton	Popup Menu



⊟ **Swing Windows**

Dialog	Frame	Color Chooser	File Chooser
Option Pane			

2 Introduction

2.1 Blank Window

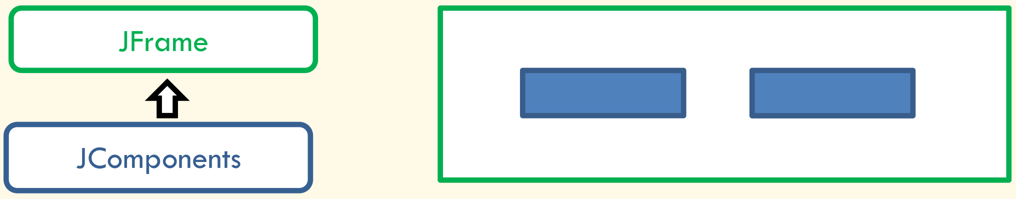
```
//Library Class - Import
import java.awt.*;
import javax.swing.*;
```

```
public class BlankWindow extends JFrame
{

     public void makeFrame()
     {
          this.setTitle("Blank Window");
          this.setSize(400, 200); //Width, Height
          this.setLocation(200, 100); //X, Y
          this.setLayout(null);
          this.setDefaultCloseOperation
                              (JFrame.EXIT_ON_CLOSE);
          this.setVisible(true);
     }


     public static void main(String[] args)
     {
          BlankWindow bw = new BlankWindow();
          bw.makeFrame();
     }
}
```

3 Basic Components

3.1 Components onto JFrame

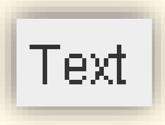
In this example, the components will be placed directly on to the JFrame, specifying the **size** and **location** for each component.



```
JLabel lblHello = new JLabel("Hello World");  //Global


public void makeFrame()
{
     this.setTitle("Blank Window");
     this.setSize(200, 100);
     this.setLayout(null);
     this.setDefaultCloseOperation
                              (JFrame.EXIT_ON_CLOSE);


     lblHello.setSize(50, 20);    //Width, Height
     lblHello.setLocation(0, 0); //X, Y
     this.add(lblHello);


     this.setVisible(true);

}
```

3.2 JLabel (including images)

A **JLabel** can contain text, an image or both.



```
//Global
JLabel lblHello = new JLabel("Hello World”);
```

```
//Change text
lblHello.setText("Goodbye World");
```

```
//Make invisible
lblHello.setVisible(false);
```

```
//Optional – Change colours
lblHello.setForeground(Color.blue); //Text colour
lblHello.setOpaque(true); //Labels: See Background
lblHello.setBackground(Color.lightGray);
```

```
//Local (inside method)
lblHello.setSize(50,20);          //Width, Height
lblHello.setLocation(0,0);        //X, Y
this.add(lblHello);               //Add to THIS WINDOW
```

```
//Global
JLabel lblLogo = new JLabel();
```

```
//Image: Change image using file path
lblLogo.setIcon( new ImageIcon("logo.jpg") );
```

```
//Make invisible
lblLogo.setVisible(false);
```

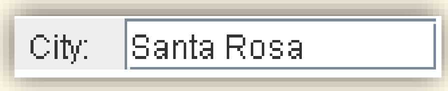
```
//Optional - Change colours
lblLogo.setOpaque(true);     //Labels: See Background
lblLogo.setBackground(Color.lightGray);
```

```
//Local (inside method)
lblLogo.setSize(50,20);      //Width, Height
lblLogo.setLocation(0,0);    //X, Y
this.add(lblLogo);           //Add to THIS WINDOW
```

3.3 JTextField

A **JTextField** or **JPasswordField** is used to display text or allow the user to enter to enter a value.

*A **FocusListener** can also be used to clear the text box when it is selected (later in the eBook)*



```
//Global
JTextField tfCity = new JTextField();
```

```
//Change text
tfCity.setText("Manchester");
```

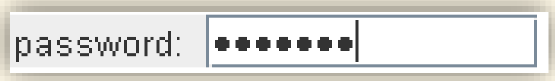
```
//Local (inside method)
tfCity.setSize(50,20);        //Width, Height
tfCity.setLocation(0, 50);    //X, Y
this.add(tfCity);             //Add to THIS WINDOW
```

```
//Get value from a Text Field
//Usually implemented in a button’s Action Listener

String textValue = tfSimpleField.getText();


//Text Field: Convert to another datatype
int intValue = Integer.parseInt(textValue);
double dblValue = Double.parseDouble(textValue))
boolean boolValue =Boolean.parseBoolean(textValue);
char charValue = textValue.charAt(0);
```

3.4 JPasswordField

A **JPasswordField** is used to allow the user to enter a value without displaying its value.

*A **FocusListener** can also be used to clear the text box when it is selected (later in the eBook)*



```
//Global
JPasswordField pwfPassword = new JPasswordField();
```

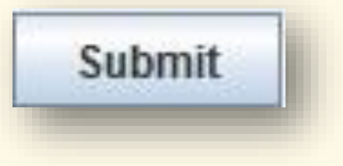
```
//Get value from Password Field
//Usually implemented in a button’s Action

String password = pwfPassword.getText();
```

```
//Local (inside method)
pwfPassword.setSize(50,20);       //Width, Height
pwfPassword.setLocation(0, 50);   //X, Y
this.add(pwfPassword);            //Add to THIS WINDOW
```

3.5 JButton

3.5.1 Create JButton

A **JButton** also requires an **ActionListener**



```
JButton btnSubmit = new JButton("Submit");
```

```
//Button: Add Event Listener
btnSubmit.addActionListener(this);
```

```
//Help when hovering over a component
btnSubmit.setToolTipText("Click to submit");
```

```
//Disable button
btnSubmit.setEnabled(false);
```

```
//Local (inside method)

btnSubmit.setSize(50,20);         //Width, Height
btnSubmit.setLocation(0,100);     //X, Y
this.add(btnSubmit);              //Add to THIS WINDOW
```

Next Page… to add an ActionListener

3.5.2 ActionListener / ActionCommand

An **ActionListener** performs an action when the **JButton** is clicked and requires the following stages:

Stage	Description
1	Import the event classes from the library
2	Implement ActionListener after your class name
3	Code (override) the actionPerformed method
4	Attach a listener to your JButton (**See JButton)**
5	Add an if statement in actionPerformed for each JButton

Stage	Description
6	(Optional) Attach a Command String to the JButton
7	(Optional) Add an if statement for the Command String

```
//1 Library Class – Import
import java.awt.event.*;
```

```
//2 Implement ActionListener
public class GUI extends JFrame implements ActionListener
```

```
//3 Override the actionPerformed Method
public void actionPerformed(ActionEvent ae)
{
     Object source = ae.getSource();
     //if statement for each button (5)
}
```

```
//4 Button: Add Event Listener
btnSubmit.addActionListener(this);
```

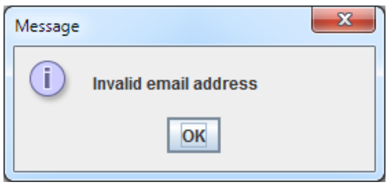
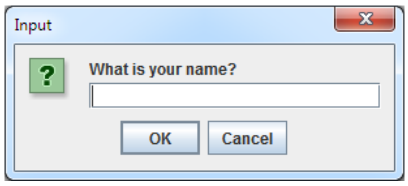
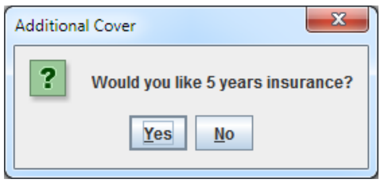
```
//5 Override the actionPerformed Method
if(source == btnSubmit)
{
     System.out.println("Submit button pressed");
     //Any other instructions
}
```

```
//6 Button: Set Action Command
btnSubmit.setActionCommand("submit");
```

```
//7 Edit actionPerformed() : Detect command string

String command = ae.getActionCommand();

if(command.equals("submit"))
{
     System.out.println("Submit button pressed");
     //Any other instructions
}
```

3.6 PopUp Window

A **JOptionPane** can prompt the user for a value or output a message.

MessageDialog	InputDialog	ConfirmDialog
 Message Invalid email address 	 Input What is your name? 	 Additional Cover Would you like 5 years insurance?

```
//MessageDialog – Output only

JOptionPane.showMessageDialog(null, "Invalid email
                                        address");
```

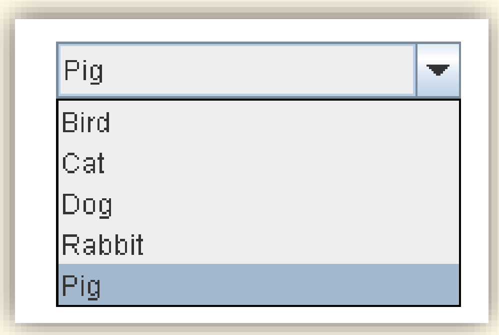
```
//InputDialog – Input as String

String reply = JOptionPane.showInputDialog("What is your
                                          name?");
```

```
/*ConfirmDialog – Answer Yes/No/Cancel/OK
Saved as….
JOptionPane.YES_OPTION
JOptionPane.NO_OPTION
JOptionPane.CANCEL_OPTION
JOptionPane.OK_OPTION
*/

String question="Would you like 5 years insurance?";
String title = "Additional Cover";

int answer = JOptionPane.showConfirmDialog(null, question,
                       title, JOptionPane.YES_NO_OPTION);
```

4 Validation Components

4.1 JComboBox

A **JComboBox** contains a list and limits the options that a user can select.



```
String[] arrayPets = new String[] {"Bird", "Cat","Dog"};
```

```
JComboBox<String> cbxPet =
                     new JComboBox<String> (arrayPets);
```

```
cbxPet.setSize(50,20);  //Pixels: X by Y
cbxPet.setLocation(0,0);     //Image: X by Y
panelOne.add(cbxPet);   //Panel: Add ComboBox
```

```
cbxPet.addItem("Rabbit");    //Add one more item
```

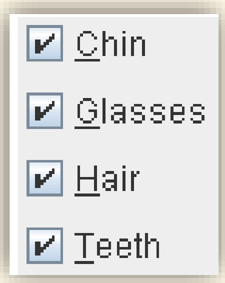
```
cbxPet.removeItem("Pig");    //Remove one item
cbxPet.removeAllItems();     //Remove all items
```

```
cbxPet.setSelectedItem("Pig");//Set option by text
cbxPet.setSelectedIndex(1); //Set option by row
```

```
//Get current text or row
String txtValue= (String) cbxPet.getSelectedItem();
int theOption=cbxPet.getSelectedIndex();
```

4.2 JCheckBox & ButtonGroup

A **JCheckBox** can be used to allow an option to be selected or unselected and will return a boolean, true or false. It can be added to a **ButtonGroup**.



```
JCheckBox cbChin = new JCheckBox("Chin");
JCheckBox cbGlasses = new JCheckBox("Glasses");
```

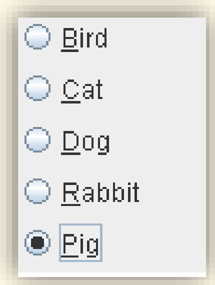
```
cbChin.setSelected(true);
```

```
boolean answer = cbChin.isSelected();
```

```
//ActionPerformed() method

if(cbChin.isSelected())
{
     //action
}

else if(cbGlasses.isSelected())
{
     //action
}
```

4.3 JRadioButton

A **JCheckBox** can be used to allow an option to be selected or unselected and will return a boolean, true or false. It can be added to a **ButtonGroup**.



```
JRadioButton rbHair = new JRadioButton("Bird");
JRadioButton rbTeeth = new JRadioButton("Cat");
```

```
rbHair.setSelected(true);
```

```
boolean answer = rbHair.isSelected();
```

```
//ActionPerformed() method

if(rbHair.isSelected())
{
     //action
}

else if(rbHair.isSelected())
{
     //action
}
```

4.4 ButtonGroup

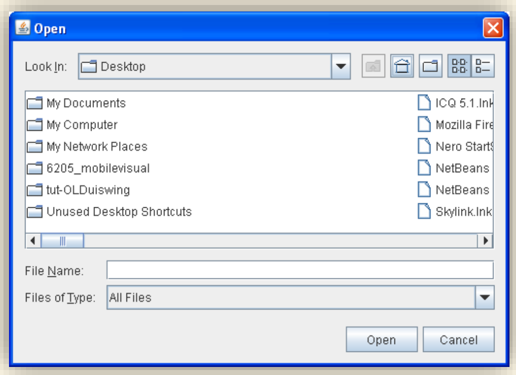
Adding buttons to a **ButtonGroup** ensures that only one item can be selected at once. A **ButtonGroup** is not a visible component, so size and location declarations are not needed.

```
ButtonGroup groupFeatures = new ButtonGroup();
```

```
groupFeatures.add(cbChin);
groupFeatures.add(cbGlasses);
groupFeatures.add(rbHair);
groupFeatures.add(rbTeeth);
```

4.5 JFileChooser

The **JFileChooser** allows the user to select a file they wish to open or save.



```
//Library Class – Import
import javax.swing.filechooser.*;
```

```
JFileChooser chooser = new JFileChooser();
```

```
//Show 'Open' File Chooser Window
int returnVal = chooser.showOpenDialog(this);
```

```
//Show 'Save' File Chooser Window
int returnVal = chooser.showSaveDialog(this);
```

```
//Store Filename
String filename = chooser.getSelectedFile().getName();
```

```
//Optional: File Filter
FileNameExtensionFilter filter
         = new FileNameExtensionFilter
                   ("Images Only", "jpg", "gif");
```

4.5.1 File Handling

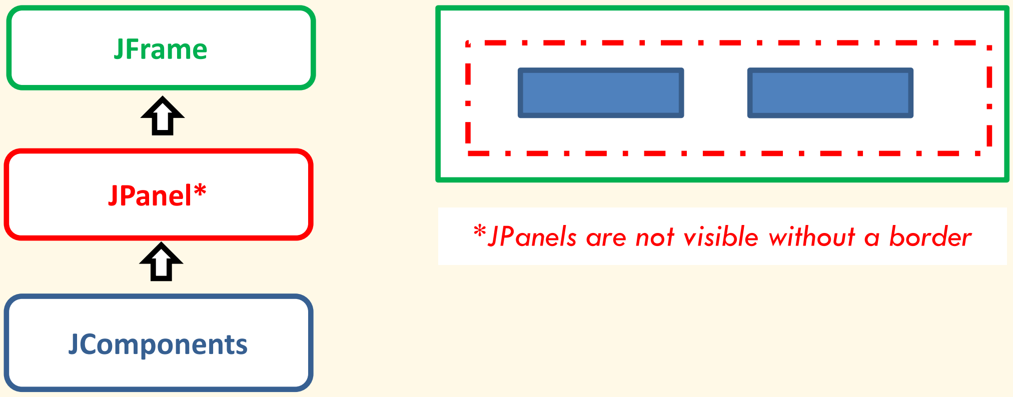
```
//Library Class – Import
import java.io.*;
```

```
//Optional: File Filter
Path FileDirectory = Paths.get("PathToFile");
//use ./ if the filepath is relative to location

Files.delete(FileDirectory);
```

```
try
{

}
catch (Exception e)
{
     System.out.println("Problem with file delete");
}
```

5 JPanel

One or more **JPanel**'s are added to a **JFrame,** A *layout* must be declared for each. Components are then added to the **JPanel**.



```
JPanel pnlHello = new JPanel();
```

```
//This Window: 1x1 Grid
this.setLayout(new GridLayout(1,1));

//This Window: Add panel
this.add(panelName);
```

```
//Panel: sizes & locations
panelName.setLayout(null);

//Component Pixels: X by Y
componentName.setSize(50, 20);

//Component Pixels: X by Y
componentName.setLocation(10, 10);

//Panel: Add component
panelName.add(componentName);
```

```java
public class BlankWindow extends JFrame
{
     JPanel pnlHello = new JPanel();
     JLabel lblHello = new JLabel("TEST");

     public void makeFrame()
     {
          makePanelOne();

          this.setTitle("First Title");
          this.setSize(200, 100);
          this.setLocation(10, 10);
          this.setLayout(new GridLayout(1,1));
          this.setDefaultCloseOperation
                                   (JFrame.EXIT_ON_CLOSE);

          this.add(panelOne);     //Window: Add panel
          this.setVisible(true);
     }

     public static void main(String[] args)
     {
          BlankWindow bw = new BlankWindow();
          bw.makeFrame();
     }
}
```

```java
public void makePanelOne()
{
     panelOne.setLayout(null);
     lblHello.setSize(50, 20);
     lblHello.setLocation(0, 0);
     panelOne.add(lblHello);
}
```

5.1.1 JScrollPane: JPanel

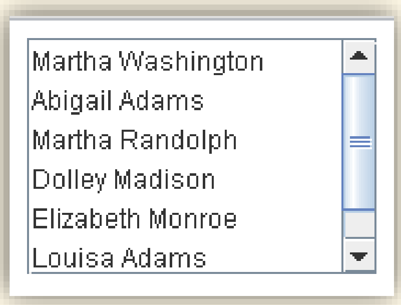
```
//make the preferred size large
panelName.setPreferredSize(new Dimension(2000, 1000));

scrollPane = new JScrollPane(panelName);
//make the size scrollpane size smaller
scrollPane.setSize(500,300);
```

6 Multi-Line Text

6.1 JList

6.1.1 Data: List Model

A simple way to manage a **JList** is by using a **ListModel.**



Martha Washington
Abigail Adams
Martha Randolph
Dolley Madison
Elizabeth Monroe
Louisa Adams

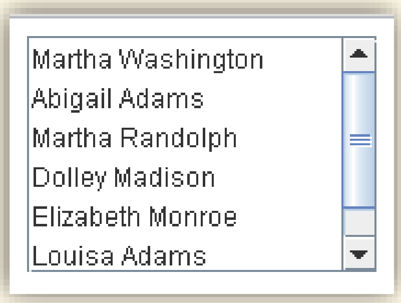
```
//Declare ListModel
DefaultListModel<String> listModel = new
                        DefaultListModel<>();
```

```
//ListModel: Add items

listModel.addElement("Mr");
listModel.addElement("Mrs");
listModel.addElement("Miss");
```

```
//ListModel: Remove items

listModel.removeElement("Mr");
listModel.removeAllElements();
```

6.1.2 **Create JList**

A **JList** displays many items and allows the user to select one or more items


Martha Washington
Abigail Adams
Martha Randolph
Dolley Madison
Elizabeth Monroe
Louisa Adams

```
JList<String> nameList = new JList<>(listModel);
```

```
//Select one item
nameList.setSelectionMode
              (ListSelectionModel.SINGLE_SELECTION);
```

```
//Change layout of list
nameList.setLayoutOrientation(JList.HORIZONTAL_WRAP);
```

```
//Number of rows visible
nameList.setVisibleRowCount(6);
```

```
//To get the row selected
int index = nameList.getSelectedIndex();
System.out.println("Index Selected: " + index);

//To get the text from the row
String value = (String) nameList.getSelectedValue();
System.out.println("Value Selected: " + value);
```

6.1.3 JScrollPane: JList

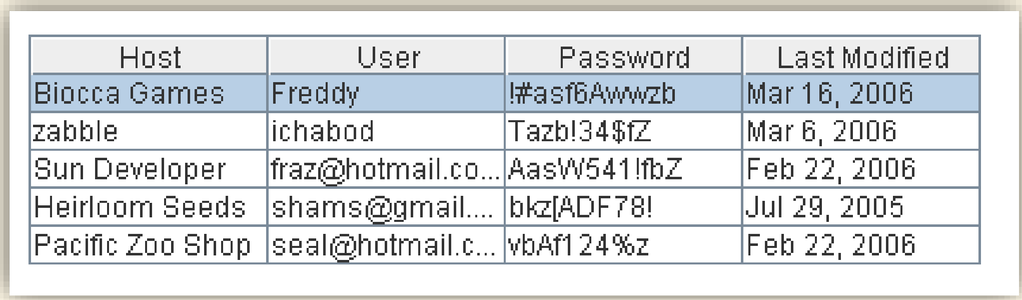
```
//Declare JScrollPane
JScrollPane listScroll = new JScrollPane(nameList);
//Comment out the setSize, setLocation & add() for the
JList
//Add the setSize, setLocation & add() for the Scroller
```

6.2 JTable

The data in a JTable is managed using a **TableModel.** The TableModel is built using **headings** and **data.** Rows can then be added or removed to the model.

A **JTable** is built using the TableModel as a parameter.

After being built, the **JTable** should be added to a JScrollPane.



Host	User	Password	Last Modified
Biocca Games	Freddy	!#asf6Awwzb	Mar 16, 2006
zabble	ichabod	Tazb!34$fZ	Mar 6, 2006
Sun Developer	fraz@hotmail.co...	AasW541!fbZ	Feb 22, 2006
Heirloom Seeds	shams@gmail....	bkz[ADF78!	Jul 29, 2005
Pacific Zoo Shop	seal@hotmail.c...	vbAf124%z	Feb 22, 2006

6.2.1 TableModel

```
//Library Class - Import
import javax.swing.table.*;
```

```
//Global
DefaultTableModel model;
```

```
String[] headings= {"Column1","Column2"};
```

```
String[][] dummyData ={
                        {"John", "Smith"},
                        {"Ivan", "Black"}
                        };
```

```
//Local
model = new DefaultTableModel(dummyData, headings);
```

6.2.2 JTable

```
JTable demoTable = new JTable(model);
```

```
//Optional: Autosort
peopleTable.setAutoCreateRowSorter(true);
```

6.2.3 JScrollPane: JTable

```
JScrollPane demoTableScroll; //Global
```

```
demoTableScroll = new JScrollPane(demoTable); //Local
```

```
demoTableScroll.setSize(300,200); //X Pixels, Y Pixels
demoTableScroll.setLocation(0,0); //X Loc, Y Loc

this.add(demoTableScroll);
```

6.2.4 Number of Rows

```
int numberRows = model.getRowCount();
```

6.2.5 Add Row

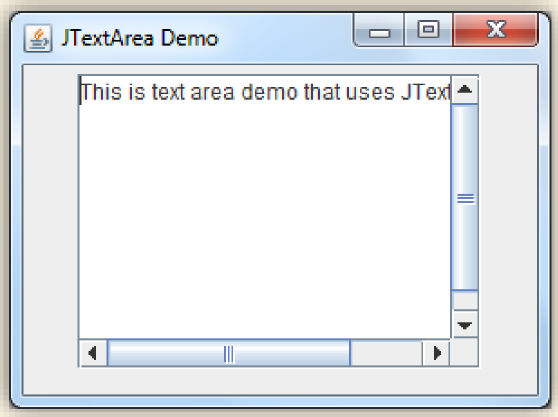
```
String[] dataToAdd = {"Helen", "Smith"};

model.addRow(dataToAdd);
```

6.2.6 Delete Row

```
model.removeRow(numberRows-1); //Remove Latest Row
```

6.3 JTextArea

A **JTextArea** holds text in rows and columns.



This is text area demo that uses JText

```
//GLOBAL

JTextArea jtaDesc = JTextArea(5, 20);
// (rows, columns)
```

```
//Change text
jtaDesc.setText("Manchester");
```

```
//Local (inside method)
jtaDesc.setSize(50,20);      //Width, Height
jtaDesc.setLocation(0, 50); //X, Y
this.add(jtaDesc);          //Add to THIS WINDOW
```

6.3.1 **JScrollPane: JTextArea**

```
//Declare JScrollPane
JScrollPane descScroll = new JScrollPane(jtaDesc);
//Comment out the setSize, setLocation & add() for the
JList
//Add the setSize, setLocation & add() for the Scroller
```

6.4 JTextPane (HTML)

Can be used to display text formatted using HTML code. This component **must be added to a JPanel**

```
//Library Class – Import
import javax.swing.JTextPane;
```

```
//GLOBAL
JTextPane tp = JTextPane();
```

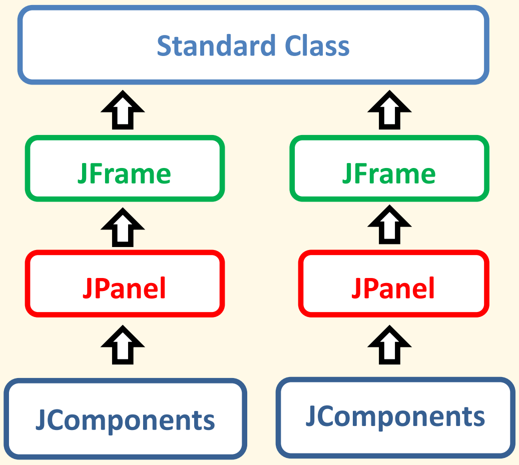
```
//Local (inside method)
tp.setContentType("text/html")        //Width, Height
tp.setText("<html><b>This is bold HTML</b></html>");
```

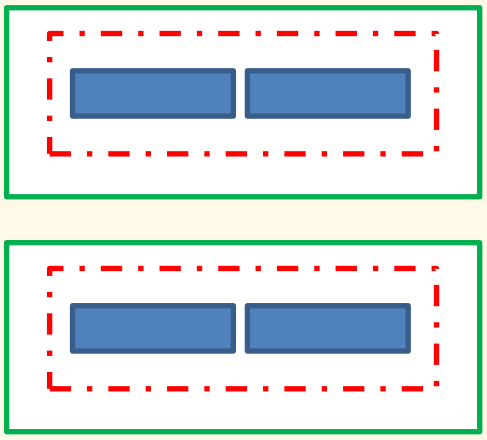
```
//Local (inside method)
tp.setSize(50,20);            //Width, Height
tp.setLocation(0, 50);        //X, Y
nameOfPanel.add(tp);          //Add to THIS WINDOW
```

7 Multiple Screens

7.1 Multiple Windows

- Create a **JFrame**
- Create each **JPanel** (with a layout) with **Components**
- Do not use the keyword '**this**'
- Refer to each JFrame by name





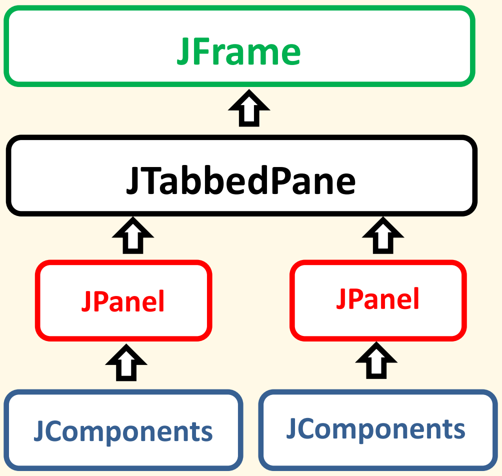
```
//Standard Class
public class FirstGUI ~~extends JFrame~~
```

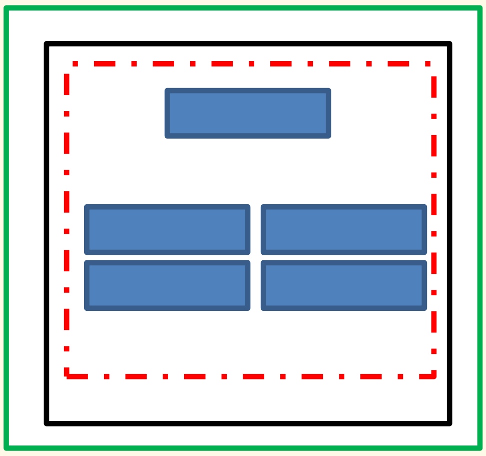
```
//A new window
JFrame windowOne = new JFrame();

//A second window
JFrame windowTwo = new JFrame();
```

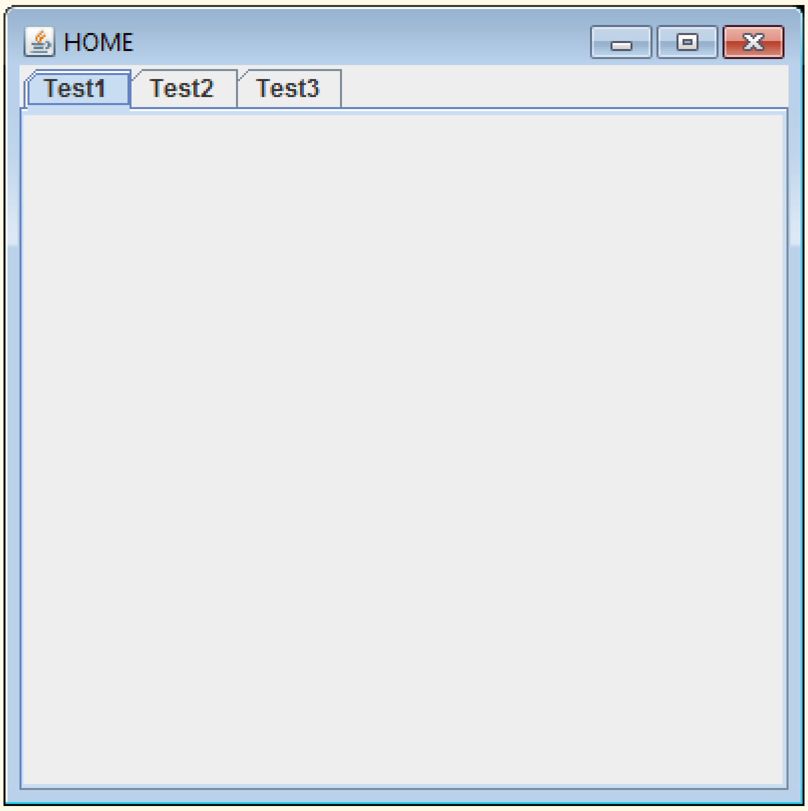
```
windowOne.setVisible(false); //Invisible

windowTwo.setVisible(true); //Visible
```

7.2 Multiple Tabs

- Create a **JFrame**
- Create each **JPanel** (with a layout) with **Components**
- Add each **JPanel** to a tab







```java
//A new TabbedPane
JTabbedPane tabs = new JTabbedPane();
```

```java
//A new TabbedPane down the left-hand side
JTabbedPane tabs = new JTabbedPane(JTabbedPane.LEFT);
```

```java
//Add Tabs
tabs.addTab("First Panel", panelOne);
tabs.addTab("Second Panel", panelTwo);
```

```java
//Add Tabs with an image and a tooltip
tabs.addTab("First Panel",imgOne,panelOne,"First");
tabs.addTab("Second Panel",imgTwo,panelTwo,"Second");
```

```java
//Change tab
tabs.setSelectedIndex(1);
```

```java
//Optional: Disable tab
tabs.setEnabledAt(0,false);
```

```java
//Add TabbedPane to this JFrame
this.add(tabs);
```

7.2.1 Hiding Tabs

An unorthodox technique to hide the actual tabs of a tabbed pane is to move them from view. They are about 50 pixels, so by setting the location of the tabs at -50, they will start out of view. **Note:** this can only be done using null layout for the frame, rather than GridLayout.

~~this.setLayout(new GridLayout(1,1));~~

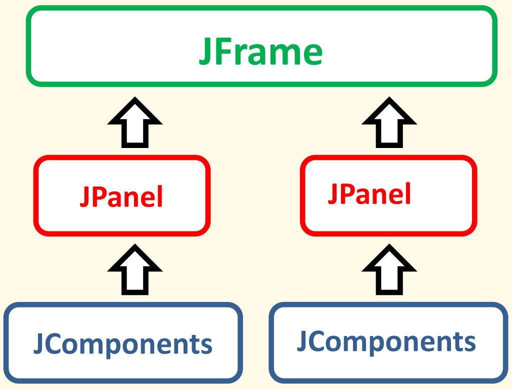
```java
this.setLayout(null);
```

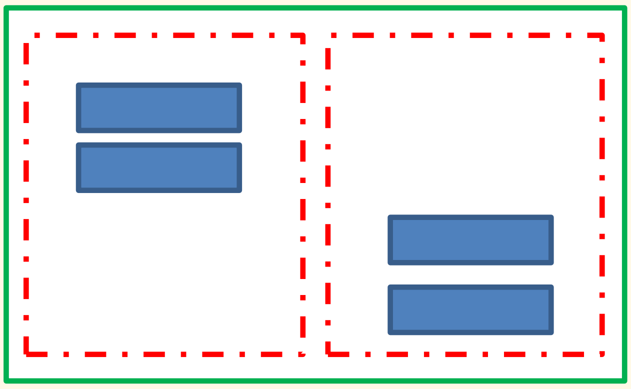
```java
tabs.setSize(600, 400); //Width, Height
tabs.setLocation(0, -50); //X, Y
```

```java
//Add TabbedPane to this JFrame
this.add(tabs);
```

7.3 Panels (One Screen)

- Create a **JFrame**
- Create each **JPanel** (with a layout) with **Components**
- Add each **JPanel** to the **JFrame**





```
//This Window: Needs sizes & locations
this.setLayout(null);
```

```
panelOne.setSize(50, 20);    //Pixels: X by Y
panelOne.setLocation(10, 10); //Panel: X by Y
this.add(panelOne);          //This Window: Add panel
```

```
panelTwo.setSize(50, 20);    //Pixels: X by Y
panelTwo.setLocation(10, 10); //Panel: X by Y
this.add(panelTwo);          //This Window: Add panel
```

8 Listeners

8.1 Focus Listener

A **FocusListener** can be used to perform an action when a JTextField is selected

```
//Library Class – Import
import java.awt.event.*;
```

```
public class FirstGUI extends JFrame
                                        implements FocusListener
```

```
textfieldName.addFocusListener(this);
```

```
public void focusGained(FocusEvent fe)
{
     if(fe.getSource() == textfieldName)
     {
          //Action
     }
}
```

```
public void focusLost(FocusEvent fe)
{
}
```

8.2 Key Listener

A **KeyListener** allows a process to be activated when a key is pressed.

```
//Library Class – Import
import java.awt.event.*;
```

```
public class FirstGUI extends JFrame
                                        implements KeyListener
```

```
textfieldName.addKeyListener(this);
```

```
public void keyTyped(KeyEvent kevt)
{
     if(kevt.getKeyChar() == KeyEvent.VK_ENTER )
     {
          //Action
     }
}
```

```
public void keyPressed(KeyEvent kevt)
{
}
```

```
public void keyReleased(KeyEvent kevt)
{
}
```

8.3 Mouse Listener

A **MouseListener** allows a process to be activated when the mouse is used.

```
import java.awt.event.*; //Library Class – Import
```

```
public class FirstGUI extends JFrame
                                implements MouseListener
```

```
tableName.addMouseListener(this);
```

```
public void mouseClicked(MouseEvent mevt)
{
     //Action
}
```

```
public void mousePressed(MouseEvent mevt)
{
}
```

```
public void mouseEntered(MouseEvent mevt)
{
}
```

```
public void mouseExited(MouseEvent mevt)
{
}
```

```
public void mouseReleased(MouseEvent mevt)
{
}
```

```
int theRow = paymentTable.rowAtPoint(mevt.getPoint());
```

9 Customisation

9.1 Theme

We can also set an overall theme known as the **Look and feel.**

```
//Library Class - Import
import javax.swing.UIManager.*;
```

```
public static void main(String[] args)
{
     try
     {
          //Add Theme Code
     }

     catch(Exception e)
     {
          System.out.println("Error with theme");
     }

     FirstGUI fg = new FirstGUI();
     fg.prepareGUI();
}
```

```
//Native Theme
UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
```

```
//Nimbus Theme
for (LookAndFeelInfo info :
          UIManager.getInstalledLookAndFeels())
{
     if ("Nimbus".equals(info.getName()))
     {
       UIManager.setLookAndFeel(info.getClassName());
       break;
     }
}
```

9.2 Font

```
//Library Class – Import
import java.awt.*;
```

```
Font customFont=new Font("Courier", Font.PLAIN,20);
//or
Font.BOLD
Font.ITALIC
```

```
lblTitle.setFont(customFont);
```

9.3 Color

```
//Library Class – Import
import java.awt.*;
```

```
Color customColour = new Color(10,10,255) //RGB
```

```
lblTitle.setForeground(customColour);
```

9.4 Border

```
//Library Class – Import
import java.awt.Color;
import javax.swing.BorderFactory;
import javax.swing.border.Border;
```

```
// create a line border with the color and width
Border blueBorder =
         BorderFactory.createLineBorder(Color.BLUE, 5);
```

```
// set the border of these components
lblName.setBorder(blueBorder);
tfName.setBorder(blueBorder);
```

9.5 Program Icon

```
this.setIconImage(new
     ImageIcon("imgLogo.jpg").getImage());
```

9.6 Remove Frame

```
this.setUndecorated(true);
this.setBackground(new Color(1.0f,1.0f,1.0f,0.5f));
```

9.7 Resize Image

```
//Library Class - Import
import java.awt.*;
```

```
ImageIcon largeLogo = new ImageIcon("logo.jpg");

Image largeLogoImg = largeLogo.getImage();

Image smallLogoImg =
         largeLogoImg.getScaledInstance(50,50,0);

ImageIcon smallLogo = new ImageIcon(smallLogoImg);
```

9.8 Close Operation Override

A **WindowListener** allows a process to be activated when the mouse is used.

```
this.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);

this.addWindowListener(new WindowAdapter()
     {
          public void windowClosing(WindowEvent we)
          {
               System.exit(0);
          }
     });
```

10 Useful Techniques

10.1 Inheritance: Timer

This class uses inheritance to extend **JLabel** and create an animated **TimerLabel**

```
//Library Class - Import
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
```

```
public class TimerLabel
         extends JLabel implements ActionListener
{
     int counter=0;
}
```

```
public TimerLabel()
{
     Timer t = new Timer(10, this);
     t.start();
}
```

```
public void actionPerformed(ActionEvent e)
{
     counter++;
     this.setText(counter+"");
}
```

10.2 Printing

Copy the **PrintUtilities.java** file from the shared drive in to the same folder as your source code.

```
//The PrintUtilities class has been written by someone
else

PrintUtilities.printComponent(panelOne);
```

10.3 Button Arrays

```
JButton[][] theButtons = new JButton[3][3]; //3x3
//Declare Button Array
```

```
JPanel panelBoard = new JPanel(new GridLayout(3,3));
//Declare Panel with a GridLayout, 1 cell per button
```

```
//Creating the Buttons - Part 1

for(int i=0;i<3;i++) //Loop through rows
{
    for(int j=0;j<3;j++)// Loop through cols
    {
        //Create a new button

        //Attach an Action Listener to the button

        //Add button in Button Array

        //Add to button to Panel
    }
}
```

```
//Creating the Buttons – Part 2

JButton temp = new JButton(i+"  "+j);
//Create a new button

temp.addActionListener(this);
//Attach an Action Listener to the button

theButtons[i][j]=temp;
//Place button in Button Array

panelBoard.add(theButtons[i][j]);
//Add to Panel
```

```
//Action Listener - Detecting a button click
//Inside actionPerformed()


for(int i=0;i<3;i++) //LOOP TO CHECK WHICH ROW
{
     for(int j=0;j<3;j++) //LOOP TO CHECK WHICH COLUMN
     {
          if(theButtons[i][j]==event.getSource())
          {
               //ACTION
          }
     }
}
```

10.4 ArrayList

An ArrayList is similar to an array but can only be used to store objects. If primitive types are stored, they will be converted using *Autoboxing*.

```
ArrayList<Person> personList=new ArrayList<Person>();
```

```
//Add Items to an ArrayList
Person temp = new Person("Ivan","Black","2221111");
personList.add(new Person(temp);
```

```
//Fill TableModel from ArrayList
for (int i = 0; i < personList.size(); i++)
{
   String fname = originalLeagueList.get(i).getForename();
   String same = originalLeagueList.get(i).getSurname();
   String phone = originalLeagueList.get(i).getPhone();

   Object[] data = {fname, sname, phone};

   tableModel.addRow(data); //Declared globally
}
```

```
//Optional - convert existing array to ArrayList
String[] stringArray = {"a","b","c","d","e"};

ArrayList<String> arrayList = new
     ArrayList<String>(Arrays.asList(stringArray));
```

10.5 Passing a JFrame

It is possible to pass a JFrame into a subclass as a parameter. This can be done using a constructor

```
SubGUI sg = new SubGUI(this);
```

```
public class SubGUI extends JFrame
{
     JFrame parentFrame;

     //constructor
     public SubGUI(JFrame tempParent)
     {
          parentFrame = tempParent;
          parentFrame.setVisible(false);
     }
}
```

Java Conventions

Conventions are rules which programmers follow to make their code more readable for themselves and other programmers. Below are several conventions you should attempt to adhere to:

Convention	Example

Class names	`public class FirstProgram`
	
	`//Starts with an uppercase letter`

Method names	`public static void main(String[] args)`
	
	`//Starts with a lowercase letter and ends with brackets`

Variable names	`int age;`
	
	`//Starts with a lowercase letter`

The Semicolon

A semicolon is used to end a statement.

Correct	Example
Output	`System.out.println("Hello World");`
Input	`name = inputScanner.nextLine();`
Calculation	`answer = number1+ number2;`
Method Call	`mp.changeName("Phil");`

Incorrect	Example
Class Declaration	`public class FirstProgram;`
Method Declaration	`public static void main(String[] args);`
IF statement	`if(iAge<18);`
WHILE loop	`while(bottles<100);`
FOR loop	`for(int i=0;i<100;i++);`