Unit-3 GUI Programming

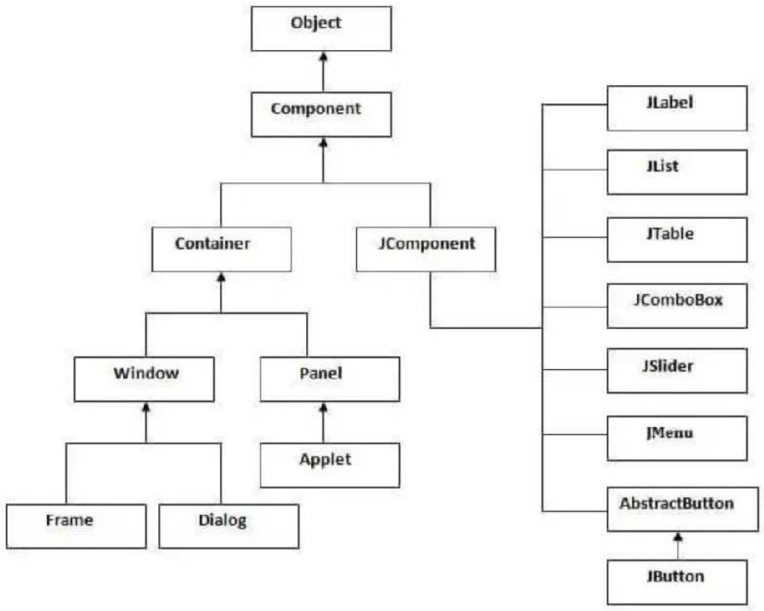
Introducing Swing:

Swing is part of Java Foundation Classes (JFC). It is used to create window-based applications which makes it suitable for developing lightweight desktop applications. It is built on the top of AWT (Abstract Window Toolkit) API (Application Programming Interface) and entirely written in java. It is a Java Graphical User Interface (GUI) toolkit. Swing provides platform-independent and lightweight components.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu etc.

First developed by Netscape Communications Corporation on December 16, 1996. After that on April 2, 1997, Sun Microsystems and Netscape Communications Corporation announced their intention to incorporate IFC (Internet Foundation Classes) with other technologies to form the Java Foundation Classes. The "Java Foundation Classes" were later renamed "Swing."

Hierarchy of Java Swing classes:



All the components in swing like JButton, JComboBox, JList, JLabel are inherited from the JComponent class which can be added to the container classes. Containers are the windows like frame and dialog boxes. Basic swing components are the building blocks of any gui application. Methods like setLayout override the default layout in each container. Containers like JFrame and JDialog can only add a component to itself.

Written by Nawaraj Prajapati (MCA From JNU)

Features of Swing:

- **Platform Independent:** It is platform-independent; the swing components used to build the program are not platform-specific. It can be used on any platform and anywhere.
- **Lightweight:** Swing components are lightweight, which helps in creating the UI lighter. The swings component allows it to plug into the operating system user interface framework, including the mappings for screens or devices and other user interactions like key press and mouse movements.
- **Plugging**: It has a powerful component that can be extended to provide support for the user interface that helps in a good look and feel to the application. It refers to the highly modular-based architecture that allows it to plug into other customized implementations and frameworks for user interfaces. Its components are imported through a package called java.swing.
- **MVC**: They mainly follow the concept of MVC, which is the Model View Controller. With the help of this, we can make changes in one component without impacting or touching other components. It is known as loosely coupled architecture as well.
- **Customizable**: Swing controls can be easily customized. It can be changed, and the visual appearance of the component application is independent of its internal representation.
- **Manageable**: It is easy to manage and configure. Its mechanism and composition pattern also allows changing the settings at run time. The uniform changes can be provided to the user interface without any changes to the application code.

Advantages of Swing:

- Swing provides set of controls or components that are purely written in Java.
- It follows Model-View-Controller (MVC) architecture.
- Swing component inherits look and feel and changes their appearance according to your base theme.
- Swing also provides all the functionality of AWT.
- Swing Application is platform independent.

AWT:

AWT (Abstract Window Toolkit) is an API to develop Graphical User Interface (GUI) or windows-based applications in Java. The AWT contains a number of classes and methods that allow you to create and manage windows. It was platform-dependent. It was heavy-weight. It has a limited set of components. The java.awt package provides classes for AWT API such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc. It was developed by Sun Microsystems In 1995.

Features of AWT:

- A set of native user interface components.
- Graphics and imaging tools, including shape, color, and font classes.
- Layout managers, for flexible window layouts that do not depend on a particular window size or screen resolution.
- A robust event-handling model.

Written by Nawaraj Prajapati (MCA From JNU)

Difference between AWT and Swing:

AWT		Swing	
1	Java AWT components are heavily weighted.	1	Java swing components are light-weighted.
2	Platform Dependent	2	Platform Independent
3	AWT does not follow MVC	3	Swing follows MVC
4	The awt in Java has slower performance as compared to swing.	4	Swing is faster than that of awt.

Containers:

1. JApplet
2. JFrame
3. JPanel

Components:

1. JtextField
2. JLabel
3. JPassword
4. JCheckBox
5. JRadioButton
6. JPromobox
7. JButton
8. JTree
9. JTable
10. JSlider
11. JProgressBar
12. JSpinner
13. JTabbedPane

Packages:

- **javax.swing**: This package contains the core components of Swing, such as JButton, JLabel, JTable, JList, and many more. It also contains the classes for creating top-level containers such as JFrame and JDialog.
- **javax.swing.event**: This package contains the classes for handling events generated by the Swing components. It includes event listener interfaces, event adapter classes, and event objects.
- **javax.swing.border**: This package contains classes for creating borders around the Swing components. It includes the classes for creating line borders, etched borders, and titled borders.
- **javax.swing.layout**: This package contains the classes for creating and managing layout managers in Swing. It includes the commonly used layout managers such as BorderLayout, FlowLayout, GridLayout, BoxLayout, and CardLayout.
- **javax.swing.plaf**: This package contains the classes for the pluggable look and feels feature of Swing. It includes the classes for creating and managing the look and feel themes, and also provides the default look and feel theme for each platform.
- **javax.swing.text**: This package contains the classes for creating and managing text components in Swing. It includes classes for creating text fields, text areas, and other text-related components.
- **javax.swing.table**: This package contains the classes for creating and managing tables in Swing. It includes the classes for creating JTable, TableModel, TableColumn, and TableCellRenderer.

Written by Nawaraj Prajapati (MCA From JNU)

Main methods:

```
setSize(width,height);
setVisible(ture/false);
setDefaultCloseOPeration(EXIT_ON_CLOSE);
setTitle();
```

JFram:

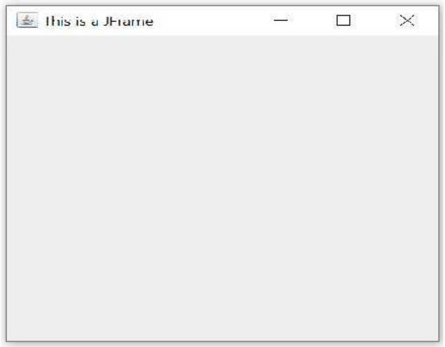
When we add a JFrame form then a class is associated with the form. The javax.swing.JFrame class is a type of container which inherits the java.awt. Frame class. JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI.

Methods of JFrom:

setVisible()	setIconImage()
setDefaultCloseOperation()	setTitle()
setSize()	setBackground()
setLocation()	setResizable()
setBounds()	

Creating a Frame:

```java
import javax.swing.JFrame;
public class JFrameDemo {


	public static void main(String[] args) {

		JFrame frame = new JFrame ();
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setBounds (100,100,500,500);
		frame.setTitle("This is a JFrame");
		frame.setVisible(true);
	}

}
```



Written by Nawaraj Prajapati (MCA From JNU)

JPanel:

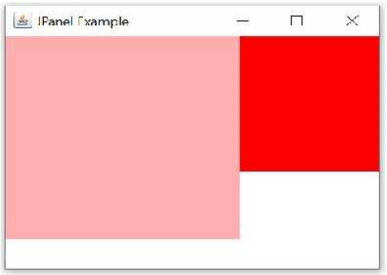
A panel is a component that is contained inside a frame window. A frame can have more than one-panel components inside it with each panel component having several other components.

Methods of JPanel:

add(Component c)	paramString()
setLayout(LayoutManager l)	updateUI()
remove(Component c)	setUI(PanelUI ui)

For example:

```
import java.awt.Color;
import java.awt.Container;
import javax.swing.JFrame;
import javax.swing.JPanel;
public class JPanelDemo {
	public static void main(String[] args) {
		JFrame frame = new JFrame();
		frame.setVisible(true);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setBounds(100, 100, 500, 400);
		frame.setTitle("JPanel Example");
		Container contner = frame.getContentPane();
		contner.setBackground(Color.WHITE);
		contner.setLayout(null);

		JPanel panelFirst = new JPanel();
		panelFirst.setBounds(0, 0, 300, 300);
		panelFirst.setBackground(Color.pink);
		contner.add(panelFirst);

		JPanel panelSecond = new JPanel();
		panelSecond.setBounds(300, 0, 200, 200);
		panelSecond.setBackground(Color.RED);
		contner.add(panelSecond);
	}
}
```



Written by Nawaraj Prajapati (MCA From JNU)

JLabel:

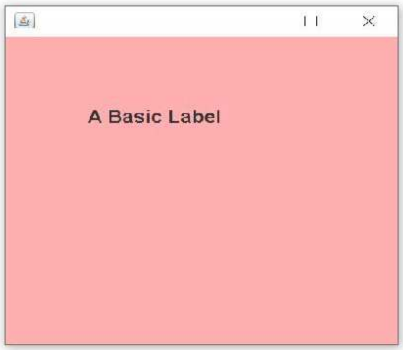
JLabel class is a component for placing text in a container. It is used to display a single line of read only text.

Method of JLabel:

setText()
setFont()

For example:

```
import java.awt.Color;
import java.awt.Container;
import java.awt.Font;
import javax.swing.JFrame;
import javax.swing.JLabel;
public class JLabelDemo {

        public static void main(String[] args) {

                JFrame frame = new JFrame ();
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setBounds (100,100,500,500);
                Container contner = frame.getContentPane();
                contner.setLayout(null);

                contner.setBackground (Color.pink);
                JLabel label=new JLabel("A Basic Label");
                Font f = new Font ("arial", Font.BOLD, 25);

                label.setFont (f);

                label.setBounds(100, 100, 300, 30);

                contner.add(label);
                frame.setVisible(true);
        }
}
```


A Basic Label

Written by Nawaraj Prajapati (MCA From JNU)

JTextField:

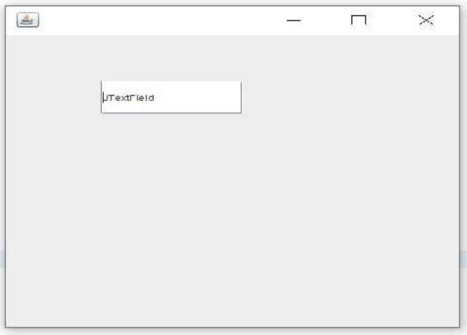
JTextField class is a text component that allows the displaying of a single line text.

Method of JTextField:

setText()
setFont()
setBackground()
setForeground()
setEditable()

For example:

```
import java.awt.Container;
import javax.swing.JFrame;
import javax.swing.JTextField;
public class TextFieldDemo {


        public static void main(String[] args) {

                JFrame frame = new JFrame();

                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setBounds(100, 100, 500, 500);
                Container contner = frame.getContentPane();
                contner.setLayout(null);

                JTextField textfield = new JTextField();
                textfield.setBounds(100, 70, 150, 50);
                textfield.setText("JTextField");

                contner.add(textfield);
                frame.setVisible(true);
        }
}
```



Written by Nawaraj Prajapati (MCA From JNU)

JPasswordField:

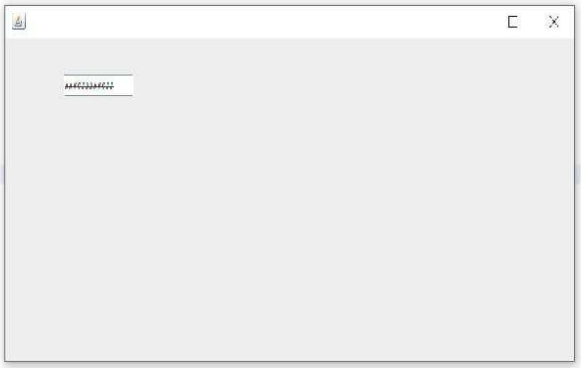
JPasswordField class is a text component specialized for password entry. For security reason, password field displays echo characters instead of the password itself. The default echo character is (*).

Method of JPasswordField

setEchoChar()

For example:

```
import java.awt.Container;
import javax.swing.JFrame;
import javax.swing.JPasswordField;

public class JPasswordDemo {

	public static void main(String[] args) {
		JFrame frame = new JFrame();
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setBounds(100, 100, 1000, 500);
		Container contner = frame.getContentPane();
		contner.setLayout(null);
		JPasswordField pass = new JPasswordField();
		pass.setBounds(100, 50, 120, 30);
		contner.add(pass);
		pass.setEchoChar('*');
		frame.setVisible(true);
	}
}
```



Written by Nawaraj Prajapati (MCA From JNU)

JTextArea

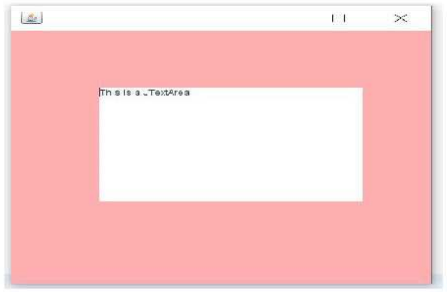
JTextArea class is a multi-line region that displays text. It allows the editing of multiple line text.

Method of JTextArea:

```
setText(String)
setFont(Font)
setEditable(boolean)
setLineWrap(boolean)
```

For example:

```
import java.awt.Color;
import java.awt.Container;
import javax.swing.JFrame;
import javax.swing.JTextArea;
public class JTextAreaDemo {

	public static void main(String[] args) {
		JFrame frame = new JFrame();
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setBounds(100, 100, 500, 500);

		Container contner = frame.getContentPane();
		contner.setLayout(null);
		contner.setBackground(Color.pink);

		JTextArea textarea=new JTextArea();
		textarea.setBounds(100, 100, 300, 200);
		contner.add(textarea);

		textarea.setText("This is a JTextArea");

		frame.setVisible(true);
	}
}
```



```
This is a JTextArea
```

Written by Nawaraj Prajapati (MCA From JNU)

JCheckBox:

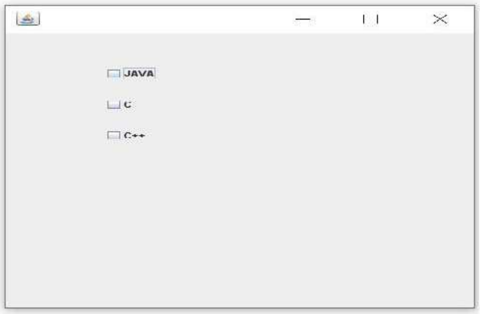
JCheckBox class is used to create a checkbox. It is used when we want to select only one option i.e true or false. When the checkbox is checked then its state is "on" (true) else it is "off"(false).

Method of JCheckBox:

setFont(Font)
setEnabled (boolean)
ButtonGroup()

For example:

```
import java.awt.Container;
import javax.swing.JCheckBox;
import javax.swing.JFrame;

public class JCheckBoxDemo {
        public static void main(String[] args) {
                JFrame frame = new JFrame();
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setBounds(100, 100, 500, 500);

                Container contner = frame.getContentPane();
                contner.setLayout(null);

                JCheckBox checkbox1=new JCheckBox("JAVA");
                JCheckBox checkbox2=new JCheckBox("C");
                JCheckBox checkbox3=new JCheckBox("C++");

                checkbox1.setBounds(100, 50, 120, 30);
                checkbox2.setBounds(100, 100, 120, 30);
                checkbox3.setBounds(100, 150, 120, 30);

                contner.add(checkbox1);
                contner.add(checkbox2);
                contner.add(checkbox3);

                frame.setVisible(true);
        }
}
```



Written by Nawaraj Prajapati (MCA From JNU)

JRadioButton:

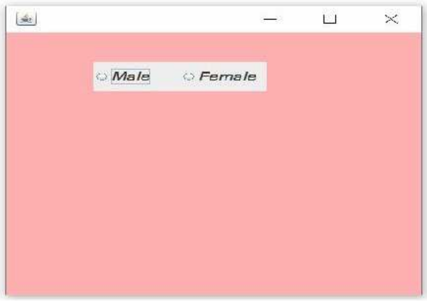
JRadioButton class is used to create a radio button. It is used to choose one option from multiple options.

Method of JRadioButton:

setFont(Font)
setEnabled(boolean)
ButtonGroup()
setSelected(boolean)

For example:

```
import java.awt.Color;
import java.awt.Container;
import java.awt.Font;
import javax.swing.ButtonGroup;
import javax.swing.JFrame;
import javax.swing.JRadioButton;

public class JRadioButtonDemo {
	public static void main(String[] args) {

		JFrame frame = new JFrame ();
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setBounds (100,100,500,500);

		Container contner = frame.getContentPane();
		contner.setLayout(null);

		contner.setBackground (Color.pink);

		JRadioButton radiobutton1=new JRadioButton("Male");
		radiobutton1.setBounds(100, 50, 100, 50);
		contner.add(radiobutton1);
		Font f = new Font ("arial", Font.ITALIC, 20);
		radiobutton1.setFont (f);

		JRadioButton radiobutton2=new JRadioButton("Female");
		radiobutton2.setBounds(200, 50, 100, 50);
		contner.add(radiobutton2);
		radiobutton2.setFont (f);

		ButtonGroup gender= new ButtonGroup();
		gender.add(radiobutton1);
		gender.add(radiobutton2);

		frame.setVisible(true);
	}
}
```



Written by Nawaraj Prajapati (MCA From JNU)

JButton:

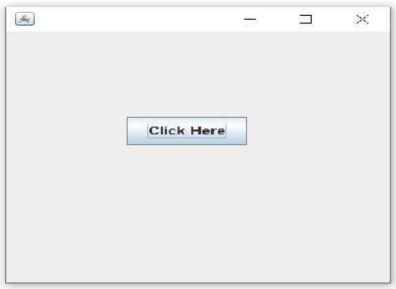
JButton class in Java is used to create push buttons that can be used to perform any ActionEvent whenever it is clicked. In Order to achieve event action, the ActionListener interface needs to be implemented. The Buttons component in Swing is similar to that of the AWT button component except that it can contain text, image or both.

Method of JButton:

- setFont()
- setText()
- setForeground()
- setBackground()
- setCursor()
- setEnabled()
- setVisible()

For example:

```
import java.awt.Container;
import java.awt.Cursor;
import java.awt.Font;
import javax.swing.JButton;
import javax.swing.JFrame;
public class JButtonDemo {

	public static void main(String[] args) {

		JFrame frame = new JFrame ();
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setBounds (100,100,500,500);
		Container contner = frame.getContentPane();
		contner.setLayout(null);

		JButton btn=new JButton("Click Here");
		btn.setSize(150, 50);
		btn.setLocation(150, 150);
		contner.add(btn);

		Font fnt=new Font("Arial", Font.BOLD, 20);
		btn.setFont(fnt);
		//btn.setText("OK");

		Cursor crsor=new Cursor(Cursor.HAND_CURSOR);
		btn.setCursor(crsor);

		//btn.setEnabled(false);
		//btn.setVisible(true);
		//btn.setVisible(false);

		frame.setVisible(true);
	}
}
```



Written by Nawaraj Prajapati (MCA From JNU)

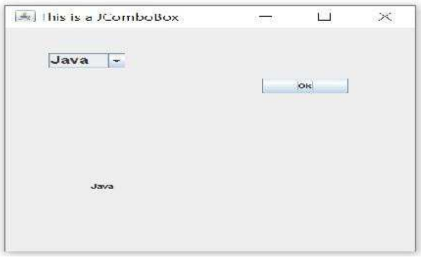
JComboBox:

JComboBox is represented by a popup menu that contains the list of elements and the user could select an option or element from that list.

Method of JComboBox:

setEditable(boolean)	setSelectedItem(String)	getSelectedItem()
setSelectedIndex(index)	setFont(Font)	getSelectedIndex()
addItem()	removeItem()	

For example:



```
import java.awt.Container;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
public class JComboBoxDemo {
        public static void main(String[] args) {
                JFrame frame = new JFrame("This is a JCombo        ");
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setBounds (100,100,500,500);
                Container contner = frame.getContentPane();
                contner.setLayout(null);

                String values[] = { "Java", "C++", "C", "PHP", "Android" };
                final JComboBox combobox = new JComboBox(values);
                combobox.setBounds(50, 50, 90, 30);
                contner.add(combobox);

                Font fnt=new Font("Arial", Font.BOLD, 20);
                combobox.setFont(fnt);

                JButton btn=new JButton("Ok");
                btn.setBounds(300, 100, 100, 30);
                contner.add(btn);

                final JLabel label=new JLabel();
                label.setBounds(100, 300, 100, 30);
                contner.add(label);

                btn.addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent arg0) {
                                String selectitem=(String)combobox.getSelectedItem();
                                label.setText(selectitem);
                        }
                });

                frame.setVisible(true);
        }    }
```

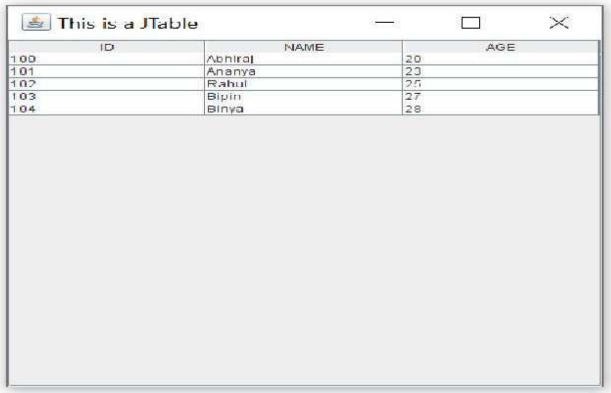
Written by Nawaraj Prajapati (MCA From JNU)

JTable:

The JTable class is used to display data in tabular form. It is composed of rows and columns.

For example:

```
import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JTable;

public class JTableDemo {


        public static void main(String[] args) {

                String [][]recod={

                                {"100","Abhiraj","20"},
                                {"101","Ananya","23"},
                                {"102","Rahul","25"},
                                {"103","Bipin","27"},
                                {"104","Binya","28"},
                };

                String []columnNames={"ID","NAME","AGE"};

                JTable table=new JTable(recod, columnNames);

                JFrame frame = new JFrame ();
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setBounds (100,100,500,500);

                frame.setTitle("This is a JTable");

                frame.add(new JScrollPane(table));

                frame.setVisible(true);


        }

}
```



This is a JTable

ID	NAME	AGE
100	Abhiraj	20
101	Ananya	23
102	Rahul	25
103	Bipin	27
104	Binya	28

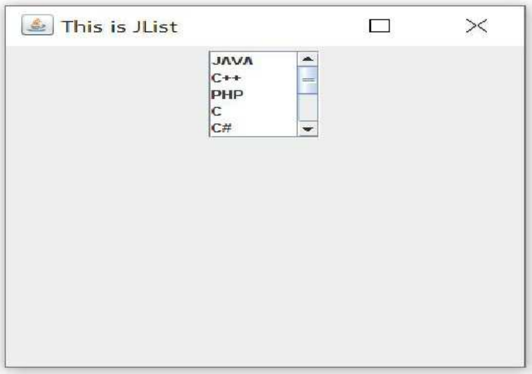
Written by Nawaraj Prajapati (MCA From JNU)

JList:

The object of JList class represents a list of text items. It is a component that displays a set of object and allows the user to select one or multiple items.

For example:

```
import javax.swing.JFrame;
import javax.swing.JList;
import javax.swing.JPanel;
import javax.swing.JScrollPane;

public class JListDemo extends JFrame {

                String[] items={
                                "JAVA","C++","PHP","C","C#",
                                "DRUPAL","PYTHON",
                                "JavaScript","HTML","CSS"
                        };
        public JListDemo() {

                setSize(400, 400);
                setDefaultCloseOperation(EXIT_ON_CLOSE);
                setLocationRelativeTo(null);
                setVisible(true);

                JList list=new JList(items);
                list.setVisibleRowCount(3);

                JPanel panel=new JPanel();
                panel.add(new JScrollPane(list));

                add(panel);
        }
        public static void main(String[] args) {

                new JListDemo();
        }
}
```



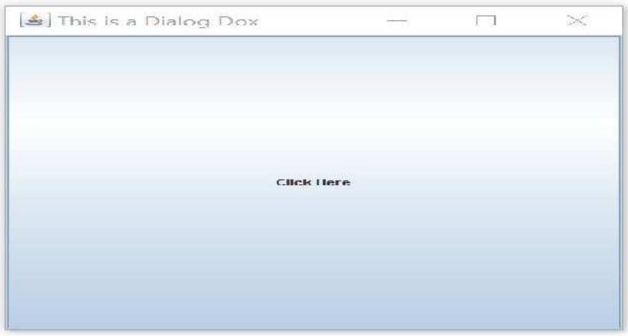
Written by Nawaraj Prajapati (MCA From JNU)

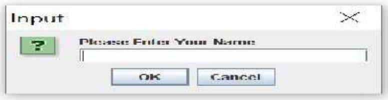
JOptionPane:

The JOptionPane class is used to provide standard dialog boxes such as message dialog box, confirm dialog box and input dialog box. These dialog boxes are used to display information or get input from the user.

For example:

```
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;

public class JOptionPaneDemo {
	public static void main(String[] args) {

		JFrame frame = new JFrame ();
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setBounds (100,100,500,500);
		frame.setTitle("This is a Dialog Dox");
		frame.setVisible(true);

		JButton btn=new JButton("Click Here");
		frame.add(btn);
		btn.addActionListener(new ActionListener() {
			@Override
			public void actionPerformed(ActionEvent arg0) {

				String name=JOptionPane.showInputDialog("Please Enter Your Name");
				if(name.length()>0){
					System.out.println("The Name is : "+name);
				}
			}
		});
	}
}
```





Written by Nawaraj Prajapati (MCA From JNU)

JMenuBar:

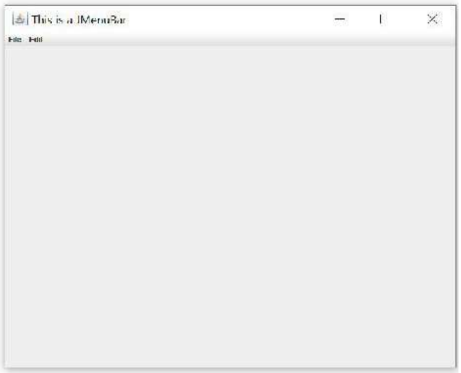
The JMenuBar class is used for displaying menubar on the frame. The JMenu Object is used for pulling down the components of the menu bar. The JMenuItem Object is used for adding the labelled menu item.

Steps to create a MenuBar:

1. create the objects of
   JManuBar
   JMenu
   JMenuItem
2. add menu items to the related menu
3. add menus to the menubar
4. add menubar to the JFrame

For example:

```
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
public class JMenuBarDemo {
	public static void main(String[] args) {
		JFrame frame = new JFrame ();
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setSize(700, 600);
		frame.setLocationRelativeTo(null);
		frame.setTitle("This is a JMenuBar");
		JMenuBar mbar=new JMenuBar();
		JMenu menu=new JMenu("File");
		JMenuItem mitem1=new JMenuItem("New");
		JMenuItem mitem2=new JMenuItem("Open");
		JMenuItem mitem3=new JMenuItem("Save");
		JMenuItem mitem4=new JMenuItem("Exit");
		menu.add(mitem1);
		menu.add(mitem2);
		menu.add(mitem3);
		menu.add(mitem4);
		mbar.add(menu);
		JMenu medit=new JMenu("Edit");
		JMenuItem mitem5=new JMenuItem("Undo");
		JMenuItem mitem6=new JMenuItem("Redo");
		medit.add(mitem5);
		medit.add(mitem6);
		mbar.add(medit);
		//menu.add(medit);
		frame.setJMenuBar(mbar);

		frame.setVisible(true);
	}

}
```



Written by Nawaraj Prajapati (MCA From JNU)

Event:

An event can be defined as changing the state of an object or behavior by performing actions. Actions can be a button click, cursor movement, key press through keyboard or page scrolling, etc.

The java.awt.event package can be used to provide various event classes.

Types of Events:

The events can be classified into two categories.

1. Foreground Events
2. Background Events

1. Foreground Events: Those events which require the direct interaction of user. They are generated as consequences of a person interacting with the graphical components in Graphical User Interface. For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page etc.

2. Background Events: Those events that require the interaction of end user are known as background events. Operating system interrupts, hardware or software failure, timer expires, an operation completion are the example of background events.

Components of Event Handling:

- **Event Handling**: Event Handling is process of responding to events that can occur at any time during execution of a program.
- **Event Source**: Event source is an object that generates an event.
- **Event Listener:** A listener is an object that listens to the event. A listener gets notified when an event occurs.

Event Classes and Interface:

The java.awt.event package provides many event classes and Listener interfaces for event handling.

Event Classes	Description	Listener Interface
ActionEvent	When a button is clicked or a list item is double-clicked, an ActionEvent is triggered.	ActionListener
MouseEvent	This event indicates a mouse action occurred in a component.	MouseListener
KeyEvent	The Key event is triggered when the character is entered using the keyboard.	KeyListener
ItemEvent	An event that indicates whether an item was selected or not.	ItemListener
TextEvent	When the value of a textarea or text field is changed.	TextListener
MouseWheelEvent	Generated when the mouse wheel is rotated.	MouseWheelListener
WindowEvent	The object of this class represents the change in the state of a window and are generated when the window is activated, deactivated, deiconified, iconified, opened or closed.	WindowListener
ComponentEvent	When a component is hidden, moved, resized, or made visible.	ComponentEventListener
ContainerEvent	When a component is added or removed from a container.	ContainerListener
AdjustmentEvent	Generated when scroll bar is manipulated	AdjustmentListener
FocusEvent	Generated when component gains or loses keyboard focus	FocusListener

Written by Nawaraj Prajapati (MCA From JNU)

Event Listener Interfaces:

ActionListener: This interface deals with the action events. Following is the event handling method available in the ActionListener interface.

void actionPerformed(ActionEvent ae)

MouseListener: This interface deals with five of the mouse events. Following are the event handling methods available in the MouseListener interface.

void mouseClicked(MouseEvent mouseEvent)

void mousePressed(MouseEvent mouseEvent)

void mouseReleased(MouseEvent mouseEvent)

void mouseEntered(MouseEvent mouseEvent)

void mouseExited(MouseEvent mouseEvent)

KeyListener: This interface deals with the key events. Following are the event handling methods available in the KeyListener interface.

void keyPressed(KeyEvent keyEvent)

void keyReleased(KeyEvent keyEvent)

void keyTyped(KeyEvent keyEvent)

ItemListener: This interface deals with the item event. Following is the event handling method available in the ItemListener interface.

void itemStateChanged(ItemEvent itemEvent)

TextListener: This interface deals with the text events. Following is the event handling method available in the TextListener interface.

void textValueChanged(TextEvent textEvent)

MouseWheelListener: This interface deals with the mouse wheel event. Following is the event handling method available in the MouseWheelListener interface.

void mouseWheelMoved(MouseWheelEvent mwe)

ComponentEventListener: This interface deals with the component events. Following are the event handling methods available in the ComponentEventListener interface.

void componentResized(ComponentEvent ce)

void componentMoved(ComponentEvent ce)

void componentShown(ComponentEvent ce)

void componentHidden(ComponentEvent ce)

Written by Nawaraj Prajapati (MCA From JNU)

WindowListener: This interface deals with seven of the window events. Following are the event handling methods available in the WindowListener interface.

```
void windowActivated(WindowEvent we)
void windowDeactivated(WindowEvent we)
void windowIconified(WindowEvent we)
void windowDeiconified(WindowEvent we)
void windowOpened(WindowEvent we)
void windowClosed(WindowEvent we)
void windowClosing(WindowEvent we)
```

ContainerListener: This interface deals with the events that can be generated on containers. Following are the event handling methods available in the ContainerListener interface.

```
void componentAdded(ContainerEvent ce)
void componentRemoved(ContainerEvent ce)
```

AdjustmentListener: This interface deals with the adjustment event generated by the scroll bar. Following is the event handling method available in the AdjustmentListener interface.

```
void adjustmentValueChanged(AdjustmentEvent ae)
```

FocusListener: This interface deals with focus events that can be generated on different components or containers. Following are the event handling methods available in the FocusListener interface.

```
void focusGained(FocusEvent fe)
void focusLost(FocusEvent fe)
```

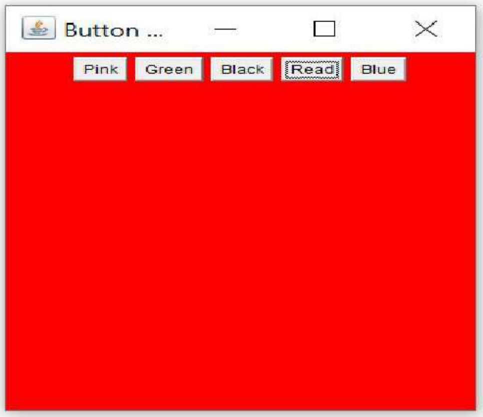
Steps in Event Handling:

- The User clicks the button and the event is generated.
- Now the object of concerned event class is created automatically and information about the source and the event get populated with in same object.
- Event object is forwarded to the method of registered listener class.
- The method is now get executed and returns.

Written by Nawaraj Prajapati (MCA From JNU)

Example of Action Listener:

```
public class ActionListenerButtonDemo extends Frame implements ActionListener {

        Button pink, green, black, read, blue;

        public ActionListenerButtonDemo() {
                FlowLayout flyout = new FlowLayout();
                setLayout(flyout);

                pink = new Button("Pink");
                green = new Button("Green");
                black = new Button("Black");
                read = new Button("Read");
                blue = new Button("Blue");

                pink.addActionListener(this);
                green.addActionListener(this);
                black.addActionListener(this);
                read.addActionListener(this);
                blue.addActionListener(this);

                add(pink);
                add(green);
                add(black);
                add(read);
                add(blue);

                setTitle("Button Action");
                setSize(350, 400);
                setVisible(true);
        }
        public static void main(String[] args) {
                ActionListenerButtonDemo obj = new ActionListenerButtonDemo();
        }
        @Override
        public void actionPerformed(ActionEvent ae) {
                String str = ae.getActionCommand();

                if (str.equals("Pink")) {
                        setBackground(Color.pink);
                } else if (str.equals("Green")) {
                        setBackground(Color.green);
```



Written by Nawaraj Prajapati (MCA From JNU)

```
		} else if (str.equals("Black")) {
			setBackground(Color.black);
		} else if (str.equals("Read")) {
			setBackground(Color.red);
		} else if (str.equals("Blue")) {
			setBackground(Color.blue);
		}
}}
```

Written by Nawaraj Prajapati (MCA From JNU)