**Syllabus**

**UNIT –I**

Swings, JavaFX and Event Handling: Swing: Introduction to swings, Comparison with AWT, Exploring Swing Components: JTextField, Jlabel, Swing buttons, JPasswordField, JTable, JComboBox, JList, JTree, JColorChooser, Dialogs and Swing Menus. Event Handling- The Delegation event model- Events, Event sources, Event Listeners, Event classes, Handling action ,mouse and keyboard events, Adapter classes, Inner classes, Anonymous Inner classes. GUI programming with JavaFX: JavaFX basic concepts, JavaFX Application Structure, JavaFX Controls and Event handling.

**UNIT –II**

Networking and Collection frame work: Networking: Networking API, Inet address, TCP/IP client sockets, URL, URL connection, HttpURL connection, Cookies, TCP/IP server sockets, Datagrams. Collections Frame work: Collection Interfaces, Collection Classes: Array Class,Vector Class, Stack Class, Dictionary class, Hash table Class. accessing using iterators, working with maps, comparators.

**UNIT – III**

Java Database Connectivity (JDBC): Introduction, JDBC Drivers, JDBC Architecture, JDBC Classes and Interfaces, Loading a Driver, Making a Connection, Execute SQL Statement, statement, prepared statement, callable statement, Retrieving Result, Getting Database Information, Scrollable and Updatable Resultset, Result Set Metadata.

**UNIT –IV**

Servlets: Servlet: Server-Side Java, Servlet Alternatives, Servlet Strengths, Servlet Architecture, Servlet Life Cycle, GenericServlet, HttpServlet, Exploring Servlet API, Handling HTTP Requests and Responses, Passing Parameters to Servlets, Retrieving Parameters, Session Tracking, Filters.

**UNIT – V**

Java Server Pages(JSP): Problem with Servlets, Life Cycle of JSP Page, JSP Processing, JSP Application Design with MVC, Setting Up the JSP Environment. JSP Directives, JSP Action elements, JSP Implicit Objects, JSP Form Processing, JSP Session and Cookies Handling, JSP Session Tracking JSP Database Access, JSP Standard Tag Libraries,JSP Custom Tag,JSP Expression Language, JSP Exception Handling, JSP XML Processing, JSTL.

**Introduction to GUI programming in Java:**

GUI applications are user friendly applications and they allow users to easily interact and use the application with out any external or specific training. The Java supports to develop GUI applications. There are two ways that we can develop GUI applications in java as follows

1. AWT

2. Swing

The AWT package comes with a collection of GUI components like label, button, text filed, text area, canvas, menubar etc…and the containers like panel, frame, Dialog etc… But due to the limitations of AWT java supposed to develop a new mechanism to build GUI applications which is known as Swings.

**Limitations of AWT:**

1. Platform dependent

2. Limited GUI Components

3. poor customization

4. complex layout management

5. performance issues

**Swings in Java:** The swing is a GUI toolkitthat is a part of theJava Foundation Classes (JFC). It is a lightweight toolkit. It has a rich set of controls and allows us to develop stand-alone applications or window-based applications.

**Features of Swings:**

**1.** **Platform Independent:** the swing components does not relay on OS GUI components and are written completely in java. The swing components behave consistently across different operating systems.

**2. Lightweight:** As the swing components are completely written in Java, they are lightweight and they don’t use any native system resources hence these components are light weight and fast in performance.

**3. MVC Architecture:** Swing components follows MVC (Model, View and Controller) Architecture which separates data, user interface and interaction or business logic.

**4. Layout managers:** The swings support additional GUI layouts like borderlayout, flowlayout, GridLayout etc… to build more beautiful GUI applications.

**5. Rich set of GUI controls:** The swings in java provides a wide range of controls like JButton, JTextField, Jpanel etc…

Package used for Swings in java: javax.swing

**Differences between AWT and Swing Components:**

|  |  |  |
| --- | --- | --- |
| **SNO** | **AWT** | **SWING** |
| **01** | **Platform Dependent** | **Platform Independent** |
| **02** | **Heavy Weight** | **Light Weight** |
| **03** | **Does not support pluggable look and feel** | **Supports pluggable look and feel** |
| **04** | **Less components** | **Rich set of components** |
| **05** | **Less controlling on UI** | **Flexibility in controlling UI** |
| **06** | **Does not follow MVC** | **Follows MVC** |
| **07** | **It is an API and part of AWT** | **It is part of JFC** |
| **08** | **Java.awt package is used** | **Javax.swing package is used** |
| **09** | **More code is needed to implement AWT Controls** | **Less code is enough to implement swing controls** |
| **10** | **Execution time is high compared with Swing** | **Execution time is less compared with Swing** |
| **11** | **Not suitable for user friendly applications** | **Suitable for user friendly applications** |
| **12** | **Low performance** | **High performance** |

**Exploring Javax.Swing:**

The javax.swing package is used for GUI applications and it has a rich set of classes, interface, enums. The most frequently useful classes, interface are as follows

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classes** | | | | |
| BoxLayout | ButtonGroup | GroupLayout | ImageIcon | InputMap |
| InputVerifier | JApplet | JButton | JCheckBox | JCheckBoxMenuItem |
| JColorChooser | JComboBox | JComponent | JDialog | JEditorPane |
| JFileChooser | JFormattedTextField | JFrame | JLabel | JList |
| JMenu | JMenubar | JOptionPane | JPanel | JPasswordField |
| JPopupMenu | JProgressBar | JRadioButton | JScrollBar | JSlider |
| JSpinner | JTabbedPane | JTable | JTextArea | JTextField |
| JToolTip | JTree | JWindow |  |  |

The class hierarchy of containers, components in swings is as shown in below figure



1. All the controls like JPanel, JLabel etc… are derived from JComponent class

2. JComponent is derived from Container

3. The Container is itself a component that can hold and organize other components

4. All the containers in swing are sub classes of JComponent

5. the package name is javax.swing.JComponent

**Containers:** In Swings we have different type of containers to hold controls as follows

**1. JFrame:** It is a main application window

**2.** **JDialog:** It gives a popup window

**3.** **JApplet:** Applet Container

**4. JPanel:** Which is generic lightweight container used with in top level containers

**JFrame:** JFrame is a fundamental class in Java's Swing library, used to create a top-level window for graphical user interface (GUI) applications. It serves as the primary container for other Swing components like buttons, text fields, labels, panels, and menus. The JFrame is inherited from java.awt.Frame and available in javax.swing package.

**Creating a window using JFrame:** By instantiating the JFrame class we can create a window and by using the methods available in JFrame class we can customize the window. The following methods will be used commonly while working with JFrame

|  |  |  |
| --- | --- | --- |
| **SNO** | **Method** | **Description** |
| 1 | JFrame() | Default constructor for creating an empty window without any title |
| 2 | JFrame(String title) | Creates a window using a title |
| 3 | setTitle(String Title) | Sets the title of the Frame |
| 4 | setSize(int W, int H) | Defines the size of the window |
| 5 | setVisible(Boolean) | Defines the visibility of the window |
| 6 | setLayout(LayoutManager manager) | Deines how to arrange GUI coponents on window |
| 7 | add(Component C) | Adds a GUI componentto Frame |
| 8 | remove(Component C) | Removes a component from Frame |
| 9 | validate() | Used for layout management |
| 10 | setIconImage(Image img) | Defines Icon Image |
| 11 | setLocation(int X,int Y) | Defines where to display on screen |
| 12 | setResizable(Boolean) | To control resizing of window |

Example Program:

import javax.swing.\*;

class create\_window

{

    public static void main(String args[])

    {

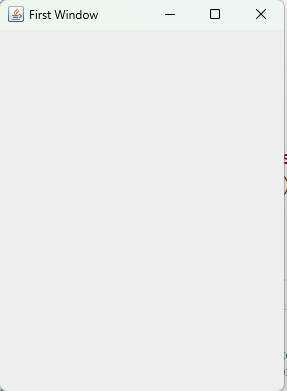
        JFrame frame=new JFrame("First Window");

        frame.setSize(300,400);

        frame.setVisible(true);

     }

}



**Adding GUI Components to Frame:**

1. Create the GUI Control

2. Customize the control

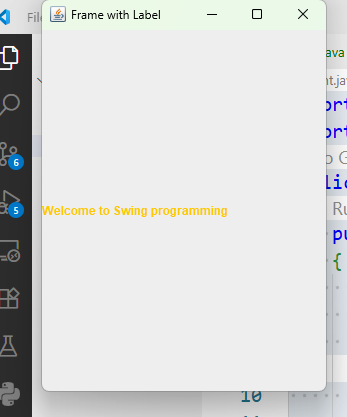
3. add to Jframe

Example: Adding a lable to JFrame window

import java.awt.Color;

import javax.swing.\*;

public class add\_component {

    public static void main(String args[])

    {

        JFrame myframe=new JFrame("Frame with Label");

        myframe.setLocation(50,0);

        myframe.setSize(300,400);

        JLabel lbl1=new JLabel("Welcome to Swing programming");

        lbl1.setForeground(Color.ORANGE);

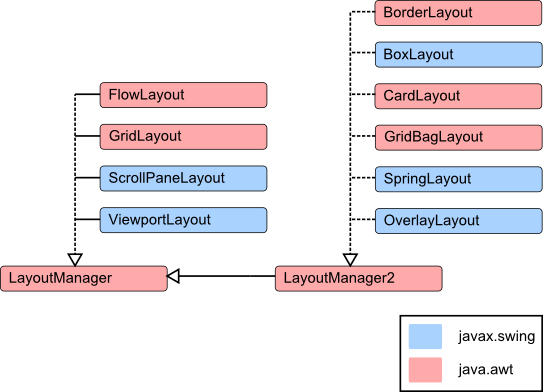
        myframe.add(lbl1);

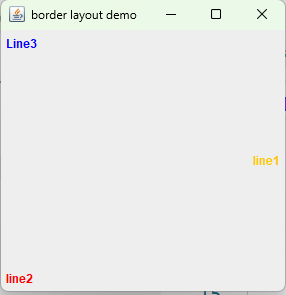
        myframe.setVisible(true);

    }

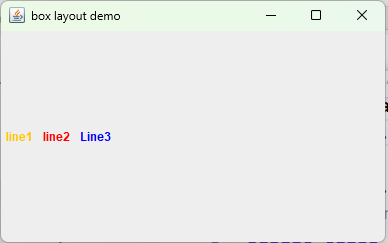
}

**Layout Manager:** The layout manager automatically positions all the components within the container. Even if you do not use the layout manager, the components are still positioned by the default layout manager. Java provides various layout managers to position the controls. Properties like size, shape, and arrangement varies from one layout manager to the other. When the size of the application window changes, the size, shape, and arrangement of the components also changes in response, i.e. the layout managers adapt to the dimensions of the application window.

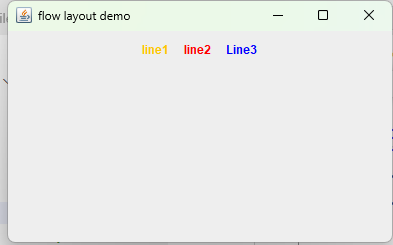


**BorderLayout:** A BorderLayout places components in five different areas like top, bottom, left, right, center. All extra space is placed in center area.

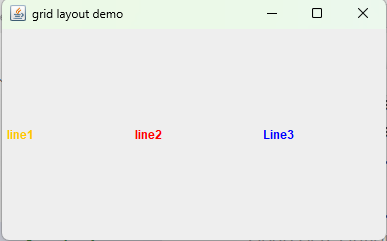
**Example program:** [clike here](https://github.com/kerlarajumca/advjava/blob/main/borderlayout.java)

**BoxLayout:** The BoxLayout class puts components in a single row or column. It respects the components' requested maximum sizes and also lets you align components.

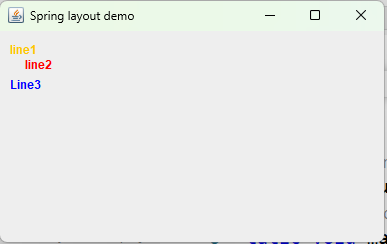
Example Program: [Click here](https://github.com/kerlarajumca/advjava/blob/main/box_layout.java)

**FlowLayout**: Flow layout is the commonly used layout. It is default layout used by JPanel. It is used to arrange components in a line or a row for example from left to right or from right to left. It arranges components in a line, if no space left remaining components goes to next line. Align property determines alignment of the components as left, right, center etc.

Example program: [click here](https://github.com/kerlarajumca/advjava/blob/main/flowlayout_example.java)

**GridLayout:** Grid layout arranges component in rectangular grid. It arranges component in cells and each cell has the same size. Components are placed in columns and rows. GridLayout(int rows, int columns) takes two parameters that is column are row.

Example: [click here](https://github.com/kerlarajumca/advjava/blob/main/grid_layout.java)



**SpringLayout:** SpringLayout is a flexible layout manager designed for use by GUI builders. It lets you specify precise relationships between the edges of components under its control. For example, you might define that the left edge of one component is a certain distance (which can be dynamically calculated) from the right edge of a second component.

**Example:** [click here](https://github.com/kerlarajumca/advjava/blob/main/spring_layout.java)

**JTextField:** JTextField is a JFC Class allows us to give single line input. It is a part of javax.swing package and it is a component. It is inherited from JTextComponent class.

**Syntax to create a text filed:**

JTextField tf=new JTextField() //creates empty text field

JTextField tf=new JTextField(10) //creates a text field with 10 columns

JTextfield tf=new JTextField(“Enter Name”) //Creates a text field with string as Enter Name

**Methods of JTextField:**

1. setColumns(int) 🡪 to define columns

2. setFont(font) 🡪 to Define font

3. int getColumns() 🡪 returns the number of columns

**JLabel:** JLabel is a class of java Swing. JLabel is used to display a short string or an image icon. JLabel can display text, image or both. JLabel is only a display of text or image and it cannot get focus. JLabel is inactive to input events such a mouse focus or keyboard focus. By default, labels are vertically centered but the user can change the alignment of label.

**Syntax:**

1. JLabel() 🡪 Empty Label

2. JLabel(String) 🡪 Label with String

3. JLabel(Icon) 🡪 Label with Icon

4. JLabel(String, Icon, Int) 🡪 Label with String, icon and alignment

**Methods:**

1. getIcon() 🡪 Returns icon

2. setIcon(Icon) 🡪 sets icon

3. getText() 🡪 Gets text

4. setText(String) 🡪 Sets Text

**JButton:** JButton in Java Swing is a fundamental graphical user interface (GUI) component that represents a push button. When a user clicks a JButton, it typically triggers an action within the application. The JButton extends AbstractButton and Implements Accessible.

**Syntax:**

1. JButton() 🡪 Empty button

2. JButton(Icon) 🡪 Button with icon

3. JButton(Action) 🡪 Button with Action

4. JButton(String) 🡪 Button with Text

5. JButton(String,Icon) 🡪 Button with string and Icon

**Example Program:**

import java.awt.FlowLayout;

import java.awt.event.\*;

import java.awt.\*;

import javax.swing.\*;

public class lbltextdemo implements ActionListener{

    public static void main(String args[])

    {

        JFrame jf=new JFrame("text demo");

        JLabel lbl1=new JLabel("Enter your name");

        JTextField txt1=new JTextField(30);

        JButton btn1=new JButton("Click me");

        btn1.setSize(50,50);

        btn1.addActionListener(new ActionListener(){

             public void actionPerformed(ActionEvent e) {

            String s1=txt1.getText();

            lbl1.setText("Hello Mr"+s1);

         }

        });

        jf.setSize(400,250);

        jf.setLayout(new FlowLayout());

        jf.add(lbl1);

        jf.add(txt1);

        jf.add(btn1);

        jf.setVisible(true);

    }

}