Introduction to Swing in Java

What is Swing?

Swing is a **GUI (Graphical User Interface) toolkit** in Java that provides a rich set of components for building user-friendly desktop applications. It is a part of **Java Foundation Classes (JFC)** and is used to create windows, buttons, text fields, menus, tables, and other UI elements.

Swing is an **extension of AWT (Abstract Window Toolkit)**, providing more powerful, flexible, and customizable components. It is built on **top of AWT** but operates independently of native platform GUI elements, making it **platform-independent**.

Key Features of Swing

Feature	Description
Lightweight	Swing components are written in Java and do not rely on OS-specific components, making them more efficient.
Pluggable Look and Feel	Swing allows changing the UI theme dynamically (e.g., Windows, Nimbus, Metal, etc.).
Rich Set of Components	Provides advanced UI elements like JTree, JTable, JTabbedPane, etc.
Platform-Independent	Swing applications run on **Windows, macOS, Linux**, etc., without modifications.
MVC Architecture	Follows **Model-View-Controller (MVC)** architecture for better separation of UI and logic.
Event-Driven Programming	Uses listeners to handle user interactions like button clicks and keyboard input.
Customizable Components	Components can be customized using colors, fonts, borders, and images.
Uses Java2D API	Enables smooth rendering of components with graphics and animations.
Supports Multithreading	Swing supports multi-threaded execution for better UI performance.

Difference Between AWT and Swing

Feature	**AWT (Abstract Window Toolkit)**	**Swing**
Package	java.awt	javax.swing
Components	Uses **native OS components**	Uses **Java-based lightweight components**
Performance	Slower because it depends on OS UI	Faster due to pure Java implementation
Look and Feel	OS-dependent (varies on Windows, macOS, etc.)	Customizable with different themes
Customization	Limited customization	Highly customizable
Event Handling	Uses **event listeners**	Uses an improved event model
Threading	**Single-threaded**, leading to potential UI freezes	Supports **multithreading**, preventing UI lags
Flexibility	Limited control over component appearance	Full control over appearance using Java2D API
Support for Advanced UI	Provides **basic UI elements** like Button, Checkbox, TextField	Provides **rich UI elements** like JTable, JTree, JTabbedPane

Conclusion:

- **AWT is simpler** but lacks advanced UI elements.
- **Swing is more powerful, flexible, and customizable**, making it suitable for modern applications.

Why Choose Swing Over AWT?

✅ **More powerful and feature-rich** than AWT
✅ **Supports pluggable Look & Feel** (Change UI style dynamically)
✅ **Lightweight and fast** (No dependency on native OS UI)
✅ **Customizable components** with icons, borders, and themes
✅ **Better event handling and multi-threading support**

Swing Components

1. Icons and Labels (JLabel & Icon)

JLabel:

- Used to display **static text** or **images** (icons).
- Cannot be edited by the user.
- Can display text, icon, or both.
- Supports alignment, font, and color customization.

Icon:

- Interface used for images in Swing (commonly ImageIcon is used).
- Icons are often used in buttons or labels.

Key Points:

- JLabel(String text) – Creates label with text.
- JLabel(Icon icon) – Creates label with image.
- setIcon(Icon) – Sets an icon for the label.
- setText(String) – Sets text of label.

2. TextField (JTextField)

JTextField:

- Allows users to **input a single line** of text.
- Supports text editing, cursor navigation, and selection.
- Often used for forms, login fields, etc.

Key Points:

- JTextField(int columns) – Creates text field with specified width.
- getText() – Retrieves text entered.
- setText(String) – Sets text.
- addActionListener() – To respond to Enter key.

3. ComboBox (JComboBox)

JComboBox:

- A **drop-down list** from which a user can choose one item.
- Can be editable or read-only.
- Supports item selection and action handling.

Key Points:

- JComboBox(String[] items) – Creates drop-down with items.
- getSelectedItem() – Gets selected value.
- setSelectedIndex(int) – Selects item by index.
- addActionListener() – Handles selection.

4. Button (JButton)

JButton:

- Represents a **clickable button**.
- Triggers **actions or events**.
- Can contain text, icons, or both.

Key Points:

- JButton(String text) – Creates button with text.
- addActionListener() – Responds to clicks.
- setEnabled(boolean) – Enables/disables button.
- Supports styling, icons, tooltips.

5. CheckBox (JCheckBox)

JCheckBox:

- Allows **multiple selections**.
- Each box can be **checked/unchecked** independently.
- Used for settings, options, and selections.

Key Points:

- JCheckBox(String label) – Creates checkbox with label.

- isSelected() – Returns true if checked.
- setSelected(boolean) – Sets checked/unchecked.

6. RadioButton (JRadioButton)

JRadioButton:

- Allows **only one selection** among a group.
- Must be grouped using **ButtonGroup**.
- Common in forms (e.g., gender selection).

Key Points:

- JRadioButton(String label) – Creates radio button.
- ButtonGroup – Groups multiple radio buttons.
- isSelected() – Checks if selected.

Summary Table:

Component	**Class**	**Key Method**	**Purpose**
Label	JLabel	setText(), setIcon()	Display text/image
Icon	ImageIcon	N/A	Image to display in GUI
Text Field	JTextField	getText(), setText()	Input single-line text
Combo Box	JComboBox	getSelectedItem()	Drop-down list
Button	JButton	addActionListener()	Perform action on click
Check Box	JCheckBox	isSelected()	Multi-option selection
Radio Button	JRadioButton	isSelected(), ButtonGroup	Single-option selection

Advance Swing Components

Components:

1. **Tabbed Pane (JTabbedPane)**
2. **Scroll Pane (JScrollPane)**
3. **Tree (JTree)**
4. **Table (JTable)**
5. **Progress Bar (JProgressBar)**
6. **Tool Tip**

1. Tabbed Pane (JTabbedPane)

Description:

- Used to create **multiple tabs**, each containing different components.
- Helps organize GUI elements into **sections**.

JTabbedPane is a GUI(Graphical User Interface) component in the Java Swing library that allows you to create a tabbed pane interface. A tabbed pane is a container that can store and organize multiple components into distinct tabs. It contains a collection of tabs. When you click on a tab, only data related to that tab will be displayed. JTabbedPane comes under the Java Swing package.

Constructors used in JTabbedPane

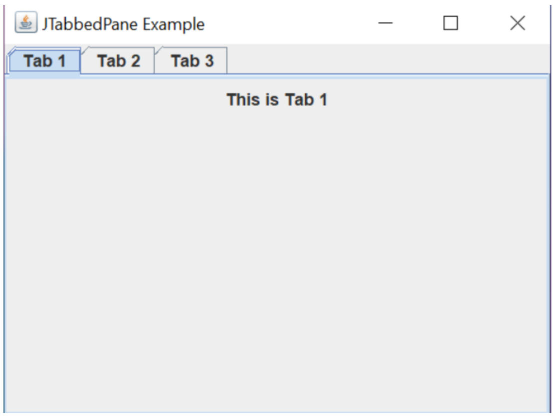
Constructor	**Description**
JTabbedPane()	The JTabbedPane class in Java Swing has a default, no-argument constructor called JTabbedPane(). This constructor initializes an empty tabbed pane with no tabs and no initial content when a JTabbedPane is created.
JTabbedPane(int tabPlacement)	The JTabbedPane(int tabPlacement) constructor allows to creation of a JTabbedPane with a defined initial location for the tabs. The tab placement option specifies whether the tabs appear at the top, bottom, left, or right of the tabbed pane.

Some commonly used methods of the JTabbedPane

Method	**Description**
addTab(String title, Component component)	Creates a new tab with the given title and content.
removeTabAt(int index)	Removes the tab at the given index.
getTabCount()	Returns the number of tabs present in the JTabbedPane.
setSelectedIndex(int index)	Sets the chosen tab to the index given.
getSelectedIndex()	Returns the index of the currently selected tab.

The classes from which JTabbedPane methods are inherited

- java.awt.Container
- javax.swing.JComponent
- javax.swing.JTabbedPane
- javax.swing.JContainer
- **Output:**



```
This is Tab 1
```

- Window titled **TabbedPane Example**.
- Tabs labeled **Tab 1** and **Tab 2**.
- Clicking each tab shows **different content**.

2. Scroll Pane (JScrollPane)

Description:

- Provides **scrollbars** for components like text areas, tables, etc.
- Useful when content is **larger than the viewable area**.

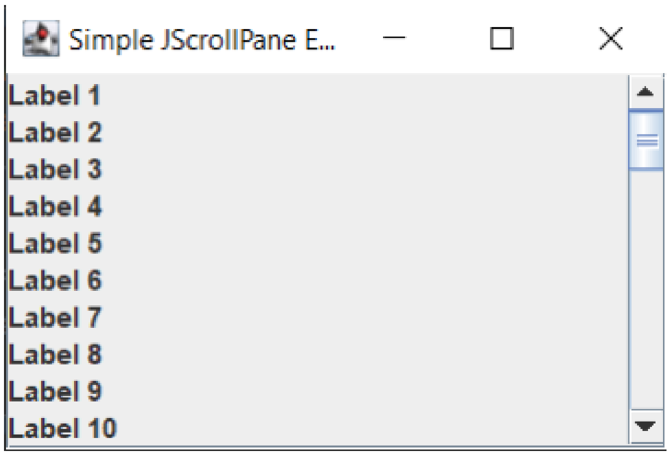
Key Methods:

- new JScrollPane(Component comp) – Wraps component.
- Automatically adds **vertical/horizontal scrollbars** as needed.

Java **JScrollPane** is a component in the Java Swing library that provides a scrollable view of another component, usually a JPanel or a JTextArea. it provides a scrolling functionality to the display for which the size changes dynamically. It is useful to display the content which exceeds the visible area of the window. In this article, we are going to see some constructors, methods, and examples of **JScrollPane**.

Methods of JScrollPane

Methods	**Description**
void setVerticalScrollBarPolicy(int vertical)	Sets the vertical scrollbar policy
void setHorizontalScrollBarPolicy(int horizontal)	Sets the horizontal scrollbar policy



Label 1
Label 2
Label 3
Label 4
Label 5
Label 6
Label 7
Label 8
Label 9
Label 10

3. Tree (JTree)

Description:

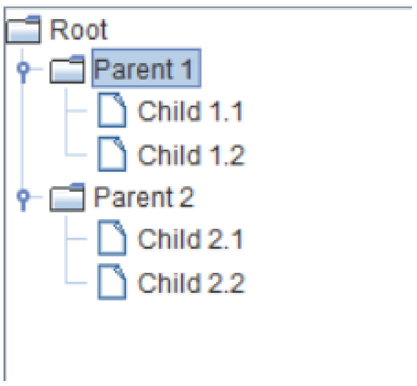
- Displays data in a **hierarchical structure** (like folders/files).
- Each item is a **node**.

Key Methods:

- new JTree() – Creates a default tree.
- Supports **expand/collapse** actions.

The **JTree** is a type of GUI(Graphic User Interface) that displays information in a hierarchical way. This intricate component part provides a quite elegant substance of representing relationships among elements in a tree-like structure. In this exploration, we'll delve into the essence of the JTree class, examining its declaration, normally used constructors and examples.

JTree()	This constructor creates a JTree with a sample model. It serves as a quick way to initialize a tree structure without specifying a custom model.
JTree(Object[] value)	JTree is created with each element of the specified array becoming a child of a new root node. This constructor is useful when you want to build a tree structure based on an array of values.
JTree(TreeNode root)	JTree is created with the specified TreeNode as its root. This allows you to define a custom structure for your tree by providing a root node explicitly.



4. Table (JTable)

Description:

- Displays data in **rows and columns** like a spreadsheet.
- Allows **editing, selection, and scrolling**.

Key Methods:

- new JTable(Object[][] data, Object[] columnNames) – Creates a table.
- Used with JScrollPane for scrolling.

The JTable class is a part of Java Swing Package and is generally used to display or edit two-dimensional data that is having both rows and columns. It is similar to a spreadsheet. This arranges data in a tabular form.

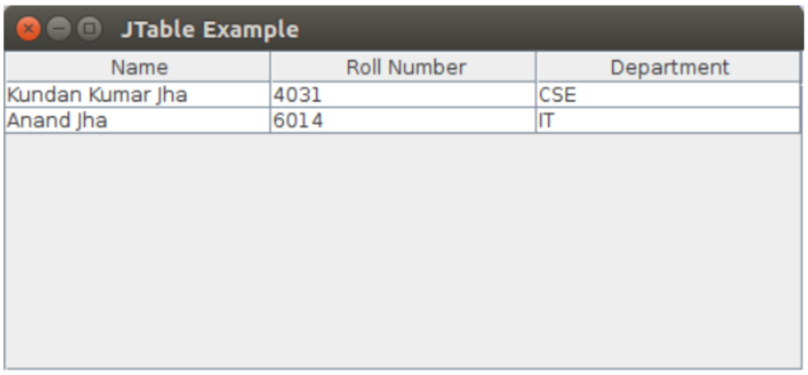
Constructors in JTable:

1. **JTable():** A table is created with empty cells.
2. **JTable(int rows, int cols):** Creates a table of size rows * cols.
3. **JTable(Object[][] data, Object []Column):** A table is created with the specified name where []Column defines the column names.

Functions in JTable:

1. **addColumn(TableColumn []column) :** adds a column at the end of the JTable.
2. **clearSelection() :** Selects all the selected rows and columns.
3. **editCellAt(int row, int col) :** edits the intersecting cell of the column number col and row number row programmatically, if the given indices are valid and the corresponding cell is editable.
4. **setValueAt(Object value, int row, int col) :** Sets the cell value as 'value' for the position row, col in the JTable.

Out put:


JTable Example

Name	Roll Number	Department
Kundan Kumar Jha	4031	CSE
Anand Jha	6014	IT

5. Progress Bar (JProgressBar)

Description:

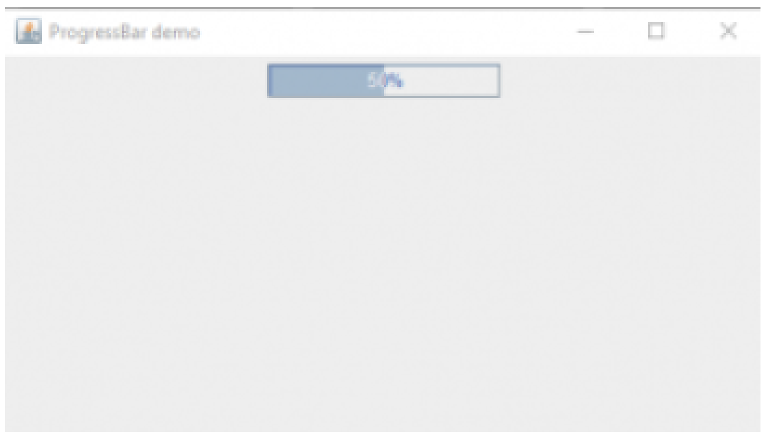
- Shows **progress of a task** (e.g., file download).
- Can be **determinate** (known progress) or **indeterminate** (unknown).

Key Methods:

- setValue(int) – Set current progress.
- setStringPainted(true) – Show percent

JProgressBar is a part of Java Swing package. JProgressBar visually displays the progress of some specified task. JProgressBar shows the percentage of completion of specified task.The progress bar fills up as the task reaches it completion. In addition to show the percentage of completion of task, it can also display some text

1. **JProgressBar()** : creates an progress bar with no text on it;
2. **JProgressBar(int orientation)** : creates an progress bar with a specified orientation. if SwingConstants.VERTICAL is passed as argument a vertical progress bar is created, if SwingConstants.HORIZONTAL is passed as argument a horizontal progress bar is created.
3. **JProgressBar(int min, int max)** : creates an progress bar with specified minimum and maximum value.
4. **JProgressBar(int orientation, int min, int max)** : creates an progress bar with specified minimum and maximum value and a specified orientation.if SwingConstants.VERTICAL is passed as argument a vertical progress bar is created, if SwingConstants.HORIZONTAL is passed as argument a horizontal progress bar is created.



6. Tool Tip

Description:

- Small popup text when mouse hovers over a component.
- Used for **user guidance**.

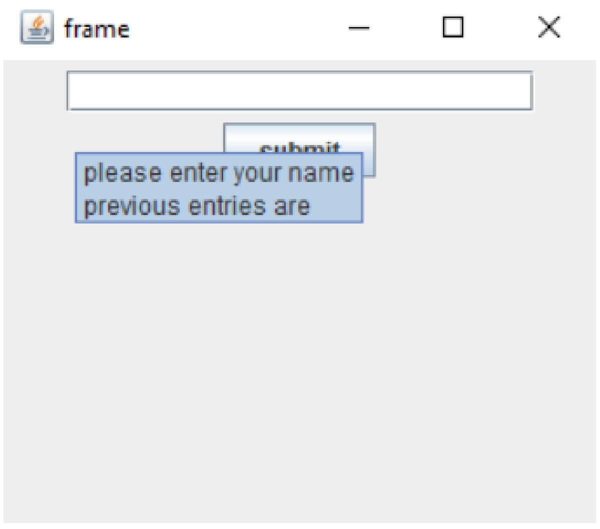
Key Method:

- setToolTipText(String text) – Assign tooltip to a component.

We can add tooltip text to almost all the components of Java Swing by using the following method setToolTipText(String s). This method sets the tooltip of the component to the specified string s. When the cursor enters the boundary of that component a popup appears and text is displayed.

Methods used:

1. getToolTipText() : returns the tooltip text for that component .
2. setToolTipText(String s) : sets the tooltip text for the component .
3. getToolTipText(MouseEvent e): returns the same value returned by getToolTipText(). Multi-part components such as JTabbedPane, JTable, and JTree override this method to return a string associated with the mouse event location.
4. getToolTipLocation(MouseEvent e) : Returns the location (in the receiving component's coordinate system) where the upper left corner of the component's tool tip appears.


frame
submit
please enter your name
previous entries are