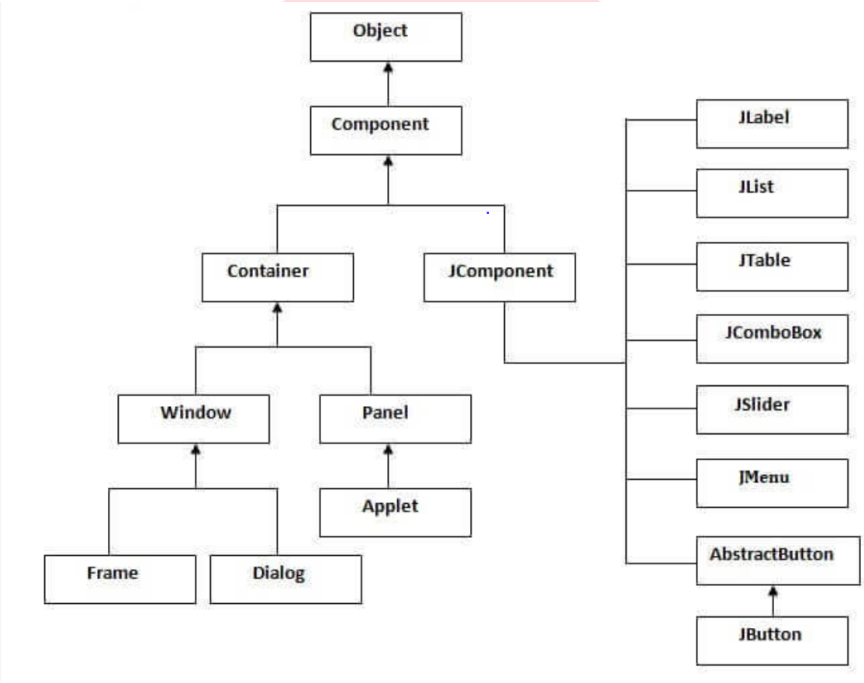
Unit 02: Swings

Java Swing is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

The Java Foundation Classes (JFC) are a set of GUI components which simplify the development of desktop applications.

Unlike AWT, Java 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, JColorChooser etc.



Difference between AWT and Swing

AWT	Swing
AWT stands for Abstract window toolkit .	Swing is also called as JFC(java Foundation classes). It is part of oracle's JFC.
AWT components require java.awt package	Swing components requires javax.swing package

AWT provides comparatively less powerful components.	Swing provides more powerful components.
MVC pattern is not supported by AWT.	MVC pattern is supported by Swing.
The components of Java AWT are platform dependent.	The components of Java Swing are platform independent.
The execution time of AWT is more than Swing.	The execution time of Swing is less than AWT.
Java AWT has comparatively less functionality as compared to Swing.	Java Swing has more functionality as compared to AWT.
The components of Java AWT are heavy weighted.	The components of Java Swing are light weighted.
Java AWT is an API to develop GUI applications in Java	Swing is a part of Java Foundation Classes and is used to create various applications.

Methods of Swing:

public void add(Component c) - add a component on another component.
public void setSize(int width,int height) -sets size of the component.
public void setLayout(LayoutManager m) - sets the layout manager for the component.
public void setVisible(boolean b) - sets the visibility of the component. It is by default false.

 # JButton

public class JButton extends AbstractButton implements Accessible

Constructor

JButton() - Creates a button with no set text or icon.
JButton(Icon icon) - Creates a button with an icon.
JButton(String text) - Creates a button with text.
JButton(String text, Icon icon) - Creates a button with initial text and an icon.

Methods

void setText(String s) - It is used to set specified text on button
String getText() - It is used to return the text of the button.
void setEnabled(boolean b) - It is used to enable or disable the button.
void setIcon(Icon b) - It is used to set the specified Icon on the button.
Icon getIcon() - It is used to get the Icon of the button.
void setMnemonic(int a) - It is used to set the mnemonic on the button.
void addActionListener(ActionListener a) - It is used to add the action listener to this object.

#ImageIcon
public class ImageIcon extends Object implements Icon, Serializable, Accessible

Constructor
ImageIcon() - Creates an uninitialized image icon.
ImageIcon(byte[] imageData) - Creates an ImageIcon from an array of bytes which were read from an image file containing a supported image format, such as GIF, JPEG, or (as of 1.3) PNG.
ImageIcon(byte[] imageData, String description) - Creates an ImageIcon from an array of bytes which were read from an image file containing a supported image format, such as GIF, JPEG, or PNG.
ImageIcon(Image image) - Creates an ImageIcon from an image object.
ImageIcon(Image image, String description) - Creates an ImageIcon from the image.
ImageIcon(String filename) - Creates an ImageIcon from the specified file.
ImageIcon(String filename, String description) - Creates an ImageIcon from the specified file.
ImageIcon(URL location) - Creates an ImageIcon from the specified URL.
ImageIcon(URL location, String description) - Creates an ImageIcon from the specified URL.

Methods
String getDescription() - Gets the description of the image.
int getIconHeight() - Gets the height of the icon.
int getIconWidth() - Gets the width of the icon.
Image getImage() - Returns this icon's Image.
void setDescription(String description) - Sets the description of the image.
void setImage(Image image) - Sets the image displayed by this icon.

#JLabel
public class JLabel extends JComponent implements SwingConstants, Accessible

Constructor
JLabel() - Creates a JLabel instance with no image and with an empty string for the title.
JLabel(String s) - Creates a JLabel instance with the specified text.
JLabel(Icon i) - Creates a JLabel instance with the specified image.
JLabel(String s, Icon i, int horizontalAlignment) - Creates a JLabel instance with the specified text, image, and horizontal alignment.

Methods
String getText() - It returns the text string that a label displays.
void setText(String text) - It defines the single line of text this component will display.

void setHorizontalAlignment(int alignment) - It sets the alignment of the label's contents along the X axis.
Icon getIcon() - It returns the graphic image that the label displays.
void setIcon(Icon i) - sets graphic image
int getHorizontalAlignment() - It returns the alignment of the label's contents along the X axis.

#JTextField
public class JTextField extends JTextComponent implements SwingConstants

Constructor
JTextField() - Creates a new TextField
JTextField(String text) - Creates a new TextField initialized with the specified text.
JTextField(String text, int columns) - Creates a new TextField initialized with the specified text and columns.
JTextField(int columns) - Creates a new empty TextField with the specified number of columns.

Methods
void addActionListener(ActionListener l) - It is used to add the specified action listener to receive action events from this textfield.
Action getAction() - It returns the currently set Action for this ActionEvent source, or null if no Action is set.
void setFont(Font f) - It is used to set the current font.
void removeActionListener(ActionListener l) - It is used to remove the specified action listener so that it no longer receives action events from this textfield.

#JComboBox
public class JComboBox extends JComponent implements ItemSelectable, ListDataListener, ActionListener, Accessible

Constructor
JComboBox() - Creates a JComboBox with a default data model.
JComboBox(Object[] items) - Creates a JComboBox that contains the elements in the specified array.
JComboBox(Vector<?> items) - Creates a JComboBox that contains the elements in the specified Vector.

Methods
void addItem(Object anObject) - It is used to add an item to the item list.
void removeItem(Object anObject) - It is used to delete an item to the item list.

void removeAllItems() - It is used to remove all the items from the list.
void setEditable(boolean b) - It is used to determine whether the JComboBox is editable.
void addActionListener(ActionListener a) - It is used to add the ActionListener.
void addItemListener(ItemListener i) - It is used to add the ItemListener.

#JRadioButton

Constructor
JRadioButton() - Creates an unselected radio button with no text.
JRadioButton(String s) - Creates an unselected radio button with specified text.
JRadioButton(String s, boolean selected) - Creates a radio button with the specified text and selected status.

Methods
void setText(String s) - It is used to set specified text on button.
String getText() - It is used to return the text of the button.
void setEnabled(boolean b) - It is used to enable or disable the button.
void setIcon(Icon b) - It is used to set the specified Icon on the button.
Icon getIcon() - It is used to get the Icon of the button.
void setMnemonic(int a) - It is used to set the mnemonic on the button.
void addActionListener(ActionListener a) - It is used to add the action listener to this object.

#JCheckBox
public class JCheckBox extends JToggleButton implements Accessible

Constructor
JCheckBox() - Creates an initially unselected check box button with no text, no icon.
JChechBox(String s) - Creates an initially unselected check box with text.
JCheckBox(String text, boolean selected) - Creates a check box with text and specifies whether or not it is initially selected.

#JTabbedPane
public class JTabbedPane extends JComponent implements Serializable, Accessible, SwingConstants

Construtor
JTabbedPane() - Creates an empty TabbedPane with a default tab placement of JTabbedPane.Top.
JTabbedPane(int tabPlacement) - Creates an empty TabbedPane with a specified tab placement.

JTabbedPane(int tabPlacement, int tabLayoutPolicy) - Creates an empty TabbedPane with a specified tab placement and tab layout policy.

Methods

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.

Constants

tabLayoutPolicy: Determines how the tabs are laid out. It can be one of the following:
JTabbedPane.WRAP_TAB_LAYOUT: Tabs will wrap to the next line if they don't fit.
JTabbedPane.SCROLL_TAB_LAYOUT: Tabs will be placed in a scrollable area if there are too many tabs to fit in the available space.

tabPlacement is an integer that determines where the tabs will be placed. It can be one of the following:
JTabbedPane.TOP: Tabs are placed at the top (default).
JTabbedPane.BOTTOM: Tabs are placed at the bottom.
JTabbedPane.LEFT: Tabs are placed on the left.
JTabbedPane.RIGHT: Tabs are placed on the right.

Program

```
import javax.swing.*;
public class TabbedPaneExample {
JFrame f;
TabbedPaneExample(){
    f=new JFrame();
    JTextArea ta=new JTextArea(200,200);
    JPanel p1=new JPanel();
    p1.add(ta);
    JPanel p2=new JPanel();
    JPanel p3=new JPanel();
    JTabbedPane tp=new JTabbedPane();
    tp.setBounds(50,50,200,200);
    tp.add("main",p1);
    tp.add("visit",p2);
    tp.add("help",p3);
```

```
    f.add(tp);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
}
public static void main(String[] args) {
    new TabbedPaneExample();
}}
```

#**JScrollPane**

Constructor

JScrollPane() It creates a scroll pane. The Component parameter, when present, sets the scroll pane's client. The two int parameters, when present, set the vertical and horizontal scroll bar policies (respectively).
JScrollPane(Component)
JScrollPane(int, int)
JScrollPane(Component, int, int)

Methods

void setColumnHeaderView(Component) - It sets the column header for the scroll pane.
void setRowHeaderView(Component) - It sets the row header for the scroll pane.
void setCorner(String, Component) - It sets or gets the specified corner.
The int parameter specifies which corner and must be one of the following constants defined in ScrollPaneConstants: UPPER_LEFT_CORNER, UPPER_RIGHT_CORNER, LOWER_LEFT_CORNER, LOWER_RIGHT_CORNER, LOWER_LEADING_CORNER, LOWER_TRAILING_CORNER, UPPER_LEADING_CORNER, UPPER_TRAILING_CORNER.
Component getCorner(String)
void setViewportView(Component) - Set the scroll pane's client.

JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED: The vertical scroll bar will appear only if the content is larger than the viewport in the vertical direction. This is the default behavior.
JScrollPane.VERTICAL_SCROLLBAR_ALWAYS: The vertical scroll bar will always be visible, regardless of whether the content is larger than the viewport.
JScrollPane.VERTICAL_SCROLLBAR_NEVER: The vertical scroll bar will never be visible, even if the content is larger than the viewport. This can be useful if you want to prevent the scroll bar from appearing under any circumstances.

JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED: The horizontal scroll bar will appear only if the content is larger than the viewport in the horizontal direction. This is the default behavior.
JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS: The horizontal scroll bar will always be visible, regardless of whether the content is larger than the viewport.
JScrollPane.HORIZONTAL_SCROLLBAR_NEVER: The horizontal scroll bar will never be visible, even if the content is larger than the viewport.

#**JProgressBar**

Constructor

JProgressBar() - It is used to create a horizontal progress bar but no string text.
JProgressBar(int min, int max) - It is used to create a horizontal progress bar with the specified minimum and maximum value.
JProgressBar(int orient) - It is used to create a progress bar with the specified orientation, it can be either Vertical or Horizontal by using SwingConstants.VERTICAL and SwingConstants.HORIZONTAL constants.
JProgressBar(int orient, int min, int max) - It is used to create a progress bar with the specified orientation, minimum and maximum value.

Methods

void setStringPainted(boolean b) - It is used to determine whether string should be displayed.
void setString(String s) - It is used to set value to the progress string.
void setOrientation(int orientation) - It is used to set the orientation, it may be either vertical or horizontal by using SwingConstants.VERTICAL and SwingConstants.HORIZONTAL constants.
void setValue(int value) - It is used to set the current value on the progress bar.