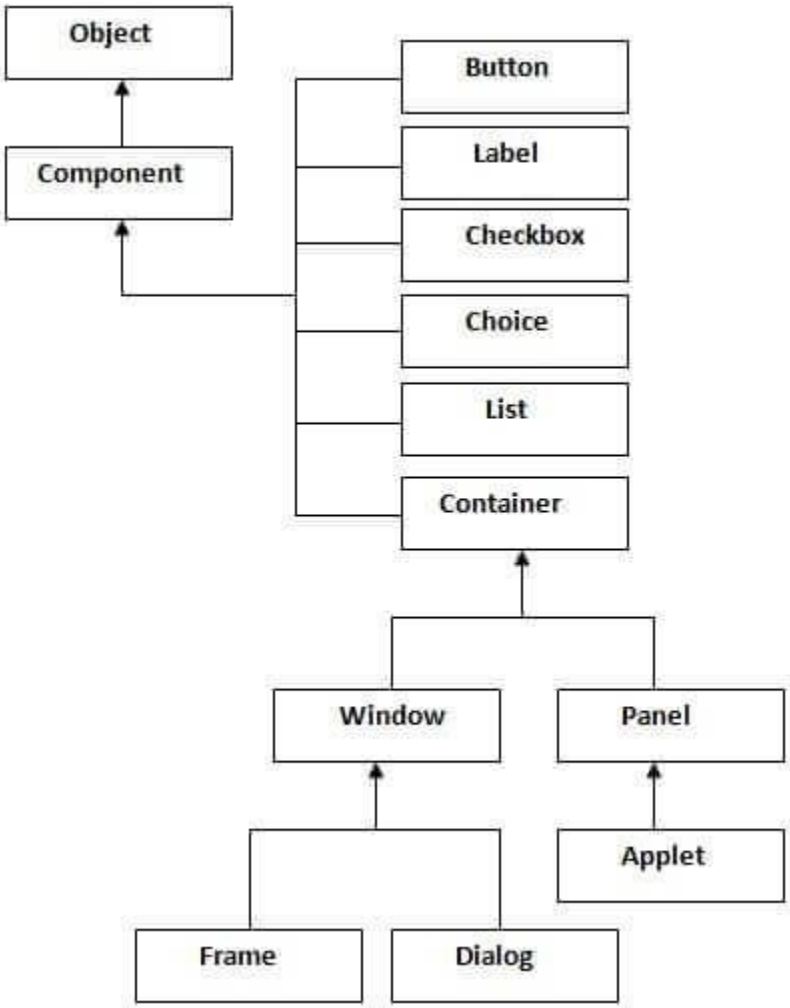
Labsheet-5

Topic: AWT (Abstract Window Toolkit)

Prof.R Gururaj

JAVA AWT

Java AWT (Abstract Window Toolkit) is an API to develop GUI or window-based applications in java.
Java AWT components are platform-dependent i.e. components are displayed according to the view of the operating system. AWT is heavyweight i.e. its components use the resources of OS.
The java.awt package provides classes for AWT api such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.
The hierarchy of java awt classes are given below:



COMPONENTS: A component is an object having a graphical representation that can be displayed on the screen and that can interact with the user. Examples of components are the buttons, checkboxes, and scrollbars of a typical graphical user interface.

CONTAINER:

Containers are an integral part of AWT GUI components. A container provides a space where a component can be located. A Container in AWT is a component itself and it adds the capability to add components to itself. Following are noticeable points to be considered.

Following is the list of commonly used containers while designed GUI using AWT:

- Frame
- Panel
- Window

FRAME:

The Frame is the container that contains the title bar and can have menu bars. It can have other components like button, textfield etc.

Constructors and Description:

- Frame() - Constructs a new instance of Frame that is initially invisible.
- Frame(String title) - Constructs a new, initially invisible Frame object with the specified title.

CONTROLS:

1. BUTTONS:

The button class is used to create a labeled button that has platform independent implementation. The application results in some action when the button is pushed.

Example:

2. CHOICE:

The object of Choice class is used to show a popup menu of choices. Choice selected by the user is shown on the top of a menu. It inherits Component class.

Example:

3. DIALOG: The Dialog control represents a top level window with a border and a title used to take some form of input from the user. It inherits the Window class. Unlike Frame, it doesn't have maximize and minimize buttons.

Example:

4. LABEL:

The object of Label class is a component for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly.

Example:

EVENT HANDLING

Change in the state of an object is known as event i.e. event describes the change in state of source. Events are generated as result of user interaction with the graphical user interface components. For example, clicking on a button, moving the mouse, , selecting entering a character through a keyboard an item from a list, scrolling the page are the activities that cause an event to happen.

Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. Java Uses the Delegation Event Model to handle the events. This model defines the standard mechanism to generate and handle the events. Let's have a brief introduction to this model.

The Delegation Event Model has the following key participants namely:

- Source - The source is an object on which event occurs. Source is responsible for providing information of the occurred event to it's handler. Java provides classes for source object.
- Listener - It is also known as event handler.Listener is responsible for generating response to an event. From a Java implementation point of view the listener is also an object. Listener waits until it receives an event. Once the event is received , the listener processes the event and then returns.

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

1. Event Classes Listener Interfaces
2. ActionEvent ActionListener
3. MouseEvent MouseListener and MouseMotionListener
4. MouseWheelEvent MouseWheelListener
5. KeyEvent KeyListener
6. ItemEvent ItemListener
7. TextEvent TextListener
8. AdjustmentEvent AdjustmentListener
9. WindowEvent WindowListener
10. ComponentEvent ComponentListener
11. ContainerEvent ContainerListener
12. FocusEvent FocusListener

Steps to perform Event Handling

Steps required to perform event handling:

1. Register the component with the Listener
2. Secondly, all the listeners interested in the XxxEvent must implement the XxxListener interface.

Registration Methods :

For registering the component with the Listener, many classes provide the registration methods.

For example:

- Button

public void addActionListener(ActionListener a){}

Java Event Handling Code

We can put the event handling code into one of the following places:

1. Within class
2. Other class
3. Anonymous class

Practice problem-1

Write a program to create a Frame (with Title MyFrame) of size 400X500 pixels and add three buttons with labels “ONE”, “TWO”, and “THREE” respectively. One label with text “Buttons”. Add these buttons and label to the frame and make it to show up when executed.

// we will discuss the code

Practice problem-2

Write a program to create a Frame with some Title of size 200X300 pixels and implement window closing event.

// we will discuss the code

Practice problem-3

Write Java code, to implement a ***Stack*** data structure (that can store integer values) using an array structure. The operations supported by the data structure are- (i) *push(int item)* to push an integer on to the stack; (ii) *pop()* to pop and return the popped item from the stack; (iii) *printStack()* operation must print the contents of the Stack from bottom to top. Note: Do not use any predefined utility classes like ArrayList etc.

// we will write code for the above

****