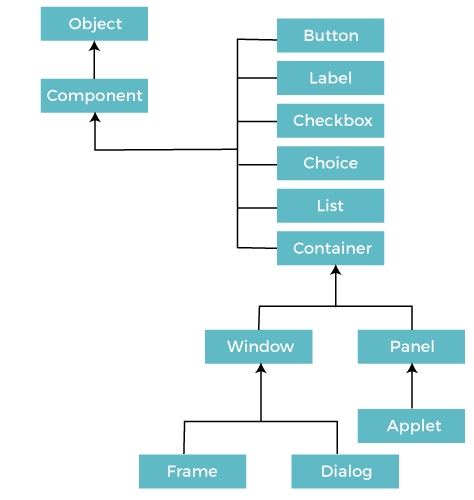
Java AWT & EVENTs

* **Java AWT** (Abstract Window Toolkit) is an API to develop Graphical User Interface (GUI) or windows-based applications in Java.
* Java AWT components are platform-dependent.
* The java.awt [package](https://www.javatpoint.com/package) provides [classes](https://www.javatpoint.com/object-and-class-in-java) such as
  + [TextField](https://www.javatpoint.com/java-awt-textfield), [Label](https://www.javatpoint.com/java-awt-label)
  + [TextArea](https://www.javatpoint.com/java-awt-textarea),
  + RadioButton
  + [CheckBox](https://www.javatpoint.com/java-awt-checkbox),
  + [Choice](https://www.javatpoint.com/java-awt-choice),
  + [List](https://www.javatpoint.com/java-awt-list)
* Java AWT Hierarchy
* The hierarchy of Java AWT classes are given below.
* 

### **Components**

All the elements like the button, text fields, scroll bars, etc. are called components.

### **Container**

The Container is a component in AWT that can contain another components like [buttons](https://www.javatpoint.com/java-awt-button), textfields, labels etc

classes that extends Container class are known as container.

**Types of containers:**

There are four types of containers in Java AWT:

1. Window
2. Panel
3. Frame
4. Dialog

## Useful Methods of Component Class

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void add(Component c) | Inserts a component on this component. |
| public void setSize(int width,int height) | Sets the size (width and height) of the component. |
| Public void setLayout(LayoutManager m) | Defines the layout manager for the component. |
| public void setVisible(boolean status) | Changes the visibility of the component, by default false. |
|  |  |

* **There are two way to perform awt program,**

Option1-By extending Frame class (**inheritance**)

Option2-By creating the object of Frame class (**association**)

EVENT AND LISTENER:

Changing the state of an object is known as an event.

For example, click on button, dragging mouse etc.

java.awt.event package provides many event classes and Listener interfaces

## Java Event classes and Listener interfaces

|  |  |
| --- | --- |
| **Event Classes** | **Listener Interfaces** |
| ActionEvent | ActionListener |
| MouseEvent | MouseListener and MouseMotionListener |
| MouseWheelEvent | MouseWheelListener |
| KeyEvent | KeyListener |
| ItemEvent | ItemListener |
| TextEvent | TextListener |
| AdjustmentEvent | AdjustmentListener |
| WindowEvent | WindowListener |
| ComponentEvent | ComponentListener |
| ContainerEvent | ContainerListener |
| FocusEvent | FocusListener |

Steps to perform Event Handling

Following steps are required to perform event handling:

1. Register the component with the Listener

Registration Methods

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

* **Button**
  + public void addActionListener(ActionListener a){}
* **MenuItem**
  + public void addActionListener(ActionListener a){}
* **TextField**
  + public void addActionListener(ActionListener a){}
  + public void addTextListener(TextListener a){}
* **TextArea**
  + public void addTextListener(TextListener a){}
* **Checkbox**
  + public void addItemListener(ItemListener a){}
* **Choice**
  + public void addItemListener(ItemListener a){}
* **List**
  + public void addActionListener(ActionListener a){}
  + public void addItemListener(ItemListener 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 clas

**Java AWT Button**

* A button is basically a control component with a label that generates an event when pushed.
* **Button** class is used to create a labeled button.
* The application result in some action when the button is pushed.
* When we press a button and release it, AWT sends an instance of **ActionEvent** to that button by calling **processEvent** on the button.
* **processEvent** method of the button receives the all the events, then it passes an action event by calling its own method **processActionEvent**.
* To perform an action on a button being pressed and released, the **ActionListener** interface needs to be implemented.
* listener can receive events from the button by calling **addActionListener** method of the button.