**Experiment No-1**

**Title:** Write a program to demonstrate use of AWT controls

**Aim:** To study different controls of AWT, its constructors and methods in detail.

**Theory:**

**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 i.e. components are displayed according to the view of operating system. AWT is heavy weight i.e. its components are using the resources of underlying operating system (OS).

The java.awt [package](https://www.javatpoint.com/package) provides [classes](https://www.javatpoint.com/object-and-class-in-java) for AWT API 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) etc.

# **Java AWT Button**

A button is basically a control component with a label that generates an event when pushed. The **Button** class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed.

When we press a button and release it, AWT sends an instance of **Action Event** to that button by calling **process Event** on the button. The **process Event** method of the button receives the all the events, then it passes an action event by calling its own method **process Action Event**. This method passes the action event on to action listeners that are interested in the action events generated by the button.

To perform an action on a button being pressed and released, the **ActionListener** interface needs to be implemented. The registered new listener can receive events from the button by calling **addActionListener** method of the button. The Java application can use the button's action command as a messaging protocol.

# **Java AWT Label**

The [object](https://www.javatpoint.com/object-and-class-in-java) of the 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 a programmer but a user cannot edit it directly.

It is called a passive control as it does not create any event when it is accessed. To create a label, we need to create the object of **Label** class.

# **Java AWT TextField**

The [object](https://www.javatpoint.com/object-and-class-in-java) of a **TextField** class is a text component that allows a user to enter a single line text and edit it. It inherits **TextComponent** class, which further inherits **Component** class.

When we enter a key in the text field (like key pressed, key released or key typed), the event is sent to **TextField**. Then the **KeyEvent** is passed to the registered **KeyListener**. It can also be done using **ActionEvent**; if the ActionEvent is enabled on the text field, then the ActionEvent may be fired by pressing return key. The event is handled by the **ActionListener** interface.

# **Java AWT TextArea**

The [object](https://www.javatpoint.com/object-and-class-in-java) of a TextArea class is a multiline region that displays text. It allows the editing of multiple line text. It inherits TextComponent class.

The text area allows us to type as much text as we want. When the text in the text area becomes larger than the viewable area, the scroll bar appears automatically which helps us to scroll the text up and down, or right and left.

# **Java AWT Checkbox**

The Checkbox class is used to create a checkbox. It is used to turn an option on (true) or off (false). Clicking on a Checkbox changes its state from "on" to "off" or from "off" to "on".

# **Java AWT CheckboxGroup**

The object of CheckboxGroup class is used to group together a set of [Checkbox](https://www.javatpoint.com/java-awt-checkbox). At a time only one check box button is allowed to be in "on" state and remaining check box button in "off" state. It inherits the [object class](https://www.javatpoint.com/object-class).

# **Java AWT Choice**

The object of Choice class is used to show [popup menu](https://www.javatpoint.com/java-awt-popupmenu) of choices. Choice selected by user is shown on the top of a menu. It inherits Component class.

# **Java AWT List**

The object of List class represents a list of text items. With the help of the List class, user can choose either one item or multiple items. It inherits the Component class.

# **Java AWT Scrollbar**

The [object](https://www.javatpoint.com/object-and-class-in-java) of Scrollbar class is used to add horizontal and vertical scrollbar. Scrollbar is a [GUI](https://www.javatpoint.com/gui-full-form) component allows us to see invisible number of rows and columns.

It can be added to top-level container like Frame or a component like Panel. The Scrollbar class extends the **Component** class.

# **Java AWT MenuItem and Menu**

The object of MenuItem class adds a simple labeled menu item on menu. The items used in a menu must belong to the MenuItem or any of its subclass.

The object of Menu class is a pull down menu component which is displayed on the menu bar. It inherits the MenuItem class.

# **Java AWT Panel**

The Panel is a simplest container class. It provides space in which an application can attach any other component. It inherits the Container class. It doesn't have title bar.

**Java AWT 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](https://www.javatpoint.com/java-awt-button).

## Frame vs Dialog

## Frame and Dialog both inherits Window class. Frame has maximize and minimize buttons but Dialog doesn't have.

**Conclusion:** In this experiment we have learnt how to place various AWT controls used to built GUI applications.

**Exercise:**

