Java AWT Tutorial

**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.

The AWT tutorial will help the user to understand Java GUI programming in simple and easy steps.

Why AWT is platform independent?

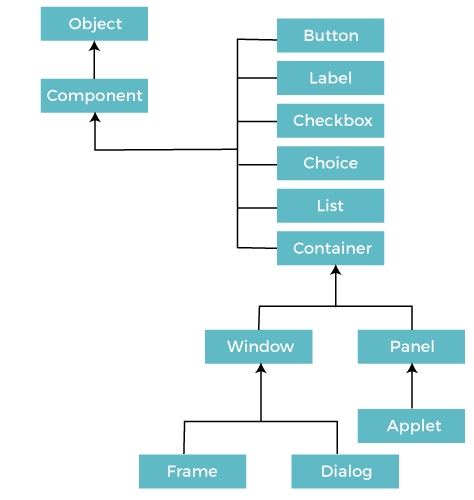
Java AWT calls the native platform calls the native platform (operating systems) subroutine for creating API components like TextField, ChechBox, button, etc.

For example, an AWT GUI with components like TextField, label and button will have different look and feel for the different platforms like Windows, MAC OS, and Unix. The reason for this is the platforms have different view for their native components and AWT directly calls the native subroutine that creates those components.

In simple words, an AWT application will look like a windows application in Windows OS whereas it will look like a Mac application in the MAC OS.

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. In Java AWT, there are classes for each component as shown in above diagram. In order to place every component in a particular position on a screen, we need to add them to a container.

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. The classes that extends Container class are known as container such as **Frame, Dialog** and **Panel**.

It is basically a screen where the where the components are placed at their specific locations. Thus it contains and controls the layout of components.

Note: A container itself is a component (see the above diagram), therefore we can add a container inside container.

**Types of containers:**

There are four types of containers in Java AWT:

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

Window

The window is the container that have no borders and menu bars. You must use frame, dialog or another window for creating a window. We need to create an instance of Window class to create this container.

Panel

The Panel is the container that doesn't contain title bar, border or menu bar. It is generic container for holding the components. It can have other components like button, text field etc. An instance of Panel class creates a container, in which we can add components.

Frame

The Frame is the container that contain title bar and border and can have menu bars. It can have other components like button, text field, scrollbar etc. Frame is most widely used container while developing an AWT application.

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. |

Java AWT Example

To create simple AWT example, you need a frame. There are two ways to create a GUI using Frame in AWT.

1. By extending Frame class (**inheritance**)
2. By creating the object of Frame class (**association**)

AWT Example by Inheritance

Let's see a simple example of AWT where we are inheriting Frame class. Here, we are showing Button component on the Frame.

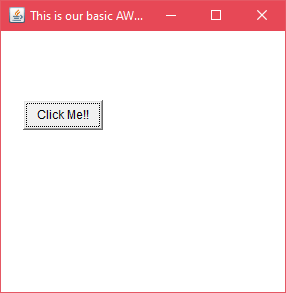
**AWTExample1.java**

1. // importing Java AWT class
2. **import** java.awt.\*;
4. // extending Frame class to our class AWTExample1
5. **public** **class** AWTExample1 **extends** Frame {
7. // initializing using constructor
8. AWTExample1() {
10. // creating a button
11. Button b = **new** Button("Click Me!!");
13. // setting button position on screen
14. b.setBounds(30,100,80,30);
16. // adding button into frame
17. add(b);
19. // frame size 300 width and 300 height
20. setSize(300,300);
22. // setting the title of Frame
23. setTitle("This is our basic AWT example");
25. // no layout manager
26. setLayout(**null**);
28. // now frame will be visible, by default it is not visible
29. setVisible(**true**);
30. }
32. // main method
33. **public** **static** **void** main(String args[]) {
35. // creating instance of Frame class
36. AWTExample1 f = **new** AWTExample1();
38. }
40. }

[download this example](https://static.javatpoint.com/src/awt/first.zip)

The setBounds(int x-axis, int y-axis, int width, int height) method is used in the above example that sets the position of the awt button.

**Output:**



AWT Example by Association

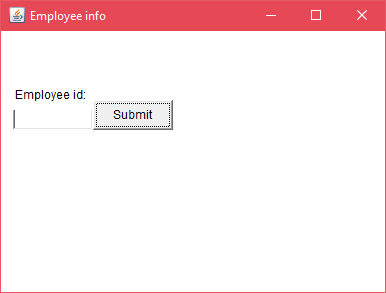
Let's see a simple example of AWT where we are creating instance of Frame class. Here, we are creating a TextField, Label and Button component on the Frame.

**AWTExample2.java**

1. // importing Java AWT class
2. **import** java.awt.\*;
4. // class AWTExample2 directly creates instance of Frame class
5. **class** AWTExample2 {
7. // initializing using constructor
8. AWTExample2() {
10. // creating a Frame
11. Frame f = **new** Frame();
13. // creating a Label
14. Label l = **new** Label("Employee id:");
16. // creating a Button
17. Button b = **new** Button("Submit");
19. // creating a TextField
20. TextField t = **new** TextField();
22. // setting position of above components in the frame
23. l.setBounds(20, 80, 80, 30);
24. t.setBounds(20, 100, 80, 30);
25. b.setBounds(100, 100, 80, 30);
27. // adding components into frame
28. f.add(b);
29. f.add(l);
30. f.add(t);
32. // frame size 300 width and 300 height
33. f.setSize(400,300);
35. // setting the title of frame
36. f.setTitle("Employee info");
38. // no layout
39. f.setLayout(**null**);
41. // setting visibility of frame
42. f.setVisible(**true**);
43. }
45. // main method
46. **public** **static** **void** main(String args[]) {
48. // creating instance of Frame class
49. AWTExample2 awt\_obj = **new** AWTExample2();
51. }
53. }

[download this example](https://static.javatpoint.com/src/awt/first2.zip)

**Output:**



**Component Class in Java**

* Last Updated : 08 Feb, 2023

 [Read](javascript:void(0))

 [Discuss](javascript:void(0))

 [Courses](javascript:void(0))

 [Practice](javascript:void(0))

 [Video](javascript:void(0))

The Component class is the superclass of all components. A component class can be linked with a page, components of web applications. Component clearly shows that is the graphical representation of an Object.

**Important methods of Component Class:**

1. **public void add(Component c):**This method inserts a component into a Container by taking a component as a parameter.
2. **public void setSize(int width, int height):**This method sets the size of the component by taking height and width as parameters.
3. **public void setLayout(LayoutManager lm):**This method sets the layout to set the components in a particular manner by taking LayoutManager as a parameter.
4. **public void setVisible(boolean status):** This method sets the visibility of a component to visible or not. If it sets to true then the component will be visible in the output else if it sets to false or not defined component won’t be visible in the output.

***Note: LayoutManager****helps us to give the positioning and size of components that should be visible.*

**Types of Components in Component Class**

The components in a component class are as follows :

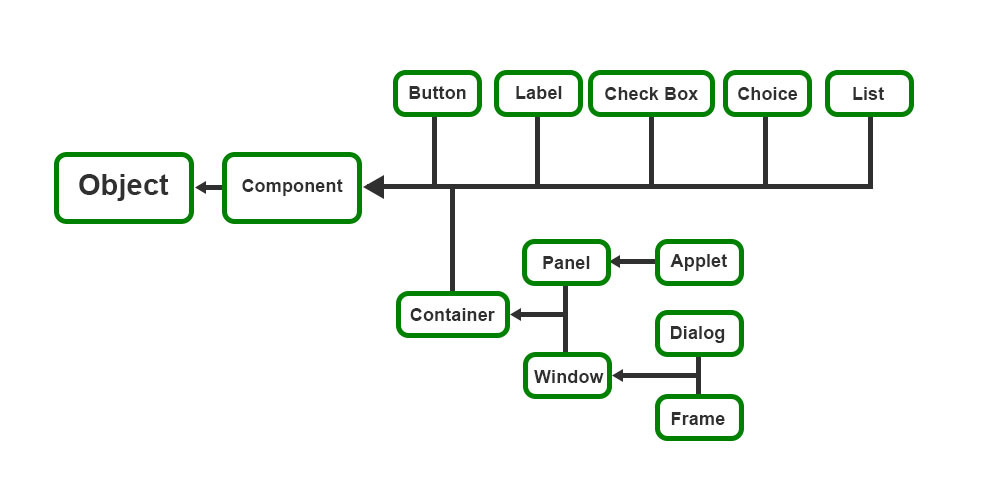
1. **Container**
2. **Button**
3. **Label**
4. **Checkbox**
5. **Choice**
6. **List**

All these components are present in **java.awt**package. We can import each of the components individually i.e., **import java.awt.Button**, **import java.awt.Container** etc.

***Note****: If we want to import all components at a time we can do that by importing like****import******java.awt.\****

The hierarchical structure of the above-listed components is below:

**Hierarchical Structure of Components**



So let us see each component in detail.

**1. Container**

The Container is a component that will be used to extend other components such as window, panel, Frame, Dialog, and Applet as shown in the above diagram.

* **Window:** The Window is a Container that doesn’t include borders and a menu bar. We must use another window, frames, and dialogue box to create a Window. Creating an instance is the way to create a Window Container.
* **Panel:** The Panel is also a Container that doesn’t include a title bar, menu, or border. It is a container that holds components like buttons, textfield, etc.  Creating an instance is the way to create a Panel Container and can add components.
* **Frame:** The Frame is a container used while creating an AWT application. It can have components like title bar, menu bars, borders and also buttons, scroll bar, etc.
* **Dialog:** The Dialog box is a container that will display the message that we want to display on the screen.

**2. Button**

A button is a labeled component when clicked performs an event. It is created by the Button class. When clicked it performs some action by sending an instance of ActionEvent by AWT. ActionEvent calls processEvent on the button and it receives all the events and performs action by calling the processActionEvent method of its own. To do such things it needs to implement ActionListener. The Declaration of Button Class will be

*public class Button extends Component implements Accessible*

It contains 2 constructors:

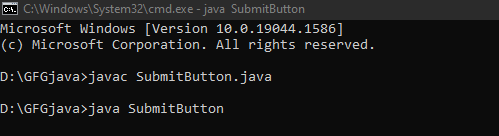
1. *Button() : This constructor will create a button with no label*
2. *Button(String label) : This constructor creates a button with label value as a value when we creates an object*

**Example:**

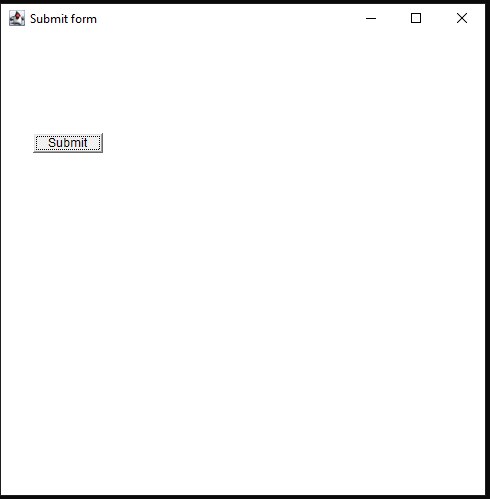
* Java

|  |
| --- |
| import java.awt.\*;  class SubmitButton extends Frame {      public static void main(String[] args)      {          // create instance of frame with label          Frame frame = new Frame("Submit form");            // create instance of button with label          Button button = new Button("Submit");            // set the position for the button in frame          button.setBounds(40, 130, 70, 20);            // adding button to the frame          frame.add(button);            // setting size for the frame          frame.setSize(500, 500);            // setting layout for the frame          frame.setLayout(null);            // visibility of frame to display the output\           // without this output will be blank          frame.setVisible(true);      }  } |

We can run it by the following commands:



**Output:**



**3. Label**

It is used to show text in the Container. It will displays text in the form of **READ-ONLY**, which cannot be changed by the user directly. We need to create an instance of Label Class to create a Label. The Declaration of Label Class will be

*public class Label extends Component implements Accessible*

It has 3 constructors:

1. *Label() : Creates an Empty Label.*
2. *Label(String labelname) : Creates a Label with labelname as parameter value.*
3. *Label(String labelname, int align) : Creates a Label with labelname as parameter value and proper alignments.*

***Note****: Align parameter aligns the text in proper alignment and it has 3 types of alignment.*

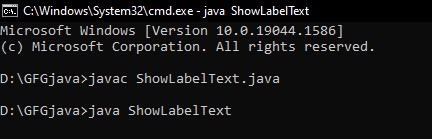
* LEFT: specifies that text should be aligned to left.
* RIGHT: specifies that text should be aligned to right.
* CENTER specifies that text should be aligned to the center.

**Example:**

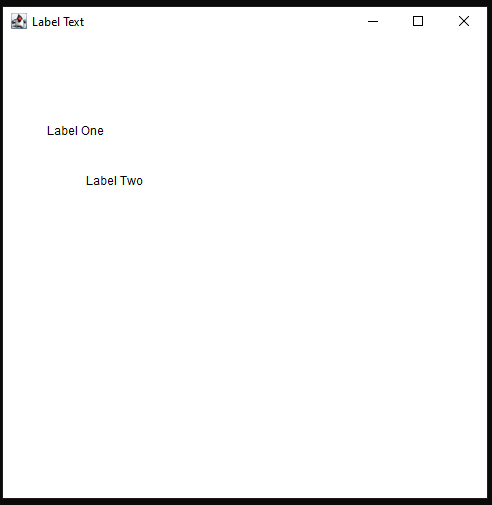
* Java

|  |
| --- |
| import java.awt.\*;    public class ShowLabelText extends Frame {      public static void main(String args[])      {            // creating objects for Frame and Label class          Frame frame = new Frame("Label Text");            // Creating label One          Label label1 = new Label("Label One", Label.LEFT);            // Creating label Two          Label label2 = new Label("Label Two", Label.RIGHT);            // set the location of label in px          label1.setBounds(50, 100, 100, 50);          label2.setBounds(50, 150, 100, 50);            // adding labels to the frame          frame.add(label1);          frame.add(label2);            // setting size, layout          // and visibility of frame          frame.setSize(500, 500);          frame.setLayout(null);          frame.setVisible(true);      }  } |

We can run it by the following commands:



**Output:**



**4. Checkbox**

It is used to create a Checkbox in the Container. It can get a value either true or false by checking and unchecking the checkbox.

* checked – returns true
* unchecked – returns false

It can be created by creating an instance of Checkbox. The Declaration of Label Class will be

*public class Checkbox extends Component implements ItemSelectable, Accessible*

It has 5 constructors:

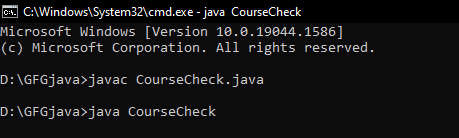
1. *Checkbox() : Creates a checkbox with empty label*
2. *Checkbox(String checkboxlabel) : Creates a Checkbox with checkboxlabel as parameter value.*
3. *Checkbox(String checkboxlabel, boolean status) : Creates a Checkbox with checkboxlabel as parameter value and sets the status either true or false.*
4. *Checkbox(String checkboxlabel, boolean status, CheckboxGroup cbgroup) : Creates a Checkbox with checkboxlabel as parameter value and sets the status to the specified checkbox group.*
5. *Checkbox(String checkboxlabel, CheckboxGroup cbgroup, boolean status) : Creates a Checkbox with checkboxlabel as parameter value for the specified cbgroup and sets the status.*

**Example 1:**

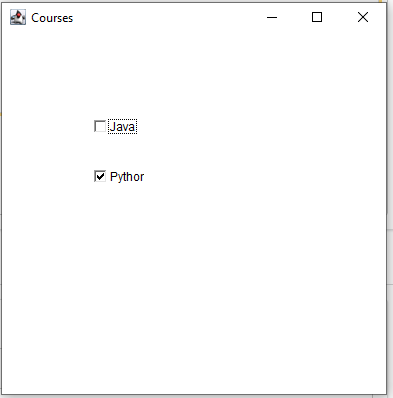
* Java

|  |
| --- |
| // importing AWT class  import java.awt.\*;    public class CourseCheck {      // main method      public static void main(String args[])      {          // creating the frame with the label          Frame frame = new Frame("Courses");            // creating checkbox java          Checkbox java = new Checkbox("Java");            // setting location of checkbox in frame          java.setBounds(100, 100, 50, 50);            // creating checkbox python with status as true          Checkbox python = new Checkbox("Python", true);            // setting location of checkbox in frame          python.setBounds(100, 150, 50, 50);            // adding checkboxes to frame          frame.add(java);          frame.add(python);            // setting size, layout and          // visibility of frame          frame.setSize(400, 400);          frame.setLayout(null);          frame.setVisible(true);      }  } |

We can run it by the following commands:



**Output:**



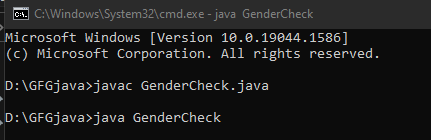
In the above output, we can select both options.

**Example 2:**

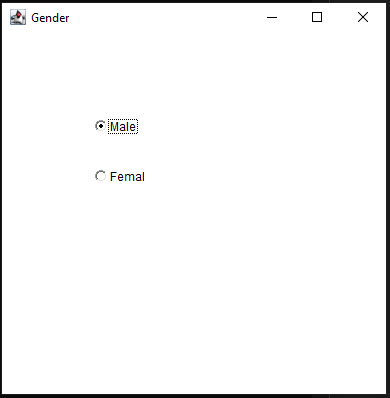
* Java

|  |
| --- |
| // importing AWT class  import java.awt.\*;    public class GenderCheck {      // main method      public static void main(String args[])      {          // creating the frame with the label          Frame frame = new Frame("Gender");            // creating a CheckboxGroup          CheckboxGroup cbgroup = new CheckboxGroup();            // creating checkbox male with          // status as true for cbgroup          Checkbox male = new Checkbox("Male", cbgroup, true);          male.setBounds(100, 100, 50, 50);            // creating checkbox female with          // status as false for cbgroup          Checkbox female = new Checkbox("Female", cbgroup, false);            // setting location of checkbox in frame          female.setBounds(100, 150, 50, 50);            // adding checkboxes to frame          frame.add(male);          frame.add(female);            // setting size, layout          // and visibility of frame          frame.setSize(400, 400);          frame.setLayout(null);          frame.setVisible(true);      }  } |

We can run it by using commands:



**Output:**



In the above output, we can select any one of the options as it works as a radio button.

**5. Choice**

It is used to show the popup menu to select any item from the menu items. The selected choice will be shown at the top of the menu bar. We need to create an instance of Choice Class to create a Choice. The Declaration of Choice Class will be

*public class Choice extends Component implements ItemSelectable, Accessible*

It has 1 constructor:

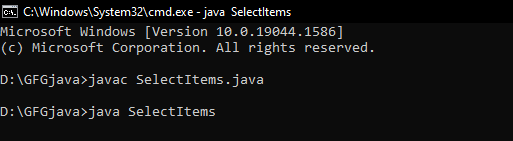
*Choice() : Creates a new Choice menu of items*

**Example:**

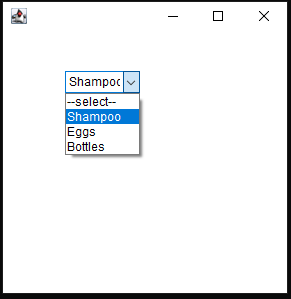
* Java

|  |
| --- |
| import java.awt.\*;  public class SelectItems {        // main method      public static void main(String args[])      {          // creating a Frame          Frame frame = new Frame();            // creating a choice component          Choice choice = new Choice();            // setting the bounds of choice menu          choice.setBounds(70, 70, 75, 75);            // adding items to the choice menu          choice.add("--select--");          choice.add("Shampoo");          choice.add("Eggs");          choice.add("Bottles");            // adding choice menu to frame          frame.add(choice);            // setting size, layout          // and visibility of frame          frame.setSize(300, 300);          frame.setLayout(null);          frame.setVisible(true);      }  } |

We can run it by the following commands:



**Output:**



**6. List**

The List Object creates a list of items in which we can choose one or multiple items at a time. We need to create an instance of List Class to create a List. The Declaration of Label Class will be

*public class List extends Component implements ItemSelectable, Accessible*

It has 3 constructors:

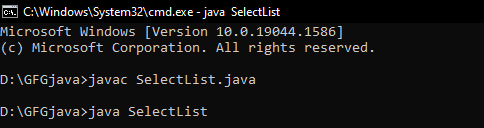
1. *List() : Creates a new Scrolling List*
2. *List(int noofrows) : Creates a new Scrolling List which displays the list of items with given no. of rows visible with parameter noofrows.*
3. *List(int noofrows, boolean multi) : Creates a new Scrolling list which displays the list of items with given no. of rows visible and allows to select multiple items at a time.*

**Example:**

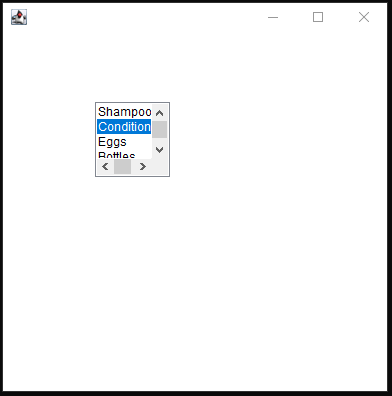
* Java

|  |
| --- |
| import java.awt.\*;  public class SelectList {        // main method      public static void main(String args[])      {          // creating frame1          Frame frame1 = new Frame();            // creating list1 with 5 rows          List list1 = new List(5);            // setting the position of list component          list1.setBounds(100, 100, 75, 75);            // adding list items to the list1          list1.add("Shampoo");          list1.add("Conditioner");          list1.add("Eggs");          list1.add("Bottles");          list1.add("Butter");            // adding the list1 to frame1          frame1.add(list1);            // setting size, layout          // and visibility of frame1          frame1.setSize(400, 400);          frame1.setLayout(null);          frame1.setVisible(true);            // creating frame2          Frame frame2 = new Frame();            // creating list2 with 5 rows          // and multi select items as true          List list2 = new List(5, true);            // setting the position of list component          list2.setBounds(100, 100, 75, 75);            // adding list items to the list2          list2.add("Shampoo");          list2.add("Conditioner");          list2.add("Eggs");          list2.add("Bottles");          list2.add("Butter");            // adding the list2 to frame2          frame2.add(list2);            // setting size, layout          // and visibility of frame2          frame2.setSize(400, 400);          frame2.setLayout(null);          frame2.setVisible(true);      }  } |

We can run it by the following commands:

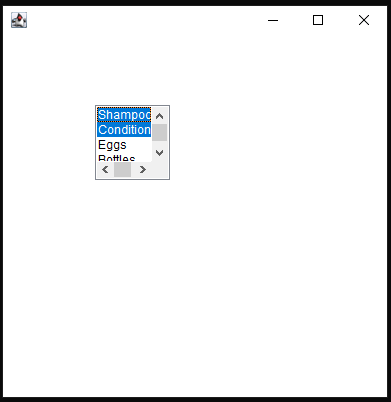


**Output 1:**



In this output List, we can select any one item at a time.

**Output 2:**



In this output List, we can select multiple items at a time. These are all the components present in the Component class and about component class.

Event and Listener (Java Event Handling)

|  |
| --- |
| Changing the state of an object is known as an event. For example, click on button, dragging mouse etc. The java.awt.event package provides many event classes and Listener interfaces for event handling. |

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 class

Java event handling by implementing ActionListener

1. **import** java.awt.\*;
2. **import** java.awt.event.\*;
3. **class** AEvent **extends** Frame **implements** ActionListener{
4. TextField tf;
5. AEvent(){
7. //create components
8. tf=**new** TextField();
9. tf.setBounds(60,50,170,20);
10. Button b=**new** Button("click me");
11. b.setBounds(100,120,80,30);
13. //register listener
14. b.addActionListener(**this**);//passing current instance
16. //add components and set size, layout and visibility
17. add(b);add(tf);
18. setSize(300,300);
19. setLayout(**null**);
20. setVisible(**true**);
21. }
22. **public** **void** actionPerformed(ActionEvent e){
23. tf.setText("Welcome");
24. }
25. **public** **static** **void** main(String args[]){
26. **new** AEvent();
27. }
28. }

**public void setBounds(int xaxis, int yaxis, int width, int height);** have been used in the above example that sets the position of the component it may be button, textfield etc.

PlayNext

Unmute

Current TimeÂ 0:00

/

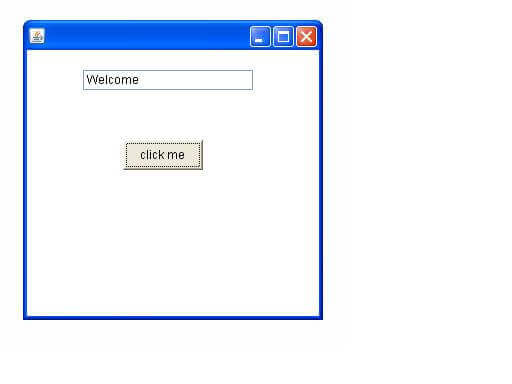
DurationÂ 18:10

Loaded: 0.37%

Â

Fullscreen

Backward Skip 10sPlay VideoForward Skip 10s



2) Java event handling by outer class

1. **import** java.awt.\*;
2. **import** java.awt.event.\*;
3. **class** AEvent2 **extends** Frame{
4. TextField tf;
5. AEvent2(){
6. //create components
7. tf=**new** TextField();
8. tf.setBounds(60,50,170,20);
9. Button b=**new** Button("click me");
10. b.setBounds(100,120,80,30);
11. //register listener
12. Outer o=**new** Outer(**this**);
13. b.addActionListener(o);//passing outer class instance
14. //add components and set size, layout and visibility
15. add(b);add(tf);
16. setSize(300,300);
17. setLayout(**null**);
18. setVisible(**true**);
19. }
20. **public** **static** **void** main(String args[]){
21. **new** AEvent2();
22. }
23. }
24. **import** java.awt.event.\*;
25. **class** Outer **implements** ActionListener{
26. AEvent2 obj;
27. Outer(AEvent2 obj){
28. **this**.obj=obj;
29. }
30. **public** **void** actionPerformed(ActionEvent e){
31. obj.tf.setText("welcome");
32. }
33. }

3) Java event handling by anonymous class

1. **import** java.awt.\*;
2. **import** java.awt.event.\*;
3. **class** AEvent3 **extends** Frame{
4. TextField tf;
5. AEvent3(){
6. tf=**new** TextField();
7. tf.setBounds(60,50,170,20);
8. Button b=**new** Button("click me");
9. b.setBounds(50,120,80,30);
11. b.addActionListener(**new** ActionListener(){
12. **public** **void** actionPerformed(){
13. tf.setText("hello");
14. }
15. });
16. add(b);add(tf);
17. setSize(300,300);
18. setLayout(**null**);
19. setVisible(**true**);
20. }
21. **public** **static** **void** main(String args[]){
22. **new** AEvent3();
23. }
24. }