**Java GUI (Applets vs Applications) : Chapter 8**

# Java Applet

Applet is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.

### Advantage of Applet

There are many advantages of applet. They are as follows:

* It works at client side so less response time.
* Secured
* It can be executed by browsers running under many plateforms, including Linux, Windows, Mac Os etc.

### Drawback of Applet

* Plugin is required at client browser to execute applet.

### Hierarchy of Applet



|  |
| --- |
| As displayed in the above diagram, Applet class extends Panel. Panel class extends Container which is the subclass of Component |

### Lifecycle of Java Applet

1. Applet is initialized.
2. Applet is started.
3. Applet is painted.
4. Applet is stopped.
5. Applet is destroyed.



### java.applet.Applet class

For creating any applet java.applet.Applet class must be inherited. It provides 4 life cycle methods of applet.

1. **public void init():** is used to initialized the Applet. It is invoked only once.
2. **public void start():** is invoked after the init() method or browser is maximized. It is used to start the Applet.
3. **public void stop():** is used to stop the Applet. It is invoked when Applet is stop or browser is minimized.
4. **public void destroy():** is used to destroy the Applet. It is invoked only once.

### java.awt.Component class

The Component class provides 1 life cycle method of applet.

1. **public void paint(Graphics g):** is used to paint the Applet. It provides Graphics class object that can be used for drawing oval, rectangle, arc etc.

### Simple example of Applet by html file:

To execute the applet by html file, create an applet and compile it. After that create an html file and place the applet code in html file. Now click the html file.

1. //First.java
2. **import** java.applet.Applet;
3. **import** java.awt.Graphics;
4. **public** **class** First **extends** Applet{
5.
6. **public** **void** paint(Graphics g){
7. g.drawString("welcome",150,150);
8. }
9.
10. }

### myapplet.html

1. <html>
2. <body>
3. <applet code="First.class" width="300" height="300">
4. </applet>
5. </body>
6. </html>

### Simple example of Applet by appletviewer tool:

To execute the applet by appletviewer tool, create an applet that contains applet tag in comment and compile it. After that run it by: appletviewer First.java. Now Html file is not required but it is for testing purpose only.

1. //First.java
2. **import** java.applet.Applet;
3. **import** java.awt.Graphics;
4. **public** **class** First **extends** Applet{
5.
6. **public** **void** paint(Graphics g){
7. g.drawString("welcome to applet",150,150);
8. }
9.
10. }
11. /\*
12. <applet code="First.class" width="300" height="300">
13. </applet>
14. \*/

To execute the applet by appletviewer tool, write in command prompt:

**c:\>**javac First.java

**c:\>**appletviewer First.java

Example of displaying image in applet:

1. **import** java.awt.\*;
2. **import** java.applet.\*;
3.
4.
5. **public** **class** DisplayImage **extends** Applet {
6.
7. Image picture;
8.
9. **public** **void** init() {
10. picture = getImage(getDocumentBase(),"sonoo.jpg");
11. }
12.
13. **public** **void** paint(Graphics g) {
14. g.drawImage(picture, 30,30, **this**);
15. }
16.
17. }

# EventHandling in Applet

|  |
| --- |
| As we perform event handling in AWT or Swing, we can perform it in applet also. Let's see the simple example of event handling in applet that prints a message by click on the button. |

## Example of EventHandling in applet:

1. **import** java.applet.\*;
2. **import** java.awt.\*;
3. **import** java.awt.event.\*;
4. **public** **class** EventApplet **extends** Applet **implements** ActionListener{
5. Button b;
6. TextField tf;
7.
8. **public** **void** init(){
9. tf=**new** TextField();
10. tf.setBounds(30,40,150,20);
11.
12. b=**new** Button("Click");
13. b.setBounds(80,150,60,50);
14.
15. add(b);add(tf);
16. b.addActionListener(**this**);
17.
18. setLayout(**null**);
19. }
20.
21. **public** **void** actionPerformed(ActionEvent e){
22. tf.setText("Welcome");
23. }
24. }

|  |
| --- |
| In the above example, we have created all the controls in init() method because it is invoked only once. |

# Java AWT Tutorial

**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 operating system. AWT is heavyweight i.e. its components are using the resources of 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 Hierarchy

The hierarchy of Java AWT classes are given below.



### 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.

### 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.

### Panel

The Panel is the container that doesn't contain title bar and menu bars. It can have other components like button, textfield etc.

### Frame

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

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.

1. **import** java.awt.\*;
2. **class** First **extends** Frame{
3. First(){
4. Button b=**new** Button("click me");
5. b.setBounds(30,100,80,30);// setting button position
6. add(b);//adding button into frame
7. setSize(300,300);//frame size 300 width and 300 height
8. setLayout(**null**);//no layout manager
9. setVisible(**true**);//now frame will be visible, by default not visible
10. }
11. **public** **static** **void** main(String args[]){
12. First f=**new** First();
13. }}

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

The setBounds(int xaxis, int yaxis, int width, int height) method is used in the above example that sets the position of the awt button.



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 |

### 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(){
6.
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);
12.
13. //register listener
14. b.addActionListener(**this**);//passing current instance
15.
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.



### Difference between AWT and Swing

There are many differences between java awt and swing that are given below.

|  |  |  |
| --- | --- | --- |
| **No.** | **Java AWT** | **Java Swing** |
| 1) | AWT components are **platform-dependent**. | Java swing components are **platform-independent**. |
| 2) | AWT components are **heavyweight**. | Swing components are **lightweight**. |
| 3) | AWT **doesn't support pluggable look and feel**. | Swing **supports pluggable look and feel**. |
| 4) | AWT provides **less components** than Swing. | Swing provides **more powerful components** such as tables, lists, scrollpanes, colorchooser, tabbedpane etc. |
| 5) | AWT **doesn't follows MVC**(Model View Controller) where model represents data, view represents presentation and controller acts as an interface between model and view. | Swing **follows MVC**. |

**What is Swing?**

Java Swing is a lightweight Graphical User Interface (GUI) toolkit that includes a rich set of widgets. It includes package lets you make GUI components for your Java applications, and It is platform independent.

The Swing library is built on top of the Java Abstract Widget Toolkit (**AWT**), an older, platform dependent GUI toolkit. You can use the Java GUI components like button, textbox, etc. from the library and do not have to create the components from scratch.

**Java Swing class Hierarchy Diagram**



### Simple Java Swing Example

Let's see a simple swing example where we are creating one button and adding it on the JFrame object inside the main() method.

*File: FirstSwingExample.java*

1. **import** javax.swing.\*;
2. **public** **class** FirstSwingExample {
3. **public** **static** **void** main(String[] args) {
4. JFrame f=**new** JFrame();//creating instance of JFrame
5.
6. JButton b=**new** JButton("click");//creating instance of JButton
7. b.setBounds(130,100,100, 40);//x axis, y axis, width, height
8.
9. f.add(b);//adding button in JFrame
10.
11. f.setSize(400,500);//400 width and 500 height
12. f.setLayout(**null**);//using no layout managers
13. f.setVisible(**true**);//making the frame visible
14. }
15. }



### Example of Swing by Association inside constructor

We can also write all the codes of creating JFrame, JButton and method call inside the java constructor.

*File: Simple.java*

1. **import** javax.swing.\*;
2. **public** **class** Simple {
3. JFrame f;
4. Simple(){
5. f=**new** JFrame();//creating instance of JFrame
6.
7. JButton b=**new** JButton("click");//creating instance of JButton
8. b.setBounds(130,100,100, 40);
9.
10. f.add(b);//adding button in JFrame
11.
12. f.setSize(400,500);//400 width and 500 height
13. f.setLayout(**null**);//using no layout managers
14. f.setVisible(**true**);//making the frame visible
15. }
16.
17. **public** **static** **void** main(String[] args) {
18. **new** Simple();
19. }
20. }

The setBounds(int xaxis, int yaxis, int width, int height)is used in the above example that sets the position of the button.

### Simple example of Swing by inheritance

We can also inherit the JFrame class, so there is no need to create the instance of JFrame class explicitly.

*File: Simple2.java*

1. **import** javax.swing.\*;
2. **public** **class** Simple2 **extends** JFrame{//inheriting JFrame
3. JFrame f;
4. Simple2(){
5. JButton b=**new** JButton("click");//create button
6. b.setBounds(130,100,100, 40);
7.
8. add(b);//adding button on frame
9. setSize(400,500);
10. setLayout(**null**);
11. setVisible(**true**);
12. }
13. **public** **static** **void** main(String[] args) {
14. **new** Simple2();
15. }}

package com.applets;

import java.applet.Applet;

import java.awt.Button;

import java.awt.Color;

import java.awt.GridLayout;

import java.awt.Label;

import java.awt.TextArea;

import java.awt.TextField;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JOptionPane;

public class EnquiryForm extends Applet implements ActionListener

{

 private static final long serialVersionUID = 1L;

 Label l1,l2,l3,l4,l5,l6;

 TextField t1,t2,t3;

 TextArea t4;

 Button b1,b2;

 @Override

 public void init()

 {

 setSize(300, 200);

 setLayout(new GridLayout(6,2));

 setBackground(Color.pink);

 l1 = new Label("Full Name");

 l2 = new Label("Phone");

 l3 = new Label("Email");

 l4 = new Label("Message");

 l5 = new Label("");

 l6 = new Label("");

 t1 = new TextField(35);

 t2 = new TextField(11);

 t3 = new TextField(35);

 t4 = new TextArea();

 b1 = new Button("Submit");

 b2 = new Button("Clear");

 add(l1); add(t1); add(l2);add(t2);add(l3);add(t3);add(l4);

 add(t4);add(l5);add(l6);add(b1);add(b2);

 b2.addActionListener(this);

 b1.addActionListener(this);

 }

 @Override

 public void actionPerformed(ActionEvent ae)

 {

 if(ae.getSource()== b2)

 {

 t1.setText("");

 t2.setText("");

 t3.setText("");

 t4.setText("");

 }

 if(ae.getSource()== b1)

 {

 if(t1.getText().equals("") || t2.getText().equals("") || t3.getText().equals("") || t4.getText().equals(""))

 {

 showStatus("All Fields data is Mandatory!!");

 }

 else

 {

 DMLOperations dbp = new DMLOperations();

 String temp=null;

 try

 {

 dbp.createConnection();

 dbp.saveData(t1.getText(),t2.getText(),t3.getText(),t4.getText());

 //dbp.createConnection();

 //temp=dbp.getEnqId(t3.getText());

 // System.out.println(temp);

 JOptionPane.showMessageDialog(this, "Your Enquiry Reached us !!");

 }

 catch (Exception e)

 {

 JOptionPane.showMessageDialog(this, "There is some network issue");

 }

 }

 }

 }

}

DML Operations:

package com.applets;

import java.sql.\*;

import java.util.HashSet;

import java.util.Set;

//import java.io.\*;

class DMLOperations

{

 public Connection con;

 public int rowsEffected;

 public PreparedStatement pstmt;

 public ResultSet rs;

 public int createConnection()throws Exception

 {

 con=null;

 Class.forName("com.mysql.jdbc.Driver");

 System.out.println("Driver Loaded.....");

 String databaseURL = "jdbc:mysql://localhost:3306/javaclass";

 try {

 con = DriverManager.getConnection(databaseURL,"root","abcd@143");

 }

 catch(SQLException e)

 {

 System.out.println("hello."+e);

 }

 System.out.println("Database Connected ");

 return 0;

 }

 public ResultSet searchDet(int s)throws Exception

 {

 pstmt=null;

 pstmt=con.prepareStatement("select \* from emp where eno=?");

 pstmt.setInt(1, s);

 rs=pstmt.executeQuery();

 rs.first();

 //String t=rs.getString(2);

 //System.out.println(t);

 return rs;

 }

 public void saveData(String s1,String s2,String s3,String s4)throws Exception

 {

 pstmt=null;

 //LAST\_INSERT\_ID

 pstmt=con.prepareStatement("insert into feedback values(?,?,?,?)");

 pstmt.setString(1, s1);

 pstmt.setString(2, s2);

 pstmt.setString(3, s3);

 pstmt.setString(4, s4);

 rowsEffected=pstmt.executeUpdate();

 System.out.println(rowsEffected + " Rows Updated ");

 pstmt.close();

 con.close();

 }

}

/\*

 \* To change this license header, choose License Headers in Project Properties.

 \* To change this template file, choose Tools | Templates

 \* and open the template in the editor.

 \*/

package com.jdbc1;

import java.sql.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

 \*

 \* @author A B Pradeep

 \*/

class Database

{

 public Connection con;

 public int rowsEffected;

 public Statement stmt;

 public ResultSet rs;

}

public class ShowData

{

 public static void main(String arg[]) throws Exception

 {

 Database d = new Database();

 d.con=null;

 Class.forName("com.mysql.jdbc.Driver");

 System.out.println("Driver Loaded.....");

 String databaseURL = "jdbc:mysql://localhost:3306/javaclass";

 d.con = DriverManager.getConnection(databaseURL,"root","abcd@143");

 System.out.println("Database Connected ");

 d.stmt=null;

 d.stmt=d.con.createStatement(); //statement object created

 d.rs=null;

 d.rs=d.stmt.executeQuery("select \* from student"); //can execute only Select statments

 while(d.rs.next()) //as long as records available

 {

 System.out.println(d.rs.getString(1) + "\t" + d.rs.getString(2)+ "\t" + d.rs.getString(3)); //get values at ordinal positions of cols

 }

 d.rs.close(); //step 4

 d.stmt.close();

 d.con.close();

 }

}