E-Content-3
Introduction to Java

Shivneet Tripathi
Department of Computer Application
UIET , CSJM University,
Kanpur

String

- String is a class not a data type in java.There are two ways to create a string object.
- 1- Implicit
- 2 – Explicit
- Implicit– When you use a string literal , like “Hello world” , java automatically create a string object .
- Ex - String s = “Hello world”;
- Explicit - When you use the new operator to initiate a string object.
- Ex String s = new String (“Hello world”);

Example

```
Import java.lang.*;
public class String1{
        public static void main(String args[]){
                String s1="java";//creating string by java string litera
l
                char ch[]={'s','t','r','i','n','g','s'};
                String s2=new String(ch);//converting char array to s
tring
                String s3=new String("example");
                System.out.println(s1);
                System.out.println(s2);
                System.out.println(s3);
                }
}
```

Package

- A **java package** is a group of similar types of classes, interfaces and sub-packages.
- Package in java can be categorized in two form, built-in package and user-defined package.
- There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

Advantage of Java Package

- 1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.
- 2) Java package provides access protection.
- 3) Java package removes naming collision.

Example

```
package pack;
public class A{
  public void msg(){System.out.println("Hello");}
}
```

```
import pack.*;
class B{
  public static void main(String args[]){
   A obj = new A();
   obj.msg();
  }
}
```

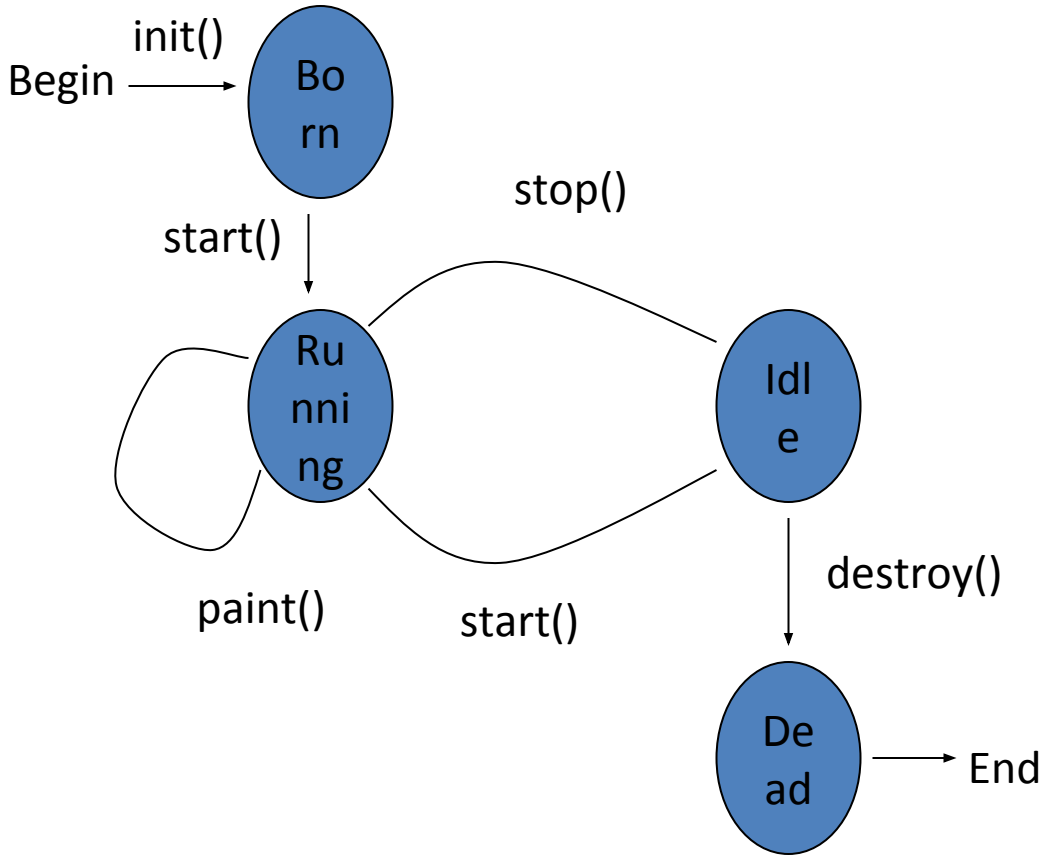
Java Applet

- Applets are small Java programs that are embedded in Web pages.
- They can be transported over the Internet from one computer (web server) to another (client computers).
- They transform web into rich media and support the delivery of applications via the Internet.
- **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.

How Applets Differ from Applications

- Although both the Applets and stand-alone applications are Java programs, there are certain restrictions are imposed on Applets due to security concerns:
  - Applets don't use the main() method, but when they are load, automatically call certain methods (init, start, paint, stop, destroy).
  - They are embedded inside a web page and executed in browsers.
  - They cannot read from or write to the files on local computer.
  - They cannot communicate with other servers on the network.
  - They cannot run any programs from the local computer.
  - They are restricted from using libraries from other languages.
- The above restrictions ensures that an Applet cannot do any damage to the local system.

Lifecycle of Java Applet



Applet Life Cycle

- Every applet inherits a set of default behaviours from the Applet class. As a result, when an applet is loaded, it undergoes a series of changes in its state. The applet states include:
  - Initialisation – invokes init()
  - Running – invokes start()
  - Display – invokes paint()
  - Idle – invokes stop()
  - Dead/Destroyed State – invokes destroy()

Applet States

- Initialisation – invokes init() – only once
  - Invoked when applet is first loaded.
- Running – invokes start() – more than once
  - For the first time, it is called automatically by the system after init() method execution.
  - It is also invoked when applet moves from idle/stop() state to active state. For example, when we return back to the Web page after temporary visiting other pages.
- Display – invokes paint() - more than once
  - It happens immediately after the applet enters into the running state. It is responsible for displaying output.
- Idle – invokes stop() - more than once
  - It is invoked when the applet is stopped from running. For example, it occurs when we leave a web page.
- Dead/Destroyed State – invokes destroy() - only once
  - This occurs automatically by invoking destroy() method when we quite the browser.

Building Applet Code: An Example

```
//HelloWorldApplet.java
import java.applet.Applet;
import java.awt.*;

public class HelloWorldApplet extends Applet {
        public void paint(Graphics g) {
                g.drawString ("Hello World of Java!",25,
    25);
        }
}
```

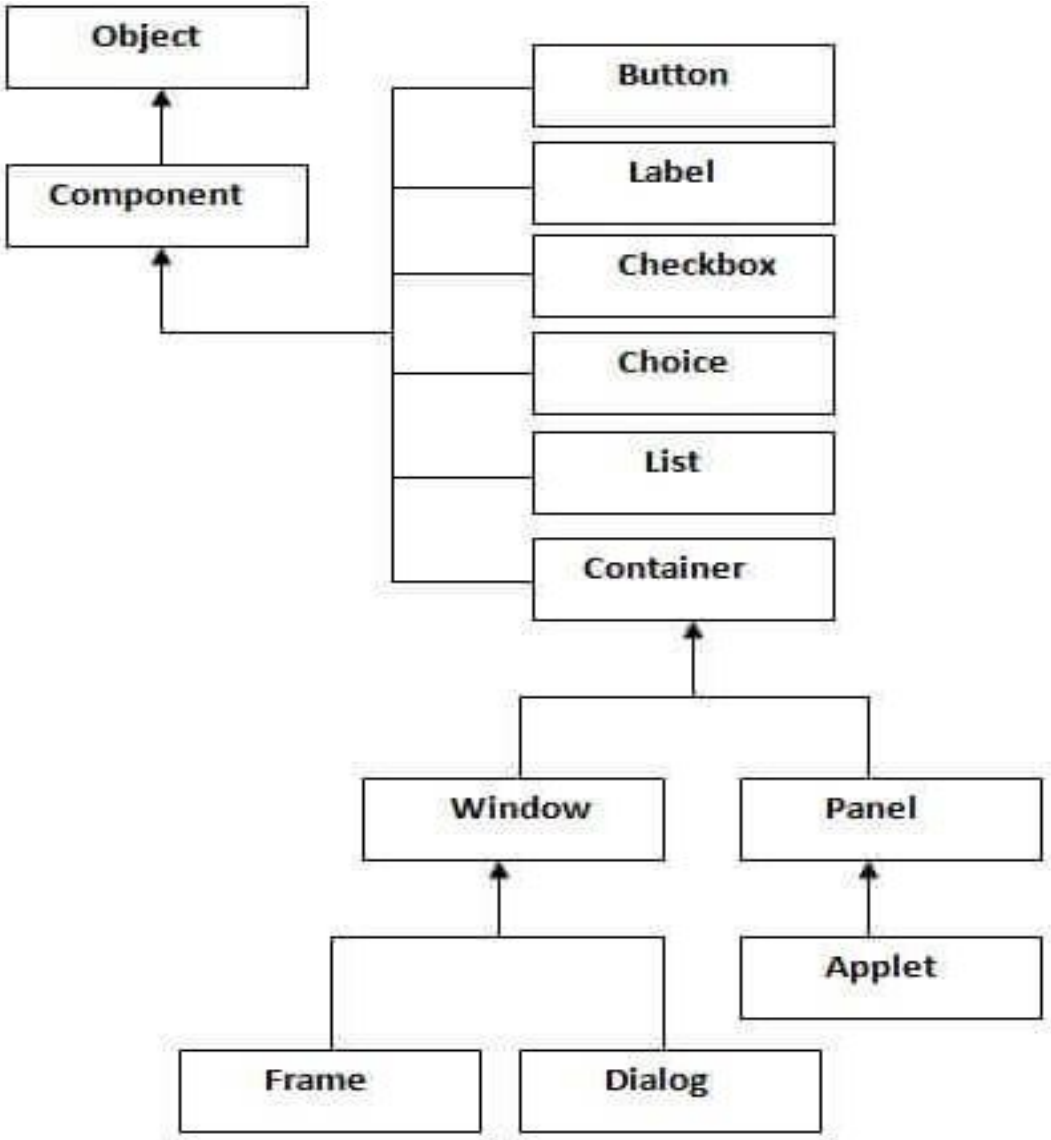
Embedding Applet in Web Page

```
<HTML>
<HEAD>
<TITLE>
  Hello World Applet
</TITLE>
</HEAD>
<body>
<h1>Hi, This is My First Java Applet on the Web!</h1>
<APPLET CODE="HelloWorldApplet.class" width=500
    height=400>
</APPLET>
</body>
</HTML>
```

Java AWT

- **Java AWT** (Abstract Window Toolkit) is *an API to develop GUI or window-based applications* in java.
- The java.awt package provides classes for AWT api such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.
- 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.

Java AWT Hierarchy



Cont…

Container

- The Container is a component in AWT that can contain another components like buttons, 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.

Example by inheritance

```
import java.awt.*;
class First extends Frame{
        First(){
            Button b=new Button("click me");
            b.setBounds(30,100,80,30);   // setting button position
            add(b);      //adding button into frame
            setSize(300,300);  //frame size 300 width and 300 height
            setLayout(null);     //no layout now bydefault BorderLayout
            setVisible(true);    //now frame willbe visible, bydefault not visible
        }
        public static void main(String args[]){
            First f=new First();
        }
}
```

Example by Association

```
import java.awt.*;
class First2{
        First2(){
            Frame f=new Frame();
            Button b=new Button("click me");
            b.setBounds(30,50,80,30);
            f.add(b);
            f.setSize(300,300);
            f.setLayout(null);
            f.setVisible(true);
            }
        public static void main(String args[]){
            First2 f=new First2();
        }
}
```

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	**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

Steps to perform Event Handling

- 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 by implementing ActionListener

```
import java.awt.*;
import java.awt.event.*;
class AEvent extends Frame implements ActionListener{
    TextField tf;
    AEvent(){
        tf=new TextField();
        tf.setBounds(60,50,170,20);
        Button b=new Button("click me");
        b.setBounds(100,120,80,30);
        b.addActionListener(this);
        add(b);add(tf);
        setSize(300,300);
        setLayout(null);
        setVisible(true);
    }
    public void actionPerformed(ActionEvent e){
        tf.setText("Welcome");
    }
    public static void main(String args[]){
        new AEvent();
    } }
```

Java AWT Button Example with ActionListener

- **import** java.awt.*;
- **import** java.awt.event.*;

```
public class ButtonExample {
        public static void main(String[] args) {
                Frame f=new Frame("Button Example");
                final TextField tf=new TextField();
                tf.setBounds(50,50, 150,20);
                Button b=new Button("Click Here");
                b.setBounds(50,100,60,30);
                b.addActionListener(new ActionListener(){
                        public void actionPerformed(ActionEvent e){
                            tf.setText("Welcome to Javatpoint.");
                          }
                });
                f.add(b);  f.add(tf);
                f.setSize(400,400);
                f.setLayout(null);
                f.setVisible(true);
        }

}
```