



Indian Institute of Technology Kharagpur

---

## Java Applets – Part I

Prof. Indranil Sen Gupta  
Dept. of Computer Science & Engg.  
I.I.T. Kharagpur, INDIA



### Lecture 28: Java applets – Part I

On completion, the student will be able to:

1. Explain the main design goals of java.
2. Justify why java is considered by many to be a language suited to be used in the Internet.
3. Explain the basic concept of a java applet, and how it is different from a java application.
4. Illustrate how an applet can be embedded in a HTML document.
5. Demonstrate the execution of simple applets.



## Introduction

---

- The language Java originated at SUN Microsystems.
  - By a small group of people who were trying to develop a new object-oriented language that would be suitable for consumer applications.
- Java is a fully object-oriented language.
- It is easier to write bug-free code in java than C++.



- Some bug-savvy features of C++ were eliminated:
  - No manual memory allocation/deallocation.
  - No pointer arithmetic.
  - No multiple inheritance.
  - No operator overloading.
  - No #define and typedef.



## Design Goals of Java

- In a published white paper, the following design goals were stated.
  - Java is simple.
  - Java is object-oriented.
  - Java is distributed.
  - Java is robust.
  - Java is secure.
  - Java is architecture neutral.
  - Java is portable.
  - Java is interpreted.



- Java is simple
  - It is basically a cleaned up version of C++.
  - Omits many rarely used and confusing features of C++.
  - The size of the basic Java runtime is under 200 Kbytes.
- Java is object-oriented
  - Features comparable with that of C++.
- Java is distributed
  - Comes with extensive class libraries for handling standard TCP/IP based applications.
  - Easy to write client-server applications.



### ➤ Java is robust

- Java compiler detects many errors that would possibly show up only at run time in some other languages.
- Use of pointers is safe; we can never access a bad pointer or make memory allocation errors.

### ➤ Java is secure

- The Java runtime disallows a number of things (to be discussed later) that may pose security threats.



### ➤ Java is architecture neutral

- Because of the fact that Java compilers generate byte code, which gets executed by Java runtime.

### ➤ Java is portable

- Java requires the sizes of primitive data to be specified.
- Binary data are stored in a fixed format, thereby avoiding the Big Endian / Little Endian confusion.
- Strings are stored in Unicode format.



### ➤ Java is interpreted

- The Java runtime (interpreter) can run on any machine / environment where it has been ported.
- The Java runtime is not platform independent; we need to have one for every possible environment.
- Existence of the Java runtime makes Java byte code platform independent.



## Platform Independence

- Java solves this problem using the concept of Java Virtual Machine (JVM).
  - JVM provides a virtual CPU and a virtual instruction set (called byte code).
  - The Java compiler transforms a Java source program into byte code.
  - A Java interpreter (called Java Runtime) converts the byte code instructions to native processor instructions and executes them.
    - Specific to environment and OS.



Java Source Program

Java  
Compiler

Java Byte Code

Java Runtime  
for  
Windows/XP

Java Runtime  
for  
Linux

Java Runtime  
for  
SUN Solaris

...



## Java and the Internet

- Java is considered to be a language well-suited to be used in the Internet.
  - In contrast with Java applications, there is something called Java applets.
    - Applets can be embedded within a HTML page.
    - Applets exist in byte code form on the server.
    - Applets are downloaded by the browser and executed on the client machine.
  - Since applets run (interpreted) on the client machine, response is much better as compared to a remote CGI script.



- A Java applet can be embedded within a HTML document using the `<APPLET>` tag.

```
<APPLET CODE=example.class  
        WIDTH=300 HEIGHT=250>  
<PARAM .....>  
</APPLET>
```



## Why do we need Applets?

- The language Java is very convenient for:
  - Programming involving graphics.
  - Interactive applications.
  - Network programming, etc.
- We need applets because:
  - They allow Java programs to be downloaded and run on the browsers themselves.
  - Attractive / interactive web pages.



- **Web based applications:**
  - **A client computer can just have the JVM.**
  - **Users can download and use whatever programs/utilities they need.**
    - **A word processor.**
    - **An email client.**
    - **Game programs.**
    - **Only limited by imagination .....**
- **The language C# is considered as an alternative to Java for developing Internet applications.**



## **Security Issues**

- **Applets are downloaded from a remote server and executed on the local machine.**
- **Following restrictions normally apply:**
  - **They cannot invoke any local executable.**
  - **They cannot access the local file system.**
  - **They can communicate with only the web server from where they were downloaded, and not with any other host.**
  - **They cannot access sensitive information on the local host, like user's name, email address, etc.**



- Not possible to have malicious code in Java applets.
  - Considered to be a plus point of Java.
  - Other alternatives are not considered to be as safe.
    - A malicious ASP code, for instance, can access sensitive information from the local file system.



## Java Applications / Applets

- Java applications are stand-alone programs like any native program.
  - Compiled as:  
`javac exampla.java`
  - Executed as:  
`java example.class`
  - The bytecode *example.class* is portable.



➤ **A simple Java application:**

```
// A rudimentary Java application
class example
{
    public static void main (String args[ ])
    {
        System.out.println ("Thank you \n");
    }
}
```



- In contrast, a Java applet is meant to be run from within a browser or using a special utility like the *appletviewer*.
  - Does not have a *main()* method.
  - Must have a HTML file from where it is linked.
  - Executed (using *appletviewer*) as:  
`appletviewer example.html`



➤ A simple Java applet:

```
// A rudimentary Java applet
import java.awt.Graphics;
import java.applet.Applet;
public class example1 extends Applet
{
    public void paint (Graphics g)
    {
        g.drawString ("Good Day!", 50,10);
    }
}
```



➤ The corresponding HTML file:

```
<HTML>
    <APPLET CODE=example1.class,
            WIDTH=150,
            HEIGHT=150>
    </APPLET>
    .....
    .....
</HTML>
```



## The <APPLET> Tag

- General syntax:

```
<APPLET CODE = applet class file
        WIDTH = pixels
        HEIGHT = pixels
        [CODEBASE = directory to look for]
        [ALT = alternate text]
        [NAME = name of current applet instance]
        [ALIGN = applet alignment]
        [VSPACE = pixels]
        [HSPACE = pixels]>
```



```
[<PARAM NAME = par. name VALUE = par. value>
.....
.....
<PARAM NAME = par. name VALUE = par. value>]

[HTML to be displayed in the absence of java]

</APPLET>
```



- **The attributes:**

- **CODE:** Specifies the name of the applet class file (no subdirectories allowed).
- **CODEBASE:** Specifies a subdirectory where the applet is to be found.
  - In general, can be a URL pointing to any web server.
- **ALT:** Specifies a text message which is displayed if the browser understands the <APPLET> tag but cannot run the applet due to some reason.
- **WIDTH, HEIGHT:** Specify the size of the applet display area in pixels.



- **NAME:** Specifies a name for the current applet instance.
  - Required when two applets on the same HTML page wants to communicate with each other.
  - To access an Applet subclass Applet\_A named “one” from another applet, we use:

```
Applet_A p = getAppletContext().getApplet (“one”);
```

Methods of Applet\_A can now be invoked using the pointer p.

- **ALIGN:** Specifies the alignment of the applet (LEFT, RIGHT, etc.).



- **VSPACE, HSPACE:** Specify in pixels the margin to be left in the vertical and horizontal directions for the applet.
- **PARAM, VALUE:** Used to pass parameters to an applet from the HTML page.

```
<APPLET CODE = "Exam.class" WIDTH=100 HEIGHT=100  
  <PARAM NAME = s_name VALUE = "Tarun Mitra">  
  <PARAM NAME = rollno VALUE = "99213">  
  <PARAM NAME = colorblind VALUE = false>  
</APPLET>
```



```
String name = getParameter ("s_name");  
int roll = Integer.parseInt (getParameter ("rollno"));  
boolean cb = Boolean.valueOf  
    (getParameter ("colorblind"));
```



## Some Applet Examples



### Example 1: draw lines

```
import java.awt.*;
import java.awt.Graphics;
import java.applet.*;

public class DrawLines extends Applet
{
    public void paint (Graphics g)
    {
        for (int i = 10; i < 300; i += 10) {
            int x1 = 300-i;  int y1 = 290;
            int x2 = 290 ;  int y2 = i;
            g.drawLine ( x1,y1, x2,y2 );
        } } }
```



## Example 2: play a sound clip

```
import java.applet.*;
import java.awt.*;
import java.net.*;

public class PlaySound extends Applet
{
    public void init()
    {
        URL sound;
        try {
            sound = new URL ("http://iitkgp.ac.in/anthem.wav");
        }
        catch (MalformedURLException e) {
            return;
        }
        play (sound);
    }
}
```



## Example 3 : display image

```
import java.applet.*;
import java.awt.*;
import java.awt.image.*;
import java.net.*;

public class SampleGif extends Applet
{
    Image picture = null;

    public void init() {
        URL url;
        try { url = new URL("http://iitkgp.ac.in/logo.gif"); }
        catch (MalformedURLException e) { return; }

        picture = getImage (url);
    }
}
```



```
public boolean imageUpdate (
    Image img, int infoflags,
    int x, int y,
    int width, int height )
{
    paint (getGraphics() );
    return true;
}

public void paint( Graphics g )
{
    if (pic != null) {
        g.drawImage (picture, 0, 0, this);
    }
}
}
```



# End of Lecture 28



## SOLUTIONS TO QUIZ QUESTIONS ON LECTURE 27



### Quiz Solutions on Lecture 27

1. How do you store and retrieve the value of a cookie?

**Example:**

```
document.cookie = "mod_date=" +  
    escape (document.lastModified);
```

2. How do you access a Java applet from Javascript code?

The applet created by an `<APPLET>` tag with a `NAME` attribute of `"myapplet"` can be referred to as `document.myapplet`



## Quiz Solutions on Lecture 27

---

3. What do the “window” and “document” objects represent?

The “document” object represents the HTML document.

The “window” object represents the window (or frame) where the document is being displayed.



## Quiz Solutions on Lecture 27

---

4. How do you change the background and foreground colors of a document?

By setting the properties *document.bgColor* and *document.fgColor* of the document object.



## QUIZ QUESTIONS ON LECTURE 28



## Quiz Questions on Lecture 28

1. Why is Java considered to be secure?
2. What makes Java programs platform independent?
3. What is the difference between Java byte code and Java run time?
4. What is the difference between a Java application and a Java applet?
5. How can you pass parameters to an applet from the HTML page?
6. Is it possible to invoke a method of some other applet from another applet on the same HTML page?