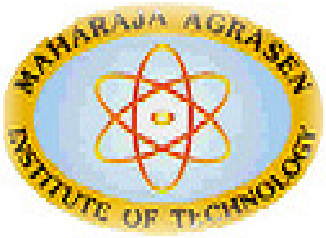
Java Programming & Website Design

ETIT 353

Lab Manual



Maharaja Agrasen Institute of Technology
PSP area,Sector – 22, Rohini, New Delhi – 110085
(Affiliated to Guru Gobind Singh Indraprastha University, Dwarka, New Delhi)

INDEX OF THE CONTENTS

1. INTRODUCTION TO THE LAB

Introduction to Java Programming and Web design lab is divided into two sections.

1. Java Programming
2. Web Design

JAVA PROGRAMMING

In java programming section, the applications of Java are taken into account. Applications of Java which are taken into details according to the syllabus prescribed by G.G.S.I.P.U for this lab are:

1. Console Based Programming
2. Applets
3. Java Beans
4. Servlets

WEB DESIGN

In web design section, HTML, DHTML & Java Script used to create a web page taken in details. Database connectivity is done by ASP. Overall the techniques explained under these sections are:

1. Html & Dhtml
2. Java Script
3. ASP

2. LAB REQUIREMENTS

For Java Programming

J2SDK 1.4
BDK/ Bean Builder
Tomcat 4.0
Java Compatible Web Browser

For Web Design

Web Browser Supporting Java Script
Windows 2000 with IIS server

3. LIST OF EXPERIMENTS (As prescribed by G.G.S.I.P.U)

Paper Code:	**P**	**C**
Paper: Java Programming Lab	**2**	**1**

1. Program to demonstrate constructor overloading and use of static members and block.

2. Program to implement multilevel inheritance and method overriding.

3. Program to illustrate class member access for packages and also implement interfaces.

4. Program for exception handling using multiple catch statements and also create your own exception.

5. Program to implement 3 threads such that first sleep for 200 ms, for 400 ms and third for 600 ms.

6. Program to create an applet of a moving banner.

7. Program to create a calculator.

8. Program to create a chatting application

9.Program to create a servlet in which user enters a name in edit box, after pressing submit button the name will be displayed on the next page

10. Program to create your own resume by using HTML

4. LIST OF ADVANCE EXPERIMENTS (Beyond the syllabus prescribed by G.G.S.I.P.U)

List of Advance experiments given to the students is summarized as below:

- Calendar
- Search & Replace in a file
- Scientific Calculator
- Address Book
- Paint Brush
- Note Pad

5. PROJECTS TO BE ALLOTED (Beyond the syllabus prescribed by G.G.S.I.P.U)

Students will be divided into a group of four/five and projects are allotted to those groups.
This project is to be submitted at the end of the semester along with a project report by the individual student.

List of projects given to the students is summarized as below:

- Dx Ball Game
- Moving ball with Java Script
- Checker Board game with Java Script
- Digital Image Processing
- Library Management System
- Global Defender Game
- Brick Game
- Rapid Roll Game
- Tic Tack Toe

Students can select project work of their own choice subject to the permission of concern faculty.

NOTE: The project is to be made in Java Language preferably.

6. FORMAT OF THE LAB RECORD TO BE PREPARED BY THE STUDENTS

1. The front page of the lab record prepared by the students should have a cover page as displayed below.

NAME OF THE LAB

Font should be(Size 20”, italics bold, Times New Roman)

Faculty name

(12”, Times Roman)

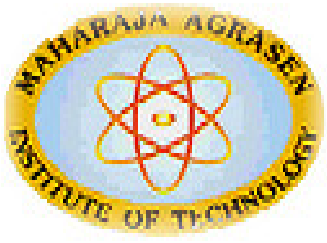
Student name Font should be

Roll No.:

Semester:

Group:

Font should be (12”, Times Roman)



Maharaja Agrasen Institute of Technology, PSP Area,

Sector – 22, Rohini, New Delhi – 110085

Font should be (18”, Times Roman)

2. The second page in the record should be the index as displayed below.

Java Programming
PRACTICAL RECORD

PAPER CODE :
Name of the student :

University Roll No. :

Branch :

Section/ Group :

PRACTICAL DETAILS

Experiments according to ITC lab syllabus prescribed by GGSIPU

Exp. no	Experiment Name	Date of performance	Date of checking	Remarks

PROJECT DETAILS

1. TITLE :
2. MEMBERS IN THE PROJECT GROUP :
3. PROJECT REPORT ATTACHED :
   a) YES b) NO
4. SOFT COPY SUBMITTED :
   a) YES b) NO

Signature of the lecturer Signature of the student

( ) ( )

3. Each practical which student is performing in the lab should have the following details :
   a) Topic Detail
   b) AIM
   c) Algorithm
   d) Source Code
   e) Output
   f) Viva questions

4. Project report should be added at last page.

7. MARKING SCHEME FOR THE PRACTICAL EXAMS

There will be two practical exams in each semester.

- Internal Practical Exam
- External Practical Exam

INTERNAL PRACTICAL EXAM

It is taken by the concerned lecturer of the batch.

MARKING SCHEME FOR THIS EXAM IS:

Total Marks: 40

Division of 40 marks is as follows

1. Regularity: 25

   - Performing program in each turn of the lab
   - Attendance of the lab
   - File

2. Viva Voice: 10

3. Project: 5

NOTE: For the regularity, marks are awarded to the student out of 10 for each experiment performed in the lab and at the end the average marks are giving out of 25.

EXTERNAL PRACTICAL EXAM

It is taken by the concerned lecturer of the batch and by an external examiner. In this exam student needs to perform the experiment allotted at the time of the examination, a sheet will be given to the student in which some details asked by the examiner needs to be written and at the last viva will be taken by the external examiner.

MARKING SCHEME FOR THIS EXAM IS:

Total Marks: 60

Division of 60 marks is as follows

1. Sheet filled by the student: 20
2. Viva Voice: 15
3. Experiment performance: 15
4. File submitted: 10

NOTE:

- Internal marks + External marks = Total marks given to the students
  (40 marks) (60 marks) (100 marks)
- Experiments given to perform can be from any section of the lab.

8. DETAILS OF THE EACH SECTION

ALONG WITH

EXAMPLES, EXERCISES

&

EXPECTED VIVA QUESTIONS

SECTION 1

Java Programming

THIS SECTION COVERS:

1. **Console Based Programming**
2. **Applets**
3. **Java Beans**
4. **Servlets**

SECTION 2

Web Programming

THIS SECTION COVERS:

1. **Html & Dhtml**

2. **Java Script**

3. **ASP**

Section 1

JAVA PROGRAMMING ENVIRONMENT (AN INTRODUCTION)

Anyone who is learning to program has to choose a programming environment that makes it possible to create and to run programs. Programming environments can be divided into two very different types: integrated development environments and command-line environments. All programming environments for Java require some text editing capability, a Java compiler, and a way to run applets and stand-alone applications. An integrated development environment, or IDE, is a graphical user interface program that integrates all these aspects of programming and probably others (such as a debugger, a visual interface builder, and project management). A command-line environment is just a collection of commands that can be typed in to edit files, compile source code, and run programs.

Command line environment is preferable for beginning programmers. IDEs can simplify the management of large numbers of files in a complex project, but they are themselves complex programs that add another level of complications to the already difficult task of learning the fundamentals of programming.

Java was developed at Sun Microsystems, Inc., and the primary source for information about Java is Sun's Java Web site, http://java.sun.com/. At this site, one can read documentation on-line and you can download documentation and software. The documentation includes the Java API reference and the Java tutorial.

The current version of Java on the Sun site is version 1.4. It is available for the Windows, Linux, and Solaris operating systems. One can download the "J2SE 1.4 SDK." This is the "Java 2 Platform Standard Edition Version 1.4 Software Development Kit." This package includes a Java compiler, a Java virtual machine that can be used to run Java programs, and all the standard Java packages. The JRE is the "Java Runtime Environment." It only includes the parts of the system that are needed to run Java programs. It does not have a compiler.

Integrated Development Environments

There are sophisticated IDEs for Java programming that are available.

- Eclipse IDE -- An increasingly popular professional development environment that supports Java development, among other things. Eclipse is itself written in Java. It is available from http://www.eclipse.org/.
- NetBeans IDE -- A pure Java IDE that should run on any system with Java 1.3 or later. NetBeans is a free, "open source" program. It is essentially the open source version of the next IDE. It can be downloaded from www.netbeans.org.
- Sun ONE Studio 4 for Java, Community Edition, for Linux, Solaris, Windows 2000, Windows NT, and Windows 98SE. This was formerly known as "Forte for Java", and it might be referred under that name. Again, it requires a lot of resources, with a 256 MB memory recommendation. Main site currently at http://www.sun.com/software/sundev/jde/index.html. It is available from there and on the J2SE download page, http://java.sun.com/j2se/1.4/download.html. The Community Edition is the free version.
- Borland JBuilder Personal Edition, for Linux, Solaris, MacOS X, Windows 2000, Windows XP, and Windows NT. Requires a lot of disk space & memory (256 MB memory recommended). Company Web page at http://www.borland.com. Jbuilder site at http://www.borland.com/jbuilder/index.html. The Personal Edition, which is free, has more than enough features for most programmers.
- BlueJ is a Java IDE written in Java that is meant particularly for educational use. It is available from http://www.bluej.org/.
- JCreator, for Windows. It looks like a nice lighter-weight IDE that works on top of Sun's SDK. There is a free version, as well as a shareware version. It is available at http://www.jcreator.com.

There are other products similar to JCreator, for Windows and for other operating systems.

Text Editors

To use a command-line environment for programming good text editor is needed. A programmer's text editor is a very different thing from a word processor. Most important, it saves work in plain text

files and it doesn't insert extra carriage returns beyond the ones you actually type. A good programmer's text editor will do a lot more than this. Here are some features to look for:

- Syntax coloring. Shows comments, strings, keywords, etc., in different colors to make the program easier to read and to help you find certain kinds of errors.
- Function menu. A pop-up menu that lists the functions in your source code. Selecting a function from this will take you directly to that function in the code.
- Auto-indentation. When you indent one line, the editor will indent following lines to match, since that's what you want more often than not when you are typing a program.
- Parenthesis matching. After typing a closing parenthesis the cursor jumps back to the matching parenthesis momentarily so one can see where it is. Alternatively, there might be a command that will highlight all the text between matching parentheses. The same thing works for brackets and braces.
- Indent Block and Unindent Block commands. These commands apply to a highlighted block of text. They will insert or remove spaces at the beginning of each line to increase or decrease the indentation level of that block of text. When you make changes in your program, these commands can help you keep the indentation in line with the structure of the program.
- Control of tabs. Don’t use tab characters for indentation. A good editor can be configured to insert multiple space characters when tab key is pressed.

There are many free text editors that have some or all of these features. **Jedit**,a programmer's text editor written entirely in Java. It requires Java 1.3 or better. It has many features listed above, and there are plug-ins available to add additional features. Since it is written in pure Java, it can be used on any operating system that supports Java 1.3. In addition to being a nice text editor, it shows what can be done with the Swing GUI. Jedit is free and can be downloaded from http://www.jedit.org.

On Linux, use **nedit**. It has all the above features, except a function menu. Under Linux, it is likely that *nedit* is included in distribution, although it may not have been installed by default. It can be downloaded from http://www.nedit.org/ and is available for many UNIX platforms in addition to Linux. Features such as syntax coloring and auto-indentation are not turned on by default. One can configure them in the Options menu. Use the "Save Options" command to make the configuration

permanent. Of course, as alternatives to nedit, the Gnome and KDE desktops for Linux have their own text editors.

Using the Java SDK

After installing Sun's Software Development Kit for Java, one can use the commands "javac", "java", and "appletviewer" for compiling and running Java programs and applets. These commands must be on the "path" where the operating system searches for commands.

Make a directory to hold Java programs. Create program with a text editor, or copy the program to be compiled into program directory

If program contains more than a few errors, most of them will scroll out of the window. In Linux and UNIX, a command window usually has a scroll bar that can be used to review the errors. In Windows 2000/NT/XP (but **not** Windows 95/98), one can save the errors in a file which can be viewed later in a text editor.

The command in Windows is

javac SourceFile.java >& errors.txt

The ">& errors.txt" redirects the output from the compiler to the file, instead of to the DOS window. It is possible to compile all the Java files in a directory at one time. Use the command "javac *.java".

After compiled class files are made, run application or applet. For running a stand-alone application -- one that has a main() routine -- use the "java" command from the SDK to run the application. If the class file that contains the main() routine is named Main. class, then run the program with the command:

java Main

SAMPLE CONSOLE BASED PROGRAM

The following program, For Demo, uses the general form of the for statement to print the numbers 1 through 10 to standard output:

Steps to write JAVA Program

1. Create JAVA file by text editor (eg vi editor).
2. Write the program as per JAVA syntax.
3. Save the file with .java extension.
4. Compile the file with JAVA compiler(javac filename.java) and create class file.
5. Run the class file with JAVA interpreter (java classname.class) and check the output.

Program Source Code:

```
class ForDemo {
    public static void main(String[] args){
        for(int i=1; i<11; i++){
            System.out.println("Count is: " + i);
        }
    }
}
```

The output of the program is:

```
Count is: 1
Count is: 2
Count is: 3
```

Count is: 4

Count is: 5

Count is: 6

Count is: 7

Count is: 8

Count is: 9

Count is: 10

JAVA APPLETS

Applet Basics

All applets are subclasses of **Applet**. Thus, all applets must import **java.applet**. Applets must also import **java.awt**. Recall that AWT stands for the Abstract Window Toolkit. Since all applets run in a window, it is necessary to include support for that window. Applets are not executed by the console-based Java run-time interpreter. Rather, they are executed by either a Web browser or an applet viewer. The figures shown in this chapter were created with the standard applet viewer, called **applet viewer**, provided by the SDK. But you can use any applet viewer or browser you like. Execution of an applet does not begin at **main( )**. Actually, few applets even have **main( )** methods. Instead, execution of an applet is started and controlled with an entirely different mechanism, which will be

explained shortly. Output to your applet's window is not performed by **System.out.println( )**. Rather, it is handled with various AWT methods, such as **drawString( )**, which outputs a string to a specified X,Ylocation. Input is also handled differently than in an application. Once an applet has been compiled, it is included in an HTML file using the APPLET tag. The applet will be executed by a Java-enabled web browser when it encounters the APPLET tag within the HTML file. To view and test an applet more conveniently, simply include a comment at the head of your Java source code file that contains the APPLET tag. This way, your code is documented with the necessary HTML statements needed by your applet, and you can test the compiled applet by starting the applet viewer with your Java source code file specified as the target.

Here is an example of such a comment:

```
/*

<applet code="MyApplet" width=200 height=60>

</applet>

*/
```

This comment contains an APPLET tag that will run an applet called **MyApplet** in a window that is 200 pixels wide and 60 pixels high.

Running Applets

If your program is an applet, then you need an HTML file to run it.

```
<applet code="MyApplet.class" width=300 height=200>
</applet>
```

The "appletviewer" command from the SDK can then be used to view the applet. If the file name is test.html, use the command

```
appletviewer test.html
```

This will only show the applet. It will ignore any text or images in the HTML file. In fact, need in the HTML file is a single applet tag. The applet will be run in a resizable window. Note also that applet can use standard output, System.out, to write messages to the command window. This can be useful for debugging applet.

Use the appletviewer command on any file, or even on a web page address. It will find all the applet tags in the file, and will open a window for each applet. If using a Web browser that does not support Java 2, one can use appletviewer to see the applets. For example, to see the applets, use the command

appletviewer http://../yourcode.html

Of course, it's also possible to view applets in a Web browser. Just open the html file that contains the applet tag for your applet. One problem with this is that if you make changes to the applet, quit the browser and restart it in order to get the changes to take effect. The browser's Reload command might not cause the modified applet to be loaded.

Sample Program

Below is the HelloWorld applet, which is a simple Java class that prints the string "Hello World" in small rectangle.

Steps to write Applet Program:

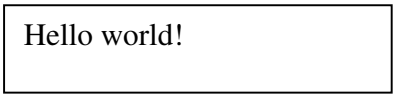
1. Create JAVA file by text editor(eg vi editor).
2. Write the program as per JAVA syntax.
3. Save the file with .java extension.
4. Compile the file with JAVA compiler (javac filename.java) and create class file.

5. Run the applet with appletviewer (appletviewer filename.java) and check the o/p.

Following is the source code for the HelloWorld applet:

```
import javax.swing.JApplet;

import java.awt.Graphics;

public class HelloWorld extends JApplet {

   public void paint(Graphics g) {

         g.drawRect(0, 0,

                  getSize().width - 1,

                  getSize().height - 1);

      g.drawString("Hello world!", 5, 15);

   }

}
```

Output for Applet Program:



JAVA BEANS

What Is a Java Bean?

A *Java Bean* is a software component that has been designed to be reusable in a variety of different environments. There is no restriction on the capability of a Bean. It may perform a simple function, such as checking the spelling of a document, or a complex function, such as forecasting the performance of a stock portfolio. A Bean may be visible to an end user. One example of this is a button on a graphical user interface. A Bean may also be invisible to a user. Software to decode a stream of multimedia information in real time is an example of this type of building block. Finally, a Bean may be designed to work autonomously on a user's workstation or to work in cooperation with a set of other distributed components. Software to generate a pie chart from a set of data points is an example of a Bean that can execute locally. However, a Bean that provides real-time price

information from a stock or commodities exchange would need to work in cooperation with other distributed software to obtain its data.

Advantages of Java Beans

Software component architecture provides standard mechanisms to deal with software building blocks. The following list enumerates some of the specific benefits that Java technology provides for a component developer:

■ A Bean obtains all the benefits of Java's "write-once, run-anywhere" paradigm.

■ The properties, events, and methods of a Bean that are exposed to an application builder tool can be controlled.

■ A Bean may be designed to operate correctly in different locales, which makes it useful in global markets.

■ Auxiliary software can be provided to help a person configure a Bean. This software is only needed when the design-time parameters for that componentare being set. It does not need to be included in the run-time environment.

■ The configuration settings of a Bean can be saved in persistent storage and restored at a later time.

■ A Bean may register to receive events from other objects and can generate events that are sent to other objects.

Bean Developer Kit (BDK)

The Bean Developer Kit (BDK), available from the JavaSoft site, is a simple example of a tool that enables you to create, configure, and connect a set of Beans. There is also a set of sample Beans with their source code. This section provides step-by-step instructions for installing and using this tool. BDK is for use with versions of Java 2 prior to 1.4.

Create a New Bean

Here are the steps that you must follow to create a new Bean:

1. Create a directory for the new Bean.

2. Create the Java source file(s).

3. Compile the source file(s).

4. Create a manifest file.

5. Generate a JAR file.

6. Start the BDK.

7. Test.

Source code for A simple Bean.

```
package sunw.demo.colors;

import java.awt.*;

import java.awt.event.*;

public class Colors extends Canvas {

transient private Color color;

private boolean rectangular;

public Colors() {

addMouseListener(new MouseAdapter() {
```

```
public void mousePressed(MouseEvent me) {
change();
}
});
rectangular = false;
setSize(200, 100);
change();
}
public boolean getRectangular() {
return rectangular;
}
public void setRectangular(boolean flag) {
this.rectangular = flag;
repaint();
}
public void change() {
color = randomColor();
repaint();
}
```

```
private Color randomColor() {
int r = (int)(255*Math.random());
int g = (int)(255*Math.random());
int b = (int)(255*Math.random());
return new Color(r, g, b);
}
public void paint(Graphics g) {
Dimension d = getSize();
int h = d.height;
int w = d.width;
g.setColor(color);
if(rectangular) {
g.fillRect(0, 0, w-1, h-1);
}
else {
g.fillOval(0, 0, w-1, h-1);
}
}
}
```

Compile the Source Code for the New Bean

Compile the source code to create a class file. Type the following:

javac Colors.java.

Create a Manifest File

You must now create a manifest file. First, switch to the **c:\bdk\demo** directory. This is the directory in which the manifest files for the BDK demos are located. Put the source code for your manifest file in the file **colors.mft**. It is shown here:

Name: sunw/demo/colors/Colors.class

Java-Bean: True

This file indicates that there is one **.class** file in the JAR file and that it is a Java Bean. Notice that the c**olors.class** file is in the package **sunw.demo.colors** and in the subdirectory **sunw\demo\colors** relative to the current directory.

Generate a JAR File

Beans are included in the ToolBox window of the BDK only if they are in JAR files in the directory **c:\bdk\jars**. These files are generated with the jar utility. Enter the following:

jar cfm ..\jars\colors.jar colors.mft sunw\demo\colors*.class

This command creates the file **colors.jar** and places it in the directory **c:\bdk\jars**.

Start the BDK

Change to the directory **c:\bdk\beanbox** and type **run**. This causes the BDK to start.You should see three windows, titled ToolBox, BeanBox, and Properties. The ToolBox window should include an entry labeled “Colors” for your new Bean.

Create an Instance of the Colors Bean

After you complete the preceding steps, create an instance of the **Colors** Bean in the BeanBox window. Test your new component by pressing the mouse anywhere within its borders. Its color immediately changes. Use the Properties window to change the **rectangular** property from **false** to **true**. Its shape immediately changes.

Create and Configure an Instance of the OurButton Bean

Create an instance of the **OurButton** Bean in the BeanBox window. Then follow these steps:

1. Go to the Properties window and change the label of the Bean to "Change". You should see that the button appearance changes immediately when this property is changed.

2. Go to the menu bar of the BeanBox and select Edit | Events | action | actionPerformed.

3. Move the cursor so that it is inside the **Colors** Bean display area, and click the left mouse button. You should see the Event Target Dialog dialog box.

4. The dialog box allows you to choose a method that should be invoked when this button is clicked. Select the entry labeled "change" and click the OK button. You should see a message box appear very briefly, stating that the tool is "Generating and compiling adaptor class."

5. Click on the button. You should see the color change. You might want to experiment with the **Colors** Bean a bit before moving on.

Using Bound Properties

A Bean that has a bound property generates an event when the property is changed. The event is of type **PropertyChangeEvent** and is sent to objects that previously registered an interest in receiving such notifications.

The **TickTock** Bean is supplied with the BDK. It generates a property change event every N seconds. N is a property of the Bean that can be changed via the Properties window of the BDK. The next

example builds an application that uses the **TickTock** Bean to automatically control the **Colors** Bean. Figure 25-4 shows how this application appears.

Steps

For this example, start the BDK and create an instance of the **Colors** Bean in the BeanBox window. Create an instance of the **TickTock** Bean. The Properties window should show one property for this component. It is "Interval" and its initial value is 5. This represents the number of seconds that elapse between property change events generated by the **TickTock** Bean. Change the value to 1.

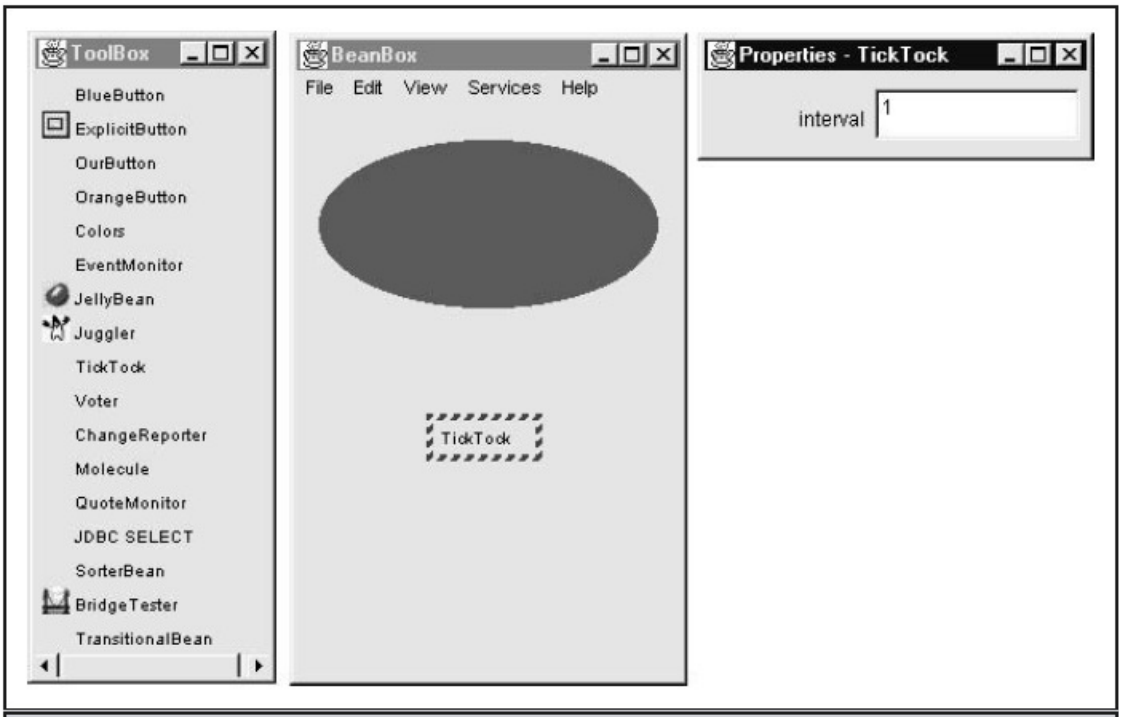


Figure 25-4. *The* Colors *and* TickTock *Beans*

Now you need to map events generated by the **TickTock** Bean into method calls onthe **Colors** Bean. Follow these steps:

1. Go to the menu bar of the BeanBox and select Edit | Events | propertyChange | propertyChange. You should now see a line extending from the button to the cursor.

2. Move the cursor so that it is inside the **Colors** Bean display area, and click the left mouse button. You should see the Event Target Dialog dialog box.

3. The dialog box allows you to choose a method that should be invoked when this event occurs. Select the entry labeled “change” and click the OK button.

You should see a message box appear very briefly, stating that the tool is “Generating and compiling adaptor class.” You should now see the color of your component change every second.

SERVLETS

Background

In order to understand the advantages of servlets, you must have a basic understanding of how Web browsers and servers cooperate to provide content to a user. Consider a request for a static Web page. A user enters a Uniform Resource Locator (URL) into a browser. The browser generates an HTTP request to the appropriate Web server. The Web server maps this request to a specific file. That file is returned in an HTTP response to the browser. The HTTP header in the response indicates the type of the content. The Multipurpose Internet Mail Extensions (MIME) are used for this purpose. For example, ordinary ASCII text has a MIME type of text/plain. The Hypertext Markup Language (HTML) source code of a Web page has a MIME type of text/html.

Now consider dynamic content. Assume that an online store uses a database to store information about its business. This would include items for sale, prices, availability, orders, and so forth. It wishes to make this information accessible to customers via Web pages. The contents of those Web pages must be dynamically generated in order to reflect the latest information in the database.

In the early days of the Web, a server could dynamically construct a page by creating a separate process to handle each client request. The process would open connections to one or more databases in order to obtain the necessary information. It communicated with the Web server via an interface known as the Common Gateway Interface (CGI). CGI allowed the separate process to read data from

the HTTP request and write data to the HTTP response. A variety of different languages were used to build CGI programs.

These included C, C++, and Perl. However, CGI suffered serious performance problems. It was expensive in terms of processor and memory resources to create a separate process for each client request. It was also expensive to open and close database connections for each client request. In addition, the CGI programs were not platform-independent. Therefore, other techniques were introduced. Among these are servlets.

Servlets offer several advantages in comparison with CGI. First, performance is significantly better. Servlets execute within the address space of a Web server. It is not necessary to create a separate process to handle each client request. Second, servlets are platform-independent because they are written in Java. A number of Web servers from different vendors offer the Servlet API. Programs developed for this API can be

moved to any of these environments without recompilation. Third, the Java security manager on the server enforces a set of restrictions to protect the resources on a server machine. You will see that some servlets are trusted and others are untrusted. Finally, the full functionality of the Java class libraries is available to a servlet. It can communicate with applets, databases, or other software via the sockets and RMI mechanisms that you have seen already.

The Life Cycle of a Servlet

Three methods are central to the life cycle of a servlet. These **are init( ), service( ),and destroy( )**. They are implemented by every servlet and are invoked at specific times by the server. Let us consider a typical user scenario to understand when these methods are called.

First, assume that a user enters a Uniform Resource Locator (URL) to a Web browser. The browser then generates an HTTP request for this URL. This request is then sent to the appropriate server. Second, this HTTP request is received by the Web server. The server maps this request to a particular servlet. The servlet is dynamically retrieved and loaded into the address space of the server. Third, the server invokes the init( ) method of the servlet. This method is invoked only when the servlet is first loaded into memory. It is possible to pass initialization parameters to the servlet so it may

configure itself. Fourth, the server invokes the service( ) method of the servlet. This method is called to process the HTTP request. You will see that it is possible for the servlet to read data that has been provided in the HTTP request. It may also formulate an HTTP response for the client. The servlet remains in the server's address space and is available to process any other HTTP requests received from clients. The service( ) method is called for each HTTP request. Finally, the server may decide to unload the servlet from its memory. The algorithms by which this determination is made are specific to each server. The server calls the destroy( ) method to relinquish any resources such as file handles that are allocated for the servlet. Important data may be saved to a persistent store. The memory allocated for the servlet and its objects can then be garbage collected.

Using Tomcat For Servlet Development

To create servlets, you will need to download a servlet development environment. The one currently recommended by Sun is Tomcat 4.0, which supports the latest servlet specification, which is 2.3. (The complete servlet specification is available for download through java.sun.com.) Tomcat replaces the old JSDK (Java Servlet Development Kit) that was previously provided by Sun. Tomcat is an open-source product maintained by the Jakarta Project of the Apache Software Foundation. It contains the class libraries, documentation, and run-time support that you will need to create and test servlets. You can download Tomcat through the Sun Microsystems Web site at java.sun.com.The current version is 4.0. Follow the instructions to install this toolkit on your machine. The examples in this chapter assume a Windows environment. The default location for Tomcat 4.0 is

C:\Program Files\Apache Tomcat 4.0

This is the location assumed by the examples in this book. If you load Tomcat in a different location, you will need to make appropriate changes to the examples. You may need to set the environmental variable JAVA_HOME to the top-level directory in which the Java Software Development Kit is installed. For Java 2, version 1.4, the default directory is C:\j2sdk1.4.0, but you will need to confirm this for your environment.

To start Tomcat, select Start Tomcat in the Start | Programs menu, or run startup.bat from the

C:\Program Files\Apache Tomcat 4.0\bin

directory. When you are done testing servlets, you can stop Tomcat by selecting Stop Tomcat in the Start | Programs menu, or run shutdown.bat. The directory

C:\Program Files\Apache Tomcat 4.0\common\lib

contains servlet.jar. This JAR file contains the classes and interfaces that are needed to build servlets. To make this file accessible, update your CLASSPATH environment variable so that it includes

C:\Program Files\Apache Tomcat 4.0\common\lib\servlet.jar.

Alternatively, you can specify this class file when you compile the servlets. For example, the following command compiles the first servlet example:

javac HelloServlet.java -classpath "C:\Program Files\Apache Tomcat

4.0\common\lib\servlet.jar"

Once you have compiled a servlet, you must copy the class file into the directory that Tomcat uses for example servlet class files. For the purposes of this chapter, you must put the servlet files into the following directory:

C:\Program Files\Apache Tomcat 4.0\webapps\examples\WEB-INF\classes

A Simple Servlet

To become familiar with the key servlet concepts, we will begin by building and testing a simple servlet. The basic steps are the following:

1. Create and compile the servlet source code.

2. Start Tomcat.

3. Start a Web browser and request the servlet.

Let us examine each of these steps in detail.

Create and Compile the Servlet Source Code

To begin, create a file named HelloServlet.java that contains the following program:

```
import java.io.*;

import javax.servlet.*;

public class HelloServlet extends GenericServlet {

public void service(ServletRequest request,

ServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter pw = response.getWriter();

pw.println("<B>Hello!");

pw.close();

}

}
```

Let's look closely at this program. First, note that it imports the javax.servlet package.This package contains the classes and interfaces required to build servlets. You will learn more about these later in this chapter. Next, the program defines HelloServlet as a subclass of GenericServlet. The GenericServlet class provides functionality that makes it easy to handle requests and responses.

Inside HelloServet, the service( ) method (which is inherited from GenericServlet) is overridden. This method handles requests from a client. Notice that the first argument is a ServletRequest object. This

enables the servlet to read data that is provided via the client request. The second argument is a ServletResponse object. This enables the servlet to formulate a response for the client.

The call to setContentType( ) establishes the MIME type of the HTTP response. In this program, the MIME type is text/html. This indicates that the browser should interpret the content as HTML source code.Next, the getWriter( ) method obtains a PrintWriter. Anything written to this stream is sent to the client as part of the HTTP response. Then println( ) is used to write some simple HTML source code as the HTTP response.

Compile this source code and place the HelloServlet.class file in the Tomcat class files directory as described in the previous section.

Start Tomcat

As explained, to start Tomcat, select Start Tomcat in the Start | Programs menu, or run startup.bat from the

C:\Program Files\Apache Tomcat 4.0\bin

directory.

Start a Web Browser and Request the Servlet Start a Web browser and enter the URL shown here:

http://localhost:8080/examples/servlet/HelloServlet

Alternatively, you may enter the URL shown here:

http://127.0.0.1:8080/examples/servlet/HelloServlet

This can be done because 127.0.0.1 is defined as the IP address of the local machine.You will observe the output of the servlet in the browser display area. **It will contain the string Hello! in bold type**

VIVA QUESTINS BASED ON JAVA PROGRAMMING

Q1. **What is the difference between an Interface and an Abstract class?**

Ans An abstract class can have instance methods that implement a default behavior. An Interface can only declare constants and instance methods, but cannot implement default behavior and all methods are implicitly abstract. An interface has all public members and no implementation. An abstract class is a class which may have the usual flavors of class members (private, protected, etc.), but has some abstract methods.

Q2. **What is the purpose of garbage collection in Java, and when is it used?**

Ans The purpose of garbage collection is to identify and discard objects that are no longer needed by a program so that their resources can be reclaimed and reused. A Java object is subject to garbage collection when it becomes unreachable to the program in which it is used.

Q3. **Describe synchronization in respect to multithreading.**

Ans With respect to multithreading, synchronization is the capability to control the access of multiple threads to shared resources. Without synchonization, it is possible for one thread to modify a shared variable while another thread is in the process of using or updating same shared variable. This usually leads to significant errors.

Q4. **Explain different way of using thread?**

Ans The thread could be implemented by using runnable interface or by inheriting from the Thread class. The former is more advantageous, 'cause when you are going for multiple inheritance..the only interface can help.

Q5. **What are pass by reference and passby value?**

Ans Pass By Reference means the passing the address itself rather than passing the value. Passby Value means passing a copy of the value to be passed.

Q6. **What is HashMap and Map?**

Ans Map is Interface and Hashmap is class that implements that.

Q7. **Difference between HashMap and HashTable?**

Ans The HashMap class is roughly equivalent to Hashtable, except that it is unsynchronized and permits nulls. (HashMap allows null values as key and value whereas Hashtable doesnt allow). HashMap does not guarantee that the order of the map will remain constant over time. HashMap is unsynchronized and Hashtable is synchronized.

Q8. **Difference between Vector and ArrayList?**

Ans Vector is synchronized whereas arraylist is not.

Q9. **Difference between Swing and Awt?**

Ans AWT are heavy-weight componenets. Swings are light-weight components. Hence swing works faster than AWT.

Q10. **What is an Iterator?**

Ans A constructor is a member function of a class that is used to create objects of that class. It has the same name as the class itself, has no return type, and is invoked using the newoperator.
A method is an ordinary member function of a class. It has its own name, a return type (which may be void), and is invoked using the dot operator.

Section 2

HTML

HTML stands for **H**yper **T**ext **M**arkup **L**anguage

An HTML file is a text file containing small **markup tags**

The markup tags tell the Web browser **how to display** the page

An HTML file must have an **htm** or **html** file extension

An HTML file can be created using a **simple text editor**

Basic HTML Tags

Tag	**Description**
<A ...> Anchor	HREF: URL you are linking to
	NAME: name a section of the page
	TARGET = "_blank" | "_parent" | "_self" | "_top" | *window name* which window the document should go in
	TITLE: suggested title for the document to be opened
	onClick: script to run when the user clicks on this anchor
	onMouseOver: when the mouse is over the link
	onMouseOut: when the mouse is no longer over the link
	ACCESSKEY
<BODY ...>	BGCOLOR: background color of the page
	BACKGROUND: background picture for the page
	TEXT: color of the text on the page
	LINK: color of links that haven't been followed yet
	VLINK: color of links that have been followed
	ALINK: color of links while you are clicking on them
	BGPROPERTIES = FIXED if the background image should not scroll
	TOPMARGIN: size of top and bottom margins
	LEFTMARGIN: size of left and right margins
	MARGINHEIGHT: size of top and bottom margins

	MARGINWIDTH: size of left and right margins
	onLoad: Script to run once the page is fully loaded
	onUnload
	onFocus
	onBlur
	STYLESRC: MS FrontPage extension
	SCROLL = YES | NO If the document should have a scroll bar
<BR ...> Line Break	CLEAR = LEFT | RIGHT | ALL | BOTH go past a picture or other object
<CAPTION ...>	ALIGN = TOP | BOTTOM | LEFT | RIGHT alignment of caption to table
	VALIGN = TOP | BOTTOM if caption should be above or below table
<CENTER ...>	
<CITE> Citation	
<CODE>	
<FONT ...>	SIZE: size of the font
	COLOR: color of the text
	FACE: set the typestyle for text
	POINT-SIZE
	WEIGHT
<FORM ...>	ACTION: URL of the CGI program
	METHOD = GET | POST how to transfer the data to the CGI
	NAME: name of this form
	ENCTYPE = "multipart/form-data" | "application/x-www-form-urlencoded" | "text/plain" what type of form this is
	TARGET = "_blank" | "_parent" | "_self" | "_top" | *frame name* what frames to put the results in
	onSubmit: script to run before the form is submitted
	onReset: script to run before the form is reset

<HR ...> Horizontal Rule	NOSHADE: don't use shadow effect
	SIZE: height
	WIDTH: horizontal width of the line
	ALIGN = LEFT | RIGHT | CENTER horizontal alignment of the line
	COLOR: color of the line
<HTML>	
<MARQUEE ...>	WIDTH: how wide the marquee is
	HEIGHT: how tall the marquee is
	DIRECTION = LEFT | RIGHT which direction the marquee should scroll
	BEHAVIOR = SCROLL | SLIDE | ALTERNATE what type of scrolling
	SCROLLDELAY: how long to delay between each jump
	SCROLLAMOUNT: how far to jump
	LOOP = INFINITE | *number of loops* how many times to loop
	BGCOLOR: background color
	HSPACE: horizontal space around the marquee
	VSPACE: vertical space around the marquee
<TABLE ...>	BORDER: size of border around the table
	CELLPADDING: space between the edge of a cell and the contents
	CELLSPACING: space between cells
	WIDTH: width of the table as a whole
	BGCOLOR: color of the background
	BACKGROUND: picture to use as background
	ALIGN = LEFT | RIGHT alignment of table to surrounding text
	HSPACE: horizontal space between table and surrounding text
	VSPACE: vertical space between table and surrounding text
	HEIGHT: height of the table as a whole

	FRAME = VOID | BOX | BORDER | ABOVE | BELOW | LHS | RHS | HSIDES | VSIDES parts of outside border that are visible
	RULES = NONE | ALL | COLS | ROWS | GROUPS if there should be internal borders
	BORDERCOLOR: color of border around the table
	BORDERCOLORLIGHT: color of "light" part of border around the table
	BORDERCOLORDARK: color of "dark" part of border around the table
	SUMMARY: Summary of the purpose of the table
<TBODY ...> Table Body Section	
<TD ...> Table Data	ALIGN = LEFT | CENTER | MIDDLE | RIGHT horizontal alignment of cell contents
	VALIGN = TOP | MIDDLE | CENTER | BOTTOM | BASELINE vertical alignment of cell contents
	WIDTH: width of cell
	HEIGHT: height of cell
	COLSPAN: number of columns to cover
	ROWSPAN: number of rows to cover
	NOWRAP: don't word wrap
	BGCOLOR: color of the background
	BORDERCOLOR: color of border around the table
	BORDERCOLORDARK: color of "dark" part of border around the table
	BORDERCOLORLIGHT: color of "light" part of border around the table
	BACKGROUND: picture to use as background
<TR ...> Table Row	ALIGN = LEFT | CENTER | RIGHT horizontal alignment of cell contents
	HALIGN = LEFT | CENTER | RIGHT
	VALIGN = TOP | MIDDLE | BOTTOM | BASELINE vertical alignment of cell contents
	BGCOLOR: background color
	BACKGROUND: background image

	BORDERCOLOR: color of border around each cell
	BORDERCOLORLIGHT: color of "light" part of border around each cell
	BORDERCOLORDARK: color of "dark" part of border around each cell

Sample Program

Table cells that span more than one row/column
(This sample program demonstrates how to define table cells that span more than one row or one column.)

Steps to write HTML Program :

1. Create HTML file by using vi editor.

2. Define Tags HTML,body tags.

3. Inside body tag ,define the table tags.

4. Save the file with .html extension

5.Open the file in the browser.Browser will show the output as shown in the sample program output**.**

Program Source Code :

```
<html>

<body>

<h4>Cell that spans two columns:</h4>

<table border="1">
```

```
<tr>
  <th>Name</th>
  <th colspan="2">Telephone</th>
</tr>
<tr>
  <td>Bill Gates</td>
  <td>555 77 854</td>
  <td>555 77 855</td>
</tr>
</table>
<h4>Cell that spans two rows:</h4>
<table border="1">
<tr>
  <th>First Name:</th>
  <td>Bill Gates</td>
</tr>
<tr>
  <th rowspan="2">Telephone:</th>
  <td>555 77 854</td>
</tr>
<tr>
  <td>555 77 855</td>
</tr>
```

```
</table>

</body>

</html>
```

Output :

Cell that spans two columns:

Name	**Telephone**	
Bill Gates	555 77 854	555 77 855

Cell that spans two rows:

First Name:	Bill Gates
Telephone:	555 77 854
	555 77 855

DHTML

DHTML stands for **D**ynamic **HTML**.

DHTML is not a standard defined by the World Wide Web Consortium (W3C). DHTML is a "marketing term" - used by Netscape and Microsoft to describe the new technologies the 4.x generation browsers would support.

DHTML is a combination of technologies used to create dynamic Web sites.

DHTML Technologies

With DHTML a Web developer can control how to display and position HTML elements in a browser window.

HTML 4.0

With HTML 4.0 all formatting can be moved out of the HTML document and into a separate **style sheet**. Because HTML 4.0 separates the presentation of the document from its structure, we have total control of presentation layout without messing up the document content.

Cascading Style Sheets (CSS)

With CSS we have a style and layout model for HTML documents.

CSS was a breakthrough in Web design because it allowed developers to control the style and layout of multiple Web pages all at once. As a Web developer you can define a style for each HTML element and apply it to as many Web pages as you want. To make a global change, simply change the style, and all elements in the Web are updated automatically.

The Document Object Model (DOM)

DOM stands for the **D**ocument **O**bject **M**odel.

The HTML DOM is the Document Object Model for HTML.

The HTML DOM defines a standard set of objects for HTML, and a standard way to access and manipulate HTML objects.

"The W3C Document Object Model (DOM) is a platform and language neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document".

JavaScript allows you to write code to control all HTML elements.

DHTML Technologies in Netscape 4.x and Internet Explorer 4.x

Netscape 4.x	**Cross-Browser DHTML**	**Internet Explorer 4.x**
JSS (JavaScript Style Sheets) (allows you to control how different HTML elements will be displayed) Layers (allows you to control	CSS1 CSS2 (allows you to control how different HTML elements will be displayed)	Visual Filters (allow you to apply visual effects to text and graphics) Dynamic CSS (allows you to control element positioning and

element positioning and visibility)	CSS Positioning (allows you to control element positioning and visibility) JavaScript	visibility

Sample Program

How to make an element invisible. Do you want the element to show or not?

Steps to write DHTML Program :

1. Create HTML file by using vi editor.

2. Define Tags HTML,body tags.

3. Inside body tag ,define the table tags.

4. Save the file with .html extension

5.Open the file in the browser.Browser will show the output as shown in the sample program output**.**

Program Source Code :

```
<html>

<head>

<style>

h1.one

{
```

```
visibility:visible;
}
h1.two
{
visibility:hidden;
}
</style>

</head>
<body>
<h1 class="one">Heading one</h1>
<h1 class="two">Heading two</h1>
<p>Where is heading two?</p>

</body>
</html>
```

Output :

Heading one

Heading two

Where is heading two?

Questions :

What is CSS?

CSS stands for Cascading Style Sheets and is a simple styling language which allows attaching style to HTML elements. Every element type as well as every occurance of a specific element within that type can be declared an unique style, e.g. margins, positioning, color or size.

What are Style Sheets?

Style Sheets are templates, very similar to templates in desktop publishing applications, containing a collection of rules declared to various selectors (elements).

What is external Style Sheet? How to link?

External Style Sheet is a template/document/file containing style information which can be linked with any number of HTML documents. This is a very convenient way of formatting the entire site as well as restyling it by editing just one file.

The file is linked with HTML documents via the LINK element inside the HEAD element. Files containing style information must have extension `.css`, e.g. `style.css`.

```
<HEAD>
<LINK REL=STYLESHEET HREF="style.css" TYPE="text/css">
</HEAD>
```

What is embedded style? How to link?

Embedded style is the style attached to one specific document. The style information is specified as a content of the `STYLE` element inside the `HEAD` element and will apply to the entire document.

```
<HEAD>
<STYLE TYPE="text/css">
<!--
P {text-indent: 10pt}
-->
</STYLE>
</HEAD>
```

Note: The styling rules are written as a HTML comment, that is, between <!-- and --> to hide the content in browsers without CSS support which would otherwise be displayed.

What is inline style? How to link?

Inline style is the style attached to one specific element. The style is specified directly in the start tag as a value of the `STYLE` attribute and will apply exclusively to this specific element occurance.

```
<P STYLE="text-indent: 10pt">Indented paragraph</P>
```

What is imported Style Sheet? How to link?

Imported Style Sheet is a sheet that can be imported to (combined with) another sheet. This allows creating one main sheet containing declarations that apply to the whole site and partial sheets containing declarations that apply to specific elements (or documents) that may require additional styling. By importing partial sheets to the main sheet a number of sources can be combined into one.

To import a style sheet or style sheets include the @import notation or notations in the STYLE element. The @import notations must come before any other declaration. If more than one sheet is imported they will cascade in order they are imported - the last imported sheet will override the next last; the next last will override the second last, and so on. If the imported style is in conflict with the rules declared in the main sheet then it will be overridden.

<LINK REL=STYLESHEET HREF="main.css" TYPE="text/css">

<STYLE TYPE="text=css">

<!--

@import url(http://www.and.so.on.partial1.css);

@import url(http://www.and.so.on.partial2.css);

.... other statements

-->

</STYLE>

What is alternate Style Sheet? How to link?

Alternate Style Sheet is a sheet defining an *alternate* style to be used in place of style(s) declared as *persistent and/or preferred* .

Persistent style is a default style that applies when style sheets are enabled but can disabled in favor of an alternate style, e.g.:

<LINK REL=Stylesheet HREF="style.css" TYPE="text/css">

Preferred style is a default style that applies automatically and is declared by setting the TITLE attribute to the LINK element. There can only be one preferred style, e.g.:

<LINK REL=Stylesheet HREF="style2.css" TYPE="text/css" TITLE="appropriate style description">

Alternate style gives an user the choice of selecting an alternative style - a very convenient way of specifying a media dependent style. Note: Each group of alternate styles must have unique TITLE, e.g.:

<LINK REL="Alternate Stylesheet" HREF="style3.css" TYPE="text/css" TITLE="appropriate style description" MEDIA=screen>
<LINK REL="Alternate Stylesheet" HREF="style4.css" TYPE="text/css" TITLE="appropriate style description" MEDIA=print>

Alternate stylesheets are not yet supported.

JavaScript

JavaScript was designed to add interactivity to HTML pages

JavaScript is a scripting language (a scripting language is a lightweight programming language)

A JavaScript consists of lines of executable computer code

A JavaScript is usually embedded directly into HTML pages

JavaScript is an interpreted language (means that scripts execute without preliminary compilation)

Everyone can use JavaScript without purchasing a license

Are Java and JavaScript the Same?

NO!

Java and JavaScript are two completely different languages in both concept and design!

Java (developed by Sun Microsystems) is a powerful and much more complex programming language - in the same category as C and C++.

What can a JavaScript Do?

JavaScript gives HTML designers a programming tool - HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages

JavaScript can put dynamic text into an HTML page - A JavaScript statement like this: document.write("<h1>" + name + "</h1>") can write a variable text into an HTML page

JavaScript can react to events - A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element

JavaScript can read and write HTML elements - A JavaScript can read and change the content of an HTML element

JavaScript can be used to validate data - A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing

JavaScript can be used to detect the visitor's browser - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser

JavaScript can be used to create cookies - A JavaScript can be used to store and retrieve information on the visitor's computer

The Advantages and Disadvantages of JavaScript

We've already seen some of the advantages of JavaScript, like cross-browser support, validating data on the client, and being able to create more sophisticated user interfaces.

JavaScript effects are also much faster to download than some other front-end technologies like Flash and Java applets. In fact, unless you're writing a massive JavaScript application, it's quite likely that no significant extra download time will be added to a page by using JavaScript on it. Nor do users need to download a plugin before they can view your JavaScript, as they would with Flash for example, they simply need a browser that supports it – and, of course, most modern browsers do.

Other advantages include the fact that you don't need any extra tools to write JavaScript, any plain text or HTML editor will do, so there's no expensive development software to buy. It's also an easy language to learn, and there's a thriving and supportive online community of JavaScript developers and information resources.

The disadvantages of JavaScript, as with most web development, are almost entirely related to browser compatibility.

While the advances in browser programmability we've seen over recent years are, generally speaking, a good thing, if you don't implement them with care you can create a lot of inconsistencies and broken pages quite unintentionally using JavaScript. Code that works just great on IE4 might not work at all on Netscape 4, what works in NN6 doesn’t always work in NN 4, and so on.

In essence, there are two main problems with JavaScript and browsers:

· The different JavaScript versions in different browsers.

· Browser programmability: the HTML elements and features of the browser that can be accessed through any scripting language. (IE4 , for example, makes most of the page and HTML accessible to scripts, but Navigator 4 limits what can be accessed and manipulated.)

The HTML <script> tag is used to insert a JavaScript into an HTML page.

How to Put a JavaScript Into an HTML Page

```
<html>

<body>

<script type="text/javascript">

document.write("Hello World!")

</script>

</body>

</html>
```

The code above will produce this output on an HTML page:

```
Hello World!
```

Example Explained

To insert a JavaScript into an HTML page, we use the <script> tag (also use the type attribute to define the scripting language).

So, the <script type="text/javascript"> and </script> tells where the JavaScript starts and ends:

```
<html>

<body>

<script type="text/javascript">

...

</script>

</body>

</html>
```

The word **document.write** is a standard JavaScript command for writing output to a page.

By entering the document.write command between the <script type="text/javascript"> and </script> tags, the browser will recognize it as a JavaScript command and execute the code line. In this case the browser will write Hello World! to the page:

```
<html>

<body>

<script type="text/javascript">

document.write("Hello World!")

</script>

</body>

</html>
```

Note: If we had not entered the <script> tag, the browser would have treated the document.write("Hello World!") command as pure text, and just write the entire line on the page.

Ending Statements With a Semicolon?

With traditional programming languages, like C++ and Java, each code statement has to end with a semicolon.

Many programmers continue this habit when writing JavaScript, but in general, semicolons are **optional**! However, semicolons are required if you want to put more than one statement on a single line.

Sample Program

Function with arguments, that returns a value

1. Create HTML file by using vi editor.

2. Define Tags HTML,body tags.

3. Inside body tag ,define the table tags.

4. Save the file with .html extension

5.Open the file in the browser.Browser will show the output as shown in the sample program output**.**

Program Source Code :

```
<html>

<head>

<script type="text/javascript">

function product(a,b)

{

return a*b

}

</script>

</head>


<body>
```

```
<script type="text/javascript">

document.write(product(4,3))

</script>

<p>The script in the body section calls a function with two parameters (4 and 3).</p>

<p>The function will return the product of these two parameters.</p>

</body>

</html>
```

Output :

12

The script in the body section calls a function with two parameters (4 and 3).

The function will return the product of these two parameters.

Questions :

Q1.How do you get the page background image to stay fixed when the page is scrolled?

Ans.The technique is called *watermarking*.

One simple way is to add bgproperties="fixed" to the body tag, like this:

```
<body bgproperties="fixed">
```

Note that this typically only works in Internet Explorer browsers.

Another way of doing it that also works in later Netscape browsers (6.x & up) is to add this style script to the <head> of your page:

```
<style>
body {background-attachment:fixed}
</style>
```

Q2.How do you call more than one JavaScript function in a body tag (or other) event handler?

Ans.Simple. End each function call with a semi-colon ;
Like this:

<body onload="someFunction();otherFunction();">

or, say, in a mouseover...

onMouseOver="someFunction();otherFunction();"

In JavaScript, the semi-colon is essentially an end-of-line marker. Within reasonable limits, you can actually write a whole script inside of an event handler.

The same thing applies to the href="javascript:etc" structure. For instance:

Click Here

Q3. How do you make a window "pop under" when it is opened?

Put this as early in the <head> of the page as possible:

<script>
self.blur();
</script>

That tells the window to "lose focus" as soon as it reads the self.blur(); -- which makes the window "jump behind" the window that is currently in focus.

Q4. How can you set a window's size when it is opened?

Put this as early in the <head> of the page as possible:

<script>
self.resizeTo(100,200);
</script>

Set the dimensions in the parentheses. The first number is the width; the second is the height.

Q5.How can you make certain a window will "come to the front" when it is loaded?

Add onload="self.focus();" to the body tag, like this:

<body onload="self.focus();">

As soon as the window is fully loaded, it will "take focus" and move in front of any other open windows.

ASP

ASP stands for **A**ctive **S**erver **P**ages

ASP is a program that runs inside **IIS**

IIS stands for **I**nternet **I**nformation **S**ervices

IIS comes as a free component with **Windows 2000**

IIS is also a part of the **Windows NT 4.0 Option Pack**

The Option Pack can be **downloaded** from Microsoft

PWS is a smaller - but fully functional - version of IIS

PWS can be found on your **Windows 95/98 CD**

ASP Compatibility

ASP is a Microsoft Technology

To run IIS you must have Windows NT 4.0 or later

To run PWS you must have Windows 95 or later

ChiliASP is a technology that runs ASP without Windows OS

InstantASP is another technology that runs ASP without Windows

What is an ASP File?

An ASP file is just the same as an HTML file

An ASP file can contain text, HTML, XML, and scripts

Scripts in an ASP file are executed on the server

An ASP file has the file extension ".asp"

How Does ASP Differ from HTML?

When a browser requests an HTML file, the server returns the file

When a browser requests an ASP file, IIS passes the request to the ASP engine. The ASP engine reads the ASP file, line by line, and executes the scripts in the file. Finally, the ASP file is returned to the browser as plain HTML

What can ASP do for you?

Dynamically edit, change or add any content of a Web page

Respond to user queries or data submitted from HTML forms

Access any data or databases and return the results to a browser

Customize a Web page to make it more useful for individual users

The advantages of using ASP instead of CGI and Perl, are those of simplicity and speed

Provide security since your ASP code can not be viewed from the browser

Clever ASP programming can minimize the network traffic

Important: Because the scripts are executed on the server, the browser that displays the ASP file does not need to support scripting at all!

How to Run ASP on your own PC

You can run ASP on your own PC without an external server. To do that, you must install Microsoft's Personal Web Server (PWS) or Internet Information Services (IIS) on your PC.

The Basic Syntax Rule

An ASP file normally contains HTML tags, just like an HTML file. However, an ASP file can also contain **server scripts**, surrounded by the delimiters **<%** and **%>**. Server scripts are **executed on the server,** and can contain any expressions, statements, procedures, or operators valid for the scripting language you prefer to use.

Write Output to a Browser

The response.write command is used to write output to a browser. The following example sends the text "Hello World" to the browser:

```
<html>

<body>

<%

response.write("Hello World!")

%>

</body>

</html>
```

There is also a shorthand method for the response.write command. The following example also sends the text "Hello World" to the browser:

```
<html>

<body>

<%="Hello World!"%>

</body>

</html>
```

VBScript

You can use several scripting languages in ASP. However, the default scripting language is VBScript:

```
<html>
```

```
<body>

<%

response.write("Hello World!")

%>

</body>

</html>
```

The example above writes "Hello World!" into the body of the document.

JavaScript

To set JavaScript as the default scripting language for a particular page you must insert a language specification at the top of the page:

```
<%@ language="javascript"%>

<html>

<body>

<%

Response.Write("Hello World!")

%>

</body>

</html>
```

Note: Unlike VBScript - JavaScript is case sensitive. You will have to write your ASP code with uppercase letters and lowercase letters when the language requires it.

Sample Program

Interact with a user in a form that uses the "get" method

Steps to write ASP Program :

1. Create HTML file by using editor.

2. Define Tags HTML, body tags.

3. Inside body tag , define the table tags.

4. Save the file with .html extension

5. Open the file in the browser. Browser will show the output as shown in the sample program output.

Program Source Code :

```
<html>
<body>
<form action="demo_reqquery.asp" method="get">
Your name: <input type="text" name="fname" size="20" />
<input type="submit" value="Submit" />
</form>
<%
dim fname
fname=Request.QueryString("fname")
If fname<>"" Then
    Response.Write("Hello " & fname & "!<br />")
    Response.Write("How are you today?")
End If
%>
</body>
</html>
```

Output :

Top of Form

Your name: 

Q1.What are session cookies?

Ans Session cookies are almost the same as normal cookies but are deleted from your computer within about 20 minutes of leaving the site or as soon as you close your web browser.

To check that you have cookies and session cookies enabled in IE go to *'Tools*' at the top of the screen and select *'Internet Options'*. Under the *'Privacy'* Tab and make sure your cookie settings are set to *'Medium'*.

Q2.Is there any inbuilt paging(for example shoping cart. which will show next 10 records without refreshing) in ASP? How will you do pating.
Ans: With the help of AJAX.

Q3.What statement has to be defined before a Response. Flush
Ans: Response. Buffer=”TRUE”

Q4.What programming model is ASP based on?
Ans: Object Based

Q4.What does connection.execute,connection.open do?
Ans: Connection.Execute- Execute Query.Connection.Open- Connect with Database

Q5.Application_start, session_start session_end application_end is the order important?
ANS: NO, the order is not important.