According to New Revised Credit System Syllabus

SPPU

Third Year (T.E.) Degree Course In

ELECTRONICS AND TELECOMMUNICATION ENGG. (Semester - VI)

ADVANCED JAVA PROGRAMMING

Mrs. MANISHA P. NAVALE

Mr. SAGAR S. MANE

Mrs. PRITI P. JORVEKAR-KUMBHAR



Includes

- Model Question Papers For Practice (In Sem-30 Marks & End Sem- 70 Marks)
- MCQ's with Answers

www.pragationline.com

niralipune@pragationline.com

www.facebook.com/niralibooks

@nirali.prakashan

NIRALI PRAKASHAN

ADVANCEMENT OF KNOWLEDGE

A TEXT BOOK OF

ADVANCED JAVA PROGRAMMING

(ELECTIVE-II)

FOR

SEMESTER – VI

THIRD YEAR DEGREE COURSE IN ELECTRONICS AND TELECOMMUNICATION ENGINEERING

Strictly According to New Revised Credit System Syllabus of Savitribai Phule Pune University

(w.e.f. June 2021)

Mrs. MANISHA P. NAVALE
M.E. (CN)
Assistant Professor,
Computer Engineering Dept.,
NBN Sinhgad School of Engineering,
Ambegaon (Bk.),
PUNE.

Mr. SAGAR S. MANE
M.E. (CN)
Assistant Professor,
Computer Engineering Dept.,
NBN Sinhgad School of Engineering,
Ambegaon (Bk.),
PUNE.

Mrs. PRITI P. JORVEKAR - KUMBHAR
M.E. (Computer Engineering)
PUNE.

NIRALI PRAKASHAN
ADVANCEMENT OF KNOWLEDGE

N6085

Advanced JAVA Programming (TE E&TC) **ISBN: 978-93-5451-596-5**

First Edition : March 2022
© : Authors

The text of this publication, or any part thereof, should not be reproduced or transmitted in any form or stored in any computer storage system or device for distribution including photocopy, recording, taping or information retrieval system or reproduced on any disc, tape, perforated media or other information storage device etc., without the written permission of Authors with whom the rights are reserved. Breach of this condition is liable for legal action.

Every effort has been made to avoid errors or omissions in this publication. In spite of this, errors may have crept in. Any mistake, error or discrepancy so noted and shall be brought to our notice shall be taken care of in the next edition. It is notified that neither the publisher nor the authors or seller shall be responsible for any damage or loss of action to any one, of any kind, in any manner, therefrom. The reader must cross check all the facts and contents with original Government notification or publications.

Published By :
NIRALI PRAKASHAN
Abhyudaya Pragati, 1312 Shivaji Nagar
Off J.M. Road, PUNE 411005
Tel : (020) 25512336/37/39
Email : niralipune@pragationline.com

Polyplate

Printed By :
YOGIRAJ PRINTERS AND BINDERS
Survey No. 10/1A, Ghule Industrial Estate
Nanded Gaon Road
Nanded, Pune 411041

DISTRIBUTION CENTRES

PUNE

Nirali Prakashan
(For orders outside Pune)
S. No. 28/27, Dhayari Narhe Road, Near Asian College
Pune 411041, Maharashtra
Tel : (020) 24690204; Mobile : 9657703143
Email : bookorder@pragationline.com

Nirali Prakashan
(For orders within Pune)
119, Budhwar Peth, Jogeshwari Mandir Lane
Pune 411002, Maharashtra
Tel : (020) 2445 2044; Mobile : 9657703145
Email : niralilocal@pragationline.com

MUMBAI

Nirali Prakashan
Rasdhara Co-op. Hsg. Society Ltd., 'D' Wing Ground Floor, 385 S.V.P. Road
Girgaum, Mumbai 400004, Maharashtra
Mobile : 7045821020, Tel : (022) 2385 6339 / 2386 9976
Email : niralimumbai@pragationline.com

DISTRIBUTION BRANCHES

DELHI

Nirali Prakashan
Room No. 2 Ground Floor
4575/15 Omkar Tower, Agarwal Road
Darya Ganj, New Delhi 110002
Mobile : 9555778814/9818561840
Email : delhi@niralibooks.com

BENGALURU

Nirali Prakashan
Maitri Ground Floor, Jaya Apartments, No. 99, 6th Cross, 6th Main, Malleswaram, Bengaluru 560003 Karnataka; Mob : 9686821074
Email : bengaluru@niralibooks.com

NAGPUR

Nirali Prakashan
Above Maratha Mandir, Shop No. 3, First Floor, Rani Jhanshi Square, Sitabuldi Nagpur 440012 (MAH)
Tel : (0712) 254 7129
Email : nagpur@niralibooks.com

KOLHAPUR

Nirali Prakashan
438/2, Bhosale Plaza, Ground Floor, Khasbag, Opp. Balgopal Talim, Kolhapur 416 012 Maharashtra
Mob : 9850046155
Email : kolhapur@niralibooks.com

JALGAON

Nirali Prakashan
34, V. V. Golani Market, Navi Peth, Jalgaon 425001, Maharashtra
Tel : (0257) 222 0395
Mob : 94234 91860
Email : jalgaon@niralibooks.com

SOLAPUR

Nirali Prakashan
R-158/2, Avanti Nagar, Near Golden Gate, Pune Naka Chowk
Solapur 413001, Maharashtra
Mobile 9890918687
Email : solapur@niralibooks.com

marketing@pragationline.com | www.pragationline.com

Also find us on www.facebook.com/niralibooks

PREFACE

It gives us great pleasure to present the book **'Advanced JAVA Programming'** for the students of **Third Year (TE) Degree Course in Electronics and Telecommunication Engineering** of Savitribai Phule Pune University. This book is strictly as per the **New Revised Credit System Syllabus 2019 Pattern** with effect from the Academic Year June 2021.

As per New Revised Examination Scheme which has been implemented from this academic year, In-Semester assessment carries 30 marks over first two units and End-Semester Examination carries 70 marks on the remaining units 3, 4, 5 and 6.

The Theory Course will have 3 Credits.

In the book all the basic concepts are explained in simplified manner. Subject matter is presented more in a conceptual manner than mathematical, as required by the new examination system. It is our objective to keep the presentation systematic, and clear through explanatory notes and figures.

Main feature of this book is, Complete Coverage of the New Credit System Syllabus with solved Examples, Exercises and Model Question Papers of In Sem. and End Sem. Exams.

We are sure that this book will cater to all needs of the students for this subject.

We take this opportunity to express our sincere thanks to Shri. Dineshbhai Furia, Shri. Jignesh Furia, Mrs. Nirali Verma, Mrs. Deepali Lachake (Co-ordinator), and entire team of Nirali Prakashan who really have taken keen interest and untiring efforts in publishing this volume.

We wish to make a special mention of the valuable contribution made by Late Shri M.P. Munde during the last two decades in reaching out to students, parents and teachers which eventually made **Nirali Prakashan, a brand of trust and quality in technical books**. Without his perseverance and zeal, successive editions would not have been possible.

The advice and suggestions of our esteemed readers, to improve the text, are most welcome, and will be highly appreciated.

Pune **Authors**

SYLLABUS

Unit I: Applet **(06 Hrs.)**

Applet Basics – Introduction, limitations of AWT, Applet architecture – HTML APPLET tag – Passing parameter to Appletget, DocumentBase() and getCodeBase(), Japplet: Icons and Labels Text Fields Buttons, Combo Boxes, Checkboxes, Tabbed Panes, Scroll Panes, Trees: Tables

Unit II: Event Handling using AWT/Swing Components **(08 Hrs.)**

Event Handling: Events, Event sources, Event classes, Event Listeners, Delegation event model, handling mouse and keyboard events, Adapter classes, inner classes. The AWT class hierarchy, user interface components- labels, button, canvas, scrollbars, text components, checkbox, checkbox groups, choices, lists panels – scroll pane, dialogs, menu bar, graphics, layout manager – layout manager types – boarder, grid, flow, card and grib bag.

Unit III: GUI Programming **(06 Hrs.)**

Designing Graphical User Interfaces in Java, Components and Containers, Basics of Components, Using Containers, Layout Managers, AWT Components, Adding a Menu to Window, Extending GUI Features Using Swing Components, Java Utilities (java.util Package) The Collection Framework: Collections of Objects, Collection Types, Sets, Sequence, Map, Understanding Hashing, and Use of Array List & Vector.

Unit IV: Database Programming using JDBC **(06 Hrs.)**

The Concept of JDBC, JDBC Driver Types & Architecture, JDBC Packages, A Brief Overview of the JDBC process, Database Connection, Connecting to non-conventional Databases Java Data Based Client/server, Basic JDBC program Concept, Statement, Result Set, Prepared Statement, Callable Statement, Executing SQL commands, Executing queries

Unit V: Remote Method Invocation (RMI) **(06 Hrs.)**

Remote Method Invocation: Architecture, RMI registry, the RMI Programming Model; Interfaces and Implementations; Writing distributed application with RMI, Naming services, Naming and Directory Services, Setting up Remote Method Invocation – RMI with Applets, Remote Object Activation; The Roles of Client and Server, Simple Client/Server Application using RMI.

Unit VI: Networking **(08 Hrs.)**

The java.net package, Connection oriented transmission – Stream Socket Class, creating a Socket to a remote host on a port (creating TCP client and server), Simple Socket Program Example.

InetAddress, Factory Methods, Instance Methods, Inet4Address and Inet6Address, TCP/IP Client Sockets. URL, URLConnection, HttpURLConnection, The URI Class, Cookies, TCP/IP Server Sockets, Datagrams, DatagramSocket, DatagramPacket, A Datagram Example.

CONTENTS

Unit I: Applet **1.1-1.18**

1.1 Introduction 1.1
1.2 Limitations of AWT 1.4
1.3 Applet Architecture 1.5
1.4 HTML <applet> Tag 1.6
1.5 Passing Parameter to Appletget 1.6
1.6 GetDocumentBase and GetCodeBase Example 1.7
1.7 JApplet Icon, Labels, Text Field Buttons 1.8
1.8 CheckBox 1.11
1.9 ComboBox 1.13
1.10 TabbedPane 1.14
1.11 ScrollPane 1.15
1.12 JTree:Tables 1.17
- **Exercise** 1.18

Unit II: Event Handling using AWT/Swing Components **2.1-2.46**

2.1 Events 2.1
2.2 Event Sources 2.1
2.3 Event Classes 2.1
2.4 Event Listeners 2.2
2.4.1 Sources of Events 2.2
2.5 Delegation Event Model 2.3
2.6 Handling Keyboard and Mouse Event 2.3
2.6.1 Mouse Event 2.3
2.6.2 Keyboard Events 2.9
2.7 Adapter Classes 2.12
2.8 Inner Classes 2.14
2.8.1 AWT Hierarchy 2.15
2.8.2 Components 2.15
2.8.3 Container 2.16
2.9 User Interface Components 2.16
2.9.1 Buttons 2.18
2.9.2 Text Fields 2.18
2.9.3 Label 2.18
2.9.4 Canvas 2.18
2.9.5 Scrollbar 2.20
2.10 Text Components 2.23
2.10.1 CheckBox 2.23
2.10.2 Checkbox Groups 2.25
2.10.3 Choices 2.26

2.11 Lists Panels 2.29
2.11.1 Dialogs 2.30
2.11.2 Menubar 2.32
2.11.3 Graphics 2.33
2.12 Layout Managers 2.34
2.12.1 FlowLayout 2.36
2.12.2 GridLayout 2.37
2.12.3 BorderLayout 2.38
2.12.4 CardLayout 2.40
2.12.5 GridLayout 2.43
- **Exercise** 2.46

Unit III: GUI Programming **3.1-3.32**

3.1 Designing Graphical User Interfaces in Java 3.1
3.2 Components and Containers 3.4
3.2.1 Basics of Components 3.5
3.2.2 Using Container 3.10
3.3 Layout Managers 3.11
3.4 AWT Components 3.12
3.5 Adding Menu to Windows in Java 3.14
3.6 Extending GUI Features Using Swing Components 3.16
3.7 Java Utilities (java.util Package) 3.21
3.8 The Collection Framework: Collections of Objects 3.23
3.8.1 Collection Types 3.24
3.8.2 Sets 3.25
3.8.3 Sequence (List) 3.27
3.8.4 Map 3.27
3.8.5 Understanding Hashing 3.29
3.8.6 Use of Array List and Vector 3.31
- **Exercise** 3.32

Unit IV: Database Programming using JDBC **4.1-4.20**

4.1 Introduction 4.1
4.1.1 JDBC Architecture 4.2
4.2 Working of JDBC 4.3
4.3 JDBC Driver 4.4
4.4 Database Connection 4.6
4.4.1 Java Database Connectivity with Oracle 4.7
4.4.2 Java Database Connectivity with MySQL 4.8
4.5 JDBC ResultSet 4.9
4.6 Prepared Statement 4.12
4.7 Callable Statement 4.14
4.8 Executing SQL Commands 4.17
4.9 Executing Queries 4.19
- **Exercise** 4.20

Unit V: Remote Method Invocation (RMI) **5.1-5.26**

5.1 Remote Method Invocation 5.1
5.1.1 RMI Application Architecture 5.1
5.1.2 RMI Registry 5.2
5.1.3 RMI Programming Model 5.2
5.2 Interfaces and Implementations 5.5
5.3 Writing Distributed Application with RMI 5.12
5.4 Naming/Registry Service 5.13
5.5 Naming and Directory Services 5.14
5.6 Setting up Remote Method Invocation – RMI with Applets 5.16
5.7 Remote Object Activation 5.19
5.8 The Roles of Client and Server 5.22
5.9 Simple Client/Server Application using RMI 5.23
- **Exercise** 5.26

Unit VI : Networking **6.1-6.30**

6.1 Introduction 6.1
6.2 The java.net Package 6.1
6.2.1 Java.net Classes 6.2
6.3 Connection Oriented Transmission – Stream Socket Class 6.5
6.4 Creating a Socket to a Remote Host on a Port (Creating TCP Client and Server) 6.6
6.5 Simple Socket Program Example 6.9
6.6 InetAddress 6.13
6.7 Factory Methods 6.13
6.8 Instance Methods 6.14
6.9 Inet4Address and Inet6Address 6.16
6.10 TCP/IP Client Sockets 6.16
6.11 URL 6.18
6.12 URLConnection 6.20
6.13 HttpURLConnection 6.22
6.14 The URI Class 6.25
6.15 Cookies 6.26
6.16 TCP/IP Server Sockets 6.27
6.17 Datagrams 6.27
6.18 DatagramSocket 6.27
6.19 DatagramPacket 6.28
6.20 A Datagram Example 6.29
- **Exercise** 6.30

- **Model Question Papers**
  - ➢ **In-Sem Exam. (30 Marks)** **P.1-P.1**
  - ➢ **End-Sem Exam. (70 Marks)** **P.2-P.2**



UNIT I

APPLET

1.1 INTRODUCTION

- 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.
- An applet is a Java program that can be embedded into a web page.
- Applets are used to make the website more dynamic and entertaining.

Advantage

- They are very secure.
- It works at client side so less response time.
- Applets can be executed by browsers running under different platforms including Linux, Windows and Mac Os etc.

Disadvantage

- Plug-ins are required at the client browser for executing applets.

Applet Life Cycle

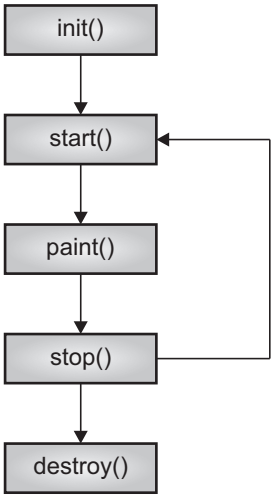


Fig. 1.1

- An applet undergoes various stages between its creation of objects and object removal as the work of the applet will get done. This cycle is known as Applet life cycle, where each state is signified by methods.
- In the life of execution, the applet exists in these 5 states. All of these methods have a name and they are called as callback methods. These methods are named so because they are called automatically by the browser when required for smooth execution of the applet. Here, programmers write the above-mentioned methods with some code but never calls.
- Following are the methods for a full applet cycle.
  - ➢ init() method
  - ➢ start() method
  - ➢ paint() method
  - ➢ stop() method
  - ➢ destroy() method

Browser Responsibilities for Applet Life Cycle

- Applet life cycle methods are callback methods because they are called implicitly by the browser for the smooth execution of the applet. The browser should provide an environment known as a container for the execution of the applet.
- Following are the responsibilities of the browser.
  - ➢ For the smooth execution, it should call the callback methods at appropriate times.
  - ➢ It is responsible to maintain the Applet Life Cycle.
  - ➢ It should have the capability to communicate between applets, applet to JavaScript and HTML, applet to browser, etc.

1. java.applet.Applet Class

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

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

2. java.awt.Component Class

The Component class provides 1 life cycle method of applet.

- **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.

How to run an Applet?

There are two ways to run an applet

1. By html file.
2. By appletViewer tool.

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.

```
//First.java
import java.applet.Applet;
import java.awt.Graphics;
public class First extends Applet{

public void paint(Graphics g){
g.drawString("welcome",150,150);
}
}
```

myapplet.html

```
<html>
<body>
<applet code="First.class" width="300" height="300">
</applet>
</body>
</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.

```
//First.java
import java.applet.Applet;
import java.awt.Graphics;
public class First extends Applet{

public void paint(Graphics g){
g.drawString("welcome to applet",150,150);
}

}
/*
<applet code="First.class" width="300" height="300">
</applet>
*/
```

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

```
c:\>javac First.java
c:\>appletviewer First.java
```

Creating Hello World Applet:

Program 1.1: Let's begin with the HelloWorld applet

```
// A Hello World Applet
// Save file as HelloWorld.java
import java.applet.Applet;
import java.awt.Graphics;
// HelloWorld class extends Applet
public class HelloWorld extends Applet
{
    // Overriding paint() method
    @Override
    public void paint(Graphics g)
    {
        g.drawString("Hello World", 20, 20);
    }
}
```

Explanation:

- The above java program begins with two import statements. The first import statement imports the Applet class from applet package. Every AWT-based (Abstract Window Toolkit) applet that you create must be a subclass of Applet class. The second statement imports the Graphics class from AWT package.

- The next line in the program declares the class HelloWorld. This class must be declared as public because it will be accessed by code that is outside the program. Inside HelloWorld, paint( ) is declared. This method is defined by the AWT and must be overridden by the applet.

Running the HelloWorld Applet:

- After you enter the source code for HelloWorld.java, compile in the same way that you have been compiling java programs (using javac command). However, running HelloWorld with the java command will generate an error because it is not an application.

```
java HelloWorld

Error: Main method not found in class HelloWorld,
please define the main method as:
public static void main(String[] args)
```

There are Two Standard Ways in which you can Run an Applet:

Executing the applet within a Java-compatible web browser.

Using an applet viewer, such as the standard tool, applet-viewer. An applet viewer executes your applet in a window. This is generally the fastest and easiest way to test your applet.

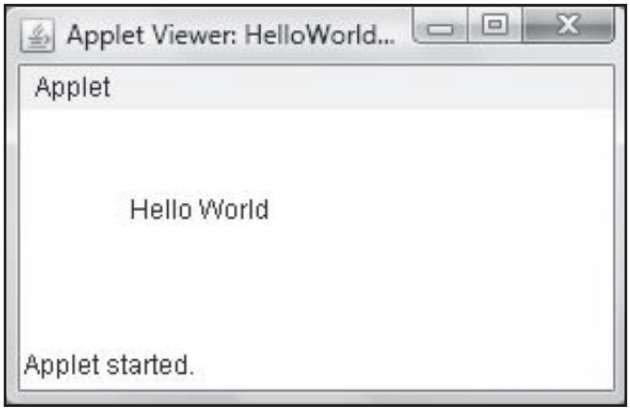
Each of these methods is described next.

1. **Using Java Enabled Web Browser:** To execute an applet in a web browser we have to write a short HTML text file that contains a tag that loads the applet. We can use APPLET or OBJECT tag for this purpose. Using APPLET, here is the HTML file that executes HelloWorld:<applet code="HelloWorld" width=200 height=60>

```
</applet>
```

   The width and height statements specify the dimensions of the display area used by the applet. The APPLET tag contains several other options. After you create this html file, you can use it to execute the applet.

2. **Using appletviewer:** This is the easiest way to run an applet. To execute HelloWorld with an applet viewer, you may also execute the HTML file shown earlier. For example, if the preceding HTML file is saved with RunHelloWorld.html, then the following command line will run HelloWorld:
   appletviewer RunHelloWorld.html



1.2 LIMITATIONS OF AWT

- The AWT defines a basic set of controls, windows, and dialog boxes that support a usable, but limited graphical interface. One reason for the limited nature of the AWT is that it translates its various visual components into their corresponding, platform-specific equivalents or peers. This means that the look and feel of a component is defined by the platform, not by java. Because the AWT components use native code resources, they are referred to as heavy weight.
- The use of native peers led to several problems. First, because of variations between operating systems, a component might look, or even act, differently on different platforms. This variability threatened java's philosophy: write once, run anywhere. Second, the look and feel of each component was fixed and could not be changed. Third, the use of heavyweight components caused some frustrating restrictions.

1.3 APPLET ARCHITECTURE

- When you write a Java application for time series data, you use the JDBC Driver to connect to the database, as shown in the following figure.

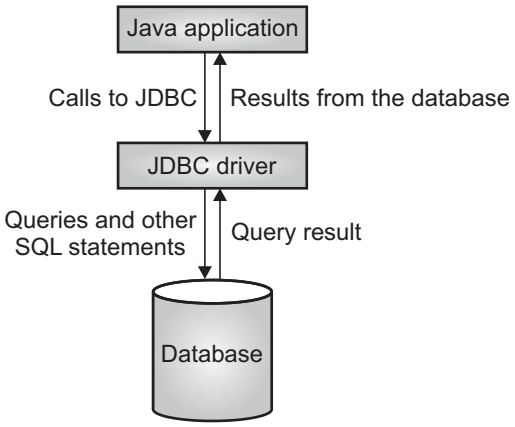


Fig. 1.2: Runtime architecture for Java programs that connect to a database

- The Java application makes calls to the JDBC driver, which sends queries and other SQL statements to the database. The database sends query results to the JDBC driver, which sends them on to the Java application.
- You can also use the time series Java classes in Java applets and servlets, as shown in the following figures.

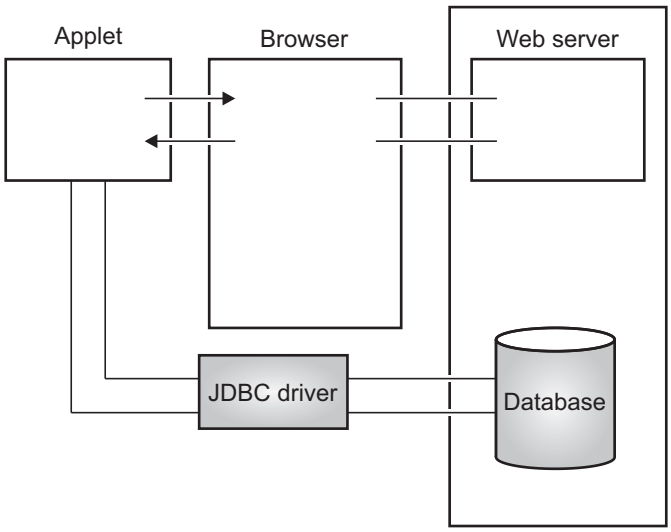


Fig. 1.3: Runtime architecture for a Java applet

- The database server is connected to the JDBC driver, which is connected to the applet. The applet is also connected to a browser, which is connected to a web server that communicates with the database.

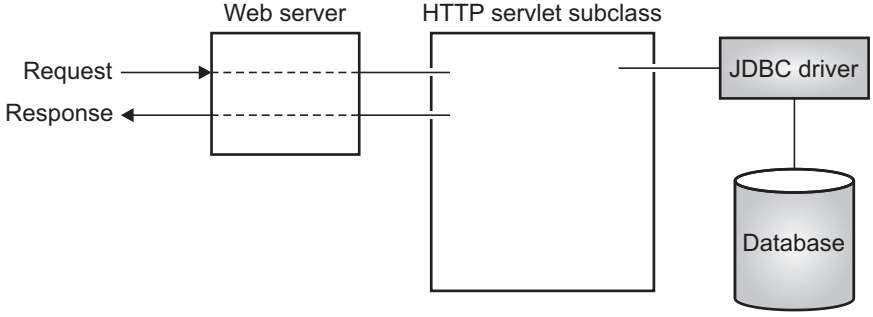


Fig. 1.4: Runtime architecture for a Java servlet

A request from an application goes through a web server, an HTTP servlet subclass, and the JDBC driver to the database. The database sends responses back along the same path.

1.4 HTML <APPLET> TAG

HTML <applet> tag was used to embed the Java applet in an HTML document.

Syntax

```
<applet code="URL" height="200" width="100">.....</applet>
```

Table 1.1

Attribute name	Value	Description
code	URL	It specifies the URL of Java applet class file.
width	pixels	It specifies the display width of the applet panel.
height	pixels	It specifies the display height of applet panel.
align	• left • right • top • middle • bottom	It specifies the position of applet application relative to surrounding content.
alt	text	It is used to display alternative text in case browser does not support Java.
archive	URL	This specifies the archived or compressed version of an applet application.
object	name	It specifies the URL or reference to a serialized representation of an applet.
codebase	URL	It specifies the exact or relative URL of applets .class file specified in the code attribute.
hspace	pixels	It specifies the horizontal space around the applet.
vspace	pixels	It specifies the vertical space around the applet.
name	name	It specifies the name for the applet

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Applet Tag</title>
 </head>
 <body>
   <p>Example of Applet Tag</p>
 <applet code="Demo.class" align="right" height="200" width="300>
     <b>Sorry! Please install Java to see this</b>
     </applet>
  </body>
</html>
```

1.5 PASSING PARAMETER TO APPLETGET

We can get any information from the HTML file as a parameter. For this purpose, Applet class provides a method named as getParameter().

Syntax:

```
public String getParameter(String parameterName)
```

Program 1.2: Example of Using Parameter in Applet

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

public class UseParam extends Applet{

public void paint(Graphics g){
String str=getParameter("msg");
g.drawString(str,50, 50);
}

}
```

myapplet.html

```
<html>
<body>
<applet code="UseParam.class" width="300" height="300">
<param name="msg" value="Welcome to applet">
</applet>
</body>
</html>
```

1.6 GETDOCUMENTBASE AND GETCODEBASE EXAMPLE

- In most of the applets, it is required to load text and images explicitly. Java enables loading data from two directories. The first one is the directory which contains the HTML file that started the applet known as the document base.
- The other one is the directory from which the class file of the applet is loaded known as the code base.
- These directories can be obtained as URL objects by using getDocumentBase() and getCodeBase()methods respectively.
- You can concatenate these URL objects with the string representing the name of the file that is to be loaded.

Program 1.3:

```
/*  <applet code="GetDocumentBaseExample" width=350 height=250>
   </applet>
   */

   import java.applet.Applet;
   import java.awt.Graphics;
   import java.net.URL;
   import java.awt.*;
   import java.awt.event.*;
      public class GetDocumentBaseExample extends Applet{

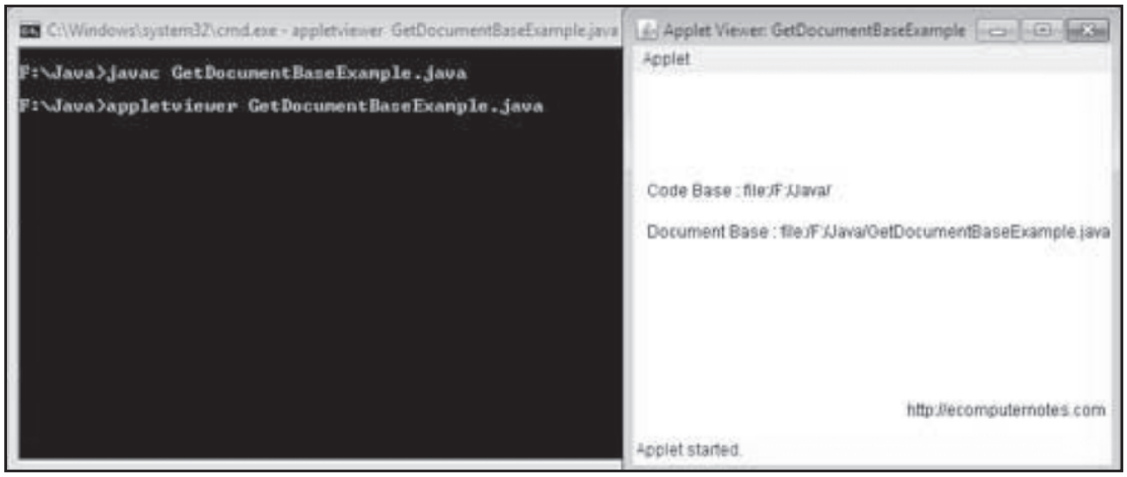
        public void paint(Graphics g){
        String message;
```

```
    //getCodeBase() method gets the base URL of the directory
    //in which contains this applet.

        URL appletCodeDir=getCodeBase();
        message = "Code Base : "+appletCodeDir.toString();
        g.drawString(message,10,90);

       // getDocumentBase() Returns an absolute URL of the Document
        URL appletDocDir = getDocumentBase();
      message="Document Base : "+appletDocDir.toString();
      g.drawString(message,10,120);
      g.drawString("https://ecomputernotes.com", 200, 250);

        }
}
```



1.7 JAPPLET ICON, LABELS, TEXT FIELD BUTTONS

- JApplet is a java swing public class designed for developers usually written in Java.
- JApplet is generally in the form of Java bytecode that runs with the help of a Java virtual machine (JVM) or Applet viewer from Sun Microsystems. It was first introduced in 1995.
- Java applets can be executed on multiple platforms which include Microsoft Windows, UNIX, Mac OS and Linux.
- JApplet can also be run as an application, though this would require a little extra coding.
- The executable applet is made available on a domain from which it needs to be downloaded. The communication of the applet is restricted only to this particular domain.
- JApplet extends the class in the form of java.applet.Applet. JApplets are executed in a tightly-controlled set of resources referred to as sandboxes. This prevents the JApplets from accessing local data like the clipboard or file system.
- The first JApplet implementations were performed by downloading an applet class by class. Classes contain many small files and so applets were considered to be slow loading components, an applet is aggregated and sent as a single, but larger file.

JApplet Class in Applet

JApplet can have all the controls of swing. The JApplet class extends the Applet class.

Program 1.4: Example of EventHandling in JApplet

```
import java.applet.*;
import javax.swing.*;
import java.awt.event.*;
public class EventJApplet extends JApplet implements ActionListener{
JButton b;
JTextField tf;
public void init(){

tf=new JTextField();
tf.setBounds(30,40,150,20);

b=new JButton("Click");
b.setBounds(80,150,70,40);

add(b);add(tf);
b.addActionListener(this);

setLayout(null);
}

public void actionPerformed(ActionEvent e){
tf.setText("Welcome");
}
}
```

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

myapplet.html

```
<html>
<body>
<applet code="EventJApplet.class" width="300" height="300">
</applet>
</body>
</html>
```

Label:

- Label is used for displaying a description which can't be change at the Run Time and which doesn't take any input from a user and used for which is used for displaying a text string that is a run-editable and for creating a label first create the object of label class.
  - ➢ **Label():** Empty label or doesn't display any String on Label.
  - ➢ **Label(String txt):** With a text string displayed on Label.
  - ➢ **Label(String txt, int align):** Which is used for first displaying the string and then its alignment using Label. LEFT or 0,1,2 etc.