

# An Internship Report On

**“ADVANCED JAVA”**

In partial fulfillment of requirements for the degree of  
Bachelor of Technology

In  
Computer Engineering

**Submitted By:**

Kalal Rohan Nitin



**Department of Computer Engineering**  
**SES's R. C. Patel Institute of Technology**

**Shirpur - 425405.**

**[2019-20]**



# R C P I T

The Shirpur Education Society's

## **R. C. Patel Institute of Technology Shirpur, Dist. Dhule (M.S.)**

### **Department of Computer Engineering**

#### ***CERTIFICATE***

*This is to certify that "Kalal Rohan Nitin" has been successfully completed internship in Advanced Java at V3 Data Solution, Nashik under supervision of experienced developers in partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Engineering of DBATU, Lonere.*

**Date of report submission:**

**Place: Shirpur**

**Prof. Ms.P.R.Patil**

**Local Guardian**

**Prof.S. N.Pardeshi**

**T&P Coordinator**

**HOD**

**Prof. Dr. Nitin N. Patil**

## ACKNOWLEDGEMENT

I would like to thank Mr. Sunil Joshi Sir from V3 Data Solution, Nasik for providing training in this internship opportunity. I have been greatly benefited by his training, valuable suggestions and ideas.

Also I would like to express my sincere gratitude to our internship coordinator Prof. S. N. Pardeshi sir for all the successful planning of this internship.

I am thankful to the Head of Department, **Prof. Dr. Nitin N. Patil** sir, for providing me an internship opportunity at V3 Data Solution, Nasik.

I am extremely thankful to our Principal, Prof. **Dr. J. B. Patil** sir for providing all the required support and guidance.

KALAL ROHAN NITIN

# INDEX

<b>Chapter No.</b>	<b>Topic</b>	<b>Page No.</b>
	<b>Abstract</b>	6
<b>1.</b>	<b>Introduction</b>	7
1.1	Tools You Will Need	7
1.2	Popular Java Editors	8
1.3	J2EE	8
<b>2.</b>	<b>Collection</b>	9
2.1	Collection Tree Part	9
2.2	Benefits of the Java collection Framework	10
<b>3.</b>	<b>Java Networking</b>	12
3.1	Advantage of JAVA Networking	12
<b>4.</b>	<b>AWT(Abstract Windowing Toolkit)</b>	13
4.1	Java AWT Heirarchy	13
<b>5.</b>	<b>Swing</b>	15
5.1	Heirarchy of Java Swing Classes	15
<b>6.</b>	<b>Java Database Connectivity</b>	16
6.1	JDBC Drivers	16
6.2	Functionally And Implementation	17

# INDEX

<b>Chapter No.</b>	<b>Topic</b>	<b>Page No.</b>
<b>7.</b>	<b>Servlet</b>	<b>21</b>
7.1	The Java Servlet API	21
7.2	Servlet Life Cycle	22
<b>8.</b>	<b>Applet</b>	<b>24</b>
8.1	Life Cycle of an Applet	25
	<b>Project : Blood Bank</b>	<b>26</b>
	<b>Conclusion</b>	<b>35</b>
	<b>Bibliography</b>	<b>36</b>

## ABSTRACT

*Java is a programming language, whereas the Advanced Java is used to build enterprise level applications. This internship report focus on generic programming, sequential and associative data structures, classic data structures, sorting and searching, exception handling, database programming with JDBC, networking programming GUI development using Swing and an overview of Multithreading. You will also explore Java Applets, web applications (Servlets), JSP (Java Server Pages), JSP tags and Java projects. The Advance Java Internship sets to create and develop Software or other web technology to enable desktop access to Information Technology sector.*

*Advanced Java is the next advanced level concept of Java programming. This high level java programming basically uses two Tier Architecture i.e Client and Server. The 'Advanced Java' comprises the very complex advanced programming.*

*The Advanced java programming covers the Swings, Socket Programming, AWT, Thread Concepts as well as the Collection objects and classes. "Advanced Java" is nothing but specialization in domains such as web, networking, data base handling. All the event handling mechanism of Java comes into the Advanced Java programming. Advanced Java is used for developing the web based application and enterprise application. It has complete solution for dynamic processes which has many frameworks design patterns servers mainly JSP.*

# CHAPTER – 1

## INTRODUCTION

---

Java is a programming language initially developed by Sun Microsystems by James Gosling and released as a principal component of Sun Microsystems' Java platform in 1995. Although the language gets much of its syntax from C and C++ it has a less complicated object model and lesser low-level services. Java applications are typically compiled to byte code (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is a general-purpose, object-oriented language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere". There were five primary goals in the creation of the Java language. [1]

1. It should be "simple, object-oriented and familiar".
2. It should be "robust and secure".
3. It should be "architecture-neutral and portable".
4. It should execute with "high performance".
5. It should be "interpreted, threaded, and dynamic".

### 1.1. Tools you will need

You will need a Pentium 200-MHz computer with a minimum of 64 MB of RAM (128 MB of RAM recommended).

You will also need the following softwares –

- Linux 7.1 or Windows xp/7/8 operating system
- Java JDK 8
- Xampp Control Panel

- Amache Tomcat Server

## 1.2. Popular JAVA Editors

To write your Java programs, you will need a text editor. There are even more sophisticated IDEs available in the market. But for now, you can consider one of the following –

- Netbeans – A Java IDE that is open-source and free which can be downloaded from <https://www.netbeans.org/index.html>.
- Eclipse – A Java IDE developed by the eclipse open-source community and can be downloaded from <https://www.eclipse.org/>

## 1.3. J2EE

Short for J2EE Is Java 2 Platform Enterprise Edition, is a platform independent, Java- centric environment from Sun for developing, building and deploying Web-based enterprise applications online. The J2EE platform consists of a set of services, APIs, and protocols that provide the functionality for developing multitier, Web-based applications.

Some of the key features and services of J2EE :

1. At the client tier, J2EE supports pure HTML, as well as Java applets or applications. It relies on Java Server Pages and servlet code to create HTML or other formatted data for the client
2. Enterprise JavaBeans (EJBs) provide another layer where the platform's logic is stored. An EJB server provides functions such as threading, concurrency, security and memory management.
3. These services are transparent to the author.
4. Java Database Connectivity (JDBC), which is the Java equivalent to ODBC, is the standard interface for Java databases.
5. The Java servlet API enhances consistency for developers without requiring a graphical user interface.



## CHAPTER – 2

### COLLECTION

---

Collection framework provides a well-designed set of interface and classes for storing and manipulating groups of data as a single unit a collection. It provides a standard programming interface to many of the most common abstractions, without blundering the programmer with too many procedure and interfaces. A collections framework is a unified architecture for representing and manipulating collections.

All collections frameworks contain the following:

1. Interfaces: These are abstract data types that represent collections. Interfaces allow collections to be manipulated independently of the details of their representation. In object-oriented languages, interfaces generally form a hierarchy.
2. Implementations: These are the concrete implementations of the collection interfaces. In essence, they are reusable data structures.
3. Algorithms: These are the methods that perform useful computations, such as searching and sorting, on objects that implement collection interfaces. The algorithms are said to be polymorphic: that is, the same method can be used on many different implementations of the appropriate collection interface. In essence, algorithms are reusable functionality. [2]

#### **2.1. Collocation tree part**

##### **2.1.1. List**

- Array List

##### **2.1.2. Set**

- Tree set

- Hash set
- Link hash set

### **2.1.3. Map**

- Hash Map
- Hash Table
- Tree Map

## **2.2. Benefits of the Java Collections Framework**

### **2.2.1. Reduces programming effort**

By providing useful data structures and algorithms, the Collections Framework frees you to concentrate on the important parts of your program rather than on the low-level "plumbing" required to make it work. By facilitating interoperability among unrelated APIs, the Java Collections Framework frees you from writing adapter objects or conversion code to connect APIs.

### **2.2.2. Increases program speed and quality**

This Collections Framework provides high-performance, high-quality implementations of useful data structures and algorithms. The various implementations of each interface are interchangeable, so programs can be easily tuned by switching collection implementations. Because you're freed from the drudgery of writing your own data structures, you'll have more time to devote to improving programs' quality and performance.

### **2.2.3. Allows interoperability among unrelated APIs**

The collection interfaces are the vernacular by which APIs pass collections back and forth. If my network administration API furnishes a collection of node names and if your GUI toolkit expects a collection of column headings, our APIs will interoperate seamlessly, even though they were written independently. [3]

#### **2.2.4. Reduces effort to learn and to use new APIs**

Many APIs naturally take collections on input and furnish them as output. In the past, each such API had a small sub-API devoted to manipulating its collections. There was little consistency among these ad hoc collections sub-APIs, so you had to learn each one from scratch, and it was easy to make mistakes when using them. With the advent of standard collection interfaces, the problem went away.

#### **2.2.5. Reduces effort to design new APIs**

This is the flip side of the previous advantage. Designers and implementers don't have to reinvent the wheel each time they create an API that relies on collections; instead, they can use standard collection interfaces.

#### **2.2.6. Fosters software reuse**

New data structures that conform to the standard collection interfaces are by nature reusable. The same goes for new algorithms that operate on objects that implement these interfaces.

## CHAPTER – 3

### JAVA NETWORKING

The term Network programming refers to writing programs that execute across multiple devices (Computers), in which the devices are all connected to each other using networks. The Java net package of the J2se API contains a collection of classes and interfaces that provide the low-level communication details, allowing you to write programs that focus on solving problem at hand.

The java net package provides support for the two common network protocols.

The java.net package provides support for the two common network protocols:-

- **TCP** - TCP stands for transmission control protocol, which allows for reliable communication between two applications. TCP is typically used over the Internet Protocol, which is referred to as TCP/IP.
- **UDP (User Datagram Protocol)** - UDP stands for User Datagram Protocol, a connection-less protocol that allows for packets of data to be transmitters between applications.

#### 3.1. Advantage of Java Networking

1. sharing resources
2. centralize software management

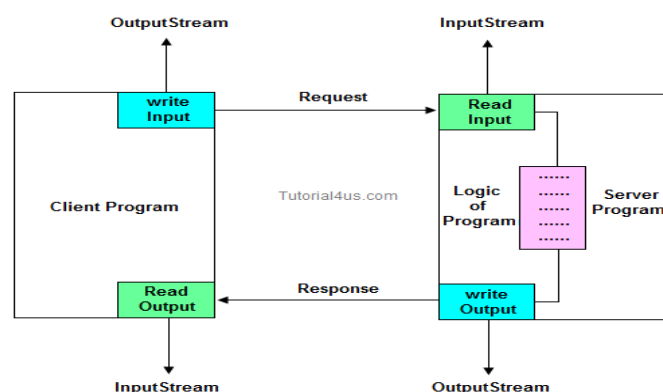


Fig. 3.1 Java Networking

### AWT (ABSTRACT WINDOWING TOOLKIT)

Java AWT (Abstract Windowing Toolkit) is an API to develop GUI or window-based application in java. Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components are using the resources of OS.

#### 4.1. Java AWT Hierarchy

The hierarchy of Java AWT classes are given below.

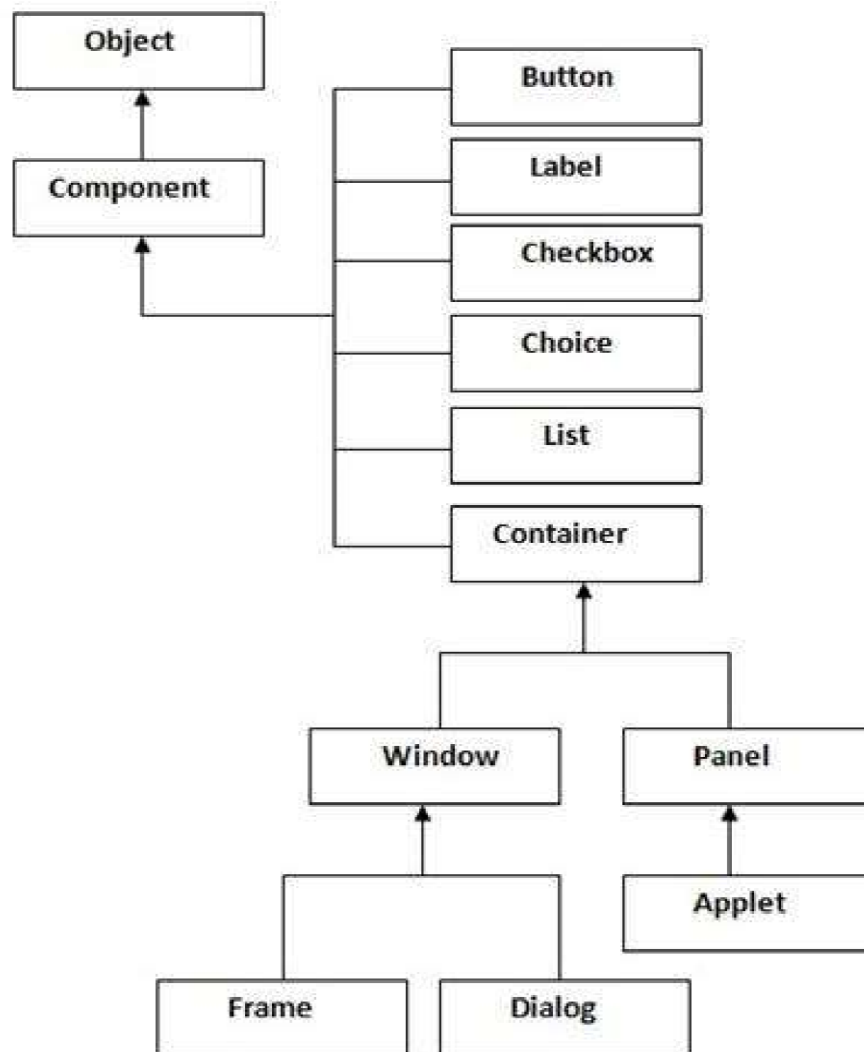


Fig. 4.1 Java AWT Hierarchy

##### 4.1.1. Container

The Container is a component in AWT that can contain another components like buttons, textfields, labels etc. The classes that extends Container class are known as container such as Frame, Dialog and Panel.

#### **4.1.2. Window**

The window is the container that have no borders and menu bars. You must use frame, dialog or another window for creating a window.

#### **4.1.3. Panel**

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

#### **4.1.4. Frame**

The Frame is the container that contain title bar and can have menu bars. It can have other components like button, textfield etc. [4]

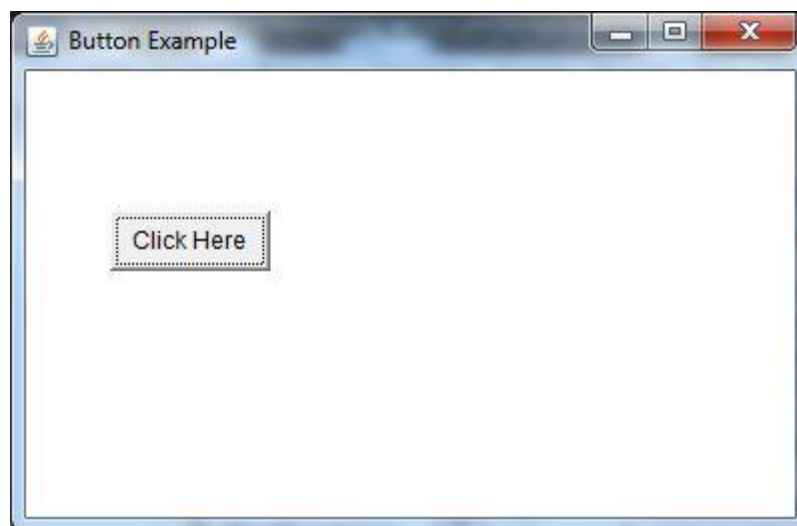


Fig. 4.2 Java AWT Button

## CHAPTER – 5

### SWING

Work Flow Java Swing is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java. Unlike AWT, Java Swing provides platform-independent and lightweight components.

- Java swing components are platform-independent.
- Swing components are lightweight.
- Swing supports pluggable look and feel.
- Swing provides more powerful components such as tables, lists, scroll panes, color chooser, tabbed pane etc.
- Swing follows MVC.

#### 5.1. Hierarchy of Java Swing classes

The hierarchy of java swing API is given below.

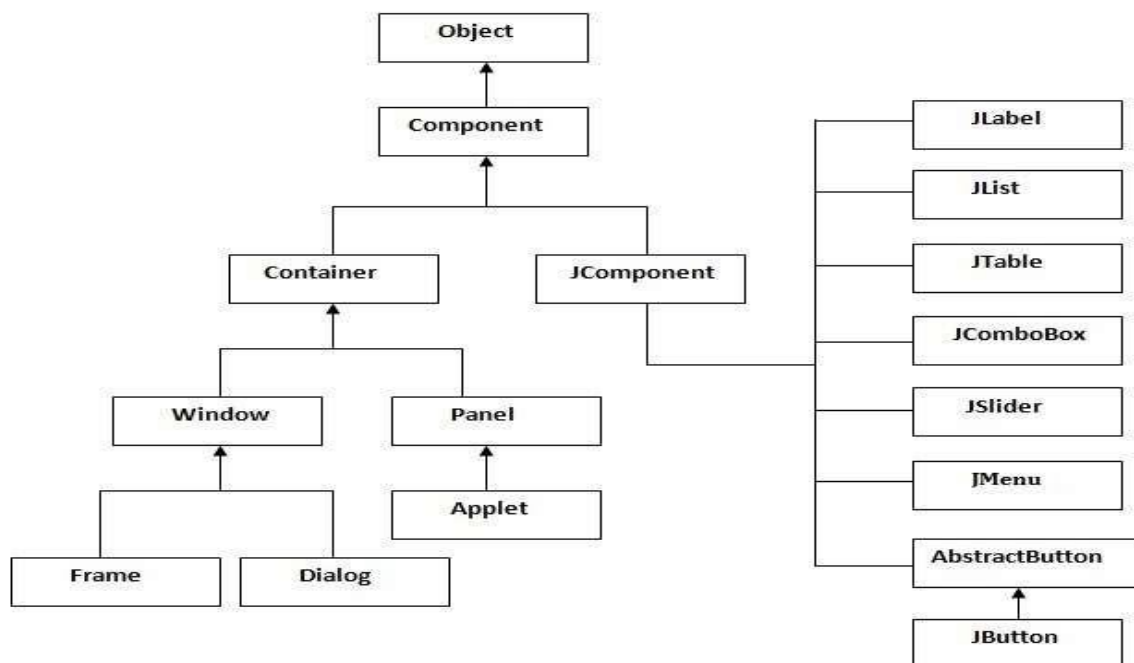


Fig. 5.1 Hierarchy of Java Swing classes

## CHAPTER – 6

### JAVA DATABASE CONNECTIVITY

---

The Java Database Connectivity (JDBC) API is the industry standard for database-independent connectivity between the Java programming language and a wide range of databases – SQL databases and other tabular data sources, such as spreadsheets or flat files. The JDBC API provides a call-level API for SQL-based database access. [5]

JDBC technology allows you to use the Java programming language to exploit "Write Once, Run Anywhere" capabilities for applications that require access to enterprise data. With a JDBC technology-enabled driver, you can connect all corporate data even in a heterogeneous environment.

#### 6.1. JDBC Drivers

There are commercial and free drivers available for most relational database servers. These drivers fall into one of the following types:

**Type 1** That calls native code of the locally available ODBC driver.

**Type 2** That calls database vendor native library on a client side. This code then talks to database over network.

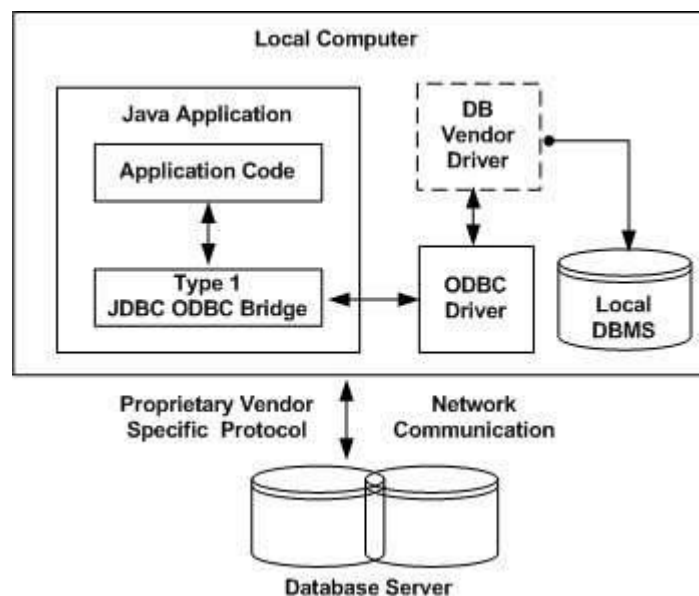


Fig. 6.1 JDBC Drivers



**Type 3**, the pure-java driver that talks with the server-side middleware those then talks to database.

**Type 4**, the pure-java driver that uses database native protocol.

## 6.2. Functionality and Implementation

JDBC allows multiple implementations to exist and be used by the same application. The API provides a mechanism for dynamically loading the correct Java packages and registering them with the JDBC Driver Manager. The Driver Manager is used as a connection factory for creating JDBC connections.

JDBC connections support Creating and executing statements. These may be update statements such as SQL's CREATE, INSERT, UPDATE and DELETE, or they may be query statements such as SELECT. Additionally, stored procedures may be invoked through a JDBC connection. JDBC represents statements using one of the following classes-

- **Statement** - the statement is sent to the database server each and every time.
- **Prepared Statement** - the statement is cached and then the execution path is predetermined on the database server allowing it to be executed multiple times in efficient manner.
- **Callable Statement** - used for executing stored procedure son the database. Update statements such as INSERT, UPDATE and DELETE return an update count that indicates how many rows were affected in the database.

These statements do not return any other information. Query statements return a JDBC row result set. The row result set is used to walk over the result set. Individual columns in a row are retrieved either by name or by column number. There may be any number of rows in the result set. The row result set has metadata that describes the names of the columns and their types.

There is an extension to the basic JDBC API in the javax.sql. The method Class.forName(String) is used to load the JDBC driver class. [6]

**Example-**Class.forName(sun.jdbc.odbc.JdbcOdbcDriver);

Used to load the JDBC-ODBC bridge driver.

When a Driver class is loaded, it creates an instance of itself and registers it with the Driver Manager. Now when a connection is needed, one of the DriverManager.getConnection() methods is used to create a JDBC connection.

**Example :-**

```
import java.sql.*;

import javax.sql.*;

public class jdbcdemo{

    public static void main(String args[]){

        String dbtime;

        String dbUrl = "jdbc:mysql://your.database.domain/yourDBname";

        String dbClass = "com.mysql.jdbc.Driver";

        String query = "Select * FROM users";

        try {

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

            Connection con = DriverManager.getConnection (dbUrl);

            Statement stmt = con.createStatement();

            ResultSet rs = stmt.executeQuery(query);

            while (rs.next()) {

                dbtime = rs.getString(1);
```

```
System.out.println(dbtime);

} //end while

con.close();

} //end try

catch(ClassNotFoundException e) {

e.printStackTrace();

}

catch(SQLException e) {

e.printStackTrace();

}

} //end main

} //end class
```

If a database operation fails, JDBC raises a `SQLException`. There is typically very little one can do to recover from such an error, apart from logging it with as much detail as possible. It is recommended that the `SQLException` be translated into an application domain exception (an unchecked one) that eventually results in a transaction rollback and a notification to the user. Finally to summarize, the following points can be stated

- JDBC API provides a database programming interface for Java programs. A Java program can send queries to a database by using the JDBC driver.
- The `java.sql` package contains classes that help in connecting to a database, sending SQL statements to the database and processing the query results.

- The Connection Object represents a connection with a database. It can be initialized using the getConnection() method of the DriverManager class.
- The PreparedStatement object allows you to execute parameterized queries. It can be initialized using the prepareStatement() method of the Connection object.
- The setString() method sets the query parameters of the PreparedStatement object.
- The executeUpdate() method executes the query statement present in the PreparedStatement object and returns the number of rows affected by the query.
- The ResultSetMetaData interface is used to obtain information about the columns stored in a ResultSet object.

## CHAPTER – 7

### SERVLET

Servlets are protocol and platform independent server-side software components, written in Java. They run inside a Java enabled server or application server, such as the WebSphere Application Server. Servlets are loaded and executed within the Java Virtual Machine (JVM) of the Web server or application server, in much the same way that applets are loaded and executed within the JVM of the Web client.

Since servlets run inside the servers, however, they do not need a graphical user interface (GUI). In this sense, servlets are also faceless objects.

Servlets more closely resemble Common Gateway Interface (CGI) scripts or programs than applets in terms of functionality. As in CGI programs, servlets can respond to user events from an HTML request, and then dynamically construct an HTML response that is sent back to the client. [7]

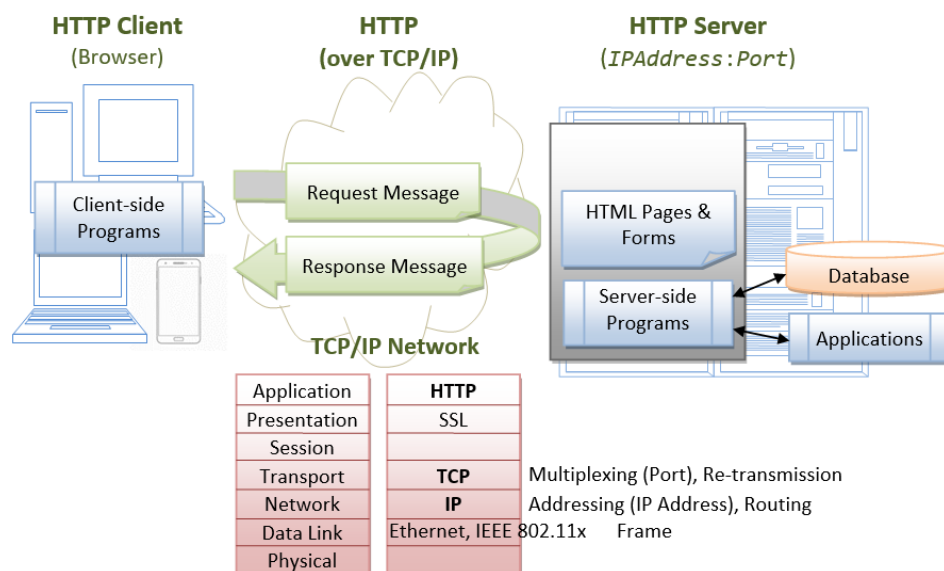


Fig. 7.1 Servlets

#### 7.1 The Java Servlet API

The Java Servlet API is a set of Java classes which define a standard interface between a Web client and a Web servlet. Client requests are made to the Web

server, which then invokes the servlet to service the request through this interface.

The API is composed of two packages:

- javax.servlet
- javax.servlet.http

The Servlet interface class is the central abstraction of the Java Servlet API. This class defines the methods which servlets must implement, including a `service()` method for the handling of requests. The `GenericServlet` class implements this interface, and defines a generic, protocol-independent servlet.

To write an HTTP servlet for use on the Web, we will use an even more specialized class of `GenericServlet` called `HttpServlet`. `HttpServlet` provides additional methods for the processing of HTTP requests such as GET (`doGet` method) and POST (`doPost` method). Although our servlets may implement a `service` method, in most cases we will implement the HTTP specific request handling methods of `doGet` and `doPost`.

## 7.2 Servlet Life Cycle

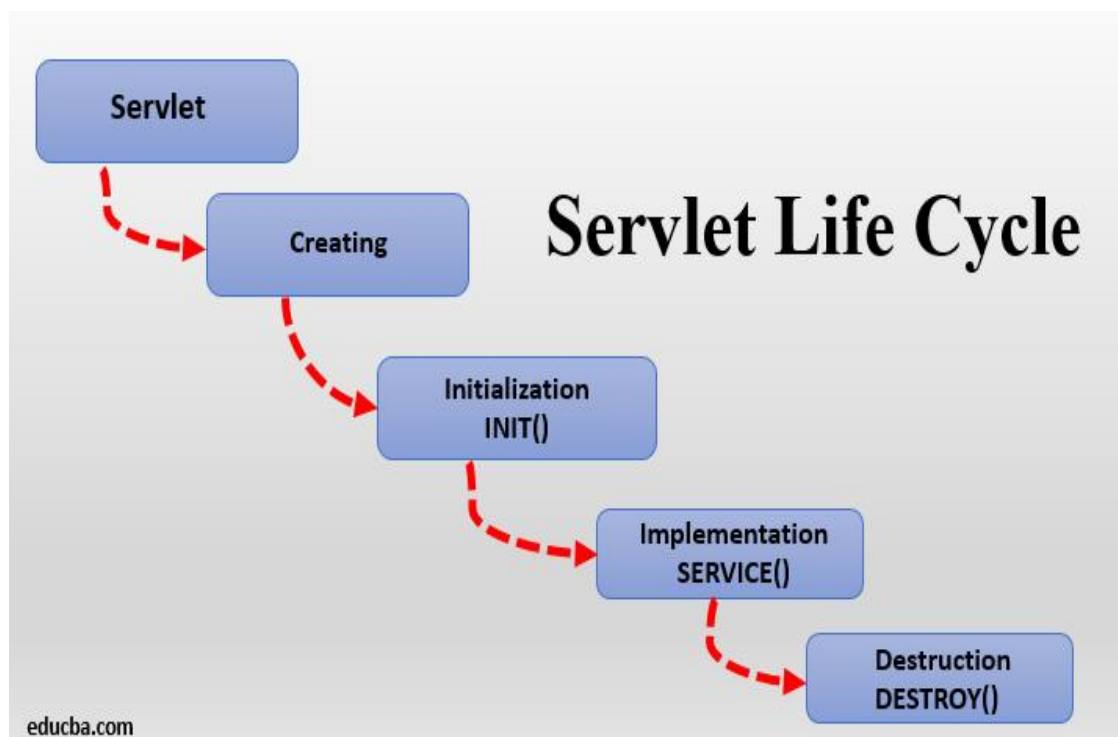


Fig. 7.2 Servlet Life Cycle

The life cycle of a servlet can be categorized into four parts:

- **Loading and Instantiation-** The servlet container loads the servlet during startup or when the first request is made. The loading of the servlet depends on the attribute `<load-onstartup>` of web.xml file. If the attribute `<load-on-startup>` has a positive value then the servlet is load with loading of the container otherwise it load when the first request comes for service. After loading of the servlet, the container creates the instances of the servlet.
- **Initialization-** After creating the instances, the servlet container calls the `init()` method and passes the servlet initialization parameters to the `init()` method. The `init()` must be called by the servlet container before the servlet can service any request. The initialization parameters persist untill the servlet is destroyed. The `init()` method is called only once throughout the life cycle of the servlet. The servlet will be available for service if it is loaded successfully otherwise the servlet container unloads the servlet.
- **Servicing the Request:** After successfully completing the initialization process, the servlet will be available for service. Servlet creates separate threads for each request. The sevlet container calls the `service()` method for servicing any request. The `service()` method determines the kind of request and calls the appropriate method (`doGet()` or `doPost()`) for handling the request and sends response to the client using the methods of the response object.
- **Destroying the Servlet-** If the servlet is no longer needed for servicing any request, the servlet container calls the `destroy()` method . Like the `init()` method this method is also called only once throughout the life cycle of the servlet. Calling the `destroy()` method indicates to the servlet container not to sent the any request for service and the servlet releases all the resources associated with it. Java Virtual Machine claims for the memory associated with the resources for garbage collection.

An applet is a Java program that runs in a Web browser. An applet can be a fully functional Java application because it has the entire Java API at its disposal. An applet is embedded in an HTML page using the APPLET or OBJECT tag and hosted on a web server. Applets are used to make the web site more dynamic and entertaining. [8]

There are some important differences between an applet and a standalone Java application, including the following –

- An applet is a Java class that extends the `java.applet.Applet` class.
- A `main()` method is not invoked on an applet, and an applet class will not define `main()`.
- Applets are designed to be embedded within an HTML page.
- When a user views an HTML page that contains an applet, the code for the applet is downloaded to the user's machine.
- A JVM is required to view an applet. The JVM can be either a plug-in of the Web browser or a separate runtime environment.
- The JVM on the user's machine creates an instance of the applet class and invokes various methods during the applet's lifetime.
- Applets have strict security rules that are enforced by the Web browser. The security of an applet is often referred to as sandbox security, comparing the applet to a child playing in a sandbox with various rules that must be followed.
- Other classes that the applet needs can be downloaded in a single Java Archive (JAR) file.



## 8.1. Life Cycle of an Applet :-

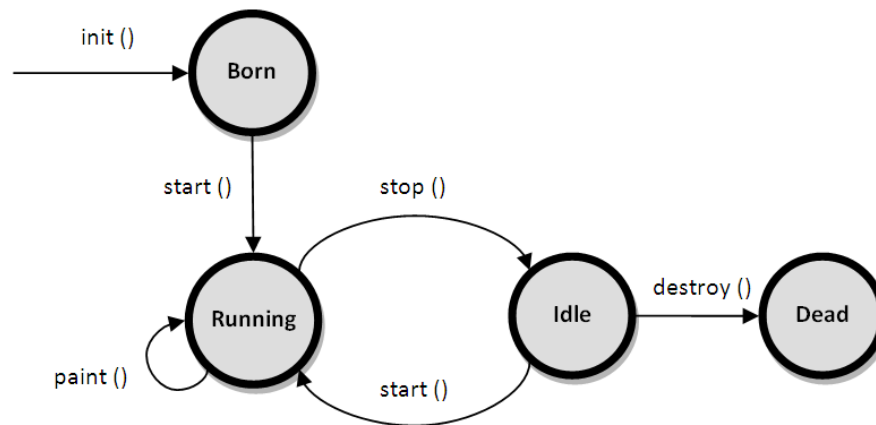


Fig. 8.1 Servlet Life Cycle

Four methods in the Applet class give you the framework on which you build any serious applet –

- i. **init** – This method is intended for whatever initialization is needed for your applet. It is called after the param tags inside the applet tag have been processed.
- ii. **start** – This method is automatically called after the browser calls the `init` method. It is also called whenever the user returns to the page containing the applet after having gone off to other pages.
- iii. **stop** – This method is automatically called when the user moves off the page on which the applet sits. It can, therefore, be called repeatedly in the same applet.
- iv. **destroy** – This method is only called when the browser shuts down normally. Because applets are meant to live on an HTML page, you should not normally leave resources behind after a user leaves the page that contains the applet.
- v. **paint** – Invoked immediately after the `start()` method, and also any time the applet needs to repaint itself in the browser. The `paint()` method is actually inherited from the `java.awt`.

# Project : Blood Bank

## RCPIT BLOOD BANK

Login Here

ADMIN LOGIN

BLOOD BANK LOGIN

USER LOGIN

## DATABASE

phpMyAdmin

Server: 127.0.0.1 » Database: bloodbank » Table: bloodbank

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 3 (4 total, Query took 0.0015 seconds.)

SELECT \* FROM 'bloodbank'

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table

bankid	bankname	bankcontact	bankaddress	bankemail	bankpwd	apos	aneg	bpos	bneg	opos	oneg	abpos	abneg
1001	Dhule Blood Bank	8857954780	Datta mandir,Dhule	dhulebb@gmail.com	12345	9	4	4	4	4	14	4	4
1003	Shirpur Blood Bank	1235446565	Nimzari Naka,Shirpur	shirpurbb@gmail.com	123	0	0	0	0	0	0	0	0
1002	Rcpit Blood Bank	9234567812	RCPIT,Shirpur	rcp@gmail.com	rcpit	1	1	1	1	1	1	1	1
10014	Sakri Blood Bank	8857954781	Sakri	sakribb@gmail.com	sakri	0	0	0	0	0	0	0	0

Show all | Number of rows: 25 | Filter rows: Search this table

Query results operations

Print Copy to clipboard Export Display chart Create view

Bookmark this SQL query

Let every user access this bookmark

Java EE - BloodBank/WebContent/index.jsp - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

- build
- RemoteSystemsTempFiles
- WebContent
  - angularjs
  - assets
  - css
  - font-awesome
  - fonts
  - images
  - js
  - META-INF
  - WEB-INF
  - webfonts
    - addadminuser.jsp
    - addbb.jsp
    - addtips.jsp
    - adduser.jsp
    - adminlogin.jsp
    - adreq.jsp
    - applyreq.jsp
    - appreq.jsp
    - bbdetails.jsp
    - bblogin.jsp
    - deleteadmin.jsp
    - deletebb.jsp
    - deletetips.jsp
    - deleteuser.jsp
    - disappreq.jsp
    - index.jsp
    - upd.jsp
    - userlogin.jsp
    - viewadmin.jsp

index.jsp

```

1 <? page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html>
4 <html lang="en">
5
6 <head>
7   <title>RCBIT BLOOD BANK</title>
8   <!-- meta tags -->
9   <meta charset="UTF-8" />
10  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
11  <meta name="keywords" content="Art Sign Up Form Responsive Widget, Audio and Video players, Login Form Web Template, Flat
12    Flat Web Templates, Login Sign-up Responsive Web Template, Smartphone Compatible Web Template, Free Web Designs for No
13  />
14  <!-- /meta tags -->
15  <!-- custom style sheet -->
16  <link href="css/style.css" rel="stylesheet" type="text/css" />
17  <!-- /custom style sheet -->
18  <!-- fontawesome css -->
19  <link href="css/fontawesome-all.css" rel="stylesheet" />
20  <!-- /fontawesome css -->
21  <!-- google fonts-->
22  <link href="//fonts.googleapis.com/css?family=Raleway:100,100i,200,200i,300,300i,400,400i,500,500i,600,600i,700,700i,800,8
23    rel="stylesheet"
24  <!-- /google fonts-->
25
26 <script>
27   function validateEmail(emailField)
28   {
29     var reg = /^[A-Za-z0-9_\-.]+\@[A-Za-z0-9_\-.]+\.[A-Za-z]{2,4}$/;
30     if (reg.test(emailField.value) == false)
31     {
32       alert('Invalid Email Address');
33       return false;
34     }
35   }

```

Java EE - BloodBank/WebContent/index.jsp - Eclipse

File Edit Navigate Search Project Run Window Help

Project Explorer

- webfonts
  - addadminuser.jsp
  - addbb.jsp
  - addtips.jsp
  - adduser.jsp
  - adminlogin.jsp
  - adreq.jsp
  - applyreq.jsp
  - appreq.jsp
  - bbdetails.jsp
  - bblogin.jsp
  - deleteadmin.jsp
  - deletebb.jsp
  - deletetips.jsp
  - deleteuser.jsp
  - disappreq.jsp
  - index.jsp
  - upd.jsp
  - userlogin.jsp
  - viewadmin.jsp
  - viewbb.jsp
  - viewbbtips.jsp
  - viewbdcamp.jsp
  - viewreq.jsp
  - viewtips.jsp
  - viewuser.jsp
  - viewuserbb.jsp
  - viewutips.jsp
- CRM
- RK
- Servers

index.jsp

```

41
42 <body>
43   <h1 style="color:Red; text-shadow: 0.5px 0.5px 0.5px red;">RCBIT BLOOD BANK</h1>
44   <div class="w3l-login-form">
45     <h2 style="color:yellow; text-shadow: 0.5px 0.5px 0.5px black;">Login Here</h2>
46
47     <form action="adminlogin.jsp">
48       <button type="submit" style="color:rgb(248, 204, 114); text-shadow: 0.5px 0.5px 0.5px red;">Admin Login</button>
49     </form>
50     <br><br><br>
51     <form action="bblogin.jsp">
52       <button type="submit" style="color:rgb(248, 204, 114); text-shadow: 0.5px 0.5px 0.5px red;">Blood Bank Login</button>
53     </form>
54     <br><br><br>
55     <form action="userlogin.jsp">
56       <button type="submit" style="color:rgb(248, 204, 114); text-shadow: 0.5px 0.5px 0.5px red;">User Login</button>
57     </form>
58   </div>
59
60 <SCRIPT language=JavaScript>
61   function isNumberKey(evt)
62   {
63     var charCode = (evt.which) ? evt.which : event.keyCode;
64
65     if (charCode > 31 && (charCode<48 || charCode>57))
66       return false;
67
68     return true;
69   }
70
71 </SCRIPT>
72
73 </body>
74

```

# RCPIT BLOOD BANK

## Admin Login

EMAIL:



admin@gmail.com

PASSWORD:



.....

LOGIN

[Home Page](#) Click Here

# RCPIT BLOOD BANK

## Blood Bank Login

EMAIL:



rcp@gmail.com

PASSWORD:



.....

LOGIN

[Home Page](#) Click Here

RCPIT BLOOD BANK
Logout

- Add Blood Bank
- View Blood Bank
- View Users
- Add Health Tips
- View Health Tips

## Add New Blood Bank

Blood Bank ID :

Blood Bank Name :

Blood Bank Contact :

Blood Bank Address :

Blood Bank Email ID :

Blood Bank Password :

Add Blood Bank

RCPIT BLOOD BANK
Logout

- Add Blood Bank
- View Blood Bank
- View Users
- Add Health Tips
- View Health Tips

## View Blood Banks

Blood Bank ID	Blood Bank Name	Contact No.	Address	Email	A+	A-	B+	B-	O+	O-	AB+	AB-	Action
1001	Dhule Blood Bank	8857954780	Datta mandir,Dhule	dhulebb@gmail.com	9	4	4	4	4	14	4	4	Delete
1003	Shirpur Blood bank	1235446565	Nimzari Naka,Shirpur	shirpurbb@gmail.com	0	0	0	0	0	0	0	0	Delete
1002	Rcpit Blood Bank	9234567812	RCPIT,Shirpur	rcp@gmail.com	0	0	0	0	0	0	0	0	Delete
10014	Sakri Blood Bank	8857954781	Sakri	sakribb@gmail.com	0	0	0	0	0	0	0	0	Delete

RCPIT BLOOD BANK
Logout

- Add Blood Bank
- View Blood Bank
- View Users
- Add Health Tips
- View Health Tips

### View Health Tips

No.	***** HEALTH TIPS *****	Action
2	Above 18 Years	<a href="#">Delete</a>
1	Take Care	<a href="#">Delete</a>

RCPIT BLOOD BANK
Logout

- Add Blood Donation Camp
- View Blood Donation Camp
- Update Stock
- Blood Bank Details
- View Health Tips
- View Requests

### View Blood Donation Camps

Blood Donation Camp ID	Blood Donation Camp Name	Camp Address	Camp Date	Camp Time	Action
1111	Dhule Camp	0502201956	10 AM	Dhule Blood Bank	<a href="#">Delete</a>

RCPIT BLOOD BANK
Logout

- Add Blood Donation Camp
- View Blood Donation Camp
- Update Stock
- Blood Bank Details
- View Health Tips
- View Requests

### Add Blood Groups Stock

A+ve Blood Groups :

A-ve Blood Groups :

B+ve Blood Groups :

B-ve Blood Groups :

O+ve Blood Groups :

O-ve Blood Groups :

AB+ve Blood Groups :

AB-ve Blood Groups :



Add Stock

RCPIT BLOOD BANK
Logout

- Add Blood Donation Camp
- View Blood Donation Camp
- Update Stock
- Blood Bank Details
- View Health Tips
- View Requests

### View Blood Banks

Blood Bank ID	Blood Bank Name	Contact No.	Address	Email	A+	A-	B+	B-	O+	O-	AB+	AB-
1002	Rcpit Blood Bank	9234567812	RCPIT,Shirpur	rcp@gmail.com	1	1	1	1	1	1	1	1

  
 Cancel

### Add New User


User ID :	<input type="text" value="1"/>
User Name :	<input type="text" value="Rohan"/>
User Contact :	<input type="text" value="7558731293"/>
User Address :	<input type="text" value="Shirpur"/>
User BloodGroup :	<input type="text" value="o+ve"/>
User Email ID :	<input type="text" value="rohan@gmail.com"/>
User Password :	<input type="password" value="...."/>

Add User


# RCPIT BLOOD BANK

## User Login

EMAIL:

 rohan@gmail.com

PASSWORD:

 ....

LOGIN

[Home Page](#)  
[Register Here](#)





View Blood Banks



View Blood Donation Camps



Apply Request



View Health Tips

## View Blood Banks

Blood Bank ID	Blood Bank Name	Contact No.	Address	Email	A+	A-	B+	B-	O+	O-	AB+	AB-
1001	Dhule Blood Bank	8857954780	Datta mandir,Dhule	dhulebb@gmail.com	9	4	4	4	4	14	4	4
1003	Shirpur Blood bank	1235446565	Nimzari Naka,Shirpur	shirpurbb@gmail.com	0	0	0	0	0	0	0	0
1002	Rcpit Blood Bank	9234567812	RCPIT,Shirpur	rcp@gmail.com	1	1	1	1	1	1	1	1
10014	Sakri Blood Bank	8857954781	Sakri	sakribb@gmail.com	0	0	0	0	0	0	0	0



View Blood Banks



View Blood Donation Camps



Apply Request



View Health Tips

## Apply Blood Group Request

Request ID :

1

User Name :

Rohan

User Contact :

7558731293

User Address :

Shirpur


User BloodGroup :

o+ve

Request

RCPIT BLOOD BANK

Logout



Add Blood Donation Camp

View Blood Donation Camp

Update Stock

Blood Bank Details


View Health Tips

View Requests

View Requests

Request ID	User Name	Contact No.	Address	Blood Group	Status
1	Rohan	7558731293	Shirpur	O+ve	Approved DisApproved


Type here to search



21:21  
31-10-2019

RCPIT BLOOD BANK

Logout



Add Blood Donation Camp

View Blood Donation Camp

Update Stock

Blood Bank Details

View Health Tips

View Requests

View Requests

Request ID	User Name	Contact No.	Address	Blood Group	Status
1	Rohan	7558731293	Shirpur	O+ve	Approved

## CONCLUSION

---

This Internship focused upon increasing our knowledge and interest in toward the java. Because java is most interesting and most used language in these days. We learnt how to create a web sites and web pages. It was a great experience. It increases our practical skills.

The Advance Java problem set, a step-by-step approach used by us, has ensured that we acquired enough knowledge required in various aspects for all the stages, before proceeding on.

During these Days, we had done the following:

- Implemented a GUI that allowed a user to view and modify the information in an XML document. XSL was used to transform XML into HTML for display in a browser.
- Wrote a Java Servlet code to handle the file access application required by the web client. These are application that cannot be handled by a Java Applet.
- Implemented the Text Editor Interface (client) using HTML, a straightforward language used to display certain interface program onto a web browser.
- Created a Text Editor Interface Application using the Java Applet. This allows greater flexibility than HTML in terms of the layout and functionality.

Having the knowledge of Java, HTML, XML, XSL, and UML proved to be immensely useful. This had also built up our confidence in dealing with feature-rich, user-friendly Windows applications.

## BIBLIOGRAPHY

- [1] Ken Arnold and James Gosling, The Java Programming Language, second ed., Addison-Wesley, 1998.
- [2] Gary Cornell and Cay S. Horstmann, Core Java, second ed., SunSoft Press, 1997.
- [3] David M. Geary, Graphic Java 2: Mastering the JFC, vol. I, AWT, third ed., Sun Microsystems Press, 1999.
- [4] Herbert Schildt, Java The Complete Reference, McGraw-Hill Publication, 9th Edition,
- [5] James Gosling and Bill Joy and Guy Steele, The Java Language Specification, Addison-Wesley, 1996.
- [6] Elliotte Rusty Harold, *Java Network Programming*, Prentice-Hall, 1997.
- [7] Jerry R. Jackson and Alan L. McClellan, Java by Example, SunSoft Press, 1996.
- [8] George Reese, *Database Programming with JDBC and Java*, O'Reilly, 1997.