INTERNSHIP REPORT

Submitted by

REEMA MANSOORA S (717821E143)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

ANNA UNIVERSITY, CHENNAI

DECEMBER 2023

KARPAGAM
COLLEGE OF ENGINEERING
Rediscover | Refine | Redefine

CERTIFICATE

This is to certify that the **Internship report** submitted by **Reema Mansoora S 717821E143** is work done by him and submitted during 2023 – 2024 academic year, in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF ENGINEERING in ELECTRICAL AND ELECTRONICS ENGINEERING**.

Department Internship Coordinator **Head of the Department**

Certified that the candidate was examined in the viva-voce examination held on…….

……..
(Internal Examiner)

CERTIFICATE FROM INDUSTRY:

N·E·A·T
प्रौद्योगिकी के लिए राष्ट्रीय शैक्षणिक सहयोग
National Educational Alliance for Technology

Ministry of Education
Government of India
सत्यमेव जयते

SkillDzire

CERTIFICATE OF INTERNSHIP

This certificate is presented to

Reema Mansoora S (717821E143)

From **Karpagam College Of Engineering**

in recognition of his/her hardwork and dedication in completing

internship in **Full Stack Java**

From **1-August-2023** To **2-September-2023**

The remarkable level of knowledge and skills demonstrated throughout the program has earned the recipient this award.

Approved by AICTE

Authorized Signature

ACKNOWLEDGEMENT

We express our sincere thanks to Karpagam educational and charitable trust for providing necessary facilities to bring out the Internship successfully. We felt greatness to record our thanks to the chairman **Dr. R. VASANTHAKUMAR, B.E., (Hons), D.Sc.,** for all his support and ray of strengthening hope extended.

It is the moment of immense pride for us to reveal our profound thanks to our respected principal, **Dr. V. KUMAR CHINNAIYAN, M.E., Ph.D.,** who happens to be striving forcein all our endeavors.

We express our sincere thanks to our **Dr. V.TAMILSELVAN, M.E., Ph.D.,** Head of the Department of Electrical and Electronics Engineering for providing us with the opportunity for this internship.

We would also like to recollect the courage and enthusiasm that was inculcated in us by our internship coordinator, **Dr. C. S. SUNDAR GANESH, M.E., Ph.D.,** Assistant Professor, Department of Electrical and Electronics Engineering for valuable guidance and support through the tenure of our internship.

We deeply express our gratitude to all the faculty members of the Department of Electrical and Electronics Engineering for their encouragement, which we received through out the semester.

TABLE OF CONTENT

ABSTRACT

This internship provides an extensive knowledge in fullstack Java development, spanning 4 weeks from 01-08-2023 to 02-09-2023. The core focus of the internship was to gain hands-on experience in fullstack Java, including Java, Servlets, Hyper Text Markup Language, Cascading Style Sheets, Java Data Base Connectivity, C, and MySQL offering a deep dive into these technologies and their practical applications. The report offers a comprehensive insight into the intern's journey and experiences while working with Java, Servlets, Hyper Text Markup Language, Cascading Style Sheets, Java Data Base Connectivity, C, and MySQL. It delves into the skills acquired and lessons learned during the internship, emphasizing the development of not only technical expertise but also crucial problem-solving skills essential in a professional context. The experience reinforced the importance of adapting to real-world scenarios and resilience. The internship played a pivotal role in enhancing the intern's proficiency in Fullstack Java development, bridging the gap between theoretical knowledge and practical application. It illuminated the intricate interplay of technologies and their roles in building robust, real-world solutions. This abstract stands as a testament to the intern's commitment to self-improvement and dedication to mastering Fullstack Java development. The internship offered a transformative journey while instilling a sense of readiness for the dynamic world of Full stack Java development.

LIST OF FIGURES

LIST OF TABLES

WEEKLY OVERVIEW OF INTERNSHIP ACTIVITIES

	DATE	DAY	NAME OF THE TOPIC/ MODULE COMPLETED
1st WEEK	01-08-2023	TUESDAY	JAVA introduction and Object -Oriented Design
	02-08-2023	WEDNESDAY	Inheritance and Abstraction
	03-08-2023	THURSDAY	JAVA string
	04-08-2023	FRIDAY	Introduction to JAVA Servlets
	05-08-2023	SATURDAY	Servlet filter methods

	DATE	DAY	NAME OF THE TOPIC/ MODULE COMPLETED
2nd WEEK	07-08-2023	MONDAY	HTML
	08-08-2023	TUESDAY	HTML Editors
	09-08-2023	WEDNESDAY	Tags in HTML
	10-08-2023	THURSDAY	HTML attributes
	11-08-2023	FRIDAY	CSS Introduction
	12-08-2023	SATURDAY	Use of CSS

	DATE	DAY	NAME OF THE TOPIC/ MODULE COMPLETED
3rd WEEK	14-08-2023	MONDAY	CSS Syntax
	15-08-2023	TUESDAY	CSS Modules
	16-08-2023	WEDNESDAY	JDBC
	17-08-2023	THURSDAY	Purpose of JDBC
	18-08-2023	FRIDAY	Architecture of JDBC
	19-08-2023	SATURDAY	Interface of JDBC API

	DATE	DAY	NAME OF THE TOPIC/ MODULE COMPLETED
4th WEEK	28-08-2023	MONDAY	Creating simple JDBC applications
	29-08-2023	TUESDAY	MySQL
	30-08-2023	WEDNESDAY	MySQL a relational database model
	31-08-2023	THURSDAY	Benefits of MySQL
	01-09-2023	FRIDAY	Use cases of MySQL
	02-09-2023	SATURDAY	Open source and choice of developers

CHAPTER 1

INTRODUCTION

The Java Development Kit (JDK) is software used for Java programming, along with the Java Virtual Machine (JVM) and the Java Runtime Environment (JRE). The JDK includes the compiler and class libraries, allowing developers to create Java programs executable by the JVM and JRE.

1.1 IDENTIFIERS

Identifiers in Java are symbolic names used for identification. They can be a class name, variable name, method name, package name, constant name, and more. However, In Java, There are some reserved words that can not be used as an identifier.

For every identifier there are some conventions that should be used before declaring them. Let's understand it with a simple Java program:

```
1. public class HelloJava {
2. public static void main(String[] args) {
3. System.out.println("Hello JavaTpoint");
4. }
5. }
```

Rules for Identifiers in Java

There are some rules and conventions for declaring the identifiers in Java. If the identifiers are not properly declared, we may get a compile-time error. Following are some rules and conventions for declaring identifiers:

- A valid identifier must have characters [A-Z] or [a-z] or numbers [0-9], and underscore(_) or a dollar sign ($). for example, @javatpoint is not a valid identifier because it contains a special character which is @.
- There should not be any space in an identifier. For example, java tpoint is an invalid identifier.
- An identifier should not contain a number at the starting. For example, 123javatpoint is an invalid identifier.
- An identifier should be of length 4-15 letters only. However, there is no limit on its length. But, it is good to follow the standard conventions.
- We can't use the Java reserved keywords as an identifier such as int, float, double, char, etc. For example, int double is an invalid identifier in Java.
- An identifier should not be any query language keywords such as SELECT, FROM, COUNT, DELETE, etc.

Java Reserved Keywords

Java reserved keywords are predefined words, which are reserved for any functionality or meaning. We can not use these keywords as our identifier names, such as class name or method name. These keywords are used by the syntax of Java for some functionality. If we use a reserved word as our variable name, it will throw an error.

1.2 DATATYPE CLASSES

Data types are different sizes and values that can be stored in the variable that is made as per convenience and circumstances to cover up all test cases. Also, let us cover up other important ailments that there are majorly two types of languages that are as follows:

1. First, one is a Statically typed language where each variable and expression type is already known at compile time. Once a variable is declared to be of a certain data type, it cannot hold values of other data types. For example C, C++, Java.
2. The other is Dynamically typed languages. These languages can receive different data types over time. For example Ruby, Python

Java has two categories in which data types are segregated

1. Primitive Data Type: such as boolean, char, int, short, byte, long, float, and double
2. Non-Primitive Data Type or Object Data type: such as String, Array, etc.

Types Of Primitive Data Types

Primitive data are only single values and have no special capabilities. There are 8 primitive data types. They are depicted below in Table 1.1.

TYPE	DESCRIPTION	DEFAULT	SIZE
boolean	True or false	False	1 bit
byte	Twos complement integer	0	8 bits
char	Unicode character	\u0000	16 bits
short	Twos complement integer	0	16 bits
int	Twos complement integer	0	32 bits
long	Twos complement integer	0	64 bits
float	IEEE 754 floating point	0.0	32 bits
double	IEEE 754 floating point	0.0	64 bits

Table 1.1 Primitive data types

1.3 OBJECTS

The Object is the real-time entity having some state and behaviour. In Java, Object is an instance of the class having the instance variables as the state of the object and the methods as the behaviour of the object. The object of a class can be created by using the new keyword in Java Programming language. A class is a template or blueprint from which objects are created. So, an object is the instance(result) of a class.

Object-oriented design started right from the moment computers were invented. Programming was there, and programming approaches came into the picture. Programming is basically giving certain instructions to the computer.

At the beginning of the computing era, programming was usually limited to machine language programming. Machine language means those sets of instructions that are specific to a particular machine or processor, which are in the form of 0's and 1's. These are sequences of bits (0100110…). But it's quite difficult to write a program or develop software in machine language.

It's actually impossible to develop software used in today's scenarios with sequences of bits. This was the main reason programmers moved on to the next generation of programming languages, developing assembly languages, which were near enough to the English language to easily understand. These assembly languages were used in microprocessors. With the invention of the microprocessor, assembly languages flourished and ruled over the industry, but it was not enough. Again, programmers came up with something new, i.e., structured and procedural programming, as shown in Fig 1.1.

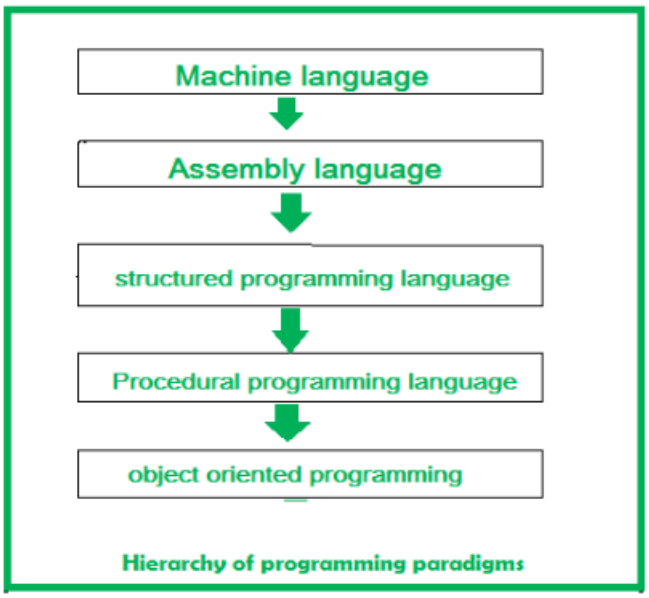

Fig1.1 Hierarchy of programming paradigms

CHAPTER 2

OBJECT ORIENTED PROGRAMMING

Object-oriented programming (OOP) is a programming paradigm that focuses on organizing and structuring code around objects, which are instances of classes. It promotes the use of objects to model and represent real-world entities and their interactions.

2.1 ABSTRACTION

Abstraction refers to the act of representing important and special features without including the background details or explanation about that feature. Data abstraction simplifies database design shown in Fig2.1.

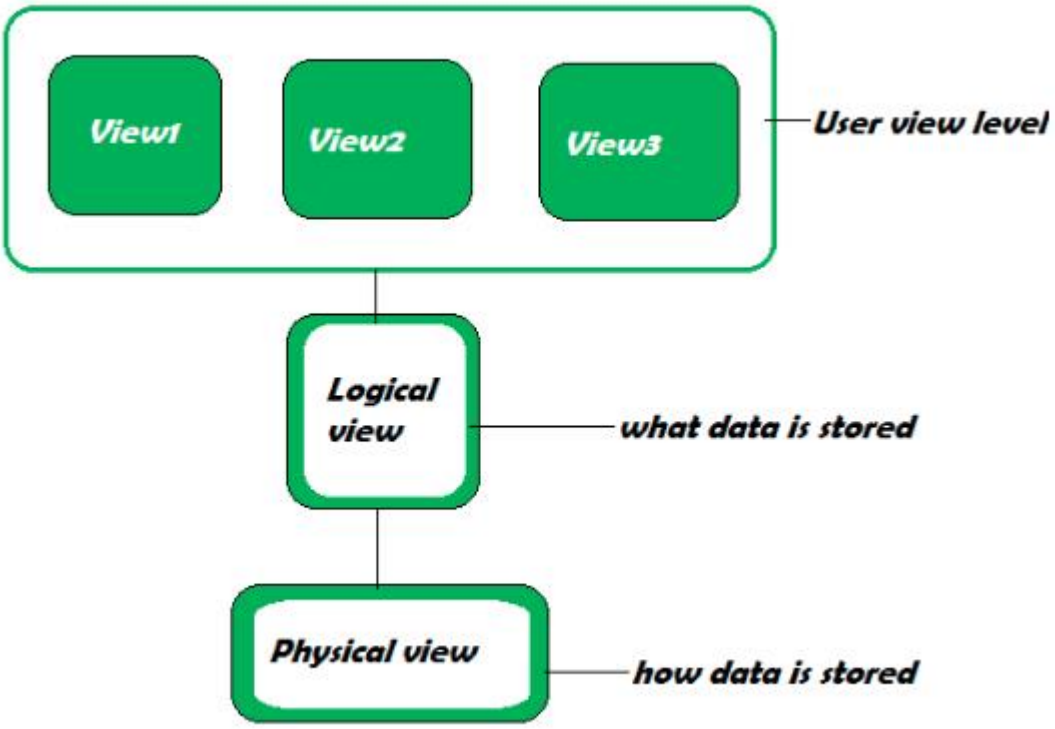

Fig2.1 Abstraction overview

2.2 ENCAPSULATION.

Encapsulation is one of the fundamental concepts in object-oriented programming (OOP). It describes the idea of wrapping data and the methods that work on data within one unit, e.g., a class in Java. This concept is often used to hide the internal state representation of an object from the outside

2.3 INHERITANCE

Inheritance is an important pillar of OOP(Object-Oriented Programming). It is the mechanism in java by which one class is allowed to inherit the features(fields and methods) of another class.

Important terminology:

- Super Class: The class whose features are inherited is known as a superclass(or a base class or a parent class).
- Sub Class: The class that inherits the other class is known as a subclass(or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass fields and methods.
- Reusability: Inheritance supports the concept of "reusability", i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

How to use inheritance in Java

The keyword used for inheritance is extends.

Types of Inheritance in Java

Below are the different types of inheritance which are supported by Java.

1. Single Inheritance: In single inheritance, subclasses inherit the features of one superclass. In the image below, class A serves as a base class for the derived class B.
2. Multilevel Inheritance: In Multilevel Inheritance, a derived class will be inheriting a base class and as well as the derived class also act as the base class to other class. In the below image, class A serves as a base class for the derived class B, which in turn serves as a base class for the derived class C. In Java, a class cannot directly access the grandparent's members.
3. Hierarchical Inheritance: In Hierarchical Inheritance, one class serves as a superclass (base class) for more than one subclass. In the below image, class A serves as a base class for the derived class B, C and D.
4. Multiple Inheritance (Through Interfaces): In Multiple inheritances, one class can have more than one superclass and inherit features from all parent classes. Please note that Java does not support multiple inheritances with classes. In java, we can achieve multiple inheritances only through Interfaces. In the image below, Class C is derived from interface A and B.
5. Hybrid Inheritance(Through Interfaces): It is a mix of two or more of the above types of inheritance. Since java doesn't support multiple inheritances with classes, hybrid inheritance is also not possible with classes. In java, we can achieve hybrid inheritance only through Interfaces.

2.4 POLYMORPHISM

Polymorphism is the ability of data to be processed in more than one form. It allows the performance of the same task in various ways. It consists of method overloading and method overriding, i.e., writing the method once and performing a number of tasks using the same method name.

2.4.1 Method Overriding in Java

If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java**.**

In other words, If a subclass provides the specific implementation of the method that has been declared by one of its parent class, it is known as method overriding.

Usage of Java Method Overriding

- Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.
- Method overriding is used for runtime polymorphism

Rules for Java Method Overriding

1. The method must have the same name as in the parent class
2. The method must have the same parameter as in the parent class.
3. There must be an IS-A relationship (inheritance).

2.4.2 Method Overloading in Java

If a class has multiple methods having same name but different in parameters, it is known as Method Overloading**.**

If we have to perform only one operation, having same name of the methods increases the readability of the program.

Suppose you have to perform addition of the given numbers but there can be any number of arguments, if you write the method such as a(int, int) for two parameters, and b(int,int,int) for three parameters then it may be difficult for you as well as other programmers to understand the behaviour of the method because its name differs.

So, we perform method overloading to figure out the program quickly.

1) Method Overloading: changing no. of arguments

In this example, we have created two methods, first add() method performs addition of two numbers and second add method performs addition of three numbers.

In this example, we are creating static methods

so that we don't need to create instance for calling methods.

2)Method Overloading: changing data type of arguments

In this example, we have created two methods that differs in data type.

The first add method receives two integer arguments and second add method receives two double arguments.

2.4.3 Important points about OOPs

1. OOP treats data as a critical element.
2. Emphasis is on data rather than procedure.
3. Decomposition of the problem into simpler modules.
4. Doesn’t allow data to freely flow in the entire system, ie localized control flow.
5. Data is protected from external functions.

2.4.4 Advantages of OOPs

- It models the real world very well.
- With OOP, programs are easy to understand and maintain.
- OOP offers code reusability. Already created classes can be reused without having to write them again.
- OOP facilitates the quick development of programs where parallel development of classes is possible.
- With OOP, programs are easier to test, manage and debug.

2.4.5 Disadvantages of OOPs

- With OOP, classes sometimes tend to be over-generalized.
- The relations among classes become superficial at times.
- The OOP design is tricky and requires appropriate knowledge. Also, one needs to do proper planning and design for OOP programming.
- To program with OOP, the programmer needs proper skills such as design, programming, and thinking in terms of objects and classes, etc.

CHAPTER 3

JAVA STRING

In Java, string is basically an object that represents sequence of char values. An array of characters works same as Java string. For example:

```
1. char[] ch={'j','a','v','a','t','p','o','i','n','t'};
2. String s=new String(ch);
3. is same as:
4. String s="javatpoint";
```

3.1 STRING METHODS

The java.lang.String class provides a lot of built-in methods that are used to manipulate string in Java. By the help of these methods, we can perform operations on String objects such as trimming, concatenating, converting, comparing, replacing strings etc.

Java String is a powerful concept because everything is treated as a String if you submit any form in window based, web based or mobile application.

Let's use some important methods of String class.

Java String toUpperCase() and toLowerCase() method

The Java String toUpperCase() method converts this String into uppercase letter and String toLowerCase() method into lowercase letter.

```
1. public class Stringoperation1
2. {
3. public static void main(String ar[])
4. {
5. String s="Sachin";
6. System.out.println(s.toUpperCase());//SACHIN
7. System.out.println(s.toLowerCase());//sachin
8. System.out.println(s);//Sachin(no change in original)
9. }
10. }
```

Java String length() Method

```
1. public class Stringoperation5
2. {
3. public static void main(String ar[])
4. {
5. String s="Sachin";
6. System.out.println(s.length());//6
```

```
}
}
```

Java String trim() method

```
public class Stringoperation2
{
public static void main(String ar[])
{
String s="  Sachin  ";
System.out.println(s);//  Sachin
System.out.println(s.trim());//Sachin
}
}
```

Java String startsWith() and endsWith() method

The method startsWith() checks whether the String starts with the letters passed as arguments and endsWith() method checks whether the String ends with the letters passed as arguments.

```
public class Stringoperation3
{
public static void main(String ar[])
{
String s="Sachin";
System.out.println(s.startsWith("Sa"));//true
System.out.println(s.endsWith("n"));//true
}
}
```

Java String charAt() Method

The String class charAt() method returns a character at specified index.

```
public class Stringoperation4
{
public static void main(String ar[])
{
String s="Sachin";
System.out.println(s.charAt(0));//S
System.out.println(s.charAt(3));//h
}
}
```

CHAPTER 4

JAVA SERVLETS

4.1 SERVLETS

Today we all are aware of the need of creating dynamic web pages i.e the ones which have the capability to change the site contents according to the time or are able to generate the contents according to the request received by the client. If you like coding in Java, then you will be happy to know that using Java there also exists a way to generate dynamic web pages and that way is Java Servlet. But before we move forward with our topic let's first understand the need for server-side extensions.

Servlets are the Java programs that run on the Java-enabled web server or application server. They are used to handle the request obtained from the webserver, process the request, produce the response, then send a response back to the webserver.

Properties of Servlets are as follows:

- Servlets work on the server-side.
- Servlets are capable of handling complex requests obtained from the webserver.

Servlet Architecture is can be depicted from the Fig4.1 as provided below:

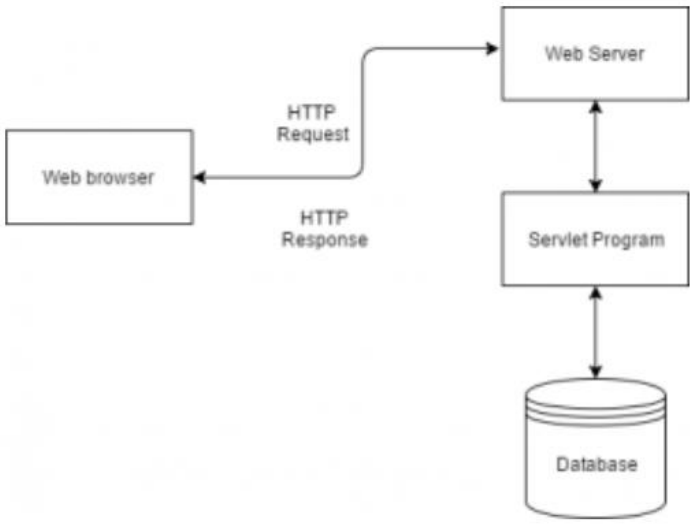

Fig4.1 Servlet Architecture

Execution of Servlets basically involves six basic steps:

1. The clients send the request to the webserver.
2. The web server receives the request.
3. The web server passes the request to the corresponding servlet.
4. The servlet processes the request and generates the response in the form of output.
5. The servlet sends the response back to the webserver.

6. The web server sends the response back to the client and the client browser displays it on the screen.

4.2 CGI

CGI is actually an external application that is written by using any of the programming languages like C or C++ and this is responsible for processing client requests and generating dynamic content.

In CGI application, when a client makes a request to access dynamic Web pages, the Web server performs the following operations ,

- It first locates the requested web page i.e the required CGI application using URL.
- It then creates a new process to service the client's request.
- Invokes the CGI application within the process and passes the request information to the application.
- Collects the response from the CGI application.
- Destroys the process, prepares the HTTP response, and sends it to the client.

A web page is the combination of many input elements such as label, text box, checkbox, options, images, etc., It can be prepared by enclosing all the input elements inside an "HTML FORM" on the server-side with java servlet. Usually, an HTML form collects data from the user via these input elements and lets us see how they will be sent to the server-side by using HTTP GET/POST methods.

4.3 HTTP GET/POST

1. Request will be sent from HTML form
2. It will be sent as either GET/POST method in the Java servlet side
3. Response will be provided from servlet as an HTML form

Following tags shown in Table 4.1 need to be known to handle the flow

TAG NAME	USAGE
<Form>	To create a form in a HTML page and it encloses all the input elements
<input>	To create a text box or password or submit etc.,
<select>	To create drop down
<textarea>	To create a text area

Table 4.1 HTML Tags

Along with the form tag, we have to specify the method either as “GET” or “POST”. Usually, sensitive data has to be passed as “POST” because, in the URL, the parameters are not sent as query parameters. Hence for sending sensitive information like “Password”, “Post” method is the good approach. Moreover

- In the “Post” method, we can pass many parameters.
- Post methods are not bookmarked
- There is no possibility of remaining in the browser history and also cannot be cached and hence safer

“POST” method is used to create or update the resources

4.4 CLIENT REQUEST

When a browser requests for a web page, it sends lot of information to the web server which cannot be read directly because this information travel as a part of header of HTTP request. You can check HTTP Protocol for more information on this.

4.5 SET HTTP RESPONSE HEADER

There are following methods in Table 4.2 which can be used to set HTTP response header in your servlet program. These methods are available with HttpServletResponse object.

S.NO	METHOD DESCRIPTION
1	String encodeRedirectURL() Encode the specified URL for use in the sendRedirect method or, if encoding is not needed , returns the URL unchanged.
2	String encodeURL() Encodes the specified URL by including the session id in it, or if encoding is not needed , returns the URL unchanged.
3	Boolean containsHeader() Returns a Boolean indicating whether the named response header has already been set
4	Boolean isCommitted() Returns a Boolean indicating the response has been committed

Table 4.2 HTTPServletResponse methods

Servlet Filters are Java classes that can be used in Servlet Programming for the following purposes −

- To intercept requests from a client before they access a resource at back end.
- To manipulate responses from server before they are sent back to the client.

There are various types of filters suggested by the specifications −

- Authentication Filters.
- Data compression Filters.
- Encryption Filters.
- Filters that trigger resource access events.
- Image Conversion Filters.
- Logging and Auditing Filters.
- MIME-TYPE Chain Filters.
- Tokenizing Filters .
- XSL/T Filters That Transform XML Content.

Filters are deployed in the deployment descriptor file web.xml and then map to either servlet names or URL patterns in your application's deployment descriptor.

When the web container starts up your web application, it creates an instance of each filter that you have declared in the deployment descriptor. The filters execute in the order that they are declared in the deployment descriptor.

4.6 SERVLET FILTER METHODS

A filter is simply a Java class that implements the javax.servlet.Filter interface. The javax.servlet.Filter interface defines three methods shown in Table 4.3.

S.NO	METHOD AND DESCRIPTION
1	Public void doFilter(ServletRequest,ServletResponse,FilterChain) This method is called by the container each time a request/response pair is passed through the chain due to a client request for a resource at the end of the chain.
2	Public void init(FilterConfig filterConfig) This method is called by the web container to indicate to a filter that is being placed into service
3	Public void destroy() This method is called by the web container to indicate to a filter that it is being taken out of service

Table 4.3 javax.servlet.Filter interface

Cookies are text files stored on the client computer and they are kept for various information tracking purpose. Java Servlets transparently supports HTTP cookies.

There are three steps involved in identifying returning users

- Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
- Browser stores this information on local machine for future use.
- When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

This chapter will teach you how to set or reset cookies, how to access them and how to delete them

Setting Cookies with Servlet

Setting cookies with servlet involves three steps −

(1) Creating a Cookie object − You call the Cookie constructor with a cookie name and a cookie value, both of which are strings.

```
Cookie cookie = new Cookie("key","value");
```

Keep in mind, neither the name nor the value should contain white space or any of the following characters −

```
[ ] ( ) = , " / ? @ : ;
```

(2) Setting the maximum age − You use setMaxAge to specify how long (in seconds) the cookie should be valid. Following would set up a cookie for 24 hours.

```
cookie.setMaxAge(60 * 60 * 24);
```

(3) Sending the Cookie into the HTTP response headers − You use response.addCookie to add cookies in the HTTP response header as follows .

```
response.addCookie(cookie);
```

HTTP is a "stateless" protocol which means each time a client retrieves a Web page, the client opens a separate connection to the Web server and the server automatically does not keep any record of previous client request.

Still there are following three ways to maintain session between web client and web server −

Cookies

A webserver can assign a unique session ID as a cookie to each web client and for subsequent requests from the client they can be recognized using the recieved cookie.

This may not be an effective way because many time browser does not support a cookie, so I would not recommend to use this procedure to maintain the sessions.

Hidden Form Fields

A web server can send a hidden HTML form field along with a unique session ID as follows −

```
<input type = "hidden" name = "sessionid" value = "12345">
```

This entry means that, when the form is submitted, the specified name and value are automatically included in the GET or POST data. Each time when web browser sends request back, then session_id value can be used to keep the track of different web browsers.

This could be an effective way of keeping track of the session but clicking on a regular (<A HREF...>) hypertext link does not result in a form submission, so hidden form fields also cannot support general session tracking.

URL Rewriting

You can append some extra data on the end of each URL that identifies the session, and the server can associate that session identifier with data it has stored about that session.

For example, with http://tutorialspoint.com/file.htm;sessionid = 12345, the session identifier is attached as sessionid = 12345 which can be accessed at the web server to identify the client.

URL rewriting is a better way to maintain sessions and it works even when browsers don't support cookies. The drawback of URL re-writing is that you would have to generate every URL dynamically to assign a session ID, even in case of a simple static HTML page.

The HttpSession Object

Apart from the above mentioned three ways, servlet provides HttpSession Interface which provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.

The servlet container uses this interface to create a session between an HTTP client and an HTTP server. The session persists for a specified time period, across more than one connection or page request from the user.

You would get HttpSession object by calling the public method getSession() of HttpServletRequest, as below −

```
HttpSession session = request.getSession();
```

You need to call request.getSession() before you send any document content to the client.

CHAPTER 5

HYPER TEXT MARKUP LANGUAGE

HTML is the language in which most websites are written**.** HTML is used to create pages and make them functional.The code used to make them visually appealing is known as CSS and we shall focus on this in a later tutorial. For now, we will focus on teaching you how to build rather than design**.**

5.1 THE HISTORY OF HTML

HTML was first created by Tim Berners-Lee, Robert Cailliau, and others starting in **1989**. It stands for Hyper Text Markup Language.Hypertext means that the document contains links that allow the reader to jump to other places in the document or to another document altogether. The latest version is known as HTML 5.

A Markup Language is a way that computers speak to each other to control how text is processed and presented. To do this HTML uses two things: tags and attributes**.**

What are Tags and Attributes?

Tags and attributes are the basis of HTML.

They work together but perform different functions – it is worth investing 2 minutes in differentiating the two**.**

5.2 TAGS

Tags are used to mark up the start of an HTML element and they are usually enclosed in angle brackets. An example of a tag is: `<h1>`.

Most tags must be opened `<h1>` and closed `</h1>` in order to function.

What are HTML Attributes?

Attributes contain additional pieces of information. Attributes take the form of an opening tag and additional info is placed inside.

5.3 EDITORS

Do not use Microsoft Word or any other word processor when writing HTML code, only an HTML editor or at the very least, your machine's built-in notepad, is suitable for the task.

Secondly, ensure that you've installed a number of different browsers such as Chrome and Firefox in order to preview your upcoming creation.

Creating Your First HTML Webpage

First off, you need to open your HTML editor, where you will find a clean white page on which to write your code.

From there you need to layout your page with the following tags.

5.4 STRUCTURE OF HTML PAGE

These tags should be placed underneath each other at the top of every HTML page that you create.

<!DOCTYPE html> — This tag specifies the language you will write on the page. In this case, the language is HTML 5.

<html> — This tag signals that from here on we are going to write in HTML code.

<head> — This is where all the metadata for the page goes — stuff mostly meant for search engines and other computer programs.

<body> — This is where the content of the page goes.

This is how your average HTML page is structured visually.

Fig5.1 HTML page structure

Adding Content

Next, we will make <body> tag.

The HTML <body> is where we add the content which is designed for viewing by human eyes.

This includes text, images, tables, forms and everything else that we see on the internet each day.

Add HTML Headings To Your Web Page

In HTML, headings are written in the following elements:

- <h1>
- <h2>

- <h3>
- <h4>
- <h5>
- <h6>

As you might have guessed <h1> and <h2> should be used for the most important titles, while the remaining tags should be used for sub-headings and less important text.

Search engine bots use this order when deciphering which information is most important on a page.

Creating Your Heading

Let's try it out. On a new line in the HTML editor, type:

<h1>Welcome to My Page</h1>

And hit save. We will save this file as "index.html" in a new folder called "my webpage."

Well let's not get carried away; we've still got loads of great features that we can add to your page.

5.5 TEXT IN HTML

Adding text to our HTML page is simple using an element opened with the tag <p> which creates a new paragraph**.** We place all of our regular text inside the element <p>.

When we write text in HTML, we also have a number of other elements we can use to control the text or make it appear in a certain way. Some are shown in Table 5.1.

ELEMENT	MEANING
<b>	Bold
	Strong
<i>	Italic
	Emphasised Text
<mark>	Marked Text
<small>	Small Text
<strike>	Striked Out Text
<u>	Underlined Text

Table 5.1 Tags to control text type

CHAPTER 6

CASCADING STYLE SHEETS

CSS (Cascading Style Sheets) allows you to create great-looking web pages, but how does it work under the hood? This article explains what CSS is with a simple syntax example and also covers some key terms about the language.

As we have mentioned before, CSS is a language for specifying how documents are presented to users, how they are styled, laid out, etc.

A document is usually a text file structured using a markup language — HTML is the most common markup language, but you may also come across other markup languages such as SVG or XML.

Presenting a document to a user means converting it into a form usable by your audience. Browsers, like Firefox, Chrome, or Edge, are designed to present documents visually, for example, on a computer screen, projector, or printer.

6.1 CSS SYNTAX

CSS is a rule-based language ,you define the rules by specifying groups of styles that should be applied to particular elements or groups of elements on your web page.

For example, you can decide to have the main heading on your page to be shown as large red text. The following code shows a very simple CSS rule that would achieve the styling described above:

```
h1 {
color: red;
font-size: 5em;
}
```

CSS properties have different allowable values, depending on which property is being specified. In our example, we have the color property, which can take various color values. We also have the font-size property. This property can take various size units as a value.

A CSS stylesheet will contain many such rules, written one after the other.

```
h1 {
color: red;
font-size: 5em;
}

p {
color: black;
}
```

6.2 CSS MODULES

As there are so many things that you could style using CSS, the language is broken down into modules. You'll see reference to these modules as you explore MDN. Many of the documentation pages are organized around a particular module. For example, you could take a look at the MDN reference to the Backgrounds and Borders module to find out what its purpose is and the properties and features it contains. In that module, you will also find a link to Specifications that defines the technology (also see the section below).

At this stage, you don't need to worry too much about how CSS is structured; however, it can make it easier to find information if, for example, you are aware that a certain property is likely to be found among other similar things, and is therefore, probably in the same specification.

For a specific example, let's go back to the Backgrounds and Borders module — you might think that it makes logical sense for the background-color and border-color properties to be defined in this module. And you'd be right.

6.3 CSS SPECIFICATIONS

All web standards technologies (HTML, CSS, JavaScript, etc.) are defined in giant documents called specifications ,which are published by standards organizations (such as the W3C, WHATWG, ECMA, or Khronos) and define precisely how those technologies are supposed to behave.

CSS is no different ,it is developed by a group within the W3C called the CSS Working Group. This group is made of representatives of browser vendors and other companies who have an interest in CSS. There are also other people, known as invited experts, who act as independent voices; they are not linked to a member organization.

New CSS features are developed or specified by the CSS Working Group sometimes because a particular browser is interested in having some capability, other times because web designers and developers are asking for a feature, and sometimes because the Working Group itself has identified a requirement. CSS is constantly developing, with new features becoming available. However, a key thing about CSS is that everyone works very hard to never change things in a way that would break old websites. A website built in 2000, using the limited CSS available then, should still be usable in a browser today!

As a newcomer to CSS, it is likely that you will find the CSS specs overwhelming — they are intended for engineers to use to implement support for the features in user agents, not for web developers to read to understand CSS. Many experienced developers would much rather refer to MDN documentation or other tutorials. Nevertheless, it is worth knowing that these specs exist and

understanding the relationship between the CSS you are using, the browser support (see below), and the specs.

6.4 BROWSER SUPPORT INFORMATION

After a CSS feature has been specified, then it is only useful for us in developing web pages if one or more browsers have implemented the feature. This means that the code has been written to turn the instruction in our CSS file into something that can be output to the screen. We'll look at this process more in the lesson How CSS works. It is unusual for all browsers to implement a feature at the same time, and so there is usually a gap where you can use some part of CSS in some browsers and not in others. For this reason, being able to check implementation status is useful.

The browser support status is shown on every MDN CSS property page in a table named "Browser compatibility". Consult the information in that table to check if the property can be used on your website. For an example, see the browser compatibility table for the CSS font-family property.

Based on your requirements, you can use the browser compatibility table to check how this property is supported across various browsers, or check if your specific browser and the version you have support the property, or if there are any caveats you should be aware of for the browser and version you are using.

CHAPTER 7

JAVA DATA BASE CONNECTIVITY

JDBC or Java Database Connectivity is a Java API to connect and execute the query with the database. It is a specification from Sun microsystems that provides a standard abstraction(API or Protocol) for java applications to communicate with various databases. It provides the language with java database connectivity standards. It is used to write programs required to access databases. JDBC, along with the database driver, can access databases and spreadsheets. The enterprise data stored in a relational database(RDB) can be accessed with the help of JDBC APIs.

Definition of JDBC(Java Database Connectivity)

JDBC is an API(Application programming interface) used in java programming to interact with databases. The classes and interfaces of JDBC allow the application to send requests made by users to the specified database.

7.1 PURPOSE OF JDBC

Enterprise applications created using the JAVA EE technology need to interact with databases to store application-specific information. So, interacting with a database requires efficient database connectivity, which can be achieved by using the ODBC(Open database connectivity) driver. This driver is used with JDBC to interact or communicate with various kinds of databases such as Oracle, MS Access, Mysql, and SQL server database.

7.2 COMPONENTS OF JDBC

There are generally four main components of JDBC through which it can interact with a database. They are as mentioned below:

1. JDBC API: It provides various methods and interfaces for easy communication with the database. It provides two packages as follows, which contain the java SE and Java EE platforms to exhibit WORA(write once run everywhere) capabilities.
2. java.sql.*;
3. It also provides a standard to connect a database to a client application.
4. JDBC Driver manager: It loads a database-specific driver in an application to establish a connection with a database. It is used to make a database-specific call to the database to process the user request.
5. JDBC Test suite: It is used to test the operation(such as insertion, deletion, updation) being performed by JDBC Drivers.

6. JDBC-ODBC Bridge Drivers**:** It connects database drivers to the database. This bridge translates the JDBC method call to the ODBC function call. It makes use of the sun.jdbc.odbc package which includes a native library to access ODBC characteristics.

Description:

1. Application: It is a java applet or a servlet that communicates with a data source.
2. The JDBC API: The JDBC API allows Java programs to execute SQL statements and retrieve results. Some of the important classes and interfaces defined in JDBC API are as follows:
3. DriverManager: It plays an important role in the JDBC architecture. It uses some database-specific drivers to effectively connect enterprise applications to databases.
4. JDBC drivers: To communicate with a data source through JDBC, you need a JDBC driver that intelligently communicates with the respective data source.

7.3 JDBC DRIVERS

JDBC drivers are client-side adapters (installed on the client machine, not on the server) that convert requests from Java programs to a protocol that the DBMS can understand. There are 4 types of JDBC drivers:

1. Type-1 driver or JDBC-ODBC bridge driver
2. Type-2 driver or Native-API driver
3. Type-3 driver or Network Protocol driver
4. Type-4 driver or Thin driver

Types of JDBC Architecture(2-tier and 3-tier)

The JDBC architecture consists of two-tier and three-tier processing models to access a database. They are as described below:

1. Two-tier model: A java application communicates directly to the data source. The JDBC driver enables the communication between the application and the data source. When a user sends a query to the data source, the answers for those queries are sent back to the user in the form of results.The data source can be located on a different machine on a network to which a user is connected. This is known as a client/server configuration, where the user's machine acts as a client, and the Machin has the data source running acts as the server.

2. Three-tier model: In this, the user's queries are sent to middle-tier services, from which the commands are again sent to the data source. The results are sent back to the middle tier, and

from there to the user. This type of model is found very useful by management information system directors.

7.4 INTERFACES OF JDBC API

A list of popular interfaces of JDBC API is given below:

- Driver interface
- Connection interface
- Statement interface
- PreparedStatement interface
- CallableStatement interface
- ResultSet interface
- ResultSetMetaData interface
- DatabaseMetaData interface
- RowSet interface

7.5 CLASSES OF JDBC API

A list of popular classes of JDBC API is given below:

- DriverManager class
- Blob class
- Clob class
- Types class

Java application that needs to communicate with the database has to be programmed using JDBC API. JDBC Driver supporting data sources such as Oracle and SQL server has to be added in java application for JDBC support which can be done dynamically at run time. This JDBC driver intelligently communicates the respective data source.

7.6 SAMPLE APPLICATION

```
package com.deepak.jdbc;
import java.sql.*;
public class JDBCDemo {
public static void main(String args[])
throws SQLException, ClassNotFoundException
{
String driverClassName
= "sun.jdbc.odbc.JdbcOdbcDriver";
String url = "jdbc:odbc:XE";
String username = "scott";
```

```
String password = "tiger";
String query
= "insert into students values(109, 'bhatt')";

// Load driver class
Class.forName(driverClassName);

// Obtain a connection
Connection con = DriverManager.getConnection(
url, username, password);

// Obtain a statement
Statement st = con.createStatement();

// Execute the query
int count = st.executeUpdate(query);
System.out.println(
"number of rows affected by this query= "
+ count);

// Closing the connection as per the
// requirement with connection is completed
con.close();
}
} // class
```

The above example demonstrates the basic steps to access a database using JDBC. The application uses the JDBC-ODBC bridge driver to connect to the database. You must import java.sql package to provide basic SQL functionality and use the classes of the package.

CHAPTER 8

MYSQL

8.1 DATABASE

MySQL is the world's most popular open source database. According to DB-Engines, MySQL ranks as the second-most-popular database, behind Oracle Database. MySQL powers many of the most accessed applications, including Facebook, Twitter, Netflix, Uber, Airbnb, Shopify, and Booking.com. Since MySQL is open source, it includes numerous features developed in close cooperation with users over more than 25 years. So it's very likely that your favorite application or programming language is supported by MySQL Database. MySQL is a relational database management system.

Databases are the essential data repository for all software applications. For example, whenever someone conducts a web search, logs in to an account, or completes a transaction, a database system is storing the information so it can be accessed in the future.

A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structure is organized into physical files optimized for speed. The logical data model, with objects such as data tables, views, rows, and columns, offers a flexible programming environment. You set up rules governing the relationships between different data fields, such as one to one, one to many, unique, required, or optional, and "pointers" between different tables. The database enforces these rules so that with a well-designed database your application never sees data that's inconsistent, duplicated, orphaned, out of date, or missing.

The "SQL" part of "MySQL" stands for "Structured Query Language." SQL is the most common standardized language used to access databases. Depending on your programming environment, you might enter SQL directly (for example, to generate reports), embed SQL statements into code written in another language, or use a language-specific API that hides the SQL syntax.

8.2 MYSQL OPEN SOURCE

Open source means it's possible for anyone to use and modify the software. Anybody can download MySQL software from the internet and use it without paying for it. You can also change its source code to suit your needs. MySQL software uses the GNU General Public License (GPL) to define what you may and may not do with the software in different situations.

If you feel uncomfortable with the GNU GPL or need to embed MySQL code into a commercial application, you can buy a commercially licensed version from Oracle. See the MySQL Licensing Information section for more information.

MySQL: the #1 choice for developers

MySQL consistently ranks as the most popular database for developers, according to surveys from Stack Overflow and JetBrains. Developers love its high performance, reliability, and ease of use.

MySQL supports the following popular development languages and drivers:

- PHP
- Python
- C
- Perl
- Ruby

MySQL works in client/server or embedded systems

MySQL Database is a client/server system that consists of a multithreaded SQL server that supports different back ends, several different client programs and libraries, administrative tools, and a wide range of application-programming interfaces (APIs). We also provide MySQL as an embedded multithreaded library that you can link into your application to get a smaller, faster, easier-to-manage standalone product.

MySQL benefits MySQL is fast, reliable, scalable, and easy to use. It was originally developed to handle large databases quickly and has been used in highly demanding production environments for many years. Although MySQL is under constant development, it offers a rich and useful set of functions. MySQL's connectivity, speed, and security make it highly suited for accessing databases on the internet.

MySQL's key benefits include

Ease of use: Developers can install MySQL in minutes, and the database is easy to manage.

Reliability: MySQL is one of the most mature and widely used databases. It has been tested in a wide variety of scenarios for more than 25 years, including by many of the world's largest companies. Organizations depend on MySQL to run business-critical applications because of its reliability.

Scalability: MySQL scales to meet the demands of the most accessed applications. MySQL's native replication architecture enables organizations such as Facebook to scale applications to support billions of users.

Performance: MySQL HeatWave is faster and less expensive than other database services, as demonstrated by multiple standard industry benchmarks, including TPC-H, TPC-DS, and CH-benCHmark.

High availability: MySQL delivers a complete set of native, fully integrated replication technologies for high availability and disaster recovery. For business-critical applications, and to meet service-level agreement commitments, customers can achieve

- Recovery point objective = 0 (zero data loss)
- Recovery time objective = seconds (automatic failover)

Security: Data security entails protection and compliance with industry and government regulations, including the European Union General Data Protection Regulation, the Payment Card Industry Data Security Standard, the Health Insurance Portability and Accountability Act, and the Defense Information Systems Agency's Security Technical Implementation Guides. MySQL Enterprise Edition provides advanced security features, including authentication/authorization, transparent data encryption, auditing, data masking, and a database firewall.

Flexibility: The MySQL Document Store gives users maximum flexibility in developing traditional SQL and NoSQL schema-free database applications. Developers can mix and match relational data and JSON documents in the same database and application.

8.3 MYSQL USE CASES

Cloud applications**:** MySQL is very popular in the cloud. MySQL HeatWave is a fully managed database service, powered by the integrated HeatWave in-memory query accelerator. It's the only cloud database service that combines transactions, analytics, and machine learning (ML) services into one MySQL Database, delivering real-time, secure analytics without the complexity, latency, and cost of ETL duplication. MySQL HeatWave is 6.5X faster than Amazon Redshift at half the cost, 7X faster than Snowflake at one-fifth the cost, and 1,400X faster than Amazon Aurora at half the cost. With MySQL HeatWave ML, developers and data analysts can build, train, and explain machine learning models in a fully automated way—25X faster than Amazon Redshift ML at 1% of the cost.

REFERENCE

[1] Kurniawan, B. (2002). Java for the Web with Servlets, JSP, and EJB. Sams Publishing.

[2] Hunter, J., & Crawford, W. (2001). Java servlet programming: Help for server side Java developers. " O'Reilly Media, Inc.".

[3] Brown, S., Dalton, S., & Raible, M. (2003). Professional jsp. APress LP.

[4] Murach, J., & Urban, M. (2014). Murach's Java Servlets and JSP. Mike Murach & Associates.

[5] Yourdon, E. (1996). Java, the Web, and software development. Computer, 29(8), 25-30.

[6] Lei, K., Ma, Y., & Tan, Z. (2014, December). Performance comparison and evaluation of web development technologies in php, python, and node. js. In 2014 IEEE 17th international conference on computational science and engineering (pp. 661-668). IEEE.

[7] Johnson, R., Hoeller, J., Arendsen, A., & Thomas, R. (2009). Professional Java development with the Spring framework. John Wiley & Sons.

[8] Arthur, J., & Azadegan, S. (2005, May). Spring Framework for rapid open source J2EE Web Application Development: A case study. In Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Network (pp. 90-95). IEEE.

[9] Prajapati, H. B., & Dabhi, V. K. (2009, March). High quality web-application development on java EE platform. In 2009 IEEE International Advance Computing Conference (pp. 1664-1669). IEEE.

[10] Shan, T. C., & Hua, W. W. (2006, October). Taxonomy of java web application frameworks. In 2006 IEEE International Conference on e-Business Engineering (ICEBE'06) (pp. 378-385). IEEE.

APPENDIX

1)BASIC JAVA PROGRAM:

```
public class HelloJava {

public static void main(String[] args) {

System.out.println("Hello JavaTpoint");

}

}
```

2)Class & Object:

```
Class car
{
int car_id;
char colour[4];
float engine_no;
double distance;

void distance_travelled();
float petrol_used();
char music_player();
void display();
}
#include <iostream>
using namespace std;
class car {
public:
int car_id;
double distance;
void distance_travelled();
void display(int a, int b)
```

```
{
cout << "car id is=\t" << a << "\ndistance travelled =\t" << b+5;
}
};

int main()
{
car c1; // Declare c1 of type car
c1.car_id = 321;
c1.distance = 12;
c1.display(321, 12);

return 0;
}
```

3)Java String toUpperCase() and toLowerCase() method

```
public class Stringoperation1
{
public static void main(String ar[])
{
String s="Sachin";
System.out.println(s.toUpperCase());//SACHIN
System.out.println(s.toLowerCase());//sachin
System.out.println(s);//Sachin(no change in original)
}
}
```

4)Java String length() Method

```
public class Stringoperation5
{
public static void main(String ar[])
{
```

```
String s="Sachin";
System.out.println(s.length());//6
}
}
```

5)Java String trim() method

```
public class Stringoperation2
{
public static void main(String ar[])
{
String s="  Sachin  ";
System.out.println(s);//  Sachin
System.out.println(s.trim());//Sachin
}
}
```

6)Java String startsWith() and endsWith() method

```
public class Stringoperation3
{
public static void main(String ar[])
{
String s="Sachin";
System.out.println(s.startsWith("Sa"));//true
System.out.println(s.endsWith("n"));//true
}
}
```

7)Java String charAt() Method

```
public class Stringoperation4
{
public static void main(String ar[])
{
String s="Sachin";
System.out.println(s.charAt(0));//S
System.out.println(s.charAt(3));//h
```

```
}
}
```

8)CSS

```
h1 {
color: red;
font-size: 5em;
}

p {
color: black;
}
```

9)SIMPLE JDBC EXAMPLE

```
import java.sql.*;

public class JDBCDemo {

public static void main(String args[])
throws SQLException, ClassNotFoundException
{
String driverClassName
= "sun.jdbc.odbc.JdbcOdbcDriver";
String url = "jdbc:odbc:XE";
String username = "scott";
String password = "tiger";
String query
= "insert into students values(109, 'bhatt')";

// Load driver class
Class.forName(driverClassName);

// Obtain a connection
Connection con = DriverManager.getConnection(
```

```
url, username, password);

// Obtain a statement
Statement st = con.createStatement();

// Execute the query
int count = st.executeUpdate(query);
System.out.println(
"number of rows affected by this query= "
+ count);

// Closing the connection as per the
// requirement with connection is completed
con.close();
}
} // class
```