VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANA SANGAMA, BELAGAVI

An Internship Report

On

"FULL STACK JAVA"

Submitted in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

In

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Submitted By

JAFAR SADIQ Z

(4GM22CS400)

Internship Carried Out

At

'Robowaves ,Banglore'

Internal Guide

Dr. B N Veerappa

Professor & Head

Department of CS&E

External Guide/mentor

Mr. Shankar Narayan

Senior Java Developer

Jspiders Rajajinagar

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GM INSTITUTE OF TECHNOLOGY, DAVANGERE

(Affiliated to VTU, Belagavi, Approved by AICTE -New Delhi & Govt. of Karnataka)

(Accredited by NBA New Delhi, Valid upto 30.06.2025)

2024-2025

Srishyla Educational Trust (R), Bheemasamudra

GM INSTITUTE OF TECHNOLOGY, DAVANGERE

(Affiliated to VTU, Belagavi, Approved by AICTE -New Delhi & Govt. of Karnataka)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(Accredited by NBA New Delhi, Valid upto 30.06.2025)

CERTIFICATE

Certified that the 8th Semester Internship titled **"Full Stack Java"** is a bonafide work carried out by **Jafar Sadiq Z (4GM22CS400),** in partial fulfillment for the award of degree of Bachelor of Engineering in Department of Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi, during the year 2024-25. The internship report has been approved as it satisfies the academic requirements with respect to the Internship work prescribed for Bachelor of Engineering Degree.

Guide	Internship Coordinator	Head of the Department
20-05-25	Kotreshi S.N 20/05/25	20-05-25
Dr. B N Veerappa	**Mr. Kotreshi S N**	**Dr. B N Veerappa**
B.E,M.Tech, ph.D	B.E, M.Tech	B.E,M.Tech,Ph.D
Professor & Head	Asst.Professor	Professor & Head

External Viva

Name of the Examiners	Signature with Date
1. Nagaraj. K	Nagaraj. K 27/5/25
2. Dr. Gangadhar. S	Ganga 27/5/25

Robowaves

CIN: U72200KA2007PTC044701 Date: 15/05/2025

TO WHOM SO EVER IT MAY CONCERN

This is to certify that **Mr/Ms. Jafar Sadiq Z** bearing **USN – "4GM22CS400"** a student of **"Bachelor Of Engineering"** from **Visvesvaraya Technological University,** had worked as intern and completed his/her internship in our company and worked on **"Java Full Stack Development"** from **17/02/2025 to 15/05/2025**. Successfully completed internship under the guidance of **Shankar Narayan (Robowaves (A Unit of Test Yantra Software Solutions (India) Pvt. Ltd.)**

During this period of this internship program with us we found he/she is sincere and hardworking.

We wish success for all his/her future endeavours.

Yours Sincerely,

For Robowaves
(A Unit of Test Yantra Software Solutions (India) Pvt. Ltd.)
"System Generated letter no need of Signature"

JSPIDERS & PYSPIDER
Development training institute
it Of Test Yantra Software Solutions Pvt. Ltd.
A-83, 5th Main, Industrial Estate
Rajajinagar 6th Block, Bengaluru-560 02

ACKNOWLEDGEMENT

The joy and satisfaction that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible.

I would like to express our gratitude to our Principal**, Dr. SANJAY PANDE M B** for providing us a congenial environment for engineering studies and also for having showed us the way to carry out the Internship work.

I consider it a privilege and honour to express our sincere thanks to**, Dr. B N VEERAPPA** Professor and Head, Department of Computer Science and Engineering for his support and invaluable guidance throughout the tenure of this Internship work.

I would like to thank our Guide **Dr. B N VEERAPPA** Professor and Head, Department of Computer Science and Engineering for support, guidance, motivation, encouragement for the successful completion of this Internship work.

I would like to thank our mentor **Mr. SHANKAR NARAYAN** Senior Java Developer , Robowaves for providing resources, sharing knowledge and providing industry working environment.

I intend to thank all the teaching and non-teaching staffs of our Department of Computer Science and Engineering for their immense help and co-operation.

Finally, I would like to express our gratitude to our parents and friends who always stood by us.

JAFAR SADIQ Z

4GM22CS400

CONTENTS

Chapter 1

Company Profile

1.1 About the Organization

Robowaves is a premier technology training institute based in Bangalore, Karnataka. The organization specializes in providing comprehensive training programs in various cutting-edge technologies such as, Data Science, Internet of Things (IoT), Devops, Cybersecurity, and Full Stack Web Development. Founded by experienced technocrats, the institute aims to bridge the gap between advanced technology and real-world applications through hands-on learning and practical experience.

1.2 About the Founder

Girish Ramanna is the CEO and Managing Director of "Robowaves" India's largest Testing Services and Training Organization. With his exemplary leadership and entrepreneurial skills he has led Test Yantra from a humble beginning of Training center to an accomplished Testing Services and Training organization with its presence across globe.

1.3 Organization Structure

The organizational structure of Robowaves includes a team of dedicated trainers and experts in various fields. The Training institute is managed by a core group of founders and supported by a diverse team of professionals who handle different aspects of training and development. The structure is designed to facilitate efficient management and delivery of training programs.

1.4 Services Offered By the Company

Robowaves offers a variety of services including:

- **Academic Training:** Comprehensive courses in Full Stack Java, Data Science,Cybersecurity, Full Stack Web Development, and Devops.
- **Corporate Training**: Experiential-based training programs designed for executives, employees, and budding engineers.

- **Consultancy Services:** Software consultancy, development, and outsourcing services.

1.5 Working Process of the Company

The working process of Robowaves involves a blend of theoretical and practical learning. The institute emphasizes hands-on experience through real-life projects and the use of diverse hardware and software tools. Training programs are designed to be interactive and immersive, allowing participants to apply their knowledge in practical scenarios.

1.6 Design Capabilities

Robowaves, through QSpiders and JSpiders, offers state-of-the-art training facilities with modern classrooms and advanced computing infrastructure. QSpiders focuses on software testing with expert trainers and ISTQB® standards. JSpiders leads in Java/J2EE training, providing hands-on development experience. Both units emphasize real-time project learning to build industry-relevant skills. Together, they train over 5000 students monthly, bridging the gap between academics and IT industry needs.

1.7 People Working in the Organization

Robowaves employs a team of experienced trainers and professionals with expertise in various technical fields. Some of the key personnel include:

- Shankar Narayan, B. E., M.Tech: Expert in Java, Advance Java & Devops.
- Shashank L , B. E., M.Tech: Expert in Full Stack Web Development, SQL, ReactJS.
- Kiran Kumar H S, B. E., M.Tech: Expert in SQl, Advance Java.
- Dharma Dhurai, B. E., M.Tech: Expert in SQL, MangoDB .
- Laxman Handevaar, B. E: Expert in Web Technologies, Devops, Node.js, express.js.
- Siddesh S B : Expert in Java Programming Logics, DSA.
- Mr. Shridhar : Expert in Java, JDBC, Hibernates.

Overall, RoboWaves is committed to providing high-quality training and consultancy services, leveraging its extensive expertise

Chapter 2

INTRODUCTION

Full Stack Java development encompasses the creation of complete web applications by working on both the front-end and back-end components using Java and related technologies. Developers skilled in Full Stack Java are proficient in building interactive user interfaces with technologies like HTML, CSS, and JavaScript frameworks, while also managing the server-side logic using Java frameworks such as Spring and Hibernate. This dual expertise allows them to handle everything from designing the user experience to implementing business logic, integrating databases, and ensuring secure and efficient communication between client and server. Full Stack Java developers play a crucial role in delivering scalable, maintainable, and high-performance applications that meet modern business needs. Due to the versatility and robustness of Java in enterprise environments, full stack Java skills are in high demand, enabling professionals to contribute across the entire software development lifecycle and bridge the gap between front-end design and back-end functionality.

2.1 Java

Java is a powerful, versatile, and widely-used programming language known for its portability, reliability, and strong security features. It follows the “write once, run anywhere” philosophy, allowing code to run on any platform with a compatible Java Virtual Machine (JVM). Java is extensively used in enterprise applications, web development, mobile apps, and large-scale systems due to its robust object-oriented principles, extensive libraries, and strong community support. Its stability and scalability make it a preferred choice for building complex and high-performance software solutions.

2.2 SQL

SQL (Structured Query Language) is a standardized programming language used to manage and manipulate relational databases. It allows users to create, read, update, and delete data efficiently through queries. SQL is essential for handling large volumes of data and supports operations like data retrieval, filtering, sorting, and aggregation. Widely used across industries, SQL enables developers and data professionals to organize data, enforce data integrity, and perform complex transactions. Its integration with various database management

systems, such as MySQL, PostgreSQL, and Oracle, makes it a foundational skill for backend development, data analysis, and application programming.

2.3 Web

Web development refers to the process of creating websites and web applications that run on the internet or intranet. It involves building the front-end, which includes the visual layout and user interface using technologies like HTML, CSS, and JavaScript, as well as the back-end, which manages server-side logic, databases, and application functionality. Web development enables businesses and individuals to present content, provide services, and interact with users online. With the rise of dynamic, responsive, and interactive web applications, web development has become a critical skill in the digital age, combining creativity and technical expertise to deliver seamless user experiences across devices and platforms.

Java powers the back-end by providing a robust, scalable, and secure platform for building server-side applications. SQL manages and manipulates the database, ensuring efficient data storage and retrieval essential for dynamic applications. Web development creates the front-end user interface using HTML, CSS, and JavaScript, enabling interactive and responsive user experiences. Together, these technologies allow full stack developers to build complete, end-to-end web applications. Mastery of all three is crucial for delivering seamless, efficient, and user-friendly software solutions.

Chapter 3

TASK PERFORMED

Week 1: Fundamentals of Web Development

Day 1-6: HTML & CSS Basics
(24-Feb-2025 to 29-Feb-2025)

HTML:

- HTML is an acronym which stands for Hyper Text Markup Language which is used for creating web pages and web applications. Let's see what is meant by Hypertext Markup Language, and Web page.

- Hyper Text: HyperText simply means "Text within Text." A text has a link within it, is a hypertext. Whenever you click on a link which brings you to a new webpage, you have clicked on a hypertext. HyperText is a way to link two or more web pages (HTML documents) with each other.

- Markup language: A markup language is a computer language that is used to apply layout and formatting conventions to a text document. Markup language makes text more interactive and dynamic. It can turn text into images, tables, links, etc.

- Web Page: A web page is a document which is commonly written in HTML and translated by a web browser. A web page can be identified by entering an URL. A Web page can be of the static or dynamic type. With the help of HTML only, we can create static web pages.

- Hence, HTML is a markup language which is used for creating attractive web pages with the help of styling, and which looks in a nice format on a web browser. An HTML document is made of many HTML tags and each HTML tag contains different content.

- Elements and Tags : HTML uses a variety of elements (e.g., „<p>“,“<h1>“,“<a>“) and tags (e.g., <tagname>) to define different types of content, such as headings, paragraphs, links, images, and lists.

- Document Structure : An HTML document is structured as a tree, with the <html> element at the root. It includes the <head> section for metadata (title, links to stylesheets or scripts) and the <body> section for the visible content of the web page.

• Attributes : HTML elements often have attributes (e.g., href, src, class) that provide additional information about the element or modify its behavior.

• Hyperlinks : HTML allows you to create hyperlinks using the <a> element, enabling users to navigate between web pages.

• Forms : HTML provides form elements (e.g., <form>, <input>, <button>) for collecting user input and sending data to a server.

• Basic Structure: <!DOCTYPE html>

```
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Internship Report</title>
<link rel="stylesheet" href="styles.css"> <!-- Optional CSS -->
</head>
<body>
    <!-- Content Goes Here -->
</body>
</html>
```

HTML Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Simple Page</title>
</head>
<body>
<h1>Welcome!</h1>
<p>This is a test paragraph.</p>
</body>
</html>
```

Week 2: CSS Basics

(1-Mar-2025 to 8-Mar-2025)

CSS

- Selectors and Properties: CSS uses selectors to target HTML elements and apply styling rules. Properties (e.g., color, font-size, margin) specify how an element should be styled.
- External and Internal Styles: CSS can be applied externally via linked stylesheets or internally within an HTML document using the <style> element.
- Cascading Style Sheets: CSS rules can cascade, meaning styles from different sources (external stylesheet, internal style block, inline styles) interact to determine the final appearance of an element.
- Layout and Positioning: CSS controls the layout and positioning of elements on a web page, including techniques for responsive designs and adapting to various screen sizes.
- Box Model: Elements are treated as boxes consisting of content, padding, borders, and margins. Understanding this model is essential for layout design.
- Flexbox and Grid Layout: Advanced CSS tools like Flexbox and Grid simplify the design of complex and flexible layouts.
- Media Queries: Media queries allow applying different styles based on device characteristics such as screen width and orientation.
- Transitions and Animations: CSS enables smooth transitions and animations to enhance user experience by adding dynamic visual effects and interactivity.

CSS Example:

```
body {
font-family: Arial;
    }

 .container
    {
    display:
    flex;
    }
```

Week 3: Javascript

(10-Mar-2025 to 11-Mar-2025)

JavaScript (js) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages.

- It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document.
- It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser. Since then, it has been adopted by all other graphical web browsers. With JavaScript, users can build modern web applications to interact directly without reloading the page every time. The traditional website uses js to provide several forms of interactivity and simplicity.
- Although, JavaScript has no connectivity with Java programming language. The name was suggested and provided in the times when Java was gaining popularity in the market. In addition to web browsers, databases such as CouchDB and MongoDB uses JavaScript as their scripting and query language.

• Application of JavaScript:

1. JavaScript is used to create interactive websites. It is mainly used for: Client-side validation, Dynamic drop-down menus, Displaying date and time,
2. Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box), Displaying clocks etc.
3. Client-Side Scripting: JavaScript is primarily a client-side scripting language, meaning it runs directly in the web browser of the user. This allows developers to control how web pages behave and respond to user actions without the need for frequent server requests.

Week 4: Introduction to SQL & Databases

(9 Mar-2025 to 13 -Mar-2025)

- **What is a Database?**

  A database is an organized collection of data that can be easily accessed, managed, and updated.

- **What is SQL?**

  Structured Query Language is the standard language used to communicate with relational Databases.

- **Uses of SQL:**
  - Creating and managing databases
  - Inserting, updating, deleting data
- **SQL is Used In:**
  - Web applications
  - Data analysis
  - Enterprise software systems

Week 4: DBMS vs RDBMS

(15 -Mar-2025 to 21 -Mar-2025)

- **DBMS (Database Management System):**

  Software that manages databases; does not support relations between tables.

  Ex: XML, File systems

- **RDBMS (Relational DBMS):**

  Manages data in tables with rows and columns. Supports relationships.

  Ex: MySQL, PostgreSQL, Oracle

Feature	DBMS	RDBMS
Data Storage	Files	Tables
Relationships	Not Supported	Supported
Normalization	No	Yes
Examples	XML, File System	MySQL, PostgreSQL, Oracle

Week 5: History of SQL & Relational Model

(23 -Apr-2025 to 28 -Apr-2025)

- Developed in 1970s at IBM
- Originally named SEQUEL by Chamberlin & Boyce
- Later renamed to SQL
- Adopted by ANSI and ISO as the standard in late 1980s
- Relational Model:
    - Based on tables (relations)
    - Each row = tuple, each column = attribute

Week 6: E.F. Codd's 12 Rules & SQL Data Types

(15-Apr-2025 to 21-Apr-2025)

- Constraints ensure data accuracy and integrity
- Types of Constraints:
    - NOT NULL: Value must be present
    - UNIQUE: No duplicate values
    - PRIMARY KEY: Unique + Not Null
    - FOREIGN KEY: References another table's primary key
    - CHECK: Validates based on conditions
    - DEFAULT: Sets default value if not provided

Example:

```
CREATE TABLE Users (
    ID INT PRIMARY KEY,
    Email VARCHAR(100) UNIQUE,
```

```
    Age INT CHECK (Age > 0),

    Status VARCHAR(10) DEFAULT 'active'

);
```

Week 7: DDL (Data Definition Language)

(22-Apr-2025 to 28-Apr-2025)

- Used to define and modify database schema
- Commands:
  - CREATE: Creates a new table or database
  - ALTER: Modifies structure of a table
  - DROP: Deletes a table or database

Example:

```
-- CREATE: Create a new table
CREATE TABLE Employees (
  ID INT,
  Name VARCHAR(50),
  Salary DECIMAL(10, 2)
);

-- ALTER: Add a new column to the existing table
ALTER TABLE Employees
ADD Department VARCHAR(50);

-- DROP: Delete the table
DROP TABLE Employees;
```

Week 8: Core Java Introduction and Basics

(30-Apr-2025 to 5-May-2025)

What is java?

Java is a high-level, object-oriented, platform-independent programming language designed for creating dynamic, secure, and distributed applications. It is used for developing a variety of software, from applications to games and web applets.

Key Features:

• Object-Oriented: Encourages modular, reusable code.

• Simple & Dynamic: Easy to learn and flexible in handling changes.

• Multithreading Support: Allows simultaneous task execution.

• Platform Independent: Runs on any system with a Java Virtual Machine (JVM).

• Secure & Robust: Offers security features and strong error handling.

• Internet Programming: Suitable for web-based applications.

History:

Java was created by James Gosling in 1991 at Sun Microsystems. Initially named Oak, it was later renamed Java. The language was designed to be platform-independent and to support networked applications. It was first released in 1995.

Primary Goals of Java:

1. Object-oriented programming methodology.
2. Cross-platform compatibility.
3. Built-in network support.
4. Secure execution of remote code.
5. Easy-to-use language with good features from other languages.

Java Platform:

The Java platform includes tools for developing and running Java applications:

• JVM (Java Virtual Machine): Executes Java bytecode on any system.

• JDK (Java Development Kit): Includes the compiler and libraries.

• JRE (Java Runtime Environment): Supports execution of Java programs.

Java Virtual Machine (JVM): It is central to the Java platform, executing Java bytecode programs, which are platform-independent. This bytecode is the same regardless of the underlying hardware or operating system. The JVM includes a Just-In-Time (JIT) compiler, which converts bytecode into native processor instructions a runtime, improving performance by caching the compiled code in memory. As a result, Java applications can run on any platform with a JVM and, after a brief startup time for compilation, perform nearly as fast as native Full stack java.

Class libraries: Java provides a comprehensive set of class libraries as part of its platform, offering reusable functions for common tasks like string parsing, list management, network, and file access. These libraries serve three main purposes: they simplify programming by offering standard functionality, provide an abstract interface for platform-dependent tasks (like networking and file handling), and ensure consistency across different platforms. If a 11 platform lacks certain features, the Java libraries can either emulate them or provide a way to .check their availability, ensuring Java applications run reliably on any system.

Platform independence: Java's automatic memory management, through garbage collection, eliminates the need for programmers to manually allocate and deallocate memory, reducing the risk of memory leaks and program instability. When objects are created, Java manages their lifecycle and automatically deletes objects that are no longer referenced by any part of the program. This prevents issues like memory leaks, which can occur in languages like C++ if memory is not properly freed. Although Java's garbage collection simplifies memory management, it can introduce performance overhead, and developers must consider this when designing applications, especially in environments with large heaps. The absence of pointer arithmetic in Java also ensures type safety and security, further enhancing reliability.

Performance: Java's performance has improved substantially since the early versions, and performance of JIT compilers relative to native compilers has in some tests been shown to be quite similar. The performance of the compilers does not necessarily indicate the performance of the compiled code; only careful testing can reveal the true performance issues in any system.

Java Runtime Environment (JRE): It is essential software that enables Java applications to run on a computer or device. It includes the Java Virtual Machine (JVM), libraries, and other components necessary for executing Java programs. For developers, the Java Development Kit (JDK) is a morecomprehensive package that includes the JRE along with additional tools like the Java compiler, debugger, and documentation generator. A key advantage of the JRE is its ability to handle exceptions without crashing the system. Through automated exception handling tools, Java can capture

detailed debugging information when errors occur, helping developers diagnose and fix issues efficiently, even in production environments.

Hello World Example:

```
public class HelloWorld {
public static void main(String[] args) {
System.out.println("Hello, Java!");
}
}
```

If-Else Example:

```
int age = 20;
if (age >= 18) {
System.out.println("Adult");
} else {
System.out.println("Minor");
}
```

Week 9: Core Java - Arrays, Strings, and Exception Handling

(7-Apr-2025 to Present)

"Core Java" is a term often used to refer to the fundamentals and core features of the Java programming language. It includes the basic concepts and libraries that are essential for any Java developer to understand.

```
Syntax: public class Main{
        public static void main(String[] args){
        System.out.println("Hello World");
        }
}
```

The core concepts and features of Java:

Identifiers:

- Java identifiers are names given to variables, methods, classes, and other program elements in Java programming language.
- Java identifiers must start with a letter, a currency character "$", or an underscore "_". The first character cannot be a digit.
- Java identifiers can contain letters, digits, underscores, and currency characters. The name can be of any length.
- Java is case-sensitive, which means that "name" and "Name" are two different identifiers.
- Identifiers should not be a Java keyword, which are reserved words in Java that have a specific meaning and cannot be used as an identifier.
- Examples of valid identifiers in Java are "myVariable", "_count", "MAX_VALUE", "calculateSum","MyClass".

Variables:

Variables are containers for storing data values.

- In Java, there are different types of variables, for example:

String - stores text, such as "Hello". String values are surrounded by double quotes

int - stores integers (whole numbers), without decimals, such as 123 or -123

float - stores floating point numbers, with decimals, such as 19.99 or 19.99

char - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes

boolean - stores values with two states: true or false.

All Java variables must be identified with unique names.

1. Names can contain letters, digits, underscores, and dollar signs.
2. Names must begin with a letter Names should start with a lowercase letter and it Lower.

3. cannot contain
4. whitespaces
5. Names can also begin with $ and _ .
6. Names are case sensitive ("myVar" and "myvar" are different variables).
7. Reserved words (like Java keywords, such as int or boolean) cannot be used as names.

Data Types:

- Data types are divided into two groups:

i. **Primitive data types** - includes byte, short, int, long, float, double, boolean and char.

ii. **Non-primitive data types** - such as String, Arrays and Classes.

• Primitive number types are divided into two groups:

i. Integer types stores whole numbers, positive or negative (such as 123 or -456),without decimals. Valid types are byte, short, int and long. Which type you should use, depends on the numeric value.

ii. Floating point types represents numbers with a fractional part, containing one or more decimals. There are two types: float and double

Integer Types:

i. The byte data type can store whole numbers from -128 to 127. This can be used instead of int or other integer types to save memory when you are certain that the value will be within -128.

ii. The short data type can store whole numbers from -32768 to 32767. The int data type can store whole numbers from -2147483648 to 2147483647. In general, the int data type is the referred data type when we create variables with a numeric value.

iii. The long data type can store whole numbers from - 9223372036854775808 to 9223372036854775807. This is used when int is not large enough to store the value. Note that you should end the value with an "L".

Floating Point Types:

The float and double data types can store fractional numbers. Note that you should end the value with an "f" for floats and "d" for doubles.

• A floating point number can also be a scientific number with an "e" to indicate the power of 10.

• Type casting is when you assign a value of one primitive data type to another type.

• In Java, there are two types of casting:

i. Widening Casting (automatically) - converting a smaller type to a larger type size

byte -> short -> char -> int -> long -> float -> double Example:

```
int myInt = 9;
double myDouble = myInt;
```

ii. Narrowing Casting (manually) - converting a larger type to a smaller size type

double -> float -> long -> int -> char -> short -> byte Example:

```
double myDouble = 9.78d;
int myInt = (int) myDouble;
```

Operators:

• Operators are used to perform operations on variables and values.

• Java divides the operators into the following groups:

i. Arithmetic operators

ii. Assignment operators

iii. Comparison operators

iv. Logical operators

v. Bitwise

Conditional Statements:

• Java has the following conditional statements:

i. Use if to specify a block of code to be executed, if a specified condition is true.

Syntax:

```
if (condition) {
// block of code to be executed if the condition is true
}
```

ii. Use else to specify a block of code to be executed, if the same condition is false.

Syntax:

```
} else {
if (condition) {
// block of code to be executed if the condition is true
// block of code to be executed if the condition is false
}
```

iii. Use else if to specify a new condition to test, if the first condition is false.

Syntax:

```
if (condition1) {
// block of code to be executed if condition1 is true
} else if (condition2) {
// block of code to be executed if the condition1 is false and condition2 is
   true }
else {
```

// block of code to be executed if the condition1 is false and condition2 is false

}

iv. Use switch to specify many alternative blocks of code to be executed.

Syntax:

switch(expression) {

case x: // code block

break; case y: //

code block

break; default: //

code block }

- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed.
- The break and default keywords are optional.

Loops:

Loops can execute a block of code as long as a specified condition is reached.

- Loops are handy because they save time, reduce errors, and they make code more readable.

i. **While Loop:** The while loop loops through a block of code as long as a specified condition is true.

Syntax:

while (condition) {

// code block to be executed

}

ii. **Do/While Loop:** The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax:

do {

// code block to be executed

}

while (condition);

iii. **For Loop:** When you know exactly how many times you want to loop through a block of code, use the for loop instead of a while loop.

Syntax:

for (statement 1; statement 2; statement 3) {

// code block to be executed

Arrays:

}

Statement 1 is executed (one time) before the execution of the code block.

Statement 2 defines the condition for executing the code block.

Statement 3 is executed (every time) after the code block has been executed.

Arrays:

- Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.
- To declare an array, define the variable type with square brackets:

Example: String[] cars;

• You can access an array element by referring to the index number.

Array Example:

```
int[] numbers = {1, 2, 3, 4, 5};
for (int num : numbers) {
System.out.println(num);
}
```

Multidimensional Arrays:

• A multidimensional array is an array of arrays.

• Multidimensional arrays are useful when you want to store data as a tabular form, like a table with rows and columns.

• To create a two-dimensional array, add each array within its own set of curly braces.

Example:

```
int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
System.out.println(myNumbers[1][2]);
// Outputs = 7
```

Methods:

• A method is a block of code which only runs when it is called.

• You can pass data, known as parameters, into a method.

• Methods are used to perform certain actions, and they are also known as functions.

• Why use methods? To reuse code: define the code once, and use it many times.

• A method must be declared within a class. It is defined with the name of the method, followed by parentheses ( ).

Example:

```
public class Main {
static void myMethod() {
// code to be executed
}
}
```

myMethod() is the name of the method. static means that the method belongs to the Main class and not an object of the Main class. You will learn more about objects and how to access methods through objects. void means that this method does not have a return value.

- To call a method in Java, write the method's name followed by two parentheses () and a semicolon;
- Information can be passed to methods as parameter. Parameters act as variables inside the method.
- Parameters are specified after the method name, inside the parentheses. You can add as many parameters as you want, just separate them with a comma.
- The following example has a method that takes a String called fname as parameter. When the method is called, we pass along a first name, which is used inside the method to print the full name:

```
public class
Main {
static void myMethod(String fname) {
System.out.println(fname + " Refsnes");
}
public static void main(String[] args) {
myMethod("Liam");
myMethod("Jenny");
```

```
myMethod("Anja");
        }
}
```

Return Values:

The void keyword, used in the examples above, indicates that the method should not return a value. If you want the method to return a value, you can use a primitive data type (such as int, char, etc.) instead of void, and use the return keyword inside the method:

```
public class Main {
static int myMethod(int x) {
return 5 + x;
}
public static void main(String[] args) {
        System.out.println(myMethod(3));
    }
}
```

Java OOP(Object Oriented Programing):

OOP stands for Object-Oriented Programming.

Procedural programming is about writing procedures or methods that perform operations on the data, while object-oriented programming is about creating objects that contain both data and methods.

• Object-oriented programming has several advantages over procedural programming:

a) OOP is faster and easier to execute. OOP provides a clear structure for the programs.

b) OOP helps to keep the Java code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug.

c) OOP makes it possible to create full reusable applications with less code and shorter development time**.**

Object:

• Any entity that has state and behavior is known as an object.

• For example, a chair, pen, table, keyboard, bike, etc. It can be physical or logical.

• An Object can be defined as an instance of a class.

• An object contains an address and takes up some space in memory.

• Objects can communicate without knowing the details of each other's data or code.

• The only necessary thing is the type of message accepted and the type of response returned by the objects.

Example: A dog is an object because it has states like color, name, breed, etc. as well as behaviors like wagging the tail, barking, eating, etc.

Class:

Collection of objects is called class. It is a logical entity.

• A class can also be defined as a blueprint from which you can create an individual object.

• Class doesn't consume any space.

• Inheritance:When one object acquires all the properties and behaviors of a parent object it is known as inheritance.

• It provides code reusability.

• It is used to achieve runtime polymorphism.

• Polymorphism: If one task is performed in different ways, it is known as polymorphism.

For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.

• In Java, we use method overloading and method overriding to achieve polymorphism.

• Another example can be to speak something; for example, a cat speaks meow, dog barks woof, etc.

• Abstraction:Hiding internal details and showing functionality is known as abstraction. For example phone call, we don't know the internal processing.

• In Java, we use abstract class and interface to achieve abstraction.

• Encapsulation Binding (or wrapping) code and data together into a single unit are know as encapsulation.

for example, a capsule, it is wrapped with different medicines.

• A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

Inheritance in Java:

- Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOPs (Object Oriented programming system).
- The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.
- Inheritance represents the IS-A relationship which is also known as a parent- child relationship.
- **Class:** A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.
- **Sub Class/Child Class:** Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.
- **Super Class/Parent Class:** Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.
- **Reusability:** As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.

Syntax:

```
class Subclass-name extends Superclass-name
 {
//methods and fields
 }
```

Types of Inheritance:

- Single Inheritance
- Multiple Inheritance
- Multi-Level Inheritance
- Hierarchical Inheritance
- Hybrid Inheritance

Abstraction in Java

- Abstraction is a process of hiding the implementation details and showing only functionality to the user.
- Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery. Abstraction lets you focus on what the object does instead of how it does it.

Ways to achieve Abstraction

There are two ways to achieve abstraction in java

1. Abstract class (0 to 100%)

2. Interface (100%)

Abstract class in Java

• A class which is declared as abstract is known as an abstract class. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

• An abstract class must be declared with an abstract keyword.

• It can have abstract and non-abstract methods.

• It cannot be instantiated.

• It can have constructors and static methods also.

• It can have final methods which will force the subclass not to change the body of the method.

Example:

```
abstract class Bike{
abstract void run();
}
class Honda4 extends Bike{
void run(){
System.out.println("running safely..");
}
public static void main(String args[]){
Bike obj = new Honda4();
obj.run();
```

Interface in Java:

• An interface in Java is a blueprint of a class. It has static constants and abstract methods.

• The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java.

• In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

- Java Interface also represents the IS-A relationship.
- Since Java 8, we can have default and static methods in an interface.
- Since Java 9, we can have private methods in an interface.
- There are mainly three reasons to use interface. They are given below. It is used to achieve abstraction. By interface, we can support the functionality of multiple inheritance. It can be used to achieve loose coupling.
- An interface is declared by using the interface keyword. It provides total abstraction; means all the methods in an interface are declared with the empty body, and all the fields are public, static and final by default. A class that implements an interface must implement all the methods declared in the interface.

Syntax:

```
Interface<interface_name> {
// declare constant fields
// declare methods that abstract
// by default.
}
```

Polymorphism:

- Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance.
- Like we specified in the previous chapter; Inheritance lets us inherit attributes and methods from another class. Polymorphism uses those methods to perform different tasks. This allows us to perform a single action in different ways.
- For example, think of a superclass called Animal that has a method called animalSound(). Subclasses of Animals could be Pigs, Cats, Dogs, Birds - And they also have their own implementation of an animal sound (the pig oinks, and the cat meows, etc.):

Example:

```
class Animal {
public void animalSound() {
System.out.println("The animal makes a sound");
} }
class Pig extends Animal {
public void animalSound() {
System.out.println("The pig says: wee wee");
} }
class Dog extends Animal {
public void animalSound() {
System.out.println("The dog says: bow wow");
}
}
```

Method Overloading:

- 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 behavior of the method because its name differs.
- So, we perform method overloading to figure out the program quickly.

Advantage of method overloading

Method overloading increases the readability of the program.

Different ways to overload the method There are two ways to overload the method in java

1. By changing number of arguments

2. By changing the data type

Example:

```
static int add(int a,int b)
{
return a+b;
}
static int add(int a,int b,int c){
return a+b+c;
} }
class TestOverloading1{
public static void main(String[] args){
System.out.println(Adder.add(11,11));
System.out.println(Adder.add(11,11,11));
}}
```

Method Overriding:

• 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

i. Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.

ii. 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).

Example:

```
class Vehicle{

void run(){

System.out.println("Vehicle is running");

} }

class Bike extends Vehicle{

public static void main(String args[]){

Bike obj = new Bike();

obj.run();

} }
```

Encapsulation:

• Encapsulation in Java is a process of wrapping code and data together into a single unit, for example, a capsule which is mixed of several medicines.

• We can create a fully encapsulated class in Java by making all the data members of the class private. Now we can use setter and getter methods to set and get the data in it.

• The Java Bean class is the example of a fully encapsulated class. Advantage of Encapsulation in Java.

- By providing only a setter or getter method, you can make the class read-only or write-only. In other words, you can skip the getter or setter methods.
- It provides you the control over the data. Suppose you want to set the value of id which should be greater than 100 only, you can write the logic inside the setter method. You can write the logic not to store the negative numbers in the setter methods.
- It is a way to achieve data hiding in Java because other class will not be able to access the data through the private data members.
- The encapsulate class is easy to test. So, it is better for unit testing.
- The standard IDE's are providing the facility to generate the getters and setters.
- So, it is easy and fast to create an encapsulated class in Java.

Java Package:

- A java package is a group of similar types of classes, interfaces and sub- packages.
- Package in java can be categorized in two form, built-in package and user- defined package.
- There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.
- Here, we will have the detailed learning of creating and using user-defined packages.

Advantage of Java Package:

1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.

2) Java package provides access protection.

3) Java package removes naming collision.

There are three ways to access the package from outside the package.

1. import package.*;
2. import package.classname;
3. fully qualified name.

Access Modifiers in Java:

• There are two types of modifiers in Java: access modifiers and non-access modifiers.

• The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class. We can change the access level of fields, constructors, methods, and class by applying the access modifier on it.

There are four types of Java access modifiers:

1) **Private:** The access level of a private modifier is only within the class. It cannot be accessed from outside the class.

2) **Default:** The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.

3) **Protected:** The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.

4) **Public:** The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

Exception Handling in Java:

• The Exception Handling in Java is one of the powerful mechanism to handle the runtime errors so that the normal flow of the application can be maintained. What is Exception in Java?

• In Java, an exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.

1) What is Exception Handling?

Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc. The core advantage of exception handling is to maintain the normal flow of the application. An exception normally disrupts the normal flow of the application; that is why we need to handle exceptions.

2) Types of Java Exceptions

There are mainly two types of exceptions: checked and unchecked. An error is considered as the unchecked exception. However, according to Oracle, there are three types of exceptions namely:

1. Checked Exception
2. Unchecked Exception
3. Error

- The classes that directly inherit the Throwable class except RuntimeException and Error are known as checked exceptions. For example, IOException, SQLException, etc. Checked exceptions are checked at compile-time.
- The classes that inherit the RuntimeException are known as unchecked exceptions. example, ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException, etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.
- Error is irrecoverable. Some example of errors are OutOfMemoryError, VirtualMachineError, AssertionError etc.
- Java provides five keywords that are used to handle the exception.

i. The "try" keyword is used to specify a block where we should place an exception code. It means we can't use try block alone. The try block must be followed by either catch or finally.

ii. The “catch” block is used to handle the exception. It must be preceded by try block which means we can‟t use catch block alone.

iii. The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is handled or not.

iv. The “throw” keyword is used to throw an exception.

v. The "throws" keyword is used to declare exceptions. It specifies that there may occur an exception in the method. It doesn't throw an exception. It is always used with method signature.

Example:

```
public class JavaExceptionExample{

public static void main(String args[]){

try{

int data=100/0;

}

catch(ArithmeticException e){

System.out.println(e);

 }}

System.out.println("rest of the code...");

}
```

Multithreading in Java:

• Multithreading in Java is a process of executing multiple threads simultaneously.

• A thread is a lightweight sub-process, the smallest unit of processing.

• Multiprocessing and multithreading, both are used to achieve multitasking.

• However, we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process. Java Multithreading is mostly used in games, animation, etc.

• Advantages of Java Multithreading

1) It doesn't block the user because threads are independent and you can perform multiple operations at the same time.

2) You can perform many operations together, so it saves time.

3) Threads are independent, so it doesn't affect other threads if an exception occurs in a single thread.

Core Java – Programming Logic and Problem Solving

Topics Covered:

- String manipulation and character-level operations
- Logical problem-solving with loops and conditions
- Hands-on coding for real-world problems:
    - Reversing a string
    - Palindrome check
    - Finding factorial
    - Swapping numbers
    - Checking for prime numbers

Program Examples:

1. Reverse a String:

```
String original = "hello";
String reversed = "";
for (int i = original.length() - 1; i >= 0; i--) {
   reversed += original.charAt(i);
}
System.out.println("Reversed: " + reversed);
```

2. Palindrome Check:

```
String word = "madam";
String reverse = "";
for (int i = word.length() - 1; i >= 0; i--) {
   reverse += word.charAt(i);
```

```
    }
    if (word.equals(reverse)) {
        System.out.println(word + " is a palindrome.");
    } else {
        System.out.println(word + " is not a palindrome.");
    }
```

3. Factorial of a Number:

```
    int num = 5;
    int factorial = 1;
    for (int i = 1; i <= num; i++) {
        factorial *= i;
    }
```

Chapter 4

REFLECTION NOTES

4.1 Technical Outcomes

Throughout my experience working with Core Java, I developed a solid understanding of object-oriented programming (OOP) concepts such as inheritance, polymorphism, abstraction, and encapsulation. I gained hands-on experience in creating classes, interfaces, and packages and learned how to design reusable and modular code.

I explored key Java features including exception handling, file handling (I/O streams), multithreading, and collections framework. I worked on small-scale projects like a library management system and a basic console-based banking application, which helped me understand the real-world application of these concepts.

Key Technical Learnings:

1) OOP principles and their implementation in Java
2) Java Collections (List, Set, Map)
3) Exception handling and custom exceptions
4) File I/O using `FileReader`, `BufferedReader`, `FileWriter`, etc.
5) Multithreading using `Thread`, `Runnable`, and synchronization
6) Basic data structures implementation in Java (Stack, Queue, LinkedList)

4.2 Personality Development

Working with Core Java helped me become more analytical and detail-oriented. Debugging code and solving logical problems taught me to think systematically and not give up easily when faced with challenges. Completing Java assignments and mini-projects boosted my confidence and enhanced my self-discipline.

I also improved my communication skills by discussing problems and solutions with peers, explaining code logic during presentations, and receiving constructive feedback.

Growth Areas: Patience, logical thinking, confidence in self-learning, and the ability to explain technical concepts clearly.

4.3 Time Management

Managing Core Java assignments alongside academic responsibilities required structured planning. I began to divide larger problems into smaller tasks and set daily coding goals. Using tools like a to-do list or Google Calendar helped me **track deadlines** and ensure consistent progress.

By consistently dedicating time to learning and practicing Java, I was able to build a habit of **regular coding** and minimize procrastination.

Lessons Learned:

- Breaking down problems into manageable parts
- Allocating fixed hours daily to code
- Tracking progress and staying consistent.

4.4 Skills

Here are the skills I developed during my Core Java journey:

Technical Skills:

1) Java syntax and best practices
2) OOP (Encapsulation, Inheritance, Polymorphism, Abstraction)
3) Java Collections (ArrayList, HashMap, HashSet)
4) Exception handling (try-catch-finally, custom exceptions)
5) File handling using Java I/O streams
6) Multithreading and synchronization
7) Use of basic algorithms and data structures in Java

Chapter 5

RESULTS & CONCLUSION

During the Core Java training/project, I focused entirely on understanding and applying fundamental programming concepts using the Java programming language. I successfully developed a console-based application (such as a Library Management System, Banking System, or Student Information System) that allowed me to implement and test all the core features of Java.

Key programming concepts used in the project included:

Object-Oriented Programming (OOP): I applied the four pillars of OOP – Encapsulation, Inheritance, Polymorphism, and Abstraction – to create clean, modular, and reusable code. I used classes and objects to model real-world entities, interfaces to define behavior, and inheritance for code reusability.

Exception Handling: I implemented robust error handling using try-catch blocks, custom exceptions, and finally blocks to prevent the application from crashing and to improve user experience.

Collections Framework: I utilized Java collections such as ArrayList, HashMap, and HashSet to store, access, and manipulate data efficiently. These collections allowed dynamic data management instead of relying on static arrays.

File I/O: For data persistence, I used Java I/O streams (FileReader, FileWriter, BufferedReader, BufferedWriter) to read from and write to text files. This allowed the application to store user data between sessions.

Multithreading (Optional, if used): I explored the basics of multithreading using the Thread class and Runnable interface to understand how Java handles concurrent execution.

Basic User Interaction: I used Scanner for user input and implemented menus, choices, and loops to guide the user through different application features.

Learning and applying Core Java has been an enriching and rewarding experience. This phase of development allowed me to lay a strong foundation in programming and problem-solving. I developed a deeper understanding of the internal workings of a Java application, particularly in the areas of OOP design, data structures, flow control, and file management.

Working on this project helped me:

1) Improve logical thinking and approach problems in a structured way.
2) Understand the importance of writing clean, readable, and reusable code.
3) Develop skills in debugging, testing, and error tracing using Java's built-in tools.
4) Gain confidence in working independently on software solutions without the use of external frameworks.

Moreover, through this project, I learned the importance of planning the software architecture before coding. I became more familiar with concepts such as class design, method decomposition, code reuse, and separation of concerns — all of which are essential in larger software systems.

From a personal development standpoint, this journey taught me patience, perseverance, and time management. I encountered bugs and logical errors that were difficult to resolve at first, but each challenge improved my problem-solving ability and made me more confident as a programmer.

REFERENCES

[1]. Oracle. (2023). *Java SE Documentation*. Oracle. https://docs.oracle.com/javase/

[2]. Schildt, H. (2019). *Java: The Complete Reference* (11th ed.). McGraw-Hill Education.

[3]. Balagurusamy, E. (2019). *Programming with Java: A Primer* (5th ed.). McGraw-Hill Education.

[4]. Eckel, B. (2006). *Thinking in Java* (4th ed.). Prentice Hall.

[5]. Flanagan, D. (2005). *Java in a Nutshell* (5th ed.). O'Reilly Media.

[6]. Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson Education.

[7]. Sierra, K., & Bates, B. (2008). *Head First Java* (2nd ed.). O'Reilly Media.

[8]. GeeksforGeeks. (2024). Java Programming Language – Tutorials and Examples. https://www.geeksforgeeks.org/java/

[9]. W3Schools. (2024). HTML, CSS, and JavaScript Tutorials. https://www.w3schools.com/

[10]. Baeldung. (2024). Java and Spring Boot Tutorials. https://www.baeldung.com/

[11]. Stack Overflow. (2024). Community Discussions on Java Programming. https://stackoverflow.com/

[12]. MySQL. (2024). MySQL Documentation. https://dev.mysql.com/doc/

[13]. Postman. (2024). Postman API Platform Documentation. https://learning.postman.com/docs/

[14]. JavaTpoint. (2024). Core Java Tutorial. https://www.javatpoint.com/java-tutorial