CHANNABASAVESHWARA INSTITUTE OF TECHNOLOGY
Affiliated to VTU,Belagavi & Approved by AICTE,New-Delhi
(NAAC Accredited & ISO-9001:2015 Certified Institution)
NH 206,(B.H ROAD) GUBBI,TUMKUR – 572 216 Karnataka

Department of Information Science & Engineering

Academic Year : 2025-2026

Advanced Java laboratory

(BIS402)

LAB MANUAL
4th SEM

Prepared By :

Ms. Varsha N
Assistant Professor
Dept.Of ISE

Reviewed By :

Mr. Arun Kumar M S
Assistant Professor
Dept.Of ISE

Approved By:

Dr Thara D K
Professor & HOD
Dept.Of ISE

CHANNABASAVESHWARA INSTITUTE OF TECHNOLOGY
Affiliated to VTU,Belagavi & Approved by AICTE,New-Delhi
(NAAC Accredited & ISO-9001:2015 Certified Institution)
NH 206,(B.H ROAD) GUBBI,TUMKUR – 572 216 Karnataka

INSTITUTE VISION

- To create centers of excellence in education and to serve the society by enhancing the quality of life through value based professional leadership.

INSTITUTE MISSION

- To provide high quality technical and professionally relevant education in a diverse learning environment.
- To provide the values that prepare students to lead their lives with personal integrity, professional ethics and civic responsibility in a global society.
- To prepare the next generation of skilled professionals to successfully compete in the diverse global market.
- To promote a campus environment that welcomes and honors women and men of all races, creeds and cultures, values and intellectual curiosity, pursuit of knowledge and academic integrity and freedom.
- To offer a wide variety of off-campus education and training programmes to individuals and groups.
- To stimulate collaborative efforts with industry, universities, government and professional societies.
- To facilitate public understanding of technical issues and achieve excellence in the operations of the institute.

QUALITY POLICY

Our organization delights customers (students, parents and society) by providing value added quality education to meet the national and international requirements. We also provide necessary steps to train the students for placement and continue to improve our methods of education to the students through effective quality management system, quality policy and quality objectives.

CHANNABASAVESHWARA INSTITUTE OF TECHNOLOGY
Affiliated to VTU,Belagavi & Approved by AICTE,New-Delhi
(NAAC Accredited & ISO-9001:2015 Certified Institution)
NH 206,(B.H ROAD) GUBBI,TUMKUR – 572 216 Karnataka

Department of Information Science & Engineering

VISION

" To educate and train students by imparting strong foundational knowledge in Information Science and Engineering, while creating centres of excellence in computing education."

MISSION

- By providing qualified and dedicated human resources, we ensure students receive the best education in Information Science and Engineering.
- By providing the state-of-the-art infrastructure, we facilitate a conducive learning environment for students pursuing excellence in the field.
- By fostering collaboration with industry and research institutions, we enhance students' exposure to practical applications and cutting-edge developments.
- By instilling a sense of social responsibility and promoting professional ethics, we prepare students to be ethical and socially conscious professionals in their careers.

CHANNABASAVESHWARA INSTITUTE OF TECHNOLOGY
Affiliated to VTU,Belagavi & Approved by AICTE,New-Delhi
(NAAC Accredited & ISO-9001:2015 Certified Institution)
NH 206,(B.H ROAD) GUBBI,TUMKUR – 572 216 Karnataka

Department of Information Science & Engineering

PROGRAM OUTCOMES

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Engineering Tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and the world:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
8. **Individual and Collaborative team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
9. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
10. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
11. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

CHANNABASAVESHWARA INSTITUTE OF TECHNOLOGY
Affiliated to VTU,Belagavi & Approved by AICTE,New-Delhi
(NAAC Accredited & ISO-9001:2015 Certified Institution)
NH 206,(B.H ROAD) GUBBI,TUMKUR – 572 216 Karnataka

Department of Information Science & Engineering

PROGRAMME EDUCATIONAL OBJECTIVES (PEO's)

PEO1: Develop the next generation of highly skilled graduates equipped with a strong knowledge in Information Science and Engineering for creating innovative solutions to society's pressing challenges.

PEO2: To motivate students to pursue higher studies in various disciplines in Information Science and Engineering with the aim of cherishing careers in research and development, academia, industry and entrepreneurship.

PEO3: To Produce engineers who are professionals' entrepreneurs and capable of self-learning to excel in their career.

PEO4: To prepare graduates who excel in diverse learns upholding professional ethics and societal responsibilities.

PROGRAMME SPECIFIC OUTCOMES (PSO's)

PSO1: Students will possess the ability to analyze, design, develop, test and apply mathematical foundations, management principles, and adapt emerging technologies effectively in the creation of computational solutions.

PSO2: Students will possess advanced skills in artificial intelligence, cybersecurity, cloud computing, project management, mobile application development, embedded systems, and big data analytics, making them highly employable, well prepared for higher education, and capable of contributing to cutting-edge research and innovation in Information Science and Engineering.

CHANNABASAVESHWARA INSTITUTE OF TECHNOLOGY
Affiliated to VTU,Belagavi & Approved by AICTE,New-Delhi
(NAAC Accredited & ISO-9001:2015 Certified Institution)
NH 206,(B.H ROAD) GUBBI,TUMKUR – 572 216 Karnataka

Department of Information Science & Engineering

COURSE OBJECTIVES

1. Understanding the fundamentals of collection framework
2. Demonstrate the fundamental concepts of String operations and Swing applications
3. Design and develop web applications using Java servlets and JSP
4. Apply database interaction through Java database Connectivity

CHANNABASAVESHWARA INSTITUTE OF TECHNOLOGY
Affiliated to VTU,Belagavi & Approved by AICTE,New-Delhi
(NAAC Accredited & ISO-9001:2015 Certified Institution)
NH 206,(B.H ROAD) GUBBI,TUMKUR – 572 216 Karnataka

Department of Information Science & Engineering

Syllabus

ADVANCED JAVA	SEMESTER	IV	
Course Code	BIS402	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	3:0:2:0	SEE Marks	50
Total Hours of Pedagogy	40 hours Theory + 8-10 Lab slots	Total Marks	100
Credits	04	Exam Hours	03

PRACTICAL COMPONENT OF IPCC

List of Programs to be Implemented and executed

Sl No.	Experiments
1	Implement a java program to demonstrate creating an ArrayList, adding elements, removing elements, sorting elements of ArrayList. Also illustrate the use of toArray() method.
2	Develop a program to read random numbers between a given range that are multiples of 2 and 5, sort the numbers according to tens place using comparator.
3	Implement a java program to illustrate storing user defined classes in collection.
4	Implement a java program to illustrate the use of different types of string class constructors.
5	Implement a java program to illustrate the use of different types of character extraction, string comparison, string search and string modification methods.
6	Implement a java program to illustrate the use of different types of StringBuffer methods
7	Demonstrate a swing event handling application that creates 2 buttons Alpha and Beta and displays the text “Alpha pressed” when alpha button is clicked and “Beta pressed” when beta button is clicked.
8	A program to display greeting message on the browser “Hello UserName”, “How Are You?”, accept username from the client using servlet.
9	A servlet program to display the name, USN, and total marks by accepting student detail
10	A Java program to create and read the cookie for the given cookie name as “EMPID” and its value as “AN2356”.
11	Write a JAVA Program to insert data into Student DATA BASE and retrieve info based on particular queries(For example update, delete, search etc…).
12	A program to design the Login page and validating the USER_ID and PASSWORD using JSP and DataBase.

CHANNABASAVESHWARA INSTITUTE OF TECHNOLOGY
Affiliated to VTU,Belagavi & Approved by AICTE,New-Delhi
(NAAC Accredited & ISO-9001:2015 Certified Institution)
NH 206,(B.H ROAD) GUBBI,TUMKUR – 572 216 Karnataka

Department of Information Science & Engineering

COURSE OUTCOMES

CO 1. Apply appropriate collection class/interface to solve the given problem

CO 2. Demonstrate the concepts of String operations in Java

CO 3. Apply the concepts of Swings to build Java applications

CO 4. Develop web based applications using Java servlets and JSP

CO 5. Use JDBC to build database applications

COURSE ASSESMENT DETAILS (Both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together.

CIE for the theory component of the IPCC (maximum marks 50)

- IPCC means practical portion integrated with the theory of the course.
- CIE marks for the theory component are **25 marks** and that for the practical component is **25 marks**.
- 25 marks for the theory component are split into **15 marks** for two Internal Assessment Tests (Two Tests, each of 15 Marks with 01-hour duration, are to be conducted) and **10 marks** for other assessment methods mentioned in 22OB4.2. The first test at the end of 40-50% coverage of the syllabus and the second test after covering 85-90% of the syllabus.
- Scaled-down marks of the sum of two tests and other assessment methods will be CIE marks for the theory component of IPCC (that is for **25 marks)**.
- The student has to secure 40% of 25 marks to qualify in the CIE of the theory component of IPCC.

CIE for the practical component of the IPCC

- **15 marks** for the conduction of the experiment and preparation of laboratory record, and **10 marks** for the test to be conducted after the completion of all the laboratory sessions.
- On completion of every experiment/program in the laboratory, the students shall be evaluated including viva-voce and marks shall be awarded on the same day.
- The CIE marks awarded in the case of the Practical component shall be based on the continuous evaluation of the laboratory report. Each experiment report can be evaluated for 10 marks. Marks of all experiments'write-ups are added and scaled down to **15 marks**.
- The laboratory test **(duration 02/03 hours)** after completion of all the experiments shall be conducted for 50 marks and scaled down to **10 marks.**
- Scaled-down marks of write-up evaluations and tests added will be CIE marks for the laboratory component of IPCC for **25 marks**.
- The student has to secure 40% of 25 marks to qualify in the CIE of the practical component of the IPCC.

SEE for IPCC Theory SEE will be conducted by University as per the scheduled timetable, with common question papers for the course (**duration 03 hours**)

1. The question paper will have ten questions. Each question is set for 20 marks.
2. There will be 2 questions from each module. Each of the two questions under a module (with a maximum of 3 sub-questions), **should have a mix of topics** under that module.
3. The students have to answer 5 full questions, selecting one full question from each module.
4. Marks scored by the student shall be proportionally scaled down to 50 Marks

The theory portion of the IPCC shall be for both CIE and SEE, whereas the practical portion will have a CIE component only. Questions mentioned in the SEE paper may include questions from the practical component.

CHANNABASAVESHWARA INSTITUTE OF TECHNOLOGY
Affiliated to VTU,Belagavi & Approved by AICTE,New-Delhi
(NAAC Accredited & ISO-9001:2015 Certified Institution)
NH 206,(B.H ROAD) GUBBI,TUMKUR – 572 216 Karnataka

Department of Information Science & Engineering

RBT LEVELS						
COS	**Remember (L1)**	**Understand (L2)**	**Apply (L3)**	**Analyze (L4)**	**Evaluate (L5)**	**Create (L6)**
CO1		√				
CO2			√			
CO3			√			
CO4			√			
CO5			√			

CO-PO MAPPING:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	2	2	2	2	2	0	0	0	0	0	0	0
CO2	1	1	2	2	2	0	0	0	0	0	0	0
CO3	2	2	2	2	2	0	0	0	0	0	0	0
CO4	2	2	2	2	2	0	0	0	0	0	0	0
CO5	2	2	2	2	2	0	0	0	0	0	0	0
Average	**1.8**	**1.8**	**2**	**2**	**2**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

CHANNABASAVESHWARA INSTITUTE OF TECHNOLOGY
Affiliated to VTU,Belagavi & Approved by AICTE,New-Delhi
(NAAC Accredited & ISO-9001:2015 Certified Institution)
NH 206,(B.H ROAD) GUBBI,TUMKUR – 572 216 Karnataka

Department of Information Science & Engineering

CO-PSO MAPPING:

COs	PSO1	PSO2	PSO3	PSO4
CO1	1	1	0	0
CO2	1	1	0	0
CO3	1	1	0	0
CO4	1	1	0	0
CO5	1	1	0	0
Average	**1**	**1**	**0**	**0**

Instructions to the Candidates

General Lab Guidelines:

• Conduct yourself in a responsible manner at all times in the laboratory. Intentional misconduct will lead to the exclusion from the lab.

• Do not wander around, or distract other students, or interfere with the laboratory experiments of other students.

• Read the handout and procedures before starting the experiments. Follow all written and verbal instructions carefully. If you do not understand the procedures, ask the instructor or teaching assistant.

• Attendance in all the labs is mandatory, absence permitted only with prior permission from Class teacher.

• The workplace has to be tidy before, during and after the experiment.

• Do not eat food, drink beverages or chew gum in the laboratory.

DO'S:

• Uniform and ID card are mandatory inside the laboratory.
• Read the experiment/program thoroughly before execution.
• Strictly follow the procedure given in the lab manual.
• Records must be submitted every week for evaluation without fail.
• Maintain silence and discipline in the laboratory.
• Verify the output with the instructor before leaving the lab.
• Shut down the system properly after completion of the session.
• Push chairs properly under the tables before leaving.
• Sign the attendance/log book when you enter/leave the laboratory.
• Inform the instructor immediately if the system is not working properly.
• Keep your personal belongings in the designated area.

DON'TS:

• Do not use mobile phones during lab hours without permission.
• Do not install any unauthorized software or applications.
• Do not change system settings (IP address, system configuration, control panel settings, etc.).
• Do not access social media, games, or unrelated websites.
• Do not copy programs from others without understanding.
• Do not use another student's login credentials.
• Do not delete or modify files belonging to other students.
• Do not plug/unplug cables (keyboard, mouse, network, power cables) unnecessarily.
• Do not shut down the system abruptly without proper exit.
• Do not eat or drink inside the laboratory.
• Do not leave the lab without faculty permission.
• Students are not allowed to work in the laboratory without the presence of the instructor/teaching staff.

TABLE OF CONTENTS

Introduction

ArrayList – Creation, Manipulation and Conversion

The first experiment demonstrates the working of the **ArrayList class**, which is a part of the Java Collections Framework. The Java Collections Framework provides a standardized architecture to store, manipulate, and retrieve groups of objects dynamically. Unlike traditional arrays in Java, which have a fixed size defined at the time of declaration, an ArrayList is dynamic in nature. This means its size can grow or shrink automatically depending on the number of elements inserted or removed.

Internally, ArrayList is implemented using a resizable array. When elements are added beyond its current capacity, the system automatically creates a new larger array and copies the existing elements into it. This resizing mechanism ensures flexibility but may involve temporary performance overhead during resizing.

In this program, elements are added using the `add()` method, accessed using `get()`, removed using `remove()`, and displayed using iteration. The sorting operation is performed using `Collections.sort()`, which arranges the elements in ascending order. This method internally uses a highly optimized sorting algorithm called TimSort. The experiment also demonstrates the use of `toArray()` method, which converts the ArrayList into a regular array. This is useful when integration with legacy APIs or array-based processing is required.

This experiment introduces students to dynamic data structures and object-based storage mechanisms, which are fundamental in modern software development.

Custom Sorting Using Comparator Interface

This experiment focuses on generating random numbers and sorting them using a custom logic. The program makes use of the `Random` class to generate numbers dynamically within a specified range. Random number generation is essential in simulations, testing, and game development scenarios.

The core concept demonstrated here is the use of the **Comparator interface**. In Java, default sorting is performed based on natural ordering (ascending order for integers). However, when sorting needs to be performed based on custom conditions — such as sorting numbers based on their tens digit — the Comparator interface is used.

The Comparator interface contains a `compare()` method that defines how two objects should be compared. In this program, the comparison logic extracts the tens place of each number by dividing it by 10 and compares those values. This shows how developers can override default behavior and implement domain-specific sorting logic.

The experiment highlights the importance of abstraction and polymorphism in Java, as it demonstrates how behavior can be customized without modifying the core data structure.

Storing User-Defined Objects in Collections

This experiment demonstrates how Java collections can store not only primitive wrapper classes but also user-defined objects. A custom class named `Address` is created with instance

variables representing name, street, city, state, and code. Objects of this class are then stored inside a LinkedList.

The key concept here is object-oriented programming combined with collections. Java stores objects in collections as references. Therefore, when an object is added to a collection, the memory reference of that object is stored.

The program overrides the `toString()` method. This is important because, by default, printing an object displays its hashcode representation. Overriding `toString()` ensures meaningful output is displayed. This demonstrates the importance of method overriding and runtime polymorphism.

The use of LinkedList instead of ArrayList introduces students to different data structure implementations. LinkedList internally uses a doubly linked list structure, making insertion and deletion operations more efficient compared to ArrayList in certain scenarios.

This experiment builds strong foundational understanding of object storage and memory reference handling in Java.

String Class Constructors

This experiment explores various ways of constructing String objects in Java. The String class in Java is immutable, meaning once a String object is created, it cannot be modified. Any modification results in the creation of a new String object in memory.

The program demonstrates multiple constructors, such as creating a String from a character array, byte array, another String, and even from StringBuilder or StringBuffer. It also demonstrates character encoding using UTF-8, which converts byte data into readable characters.

Understanding immutability is critical. Since Strings are widely used, making them immutable ensures security, thread-safety, and efficient memory usage through the String Constant Pool. When a String literal is created, it is stored in a special memory area called the String pool to avoid duplication.

This experiment explains how Java manages textual data internally and how memory efficiency is maintained.

String Manipulation and Comparison

This experiment deeply explores various string manipulation techniques. It demonstrates character extraction using `charAt()` and `getChars()`, conversion methods like `toCharArray()` and `getBytes()`, and comparison techniques such as `equals()`, `equalsIgnoreCase()`, and `compareTo()`.

A very important concept demonstrated is the difference between `==` and `equals()`. The `==` operator checks reference equality (whether two references point to the same memory location), while `equals()` checks content equality. This distinction is fundamental in Java programming.

The program also shows searching techniques using `indexOf()` and `lastIndexOf()`, and modification methods like `replace()`, `trim()`, and `concat()`. Since String is immutable, each modification creates a new object.

This experiment builds a strong understanding of text processing and memory behavior in Java.

StringBuffer and Mutable Strings

Unlike the String class, StringBuffer is mutable. This means modifications happen on the same object without creating new memory locations. This makes StringBuffer more efficient when multiple modifications are required.

The experiment demonstrates methods like `append()`, `insert()`, `delete()`, `reverse()`, and `replace()`. It also explains capacity and length. Capacity refers to allocated memory, whereas length refers to actual characters stored.

StringBuffer is thread-safe, meaning its methods are synchronized, making it suitable for multi-threaded applications. However, this synchronization may slightly reduce performance compared to StringBuilder.

This experiment teaches memory optimization and performance-aware programming.

Swing Event Handling

This experiment introduces GUI programming using Swing. Swing is part of Java Foundation Classes and is used to build platform-independent desktop applications.

The program demonstrates event-driven programming. When a user clicks a button, an event is generated. This event is captured by an ActionListener, and the corresponding `actionPerformed()` method executes. This is known as the Delegation Event Model.

The use of `EventQueue.invokeLater()` ensures that GUI components are created and updated in the Event Dispatch Thread, which maintains thread safety in graphical applications.

This experiment introduces user interaction handling and graphical programming concepts.

Servlets – Handling Client Requests

Servlets are server-side programs that handle HTTP requests and generate dynamic responses. They run inside a web container like Apache Tomcat.

When a client submits a form, the request object carries user input to the servlet. The servlet processes the input and sends a response using the response object. The lifecycle of a servlet includes initialization, service handling, and destruction.

These experiments demonstrate form handling, parameter retrieval using `getParameter()`, and dynamic HTML generation using PrintWriter.

This introduces students to web application development fundamentals.

Cookies in Servlets

Cookies are small pieces of information stored on the client's browser. They help maintain user state between multiple HTTP requests.

The experiment shows how to create a Cookie object, set its maximum age, add it to the response, and later retrieve it from the request. Since HTTP is stateless, cookies help in session tracking.

This experiment introduces state management concepts in web programming.

CRUD Application

This experiment simulates database operations using ArrayList. CRUD stands for Create, Read, Update, and Delete.

The program uses a menu-driven approach with Scanner for user input. Objects of Student class are stored in an ArrayList. Operations are performed dynamically, demonstrating control flow, data manipulation, and object-oriented design.

This forms the base for understanding real database-driven applications.

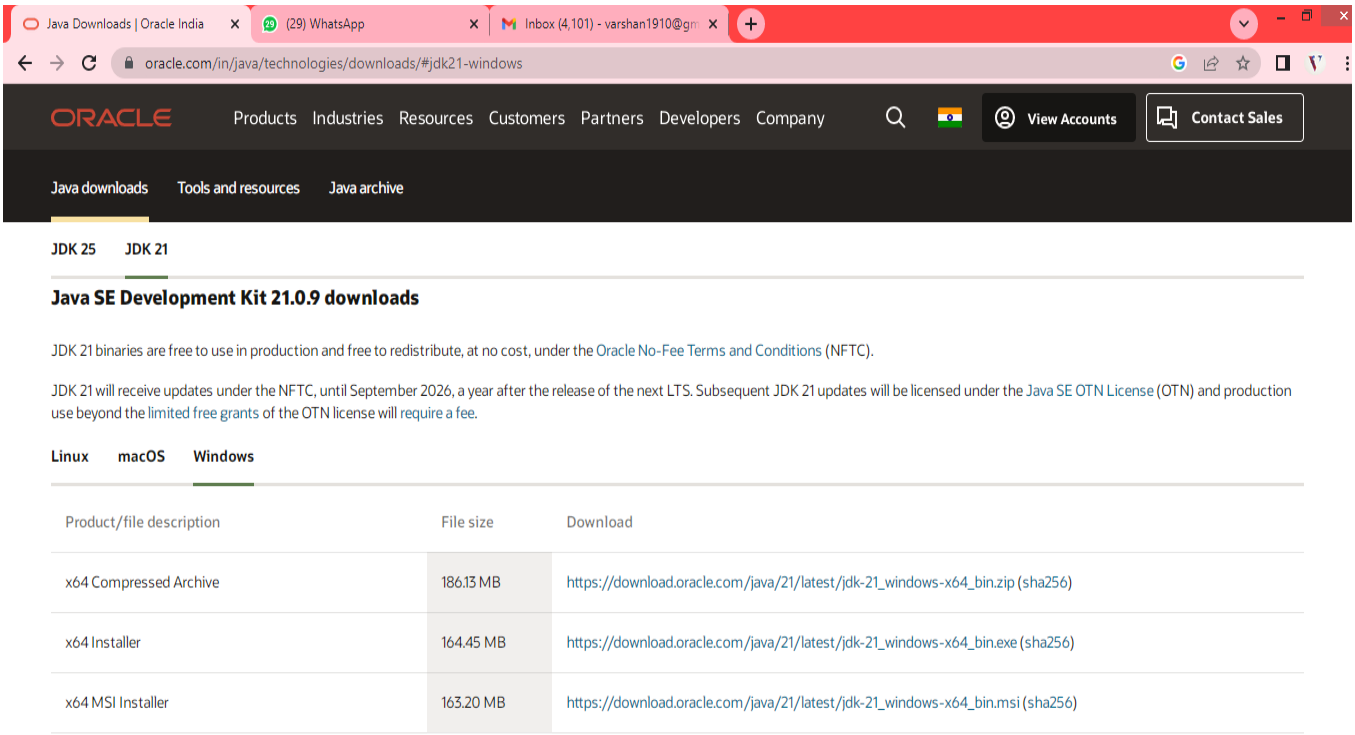
JSP Login Validation

JSP simplifies servlet-based web development by allowing Java code inside HTML pages. When a JSP page is executed, it is internally converted into a servlet.

The program demonstrates retrieving form data, validating credentials, and dynamically generating responses. Scriptlet tags `<% %>` embed Java code, while expression tags `<%= %>` display dynamic data.

This experiment introduces server-side scripting and web authentication logic.

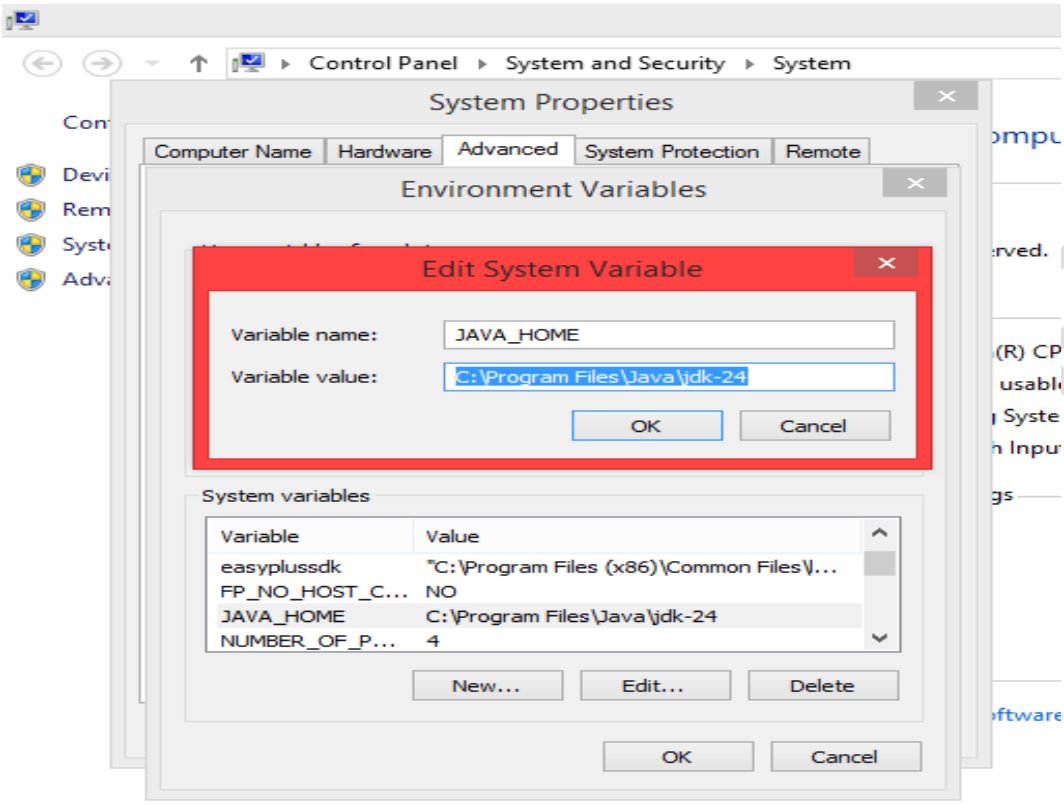
Installation Procedure



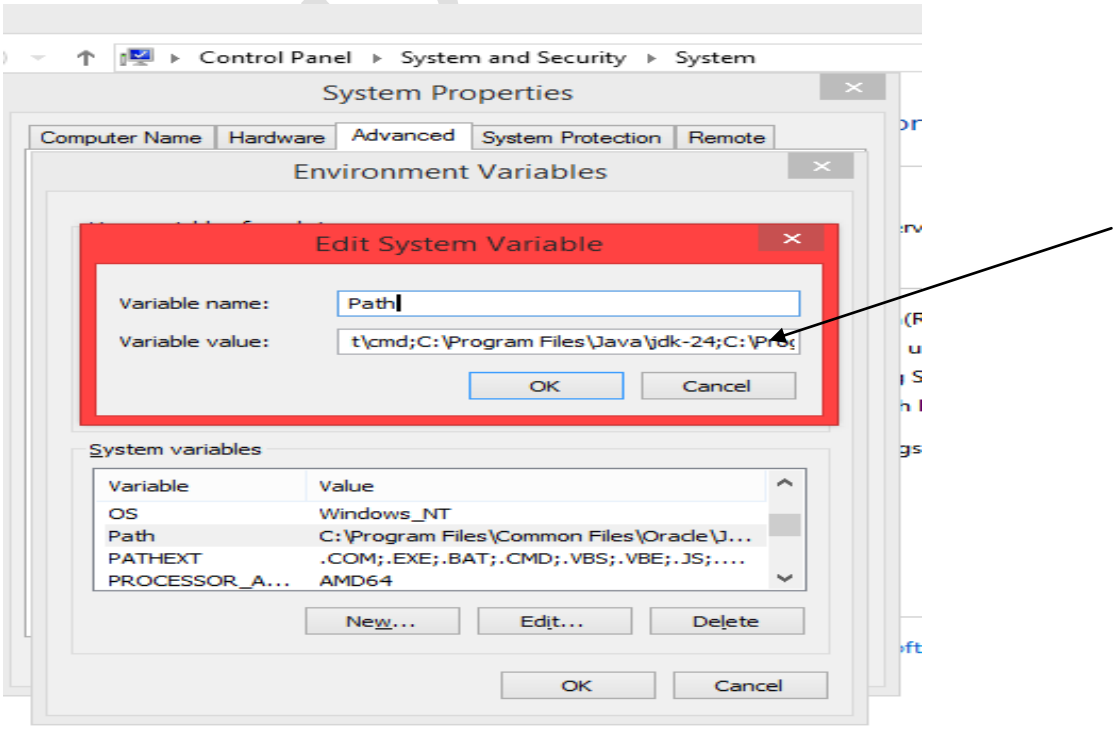
1) JDK Download

Download only "https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.**exe** (sha256)" this link

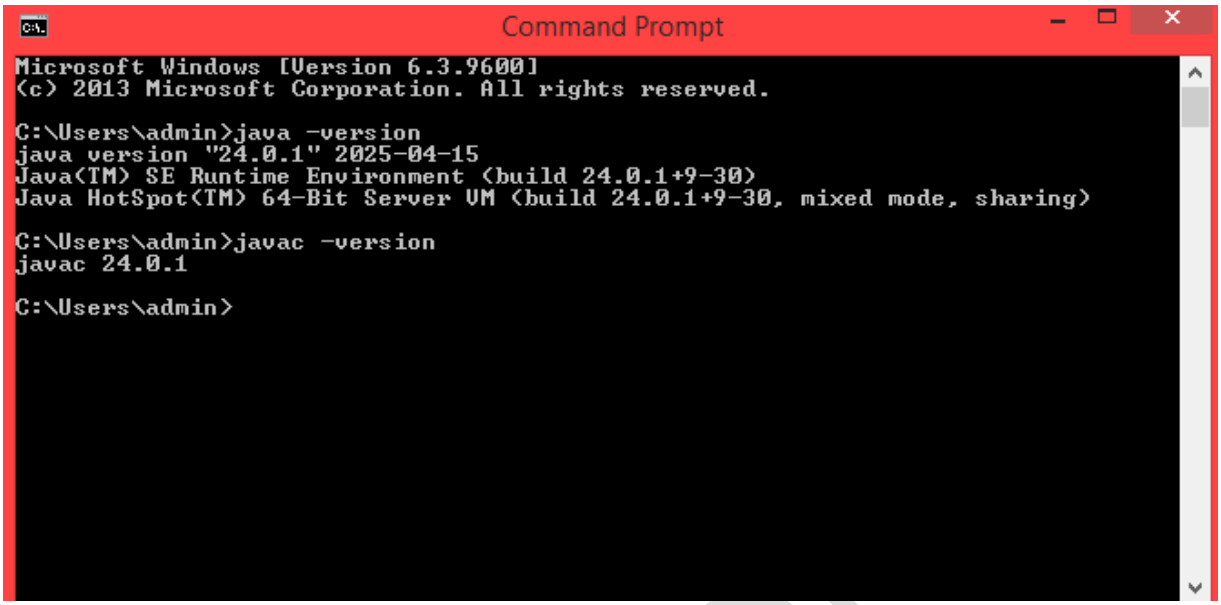
Set JAVA_HOME path in system properties > advanced system settings > Environment variables > New



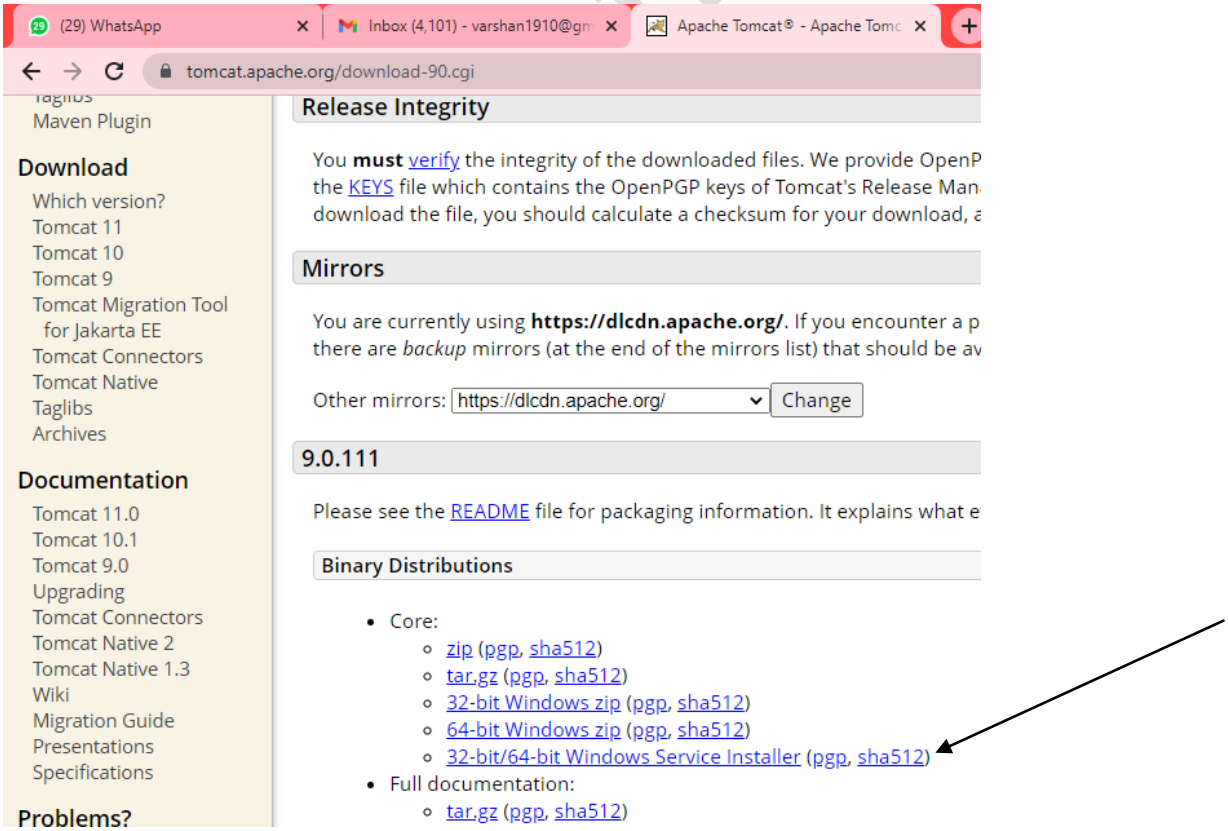
Set Path(path variable already exists, just add " ; " and add jdk path) Exactly as shown as below :



In Cmd Prompt Check “java –version” and “javac –version” , Both should give the same version



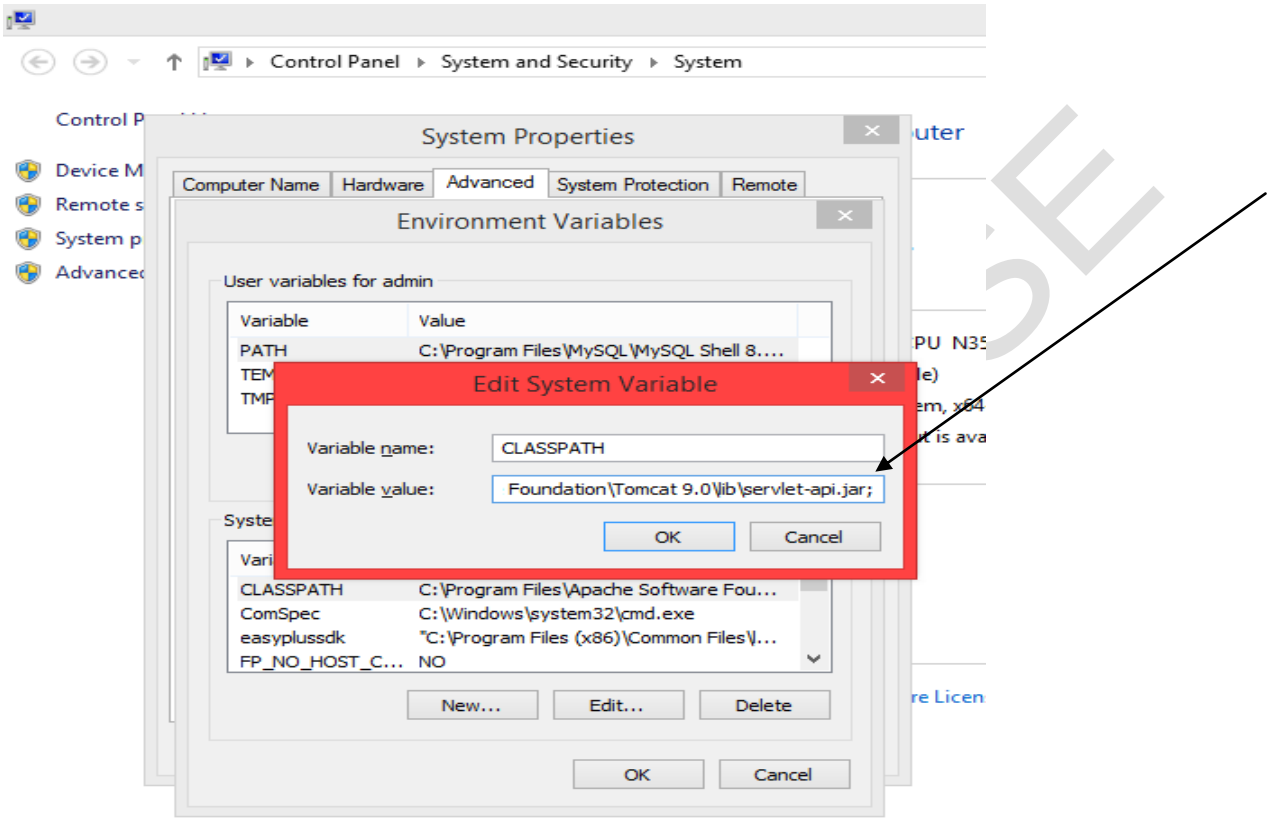
2) Apache Tomcat download



Download only “32-bit/64-bit Windows Service Installer (pgp, sha512)”

Set CLASSPATH in system properties > advanced system settings > Environment variables > New

Add "C:\Program Files\Apache Software Foundation\Tomcat 9.0\lib\servlet-api.jar;" this path



All of your files will be installed in PROGRAM FILES by Default

1) Implement a java program to demonstrate creating an ArrayList, adding elements, removing elements ,sorting elements of ArrayList. Also illustrate the use of toArray() method.

```
import java.util.*;

public class prgram1 {
    public static void main(String[] args) {

        // Create ArrayLists
        ArrayList<Integer> list = new ArrayList<Integer>();
        ArrayList<Integer> list1 = new ArrayList<Integer>();

        // Add elements to ArrayList
        list.add(30);
        list.add(20);
        list.add(10);
        list.add(40);

        // Display the ArrayList
        System.out.println("ArrayList: " + list);

        // Remove an element from ArrayList (index 2)
        list.remove(2);
        System.out.println("Updated ArrayList: " + list);

        // Access an element
        int element = list.get(1);
        System.out.println("Accessed Element: " + element);

        // Iterate through ArrayList
        System.out.print("Iterating over ArrayList: ");
        for (int i = 0; i < list.size(); i++) {
            System.out.print(list.get(i) + " ");
        }
        System.out.println();

        // Before sorting
        System.out.println("Before sorting: " + list);

        // Sort the ArrayList
        Collections.sort(list);
        System.out.println("After sorting: " + list);

        // Convert ArrayList to Array
        Object[] arr = list.toArray();
        System.out.println("Elements of ArrayList as Array: " +
Arrays.toString(arr));
    }
}
```

OUTPUT :

```
ArrayList: [30, 20,  10, 40]
Updated ArrayList: [30, 20, 40]
Accessed Element: 20
Iterating over ArrayList: 30 20 40
Before sorting: [30, 20, 40]
After sorting: [20, 30, 40]
Elements of ArrayList as Array: [20, 30, 40]
```

2) Develop a program to read random numbers between a given range that are multiples of 2 and 5, sort the numbers according to tens place using comparator.

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.Random;

public class Main {

    public static void main(String[] args) {

        ArrayList<Integer> numbers = new ArrayList<Integer>();
        Random random = new Random();

        // Generate 10 random numbers between 10 and 100
        // that are multiples of both 2 and 5
        while (numbers.size() < 10) {

            int number = random.nextInt(21) * 5; // generates multiples of
5 (0 to 100)

            if (number >= 10 && number <= 100 && number % 2 == 0) {
                numbers.add(number);
            }
        }

        // Sort the numbers based on the tens place
        Collections.sort(numbers, new Comparator<Integer>() {
            @Override
            public int compare(Integer o1, Integer o2) {
                return Integer.compare(o1 / 10, o2 / 10);
            }
        });

        // Display the sorted numbers
        System.out.println("Numbers sorted based on tens place:");
        for (int number : numbers) {
            System.out.println(number);
        }
    }
}
```

OUTPUT :

```
Numbers sorted based on tens place:
 100
 70
 75
 50
 55
 60
 85
 10
 35
 25
```

3) Implement a java program to illustrate storing user defined classes in collection.

```
// A simple mailing list example
import java.util.LinkedList;

class Address {

    private String name;
    private String street;
    private String city;
    private String state;
    private String code;

    // Constructor
    Address(String n, String s, String c, String st, String cd) {
        name = n;
        street = s;
        city = c;
        state = st;
        code = cd;
    }

    // Convert Address object to String
    public String toString() {
        return name + "\n" +
               street + "\n" +
               city + " " + state + " " + code;
    }
}

public class MailList {

    public static void main(String[] args) {

        // Create a LinkedList of Address objects
        LinkedList<Address> ml = new LinkedList<Address>();

        // Add elements to the LinkedList
        ml.add(new Address("J.W. West", "11 Oak Ave", "Urbana", "IL",
"61801"));
        ml.add(new Address("Ralph Baker", "1142 Maple Lane", "Mahomet",
"IL", "61853"));
        ml.add(new Address("Tom Carlton", "867 Elm St", "Champaign", "IL",
"61820"));

        // Display the mailing list
        System.out.println("Mailing List:\n");
        for (Address element : ml) {
            System.out.println(element);
            System.out.println();
        }
    }
}
```

OUTPUT :

```
Mailing List:

J.W. West
11 Oak Ave
Urbana  IL  61801

Ralph Baker
1142 Maple  Lane
Mahomet  IL  618553

Tom Carlton
867 Elm St
Champaign  IL 61820

for Address element : mi)
```

4) Implement a java program to illustrate the use of different types of string class constructors.

```
import java.io.UnsupportedEncodingException;

public class Program4 {

    public static void main(String[] args) throws
UnsupportedEncodingException {

        // Character array
        char c[] = {'j', 'a', 'v', 'a'};

        // Default String constructor
        String s1 = new String();
        System.out.println("Default constructor: " + s1);

        // Construct one string from char array
        String s2 = new String(c);
        String s3 = new String(s2);
        System.out.println("String object: " + s2);
        System.out.println("String object: " + s3);

        // Construct string from byte array
        byte ascii[] = {65, 66, 67, 68, 69, 70};
        String s4 = new String(ascii);
        System.out.println(s4);

        // Construct string from subset of byte array
        String s5 = new String(ascii, 2, 3);
        System.out.println(s5);

        // Construct string from StringBuilder
        StringBuilder s6 = new StringBuilder("Hello");
        System.out.println(s6.toString());
        s6.append(" World!");
        System.out.println(s6.toString());

        // StringBuffer example
        StringBuffer s7 = new StringBuffer("Hello ");
        s7.insert(1, "Java");   // modifies original string
        System.out.println("Now original string is: " + s7);

        // Construct string using byte array with offset, length and
charset
        byte[] byteArrayOffsetCharset = {72, 101, 108, 108, 111, 113, 117};
        String str8 = new String(byteArrayOffsetCharset, 6, 1, "UTF-8");
        System.out.println(
            "Constructor with byte array, offset, length and charset: " +
str8
        );
    }
}
```

OUTPUT:

```
Default constructor:
String object: java
String object: java
ABCDEF
Construct one string from char array
ABCDEF
Construct string from byte array
Hello World!
Hello World!
Now original string is: HJavaelIo
Constructor with byte array, offset, length and charset: q
```

5) Implement a java program to illustrate the use of different types of character extraction, string comparison, string search and string modification methods.

```
public class StringDemo {
    public static void main(String[] args) {

        String s = "This is a demo of the getChars method";
        int start = 10;
        int end = 14;

        char buf[] = new char[end - start];
        s.getChars(start, end, buf, 0);
        System.out.println(buf);

        char ch;
        ch = "abc".charAt(1);
        System.out.println(ch);

        String s1 = "Hello";

        byte[] barr = s1.getBytes();
        for (int i = 0; i < barr.length; i++) {
            System.out.println(barr[i]);
        }

        char[] sh = s1.toCharArray();
        for (int i = 0; i < sh.length; i++) {
            System.out.println(sh[i]);
        }

        String s2 = "good bye";
        String s3 = "Hello";

        System.out.println(s1 + " equals " + s2 + " -> " + s1.equals(s2));
        System.out.println(s1 + " equalsIgnoreCase " + s3 + " -> " +
s1.equalsIgnoreCase(s3));

        System.out.println(s1.endsWith("lo"));
        System.out.println(s1.startsWith("He"));

        System.out.println(s1 + " == " + s2 + " -> " + (s1 == s2));

        String arr[] = {"Now", "is", "the", "time"};

        // Sorting strings using compareTo
        for (int j = 0; j < arr.length; j++) {
            for (int i = j + 1; i < arr.length; i++) {
                if (arr[i].compareTo(arr[j]) < 0) {
                    String t = arr[j];
                    arr[j] = arr[i];
                    arr[i] = t;
                }
            }
            System.out.println(arr[j]);
        }

        System.out.println("IndexOf(e) = " + s1.indexOf('e'));
        System.out.println("lastIndexOf(bye) = " + s2.lastIndexOf("bye"));

        String s4 = "this is a test. this is too";
```

```
        String search = "is";
        String sub = "was";

        // Replace first occurrence
        String result = s4.replaceFirst(search, sub);
        System.out.println(result);

        String s5 = s1.concat(" two");
        System.out.println(s5);

        String s6 = "Hello".replace('l', 'w');
        System.out.println(s6);

        String s7 = " Helloworld ".trim();
        System.out.println(s7);

        String upper = s1.toUpperCase();
        String lower = s1.toLowerCase();

        System.out.println(upper);
        System.out.println(lower);
    }
}
```

OUTPUT :

```
demo
b
72
101
108
108
111
H
e
l
l
o
Helloequalsgood bye->false
HelloequalsIsIgnoreCaseHello->true
true
true
Hello==good bye->false
Now
Is
the
time
Indexof(e)=1
lastIndexOf(bye)=5
Hellotwo
Hello
Helloworld
HELLO
hello
```

6) Implement a java program to illustrate the use of different types of StringBuffer methods

```
public class Program6Demo {

    public static void main(String[] args) {

        StringBuffer sb = new StringBuffer("Hello");

        // Display initial buffer
        System.out.println("Buffer: " + sb);
        System.out.println("Length: " + sb.length());
        System.out.println("Capacity: " + sb.capacity());

        // charAt and setCharAt
        System.out.println("charAt(1) before = " + sb.charAt(1));
        sb.setCharAt(1, 'i');
        System.out.println("charAt(1) after  = " + sb.charAt(1));

        // append using another StringBuffer
        int a = 42;
        StringBuffer sbb = new StringBuffer(40);
        String s = sbb.append("a=").append(a).append("!").toString();
        System.out.println(s);

        // insert operation
        sb.insert(2, "like");
        System.out.println("After insert: " + sb);

        // reverse operation
        sb.reverse();
        System.out.println("After reverse: " + sb);

        // delete characters
        sb.delete(4, 7);
        System.out.println("After delete: " + sb);

        // delete character at index
        sb.deleteCharAt(0);
        System.out.println("After deleteCharAt: " + sb);

        // replace characters
        sb.replace(5, 7, "wo");
        System.out.println("After replace: " + sb);

        // substring
        System.out.println("Substring from index 1: " + sb.substring(1));

        // indexOf and lastIndexOf
        int i = sb.indexOf("One");
        System.out.println("First Index of 'One': " + i);

        i = sb.lastIndexOf("One");
        System.out.println("Last Index of 'One': " + i);
    }
}
```

OUTPUT :

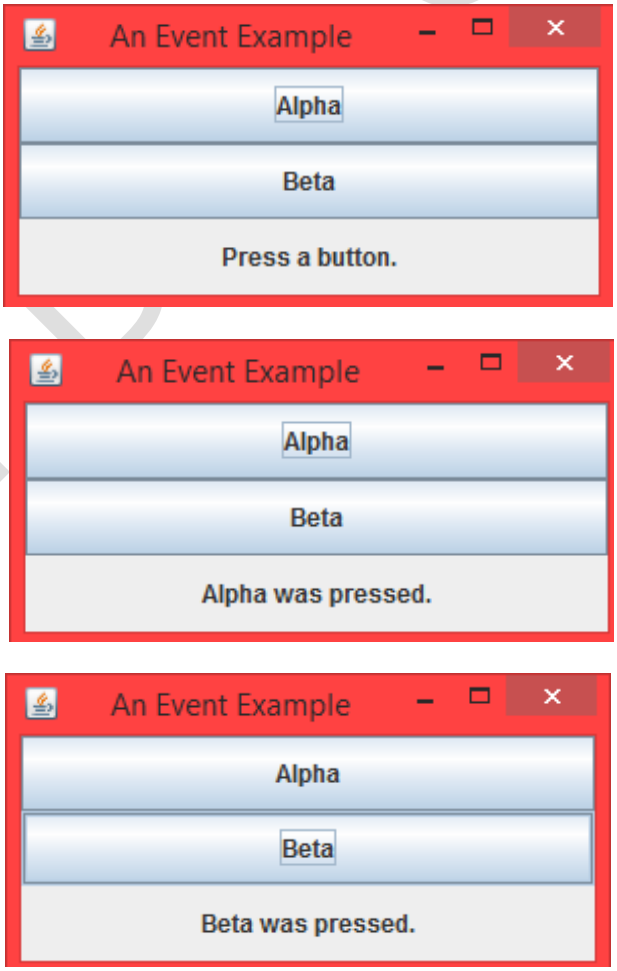
```
Buffer:Hello
Length:5
Capacity:21
charAt(1)before=e
charAt(1)After=i
a=42!
Hilikello
ollekiliH
After delete:olleiH
After delete charAt:lleiH
After replace:lleiHwo
String subString:leiHwo
First Index:-1
Last Index:-1
```

7) Demonstrate a swing event handling application that creates 2 buttons Alpha and Beta and displays the text “Alpha pressed” when alpha button is clicked and “Beta pressed” when beta button is clicked.

```
import java.awt.EventQueue;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;

public class Program7 {

    private JFrame frame;
    private JLabel jlab;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Program7 window = new Program7();
                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the application.
     */
    public Program7() {
        initialize();
    }

    /**
     * Initialize the contents of the frame.
     */
    private void initialize() {

        frame = new JFrame("An Event Example");

        // Set GridLayout
        frame.setLayout(new GridLayout(3, 1));

        // Set frame size
        frame.setSize(300, 150);

        // Close operation
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Create buttons
        JButton jbtnAlpha = new JButton("Alpha");
```

```
        JButton jbtnBeta = new JButton("Beta");

        // Create label
        jlab = new JLabel("Press a button.", JLabel.CENTER);

        // Add ActionListener for Alpha button
        jbtnAlpha.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent ae) {
                jlab.setText("Alpha was pressed.");
            }
        });

        // Add ActionListener for Beta button
        jbtnBeta.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent ae) {
                jlab.setText("Beta was pressed.");
            }
        });

        // Add components to frame
        frame.add(jbtnAlpha);
        frame.add(jbtnBeta);
        frame.add(jlab);
    }
}
```

OUTPUT :



8) A program to display greeting message on the browser “Hello UserName”, “How Are You?”, accept username from the client using servlet.

Step 1: Create HTML Form (index.html)

Save this inside Web Content (or webapp folder)

```
<!DOCTYPE html>
<html>
<head>
    <title>Greeting Form</title>
</head>
<body>
    <h2>Enter Your Name</h2>
    <form action="GreetServlet" method="post">
        User Name: <input type="text" name="username" required>
        <br><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

Step 2: Create Servlet (GreetServlet.java)

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class GreetServlet extends HttpServlet {

    protected void doPost(HttpServletRequest request,
                          HttpServletResponse response)
                          throws ServletException, IOException {

        // Set content type
        response.setContentType("text/html");

        // Get username from form
        String name = request.getParameter("username");

        // Get PrintWriter object
        PrintWriter out = response.getWriter();

        // Display greeting message
        out.println("<html>");
        out.println("<body>");
        out.println("<h2>Hello " + name + "</h2>");
        out.println("<h3>How Are You?</h3>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

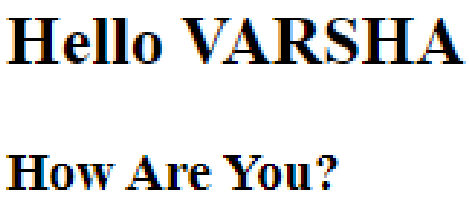
Step 3: Configure web.xml

```
<web-app>
    <servlet>
        <servlet-name>GreetServlet</servlet-name>
        <servlet-class>GreetServlet</servlet-class>
    </servlet>
```

```
    <servlet-mapping>
        <servlet-name>GreetServlet</servlet-name>
        <url-pattern>/GreetServlet</url-pattern>
    </servlet-mapping>
</web-app>
```

Run On The Server : http://localhost:8080/GreetApp/index.html

OUTPUT :



9) A servlet program to display the name, USN, and total marks by accepting student detail

Step 1: HTML Form (student.html)

Place this file in your webapp folder:

```
<!DOCTYPE html>
<html>
<head>
    <title>Student Details</title>
</head>
<body>
    <h2>Enter Student Details</h2>
    <form action="student" method="post">
        Name: <input type="text" name="name" required><br><br>
        USN: <input type="text" name="usn" required><br><br>
        Total Marks: <input type="number" name="marks" required><br><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

Step 2: Servlet (StudentServlet.java)

Place this class in your src/demo package (or similar):

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class StudentServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        String name = request.getParameter("name");
        String usn = request.getParameter("usn");
        String marks = request.getParameter("marks");
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><head><title>Student Details</title></head><body>");
        out.println("<h2>Student Details Submitted</h2>");
        out.println("<p>Name: " + name + "</p>");
        out.println("<p>USN: " + usn + "</p>");
        out.println("<p>Total Marks: " + marks + "</p>");
        out.println("<a href='student.html'>Back to Form</a>");
        out.println("</body></html>");
    }
}
```

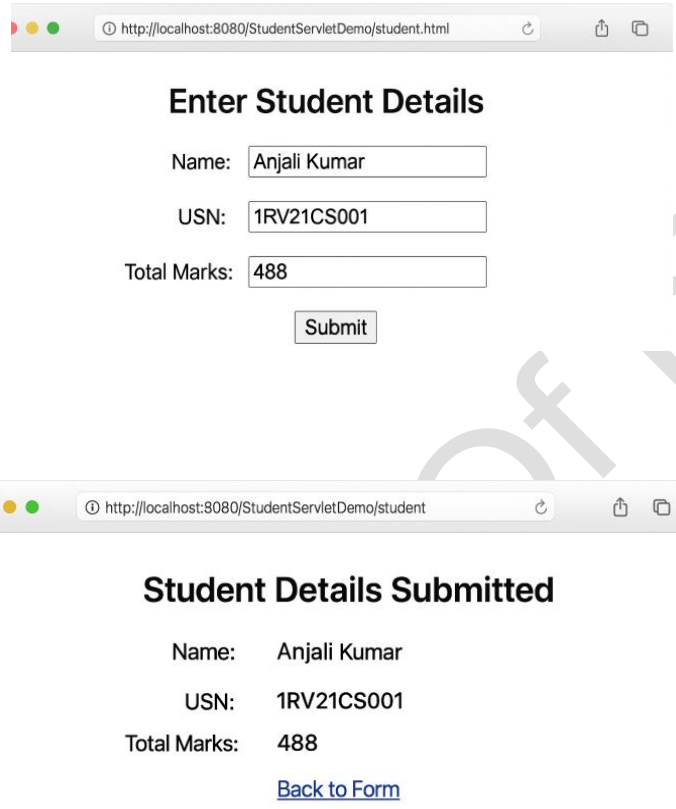
Step 3: web.xml Configuration

In WebContent/WEB-INF/web.xml, add:

```
<servlet>
    <servlet-name>StudentServlet</servlet-name>
    <servlet-class>demo.StudentServlet</servlet-class>
</servlet>
```

```
<servlet-mapping>
    <servlet-name>StudentServlet</servlet-name>
    <url-pattern>/student</url-pattern>
</servlet-mapping>
```

OUTPUT :



10) A Java program to create and read the cookie for the given cookie name as “EMPID” and its value as “AN2356”.

Servlet Program (Create and Read Cookie)

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.WebServlet;

@WebServlet("/CookieServlet")
public class CookieServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request,
                         HttpServletResponse response)
                         throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        // Create Cookie
        Cookie empCookie = new Cookie("EMPID", "AN2356");

        // Set cookie expiry time (1 hour)
        empCookie.setMaxAge(60 * 60);

        // Add cookie to response
        response.addCookie(empCookie);

        out.println("<h3>Cookie Created Successfully!</h3>");

        // Read Cookie
        Cookie[] cookies = request.getCookies();

        if (cookies != null) {
            for (Cookie c : cookies) {
                if (c.getName().equals("EMPID")) {
                    out.println("<h3>Cookie Name: " + c.getName() + "</h3>");
                    out.println("<h3>Cookie Value: " + c.getValue() + "</h3>");
                }
            }
        } else {
            out.println("<h3>No Cookies Found</h3>");
        }
    }
}
```

web.xml

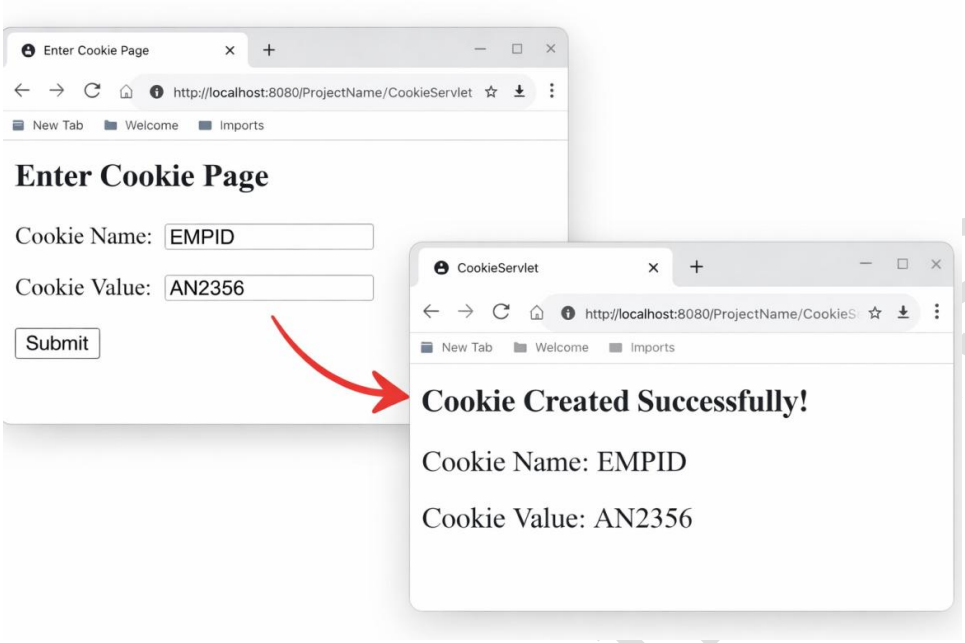
```
<web-app>
    <servlet>
        <servlet-name>CookieServlet</servlet-name>
        <servlet-class>CookieServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>CookieServlet</servlet-name>
        <url-pattern>/CookieServlet</url-pattern>
    </servlet-mapping>
</web-app>
```

Run in Browser

Start Tomcat and open:

http://localhost:8080/ProjectName/CookieServlet

OUTPUT:



11) Write a JAVA Program to insert data into Student DATA BASE and retrieve info based on particular queries(For example update, delete, search etc…).

```
import java.util.ArrayList;
import java.util.Scanner;
class Student {
    String name;
    String usn;
    int marks;
    Student(String name, String usn, int marks) {
        this.name = name;
        this.usn = usn;
        this.marks = marks;
    }
    @Override
    public String toString() {
        return "Name: " + name + ", USN: " + usn + ", Marks: " + marks;
    }
}
public class StudentCRUD {
    private static ArrayList<Student> students = new ArrayList<>();
    private static Scanner scanner = new Scanner(System.in);
    public static void main(String[] args) {
        while (true) {
            System.out.println("\nStudent CRUD Application");
            System.out.println("1. Add Student");
            System.out.println("2. View All Students");
            System.out.println("3. Update Student");
            System.out.println("4. Delete Student");
            System.out.println("5. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // consume newline
            switch (choice) {
                case 1:
                    addStudent();
                    break;
                case 2:
                    viewStudents();
                    break;
                case 3:
                    updateStudent();
                    break;
                case 4:
                    deleteStudent();
                    break;
                case 5:
                    System.out.println("Exiting...");
                    scanner.close();
                    return;
                default:
                    System.out.println("Invalid choice! Try again.");
            }
        }
    }
    private static void addStudent() {
        System.out.print("Enter Name: ");
        String name = scanner.nextLine();
        System.out.print("Enter USN: ");
        String usn = scanner.nextLine();
```

```
        System.out.print("Enter Marks: ");
        int marks = scanner.nextInt();
        students.add(new Student(name, usn, marks));
        System.out.println("Student added successfully!");
    }
    private static void viewStudents() {
        if (students.isEmpty()) {
            System.out.println("No students found!");
            return;
        }
        System.out.println("\nList of Students:");
        for (int i = 0; i < students.size(); i++) {
            System.out.println((i + 1) + ". " + students.get(i));
        }
    }
    private static void updateStudent() {
        viewStudents();
        if (students.isEmpty()) return;
        System.out.print("Enter student number to update: ");
        int index = scanner.nextInt() - 1;
        scanner.nextLine(); // consume newline
        if (index < 0 || index >= students.size()) {
            System.out.println("Invalid student number!");
            return;
        }
        Student s = students.get(index);
        System.out.print("Enter new Name (current: " + s.name + "): ");
        String name = scanner.nextLine();
        System.out.print("Enter new USN (current: " + s.usn + "): ");
        String usn = scanner.nextLine();
        System.out.print("Enter new Marks (current: " + s.marks + "): ");
        int marks = scanner.nextInt();
        students.set(index, new Student(name, usn, marks));
        System.out.println("Student updated successfully!");
    }
    private static void deleteStudent() {
        viewStudents();
        if (students.isEmpty()) return;
        System.out.print("Enter student number to delete: ");
        int index = scanner.nextInt() - 1;
        if (index < 0 || index >= students.size()) {
            System.out.println("Invalid student number!");
            return;
        }
        students.remove(index);
        System.out.println("Student deleted successfully!");
    }
}}
```

OUTPUT:

```
Student CRUD Application
1. Add Student
2. View All Students
3. Update Student
4. Delete Student
5. Exit
Enter your choice:
```

User Input 1

```
Enter your choice: 1
Enter Name: Alice
Enter USN: 1DS21CS001
Enter Marks: 95
```

O\P

```
Student added successfully!
```

User Input 2

```
Enter your choice: 2
```

O\P

```
List of Students:
1. Name: Alice, USN: 1DS21CS001, Marks: 95
2. Name: Bob, USN: 1DS21CS002, Marks: 89
```

12) A program to design the Login page and validating the USER_ID and PASSWORD using JSP and DataBase.

1. login.jsp (Login Form)

Place this file in your WebContent folder.

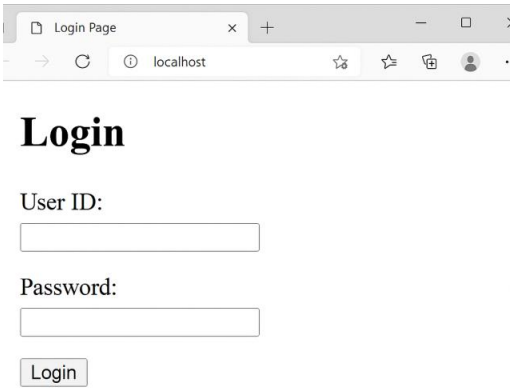
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
    <title>Login Page</title>
</head>
<body>
    <h2>Login</h2>
    <form action="checkLogin.jsp" method="post">
        User ID: <input type="text" name="userid" required><br><br>
        Password: <input type="password" name="password" required><br><br>
        <input type="submit" value="Login">
    </form>
</body>
</html>
```

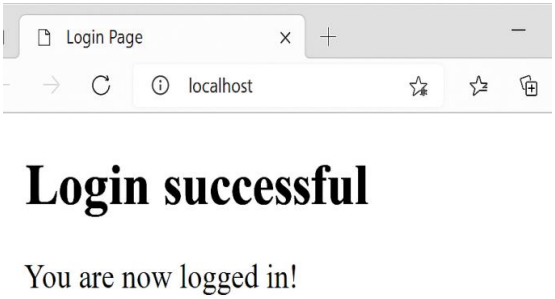
2. checkLogin.jsp (Validation Logic)

Place this file in your WebContent folder.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>
<html>
<head>
    <title>Login Check</title>
</head>
<body>
    <%
        String userid = request.getParameter("userid");
        String password = request.getParameter("password");

        // Hardcoded valid credentials (for demo only)
        if (userid.equals("admin") && password.equals("admin123")) {
    %>
            <h2>Welcome, <%= userid %>!</h2>
            <p>Login successful.</p>
    <%
        } else {
    %>
            <h2>Invalid User ID or Password!</h2>
            <a href="login.jsp">Try again</a>
    <%
        }
    %>
</body>
</html>
```

OUTPUT:





BEYOND THE SYLLABUS PROGRAMS

1) Write a Java program to print even and odd numbers using two separate threads.

```
class EvenThread extends Thread {
    public void run() {
        for(int i = 2; i <= 10; i += 2) {
            System.out.println("Even: " + i);
        }
    }
}

class OddThread extends Thread {
    public void run() {
        for(int i = 1; i <= 10; i += 2) {
            System.out.println("Odd: " + i);
        }
    }
}

public class ThreadDemo {
    public static void main(String[] args) {
        EvenThread t1 = new EvenThread();
        OddThread t2 = new OddThread();

        t1.start();
        t2.start();
    }
}
```

OUTPUT :

```
Even: 2
Odd: 1
Even: 4
Odd: 3
Even: 6
Odd: 5
Even: 8
Odd: 7
Even: 10
Odd: 9
```

2) Develop a servlet program to demonstrate session tracking using HttpSession.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SessionDemo extends HttpServlet {

    protected void doGet(HttpServletRequest request,
                         HttpServletResponse response)
                         throws ServletException, IOException {

        response.setContentType("text/html");

        HttpSession session = request.getSession();

        Integer count = (Integer) session.getAttribute("visitCount");

        if(count == null) {
            count = 1;
        } else {
            count++;
        }

        session.setAttribute("visitCount", count);

        PrintWriter out = response.getWriter();
        out.println("<h2>Visit Count: " + count + "</h2>");
    }
}
```

OUTPUT :

```
Visit Count: 1
Visit Count: 2
Visit Count: 3
```

VIVA

1. What is the Java Collections Framework?
Java Collections Framework is a unified architecture for storing and manipulating groups of objects using interfaces like List, Set, and Map.

2. What is an ArrayList?
ArrayList is a resizable array implementation of the List interface that allows dynamic storage of elements.

3. How is ArrayList different from an array?
Array size is fixed, whereas ArrayList grows and shrinks dynamically.

4. What happens internally when ArrayList capacity is exceeded?
A new larger array is created and existing elements are copied into it.

5. What is the difference between ArrayList and LinkedList?
ArrayList uses a dynamic array, LinkedList uses a doubly linked list. LinkedList is better for frequent insertions/deletions.

6. What is the purpose of Collections.sort()?
It sorts elements in ascending order using natural ordering or Comparator.

7. What is Comparator?
Comparator is an interface used to define custom sorting logic.

8. What is the difference between Comparable and Comparator?
Comparable provides natural ordering, Comparator provides custom ordering.

9. Can collections store user-defined objects?
Yes, collections store object references, including user-defined class objects.

10. Why do we override toString() in user-defined classes?
To print meaningful object information instead of memory reference.

11. What is String immutability?
Once a String object is created, its value cannot be changed.

12. Why are Strings immutable in Java?
For security, thread safety, and efficient memory usage.

13. What is the String Constant Pool?
A special memory area where string literals are stored to avoid duplication.

14. What is the difference between == and equals()?
== compares references; equals() compares content.

15. What is StringBuffer?
StringBuffer is a mutable and thread-safe string class.

16. What is the difference between StringBuffer and StringBuilder?
StringBuffer is synchronized (thread-safe), StringBuilder is not synchronized.

17. What is capacity in StringBuffer?
Capacity is the total allocated memory for storing characters.

18. What is length in StringBuffer?
Length is the actual number of characters stored.

19. What does trim() method do?
It removes leading and trailing spaces.

20. What is compareTo() method used for?
It compares two strings lexicographically.

21. What is Swing?
Swing is a Java GUI toolkit used to build platform-independent desktop applications.

22. What is JFrame?
JFrame is a top-level container used to create windows.

23. What is event handling in Java?
It is a mechanism to handle user actions like button clicks using listeners.

24. What is ActionListener?
It is an interface that handles action events like button clicks.

25. What is the Event Delegation Model?
It is a design pattern where event handling is delegated to listener objects.

26. What is a Servlet?
A Servlet is a server-side Java program that handles HTTP requests and responses.

27. Which package is used for servlet development?
javax.servlet and javax.servlet.http

28. What are the lifecycle methods of a servlet?
init(), service(), destroy()

29. What is the difference between doGet() and doPost()?
doGet() handles GET requests; doPost() handles POST requests.

30. How do you retrieve form data in a servlet?
Using request.getParameter() method.

31. What is web.xml?
Deployment descriptor used to configure servlets and URL mappings.

32. What is HttpServletRequest?
It is an object that represents client request information.

33. What is HttpServletResponse?
It is an object used to send response to the client.

34. What is a cookie?
A cookie is small data stored on the client browser.

35. How do you create a cookie?
Using Cookie class and response.addCookie().

36. How do you retrieve cookies?
Using request.getCookies() method.

37. What is HttpSession?
HttpSession is used to maintain user data across multiple requests.

38. Why is session tracking required?
Because HTTP protocol is stateless.

39. What is JDBC?
JDBC is an API that connects Java applications to databases.

40. What are the steps in JDBC?
Load driver → Establish connection → Create statement → Execute query → Close connection.

REFERENCES

Suggested Learning Resources	
Books	1. Y. Daniel Liang: Introduction to JAVA Programming, 7th Edition, Pearson Education, 2007.
	2. Stephanie Bodoff et al: The J2EE Tutorial, 2nd Edition, Pearson Education,2004.
	3. Uttam K Roy, Advanced JAVA programming, Oxford University press, 2015.
Web links and Video Lectures (e-Resources):	1. https://nptel.ac.in/courses/106/105/106105191/
	2. https://nptel.ac.in/courses/106/105/106105225/
	3. https://youtu.be/qGMxs-PbFPk