1)Write a java program to accept the empno from the user and update the salary of employee and display the updated Record on the screen. Employee table having fields empno, ename and salary.

✅ Accepts employee number from user
✅ Updates the salary of the employee
✅ Displays the updated record

It uses **JDBC** to connect to a database table called Employee with columns: empno, ename, salary.

✅ **Prerequisites:**

- MySQL database with a table Employee(empno INT, ename VARCHAR(50), salary DOUBLE)
- JDBC Driver for MySQL (usually mysql-connector-java.jar added to classpath)

💻 **Java Program (Terminal Ready)**

java

CopyEdit

```
import java.sql.*;

import java.util.Scanner;


public class UpdateEmployeeSalary {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);


        // Database details

        String url = "jdbc:mysql://localhost:3306/your_database_name";

        String user = "root";

        String password = "your_password";
```

```
try {
    // Load JDBC driver
    Class.forName("com.mysql.cj.jdbc.Driver");

    // Establish connection
    Connection con = DriverManager.getConnection(url, user, password);

    // Get employee number from user
    System.out.print("Enter Employee Number: ");
    int empno = scanner.nextInt();

    // Get new salary from user
    System.out.print("Enter New Salary: ");
    double newSalary = scanner.nextDouble();

    // Update query
    String updateQuery = "UPDATE Employee SET salary = ? WHERE empno = ?";
    PreparedStatement pstmt = con.prepareStatement(updateQuery);
    pstmt.setDouble(1, newSalary);
    pstmt.setInt(2, empno);

    int rowsUpdated = pstmt.executeUpdate();

    if (rowsUpdated > 0) {
        System.out.println("Salary updated successfully!");
```

```
        // Display updated record
        String selectQuery = "SELECT * FROM Employee WHERE empno = ?";
        PreparedStatement selectStmt = con.prepareStatement(selectQuery);
        selectStmt.setInt(1, empno);
        ResultSet rs = selectStmt.executeQuery();

        while (rs.next()) {
            System.out.println("Updated Employee Record:");
            System.out.println("Emp No: " + rs.getInt("empno"));
            System.out.println("Name  : " + rs.getString("ename"));
            System.out.println("Salary: " + rs.getDouble("salary"));
        }
        rs.close();
        selectStmt.close();
    } else {
        System.out.println("Employee not found!");
    }

    pstmt.close();
    con.close();
    scanner.close();

} catch (Exception e) {
    e.printStackTrace();
```

```
    }
  }
}
```

□ Sample Output

yaml

CopyEdit

Enter Employee Number: 102

Enter New Salary: 45000

Salary updated successfully!

Updated Employee Record:

Emp No: 102

Name : Arya

Salary: 45000.0

2) Write a servlet program using jdbc to display all the records from student table.(assume at least 5 suitable fields)

✅ **Prerequisites:**

- Apache Tomcat server (e.g., Tomcat 9 or 10) installed
- MySQL database with a Student table
- mysql-connector-java.jar in your project's lib folder
- Proper web.xml configuration (or use @WebServlet annotation)

🛠 **MySQL Table (for reference):**

sql

CopyEdit

```
CREATE TABLE Student (
   id INT PRIMARY KEY,
   name VARCHAR(50),
   age INT,
   course VARCHAR(50),
   marks DOUBLE
);


-- Insert sample data
INSERT INTO Student VALUES
(1, 'Arya', 21, 'MCA', 88.5),
(2, 'Rahul', 22, 'BCA', 76.2),
(3, 'Sneha', 20, 'BSc', 91.0),
(4, 'Ravi', 23, 'MSc', 85.3),
(5, 'Priya', 21, 'MCA', 78.9);
```

💻 Java Servlet Code (DisplayStudentServlet.java)

java

CopyEdit

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

```
public class DisplayStudentServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

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

        // Database details
        String url = "jdbc:mysql://localhost:3306/your_database_name";
        String user = "root";
        String password = "your_password";

        try {
            // Load JDBC Driver
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Connect to database
            Connection con = DriverManager.getConnection(url, user, password);

            // Query to get all student records
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM Student");

            out.println("<html><body>");
```

```
            out.println("<h2>Student Records</h2>");
            out.println("<table border='1'>");

out.println("<tr><th>ID</th><th>Name</th><th>Age</th><th>Course</th><th>Marks</th></tr>");

            while (rs.next()) {
                out.println("<tr>");
                out.println("<td>" + rs.getInt("id") + "</td>");
                out.println("<td>" + rs.getString("name") + "</td>");
                out.println("<td>" + rs.getInt("age") + "</td>");
                out.println("<td>" + rs.getString("course") + "</td>");
                out.println("<td>" + rs.getDouble("marks") + "</td>");
                out.println("</tr>");
            }

            out.println("</table>");
            out.println("</body></html>");

            rs.close();
            stmt.close();
            con.close();

        } catch (Exception e) {
            out.println("<p>Error: " + e.getMessage() + "</p>");
        }
    }
```

}

📄 **web.xml Entry (if not using annotation)**

xml

CopyEdit

```
<servlet>
   <servlet-name>DisplayStudentServlet</servlet-name>
   <servlet-class>DisplayStudentServlet</servlet-class>
</servlet>

<servlet-mapping>
   <servlet-name>DisplayStudentServlet</servlet-name>
   <url-pattern>/display-students</url-pattern>
</servlet-mapping>
```

3) : Write a servlet program illustrating its life cycle.(take counters for each method of life cycle).

🚀 **Servlet Life Cycle Methods:**

1. init() – Called once when the servlet is initialized
2. service() – Called every time the servlet is accessed
3. destroy() – Called when the servlet is being taken down

✅ **Servlet Program with Counters**

java

CopyEdit

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

public class LifeCycleServlet extends HttpServlet {

   // Counters for each life cycle method
   int initCounter = 0;
   int serviceCounter = 0;
   int destroyCounter = 0;

   @Override
   public void init() throws ServletException {
      initCounter++;
      System.out.println("init() called " + initCounter + " time(s)");
   }

   @Override
   protected void service(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {

      serviceCounter++;
      System.out.println("service() called " + serviceCounter + " time(s)");

      response.setContentType("text/html");
```

```
        PrintWriter out = response.getWriter();

        out.println("<html><body>");
        out.println("<h2>Servlet Life Cycle Counters</h2>");
        out.println("<p>init() called: " + initCounter + " time(s)</p>");
        out.println("<p>service() called: " + serviceCounter + " time(s)</p>");
        out.println("<p>destroy() called: " + destroyCounter + " time(s)</p>");
        out.println("</body></html>");
    }

    @Override
    public void destroy() {
        destroyCounter++;
        System.out.println("destroy() called " + destroyCounter + " time(s)");
    }
}
```

---

web.xml Entry (if not using annotation)

```
xml
CopyEdit
<servlet>
    <servlet-name>LifeCycleServlet</servlet-name>
    <servlet-class>LifeCycleServlet</servlet-class>
</servlet>
```

```
<servlet-mapping>
    <servlet-name>LifeCycleServlet</servlet-name>
    <url-pattern>/life</url-pattern>
</servlet-mapping>
```

🔍 Browser URL

bash

CopyEdit

http://localhost:8080/your_project_name/life

📝 Output Example (Browser)

scss

CopyEdit

Servlet Life Cycle Counters

init() called: 1 time(s)

service() called: 3 time(s)

destroy() called: 0 time(s)

📟 Output Example (Server Console)

scss

CopyEdit

init() called 1 time(s)

service() called 1 time(s)

service() called 2 time(s)

service() called 3 time(s)

4) Write a jdbc application to display the records of employee table.

Connects to a MySQL database

✅ Retrieves and displays **all records from the Employee table**

✅ **Employee Table Structure (for reference):**

sql

CopyEdit

```
CREATE TABLE Employee (
   empno INT PRIMARY KEY,
   ename VARCHAR(50),
   job VARCHAR(30),
   salary DOUBLE,
   deptno INT
);


-- Sample data
INSERT INTO Employee VALUES
(101, 'Arya', 'Developer', 55000, 10),
(102, 'Rahul', 'Manager', 65000, 20),
(103, 'Sneha', 'Analyst', 58000, 30);
```

💻 **JDBC Java Program (DisplayEmployee.java)**

java

CopyEdit

```
import java.sql.*;
```

```
public class DisplayEmployee {
    public static void main(String[] args) {
        // DB details
        String url = "jdbc:mysql://localhost:3306/your_database_name";
        String user = "root";
        String password = "your_password";

        try {
            // Load JDBC driver
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Connect to database
            Connection con = DriverManager.getConnection(url, user, password);

            // Create statement
            Statement stmt = con.createStatement();

            // Execute query
            ResultSet rs = stmt.executeQuery("SELECT * FROM Employee");

            System.out.println("Employee Records:");
            System.out.println("EmpNo | Name     | Job       | Salary   | DeptNo");
            System.out.println("-----------------------------------------------");

            // Display results
```

```
            while (rs.next()) {
                int empno = rs.getInt("empno");
                String ename = rs.getString("ename");
                String job = rs.getString("job");
                double salary = rs.getDouble("salary");
                int deptno = rs.getInt("deptno");

                System.out.printf("%5d | %-8s | %-9s | %8.2f | %6d\n", empno, ename, job, salary, deptno);
            }

            // Close connections
            rs.close();
            stmt.close();
            con.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

□ Sample Output (Terminal)

markdown

CopyEdit

Employee Records:

EmpNo | Name | Job | Salary | DeptNo

---

101 | Arya | Developer | 55000.00 | 10

102 | Rahul | Manager | 65000.00 | 20

103 | Sneha | Analyst | 58000.00 | 30

5) Write a JAVA program to accept the details of Employee (Eno , Ename, Sal) from the user, store it into the database and display that details on the screen .

✅ Accepts Employee details (eno, ename, sal) from the user
✅ Inserts the data into a MySQL database
✅ Displays the inserted data back on the screen

✅ **Step 1: MySQL Table Setup**

Run this SQL in your database first:

sql

CopyEdit

```
CREATE TABLE Employee (
    eno INT PRIMARY KEY,
    ename VARCHAR(50),
    sal DOUBLE
);
```

💻 **Java Program: InsertAndDisplayEmployee.java**

java

CopyEdit

```
import java.sql.*;
import java.util.Scanner;

public class InsertAndDisplayEmployee {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Database credentials
        String url = "jdbc:mysql://localhost:3306/your_database_name";
        String user = "root";
        String password = "your_password";

        try {
            // Load JDBC driver
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Connect to the database
            Connection con = DriverManager.getConnection(url, user, password);

            // Accept employee details
            System.out.print("Enter Employee Number (e.g., 1001): ");
            int eno = scanner.nextInt();
            scanner.nextLine();  // consume newline
```

```
System.out.print("Enter Employee Name (e.g., Harry): ");
String ename = scanner.nextLine();

System.out.print("Enter Employee Salary (e.g., 50000): ");
double sal = scanner.nextDouble();

// Insert into database
String insertQuery = "INSERT INTO Employee VALUES (?, ?, ?)";
PreparedStatement pstmt = con.prepareStatement(insertQuery);
pstmt.setInt(1, eno);
pstmt.setString(2, ename);
pstmt.setDouble(3, sal);

int rowsInserted = pstmt.executeUpdate();
if (rowsInserted > 0) {
   System.out.println("Employee inserted successfully!\n");
}

// Display inserted employee
String selectQuery = "SELECT * FROM Employee WHERE eno = ?";
PreparedStatement selectStmt = con.prepareStatement(selectQuery);
selectStmt.setInt(1, eno);
ResultSet rs = selectStmt.executeQuery();
```

```
            System.out.println("Inserted Employee Details:");
            System.out.println("Eno\tEname\t\tSalary");
            System.out.println("---------------------------------");
            while (rs.next()) {
               System.out.printf("%d\t%-10s\t%.2f\n",
                   rs.getInt("eno"),
                   rs.getString("ename"),
                   rs.getDouble("sal"));
            }

            // Close all resources
            rs.close();
            selectStmt.close();
            pstmt.close();
            con.close();
            scanner.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

□ Sample Output

yaml

CopyEdit

Enter Employee Number (e.g., 1001): 1002

Enter Employee Name (e.g., Harry): Hermione

Enter Employee Salary (e.g., 50000): 60000

Employee inserted successfully!

Inserted Employee Details:

Eno Ename Salary

1002 Hermione 60000.00

6) Accept table name (as Command Line Argument)(assume any table like emp, student etc.) and display number of columns along with name. (Hint:ResultSetMetaData)

✅ Accepts the **table name from the command line**
✅ Connects to a MySQL database
✅ Uses ResultSetMetaData to display the **number of columns** and their **names**

💻 **Java Program: TableColumnInfo.java**

java

CopyEdit

```
import java.sql.*;


public class TableColumnInfo {
    public static void main(String[] args) {
        if (args.length != 1) {
            System.out.println("Usage: java TableColumnInfo <table_name>");
```

```
    return;
}

String tableName = args[0];

// DB Credentials
String url = "jdbc:mysql://localhost:3306/your_database_name";
String user = "root";
String password = "your_password";

try {
    // Load JDBC driver
    Class.forName("com.mysql.cj.jdbc.Driver");

    // Connect to database
    Connection con = DriverManager.getConnection(url, user, password);

    // Execute dummy select to get metadata
    Statement stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery("SELECT * FROM " + tableName + " LIMIT 1");

    ResultSetMetaData rsmd = rs.getMetaData();
    int columnCount = rsmd.getColumnCount();

    // Display metadata info
```

```
            System.out.println("Table: " + tableName);
            System.out.println("Number of columns: " + columnCount);
            System.out.println("Column Names:");
            System.out.println("----------------------------");

            for (int i = 1; i <= columnCount; i++) {
                System.out.println(i + ". " + rsmd.getColumnName(i));
            }

            // Close connections
            rs.close();
            stmt.close();
            con.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

🏃‍♂️ How to Run in Terminal

bash

CopyEdit

javac TableColumnInfo.java

java TableColumnInfo Employee

🔁 Replace Employee with your table name like Student, Emp, etc.

□ Sample Output

markdown

CopyEdit

Table: Employee

Number of columns: 3

Column Names:

---

1. eno

2. ename

3. sal

7) Write a servlet displaying current date & time also wish the user accordingly (like a.m. “Good Morning”).

✅ Displays the **current date and time**
✅ Wishes the user with a greeting based on the time of day (e.g., **Good Morning**, **Good Afternoon**, etc.)

💻 **Servlet Code: TimeGreetingServlet.java**

java

CopyEdit

import java.io.*;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

import javax.servlet.*;

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

public class TimeGreetingServlet extends HttpServlet {

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

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

      // Get current date and time
      LocalDateTime now = LocalDateTime.now();
      DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");
      String currentTime = now.format(formatter);

      // Determine greeting
      int hour = now.getHour();
      String greeting;

      if (hour >= 5 && hour < 12) {
         greeting = "Good Morning, Harry!";
      } else if (hour >= 12 && hour < 17) {
         greeting = "Good Afternoon, Hermione!";
```

```
        } else if (hour >= 17 && hour < 21) {
            greeting = "Good Evening, Ron!";
        } else {
            greeting = "Good Night, Draco!";
        }

        // Output
        out.println("<html><body>");
        out.println("<h2>Current Date & Time</h2>");
        out.println("<p>" + currentTime + "</p>");
        out.println("<h3>" + greeting + "</h3>");
        out.println("</body></html>");
    }
}
```

---

web.xml Entry (if using web.xml)

xml

CopyEdit

```
<servlet>
    <servlet-name>TimeGreetingServlet</servlet-name>
    <servlet-class>TimeGreetingServlet</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>TimeGreetingServlet</servlet-name>
```

```
    <url-pattern>/greet</url-pattern>
</servlet-mapping>
```

□ Sample Output in Browser

sql

CopyEdit

Current Date & Time

10-04-2025 08:52:13

Good Morning, Harry!

8) Write a html code to accept the details of student (Rno , SName , Percentage) from the user and insert it into the table using jsp. (use PreparedStatement)

✅ An HTML form to accept **Student Details** (Rno, SName, Percentage)
✅ A **JSP file** that inserts the data into the database using **PreparedStatement**

🌐 **student_form.html – HTML Form**

html

CopyEdit

```
<!DOCTYPE html>
<html>
<head>
    <title>Student Entry</title>
</head>
<body>
    <h2>Enter Student Details</h2>
    <form action="insertStudent.jsp" method="post">
```

```
    Roll No: <input type="number" name="rno" required><br><br>
    Name: <input type="text" name="sname" required><br><br>
    Percentage: <input type="text" name="percentage" required><br><br>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

⚙ **insertStudent.jsp – JSP to Insert Data into DB**

jsp

CopyEdit

```
<%@ page import="java.sql.*" %>
<%
  int rno = Integer.parseInt(request.getParameter("rno"));
  String sname = request.getParameter("sname");
  double percentage = Double.parseDouble(request.getParameter("percentage"));

  String url = "jdbc:mysql://localhost:3306/your_database_name";
  String user = "root";
  String password = "your_password";

  Connection con = null;
  PreparedStatement pst = null;

  try {
```

```
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection(url, user, password);

        String query = "INSERT INTO Student (rno, sname, percentage) VALUES (?, ?, ?)";
        pst = con.prepareStatement(query);
        pst.setInt(1, rno);
        pst.setString(2, sname);
        pst.setDouble(3, percentage);

        int rows = pst.executeUpdate();

        if (rows > 0) {
%>
            <h3>Student <%= sname %> inserted successfully!</h3>
<%
        } else {
%>
            <h3>Insertion failed!</h3>
<%
        }
    } catch (Exception e) {
        out.println("Error: " + e);
    } finally {
        if (pst != null) pst.close();
        if (con != null) con.close();
```

```
    }
%>
```

🗃️ Sample SQL Table Creation

```
sql
CopyEdit
CREATE TABLE Student (
    rno INT PRIMARY KEY,
    sname VARCHAR(50),
    percentage DOUBLE
);
```

⌨️ Sample Input (via HTML Form)

```
yaml
CopyEdit
Roll No: 101
Name: Lucy
Percentage: 89.5
```

🖥️ Output on Browser (from JSP):

```
nginx
CopyEdit
Student Lucy inserted successfully!
```

9) Explain this() for constructor overloading and super() for superclass's constructor calling with example

🔁 this() – Constructor Overloading in Same Class

- Used to call **another constructor in the same class**.
- It must be the **first statement** in the constructor.

✅ **Example: this() – Constructor Overloading**

java

CopyEdit

```
class Wizard {
    String name;
    int age;

    // Constructor 1
    Wizard() {
        this("Harry", 17);  // calls Constructor 2
    }

    // Constructor 2
    Wizard(String name, int age) {
        this.name = name;
        this.age = age;
    }

    void display() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}
```

```
public class Main1 {
    public static void main(String[] args) {
        Wizard w = new Wizard();  // uses default constructor
        w.display();
    }
}
```

□ Output:

```
makefile
CopyEdit
Name: Harry
Age: 17
```

□ super() – Calling Parent Class Constructor

- Used to call the **constructor of the superclass**.
- Must be the **first statement** in the subclass constructor.

✅ Example: super() – Superclass Constructor Calling

```
java
CopyEdit
class Creature {
    String species;

    // Superclass constructor
    Creature(String species) {
        this.species = species;
```

```
        System.out.println("Creature constructor called.");
    }
}

class Elf extends Creature {
    String name;

    // Subclass constructor
    Elf(String name) {
        super("Elf");  // calls Creature(String species)
        this.name = name;
    }

    void show() {
        System.out.println("Name: " + name);
        System.out.println("Species: " + species);
    }
}

public class Main2 {
    public static void main(String[] args) {
        Elf e = new Elf("Legolas");
        e.show();
    }
}
```

□ Output:

makefile

CopyEdit

Creature constructor called.

Name: Legolas

Species: Elf

10) Write a JDBC program to create "Student_Master" table in MySQL having fields roll no, name, marks for 3 subjects. Calculate total, percentage for each student. Insert minimum 3 records. Display the marksheet for each student in proper format.

✅ Creates a Student_Master table
✅ Inserts at least 3 student records (with marks in 3 subjects)
✅ Calculates **total** and **percentage**
✅ Displays a formatted **marksheet** for each student

💻 **Full Java Program: StudentMarksheet.java**

java

CopyEdit

```
import java.sql.*;


public class StudentMarksheet {

    public static void main(String[] args) {

        String url = "jdbc:mysql://localhost:3306/your_database_name";

        String user = "root";

        String password = "your_password";
```

```
try {
    // Load MySQL JDBC driver
    Class.forName("com.mysql.cj.jdbc.Driver");

    // Connect to DB
    Connection con = DriverManager.getConnection(url, user, password);
    Statement stmt = con.createStatement();

    // 1. Create Table
    String createTable = "CREATE TABLE IF NOT EXISTS Student_Master (" +
        "roll_no INT PRIMARY KEY," +
        "name VARCHAR(50)," +
        "sub1 INT," +
        "sub2 INT," +
        "sub3 INT," +
        "total INT," +
        "percentage DOUBLE)";
    stmt.executeUpdate(createTable);

    // 2. Insert 3 Records
    String insertQuery = "INSERT INTO Student_Master (roll_no, name, sub1, sub2, sub3, total,
percentage) VALUES (?, ?, ?, ?, ?, ?, ?)";
    PreparedStatement pst = con.prepareStatement(insertQuery);

    // Student 1
    int s1 = 78, s2 = 82, s3 = 90;
```

```
int total = s1 + s2 + s3;
double percentage = total / 3.0;
pst.setInt(1, 1);
pst.setString(2, "Hermione");
pst.setInt(3, s1);
pst.setInt(4, s2);
pst.setInt(5, s3);
pst.setInt(6, total);
pst.setDouble(7, percentage);
pst.executeUpdate();

// Student 2
s1 = 65; s2 = 70; s3 = 75;
total = s1 + s2 + s3;
percentage = total / 3.0;
pst.setInt(1, 2);
pst.setString(2, "Ron");
pst.setInt(3, s1);
pst.setInt(4, s2);
pst.setInt(5, s3);
pst.setInt(6, total);
pst.setDouble(7, percentage);
pst.executeUpdate();

// Student 3
```

```
        s1 = 50; s2 = 60; s3 = 55;
        total = s1 + s2 + s3;
        percentage = total / 3.0;
        pst.setInt(1, 3);
        pst.setString(2, "Draco");
        pst.setInt(3, s1);
        pst.setInt(4, s2);
        pst.setInt(5, s3);
        pst.setInt(6, total);
        pst.setDouble(7, percentage);
        pst.executeUpdate();

        // 3. Display Marksheet
        ResultSet rs = stmt.executeQuery("SELECT * FROM Student_Master");

        System.out.println("----------------------------------------------------------");
        System.out.printf("%-5s %-10s %-6s %-6s %-6s %-6s %-10s\n", "RNo", "Name", "Sub1", "Sub2",
"Sub3", "Total", "Percentage");
        System.out.println("----------------------------------------------------------");

        while (rs.next()) {
           System.out.printf("%-5d %-10s %-6d %-6d %-6d %-6d %-10.2f\n",
                rs.getInt("roll_no"),
                rs.getString("name"),
                rs.getInt("sub1"),
                rs.getInt("sub2"),
```

```
                rs.getInt("sub3"),
                rs.getInt("total"),
                rs.getDouble("percentage"));
        }

        // Close everything
        rs.close();
        pst.close();
        stmt.close();
        con.close();

      } catch (Exception e) {
        e.printStackTrace();
      }
   }
}
```

📦 **What You Need Before Running:**

- ✅ MySQL running
- ✅ Database created (your_database_name)
- ✅ JDBC driver added to your classpath

□ Sample Output:

markdown

CopyEdit

RNo Name Sub1 Sub2 Sub3 Total Percentage

---

1 Hermione 78 82 90 250 83.33

2 Ron 65 70 75 210 70.00

3 Draco 50 60 55 165 55.00

11): Write Servlet application to check whether entered no. is palindrome no. or not.

HTML Page: palindrome.html

This page takes input from the user:

html

CopyEdit

```
<!DOCTYPE html>
<html>
<head>
  <title>Palindrome Checker</title>
</head>
<body>
  <h2>Enter a number to check if it's a Palindrome</h2>
  <form action="PalindromeServlet" method="post">
    Enter Number: <input type="text" name="number">
    <input type="submit" value="Check">
  </form>
</body>
</html>
```

Java Servlet: PalindromeServlet.java

java

CopyEdit

```
import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;


public class PalindromeServlet extends HttpServlet {

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

      response.setContentType("text/html");

      PrintWriter out = response.getWriter();


      int num = Integer.parseInt(request.getParameter("number"));

      int original = num;

      int reversed = 0;


      while (num != 0) {

         int digit = num % 10;

         reversed = reversed * 10 + digit;

         num = num / 10;

      }


      out.println("<html><body>");

      out.println("<h2>Palindrome Checker</h2>");
```

```
        if (original == reversed) {
            out.println("<p>The number <strong>" + original + "</strong> is a Palindrome.</p>");
        } else {
            out.println("<p>The number <strong>" + original + "</strong> is NOT a Palindrome.</p>");
        }

        out.println("</body></html>");
    }
}
```

🛠️ **Deployment Steps:**

1. Place palindrome.html in your webapp or public_html directory.
2. Place PalindromeServlet.java in the src folder.
3. Map servlet in web.xml or use @WebServlet("/PalindromeServlet").
4. Compile and deploy on Tomcat.

□ Sample Input/Output

Input: 121
Output: The number **121** is a Palindrome.

Input: 123
Output: The number **123** is NOT a Palindrome.

12): Write a JAVA program using servlet to count the no of times a servlet has been invoked.

✅ **Java Servlet: VisitCounterServlet.java**

java

CopyEdit

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

@WebServlet("/VisitCounterServlet")
public class VisitCounterServlet extends HttpServlet {
   private int count;

   @Override
   public void init() throws ServletException {
      count = 0;  // initialize counter
   }

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

      count++;  // increment on each call

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

```
        out.println("<html><body>");
        out.println("<h2>Servlet Visit Counter</h2>");
        out.println("<p>This servlet has been accessed <strong>" + count + "</strong> times.</p>");
        out.println("</body></html>");
    }

    @Override
    public void destroy() {
        // Any cleanup if needed
    }
}
```

🛠️ **How to Run:**

1. Use any **Java EE-supported server** (e.g., Apache Tomcat).
2. Deploy this servlet by placing it in your project source folder.
3. Use annotations (@WebServlet) or add mapping to web.xml (if needed).

🌐 **URL to Access:**

arduino

CopyEdit

http://localhost:8080/YourAppName/VisitCounterServlet

✅ **Sample Output:**

rust

CopyEdit

Servlet Visit Counter

This servlet has been accessed 1 times.

Next time:

rust

CopyEdit

Servlet Visit Counter

This servlet has been accessed 2 times.

13) Write a JSP and JDBC application for displaying list of scholarship holders students

Here's a complete **JSP + JDBC** application to display a **list of scholarship-holder students** from a database.

Assumptions:

We assume a **MySQL table** like this:

Table: students

roll_no	name	course	percentage	scholarship
1	Harry	BSc	85	Yes
2	Ron	BCom	72	No
3	Hermione	BSc	91	Yes

Step 1: Create the MySQL Table

sql

CopyEdit

```
CREATE TABLE students (
   roll_no INT PRIMARY KEY,
   name VARCHAR(50),
   course VARCHAR(50),
   percentage DOUBLE,
   scholarship VARCHAR(5)
);

-- Insert Sample Data
INSERT INTO students VALUES
(1, 'Harry', 'BSc', 85, 'Yes'),
(2, 'Ron', 'BCom', 72, 'No'),
(3, 'Hermione', 'BSc', 91, 'Yes');
```

✅ **Step 2: JSP File – scholarship.jsp**

```
jsp
CopyEdit
<%@ page import="java.sql.*" %>
<!DOCTYPE html>
<html>
<head>
   <title>Scholarship Holders</title>
</head>
<body>
   <h2>List of Scholarship Holder Students</h2>
```

```
<table border="1" cellpadding="10">
    <tr>
        <th>Roll No</th>
        <th>Name</th>
        <th>Course</th>
        <th>Percentage</th>
    </tr>

    <%
        String url = "jdbc:mysql://localhost:3306/your_database_name";
        String user = "root";
        String password = "your_password";

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection(url, user, password);

            String sql = "SELECT roll_no, name, course, percentage FROM students WHERE scholarship = 'Yes'";
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(sql);

            while (rs.next()) {
    %>
                <tr>
                    <td><%= rs.getInt("roll_no") %></td>
```

```
                <td><%= rs.getString("name") %></td>
                <td><%= rs.getString("course") %></td>
                <td><%= rs.getDouble("percentage") %></td>
            </tr>
        <%
            }

            rs.close();
            stmt.close();
            con.close();

        } catch (Exception e) {
            out.println("<p>Error: " + e.getMessage() + "</p>");
        }
    %>
  </table>
</body>
</html>
```

□ Output (on browser):

pgsql

CopyEdit

List of Scholarship Holder Students

| Roll No | Name | Course | Percentage |

|---------|----------|--------|------------|

| 1 | Harry | BSc | 85.0 |

| 3 | Hermione | BSc | 91.0 |

📌 **Notes:**

- Replace your_database_name, your_password accordingly.
- Make sure MySQL server is running.
- Deploy this JSP on a server like **Apache Tomcat**.

14) Write servlet application to insert record into teacher table and display record in tabular format assume suitable structure [write HTML, web.xml, servlet class]

Here's a complete **Servlet-based Java web application** to **insert and display teacher records** using **JDBC** and **MySQL**. It includes:

- HTML form to take input
- Servlet class for insertion and display
- web.xml configuration

🗃️ **Step 1: Database Table**

sql

CopyEdit

```
CREATE TABLE teacher (
    id INT PRIMARY KEY,
    name VARCHAR(50),
    subject VARCHAR(50),
    salary DOUBLE
```

);

📄 Step 2: HTML Form – teacher.html

html

CopyEdit

```
<!DOCTYPE html>
<html>
<head>
  <title>Teacher Entry Form</title>
</head>
<body>
  <h2>Enter Teacher Details</h2>
  <form action="TeacherServlet" method="post">
    ID: <input type="text" name="id"><br><br>
    Name: <input type="text" name="name"><br><br>
    Subject: <input type="text" name="subject"><br><br>
    Salary: <input type="text" name="salary"><br><br>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

☕ Step 3: Servlet – TeacherServlet.java

java

CopyEdit

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

@WebServlet("/TeacherServlet")
public class TeacherServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

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

        int id = Integer.parseInt(request.getParameter("id"));
        String name = request.getParameter("name");
        String subject = request.getParameter("subject");
        double salary = Double.parseDouble(request.getParameter("salary"));

        Connection conn = null;
        PreparedStatement pst = null;
        Statement stmt = null;
        ResultSet rs = null;

        try {
```

```
        Class.forName("com.mysql.cj.jdbc.Driver");
        conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/your_database", "root",
"your_password");

        // Insert record
        pst = conn.prepareStatement("INSERT INTO teacher VALUES (?, ?, ?, ?)");
        pst.setInt(1, id);
        pst.setString(2, name);
        pst.setString(3, subject);
        pst.setDouble(4, salary);
        pst.executeUpdate();

        // Display all records
        out.println("<h2>All Teacher Records</h2>");
        out.println("<table
border='1'><tr><th>ID</th><th>Name</th><th>Subject</th><th>Salary</th></tr>");

        stmt = conn.createStatement();
        rs = stmt.executeQuery("SELECT * FROM teacher");

        while (rs.next()) {
           out.println("<tr><td>" + rs.getInt("id") + "</td><td>" +
                rs.getString("name") + "</td><td>" +
                rs.getString("subject") + "</td><td>" +
                rs.getDouble("salary") + "</td></tr>");
        }
```

```
            out.println("</table>");
        } catch (Exception e) {
            out.println("<p>Error: " + e.getMessage() + "</p>");
        } finally {
            try { if (pst != null) pst.close(); } catch (Exception e) {}
            try { if (stmt != null) stmt.close(); } catch (Exception e) {}
            try { if (conn != null) conn.close(); } catch (Exception e) {}
        }
    }
}
```

⚙️ **Step 4: web.xml Configuration**

xml

CopyEdit

```
<web-app xmlns="http://jakarta.ee/xml/ns/jakartaee" version="5.0">
    <servlet>
        <servlet-name>TeacherServlet</servlet-name>
        <servlet-class>TeacherServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>TeacherServlet</servlet-name>
        <url-pattern>/TeacherServlet</url-pattern>
    </servlet-mapping>
```

</web-app>

✅ **Run Instructions:**

1. Create DB and table.
2. Place HTML in your project's web folder.
3. Compile and deploy TeacherServlet.java in src.
4. Update DB URL, username, and password.
5. Access teacher.html via your browser.

15) Write a program to implement stack using Array with PUSH, POP operations

Here's a **Java program** to implement a **Stack using an Array**, supporting basic operations like **PUSH** and **POP**, with user interaction via terminal.

✅ **StackUsingArray.java**

java

CopyEdit

```
import java.util.Scanner;


public class StackUsingArray {
    int top = -1;
    int maxSize;
    int[] stack;

    // Constructor
    StackUsingArray(int size) {
```

```
    maxSize = size;
    stack = new int[maxSize];
}

// Push operation
void push(int value) {
    if (top == maxSize - 1) {
        System.out.println("Stack Overflow! Cannot insert " + value);
    } else {
        stack[++top] = value;
        System.out.println("Pushed: " + value);
    }
}

// Pop operation
void pop() {
    if (top == -1) {
        System.out.println("Stack Underflow! No elements to pop.");
    } else {
        System.out.println("Popped: " + stack[top--]);
    }
}

// Display stack
void display() {
```

```
    if (top == -1) {
        System.out.println("Stack is empty.");
    } else {
        System.out.print("Stack elements: ");
        for (int i = 0; i <= top; i++) {
            System.out.print(stack[i] + " ");
        }
        System.out.println();
    }
}

// Main method
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    System.out.print("Enter size of stack: ");
    int size = sc.nextInt();

    StackUsingArray stack = new StackUsingArray(size);

    while (true) {
        System.out.println("\n1. PUSH\n2. POP\n3. DISPLAY\n4. EXIT");
        System.out.print("Enter your choice: ");
        int choice = sc.nextInt();
```

```
            switch (choice) {
                case 1:
                    System.out.print("Enter value to PUSH: ");
                    int val = sc.nextInt();
                    stack.push(val);
                    break;
                case 2:
                    stack.pop();
                    break;
                case 3:
                    stack.display();
                    break;
                case 4:
                    System.out.println("Exiting program.");
                    sc.close();
                    return;
                default:
                    System.out.println("Invalid choice. Try again.");
            }
        }
    }
}
```

□ Sample Output

yaml

CopyEdit

Enter size of stack: 3

1. PUSH

2. POP

3. DISPLAY

4. EXIT

Enter your choice: 1

Enter value to PUSH: 10

Pushed: 10

Enter your choice: 1

Enter value to PUSH: 20

Pushed: 20

Enter your choice: 3

Stack elements: 10 20

Enter your choice: 2

Popped: 20

Enter your choice: 4

Exiting program.

16): Write programs to explain at least 3 types of nested and inner classes

✅ 1. Member Inner Class

A non-static class declared inside another class.

java

CopyEdit

```
public class OuterClass {
    String outerName = "Outer - Harry";

    // Member Inner Class
    class InnerClass {
        void display() {
            System.out.println("Inside Inner Class. Accessing: " + outerName);
        }
    }

    public static void main(String[] args) {
        OuterClass outer = new OuterClass();
        OuterClass.InnerClass inner = outer.new InnerClass();
        inner.display();
    }
}
```

✅ 2. Static Nested Class

A static class declared inside another class (doesn't need outer object to be created).

java

CopyEdit

```
public class OuterStaticClass {
    static String msg = "Static Nested Class - Hermione";

    // Static Nested Class
    static class Nested {
        void print() {
            System.out.println("Message: " + msg);
        }
    }

    public static void main(String[] args) {
        OuterStaticClass.Nested obj = new OuterStaticClass.Nested();
        obj.print();
    }
}
```

✅ 3. Local Inner Class

A class declared inside a method.

java

CopyEdit

```
public class LocalInnerExample {
    void show() {
        String localVar = "Ron Weasley";
```

```
        // Local Inner Class
        class Local {
            void display() {
                System.out.println("Accessing Local Var: " + localVar);
            }
        }

        Local obj = new Local();
        obj.display();
    }

    public static void main(String[] args) {
        LocalInnerExample example = new LocalInnerExample();
        example.show();
    }
}
```

💡 **Bonus (Anonymous Inner Class – Just for fun!)**

Used when you want to implement a class without declaring it.

java

CopyEdit

```
abstract class Animal {
    abstract void sound();
}
```

```
public class AnonymousInner {
    public static void main(String[] args) {
        Animal dog = new Animal() {
            void sound() {
                System.out.println("Draco's dog barks!");
            }
        };
        dog.sound();
    }
}
```

17) Define a class Student (name, roll_no, class and marks of 6 subjects). Create an array of 5 Student objects. Calculate the percentage of each student using a method per().

This is a complete Java program that defines a Student class with the required details, creates an array of 5 students, calculates their percentage using a method per() and displays everything nicely.

✅ **StudentPercentage.java**

java

CopyEdit

```
import java.util.Scanner;

class Student {
    String name;
    int roll_no;
    String studentClass;
```

```
int[] marks = new int[6];


// Constructor
Student(String name, int roll_no, String studentClass, int[] marks) {
    this.name = name;
    this.roll_no = roll_no;
    this.studentClass = studentClass;
    this.marks = marks;
}


// Method to calculate percentage
double per() {
    int total = 0;
    for (int mark : marks) {
        total += mark;
    }
    return (total / 6.0);
}


// Method to display student info
void display() {
    System.out.println("Name      : " + name);
    System.out.println("Roll No   : " + roll_no);
    System.out.println("Class     : " + studentClass);
    System.out.print("Marks     : ");
```

```
        for (int m : marks) {
            System.out.print(m + " ");
        }
        System.out.printf("\nPercentage: %.2f%%\n", per());
        System.out.println("--------------------------");
    }
}

public class StudentPercentage {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Student[] students = new Student[5];

        System.out.println("Enter details for 5 students:\n");

        for (int i = 0; i < 5; i++) {
            System.out.println("Student " + (i + 1));
            System.out.print("Enter name: ");
            String name = sc.nextLine();

            System.out.print("Enter roll number: ");
            int roll_no = Integer.parseInt(sc.nextLine());

            System.out.print("Enter class: ");
            String studentClass = sc.nextLine();
```

```
            int[] marks = new int[6];
            for (int j = 0; j < 6; j++) {
                System.out.print("Enter mark for subject " + (j + 1) + ": ");
                marks[j] = Integer.parseInt(sc.nextLine());
            }

            students[i] = new Student(name, roll_no, studentClass, marks);
            System.out.println();
        }

        System.out.println("------ Student Results ------");
        for (Student s : students) {
            s.display();
        }

        sc.close();
    }
}
```

📌 **Sample Output**

yaml

CopyEdit

Enter details for 5 students:

Student 1

Enter name: Hermione

Enter roll number: 1

Enter class: 12A

Enter mark for subject 1: 90

Enter mark for subject 2: 88

Enter mark for subject 3: 92

Enter mark for subject 4: 85

Enter mark for subject 5: 94

Enter mark for subject 6: 89

... (input for other 4 students)

------ Student Results ------

Name : Hermione

Roll No : 1

Class : 12A

Marks : 90 88 92 85 94 89

Percentage: 89.67%

---

18): Define a class Staff with members' id, name, DOB, joining_date and salary. . Define class TeachingStaff with subjects[], experience and extends Staff. Using array of objects store details of teaching staff. Find senior staff member from the teaching staff (hint. Use joining_date).

Here's a **complete Java program** that:

✅ Defines a base class Staff
✅ Defines a derived class TeachingStaff that extends Staff
✅ Uses an array of TeachingStaff objects
✅ Compares joining_date to find the **senior-most** teaching staff

---

✅ **SeniorTeachingStaff.java**

java

CopyEdit

```
import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.*;


class Staff {

   int id;

   String name;

   String dob;

   Date joining_date;

   double salary;


   // Constructor

   Staff(int id, String name, String dob, String joiningDateStr, double salary) throws ParseException {

      this.id = id;

      this.name = name;

      this.dob = dob;

      SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
```

```
        this.joining_date = sdf.parse(joiningDateStr);
        this.salary = salary;
    }
}

class TeachingStaff extends Staff {
    String[] subjects;
    int experience;

    // Constructor
    TeachingStaff(int id, String name, String dob, String joiningDateStr, double salary, String[] subjects, int experience)
            throws ParseException {
        super(id, name, dob, joiningDateStr, salary);
        this.subjects = subjects;
        this.experience = experience;
    }

    // Display method
    void display() {
        System.out.println("Name: " + name);
        System.out.println("ID: " + id);
        System.out.println("DOB: " + dob);
        System.out.println("Joining Date: " + new SimpleDateFormat("dd-MM-yyyy").format(joining_date));
        System.out.println("Salary: ₹" + salary);
        System.out.println("Experience: " + experience + " years");
```

```
        System.out.print("Subjects: ");
        for (String subject : subjects) {
            System.out.print(subject + " ");
        }
        System.out.println("\n-------------------------");
    }
}

public class SeniorTeachingStaff {
    public static void main(String[] args) throws ParseException {
        Scanner sc = new Scanner(System.in);
        TeachingStaff[] ts = new TeachingStaff[3];

        System.out.println("Enter details for 3 Teaching Staff:\n");

        for (int i = 0; i < ts.length; i++) {
            System.out.println("Staff " + (i + 1));
            System.out.print("Enter ID: ");
            int id = Integer.parseInt(sc.nextLine());

            System.out.print("Enter Name: ");
            String name = sc.nextLine();

            System.out.print("Enter DOB (dd-mm-yyyy): ");
            String dob = sc.nextLine();
```

```
    System.out.print("Enter Joining Date (dd-mm-yyyy): ");
    String joining_date = sc.nextLine();

    System.out.print("Enter Salary: ");
    double salary = Double.parseDouble(sc.nextLine());

    System.out.print("Enter number of subjects: ");
    int n = Integer.parseInt(sc.nextLine());
    String[] subjects = new String[n];

    for (int j = 0; j < n; j++) {
        System.out.print("Enter subject " + (j + 1) + ": ");
        subjects[j] = sc.nextLine();
    }

    System.out.print("Enter Experience in years: ");
    int exp = Integer.parseInt(sc.nextLine());

    ts[i] = new TeachingStaff(id, name, dob, joining_date, salary, subjects, exp);
    System.out.println();
}

// Find senior-most staff
TeachingStaff senior = ts[0];
```

```
        for (int i = 1; i < ts.length; i++) {
            if (ts[i].joining_date.before(senior.joining_date)) {
                senior = ts[i];
            }
        }

        System.out.println("------ All Teaching Staff ------");
        for (TeachingStaff t : ts) {
            t.display();
        }

        System.out.println("\n🏆 Senior Most Staff Member:");
        senior.display();

        sc.close();
    }
}
```

📌 **Sample Output (if you use names like Harry, Hermione, Draco)**

mathematica

CopyEdit

Enter details for 3 Teaching Staff:

Staff 1

Enter ID: 101

Enter Name: Hermione

Enter DOB (dd-mm-yyyy): 12-03-1990

Enter Joining Date (dd-mm-yyyy): 15-06-2010

Enter Salary: 50000

Enter number of subjects: 2

Enter subject 1: Math

Enter subject 2: Physics

Enter Experience in years: 13

...

🏰 Senior Most Staff Member:

Name: Hermione

...

Joining Date: 15-06-2010

20): Write servlet application to insert record into teacher table and display record in tabular format assume suitable structure [write HTML, web.xml, servlet class]

Here's a complete **Servlet application** that:

✅ Accepts teacher details from an HTML form
✅ Inserts the record into the teacher table using JDBC
✅ Displays all teacher records in a table format

We'll use:

- ✅ index.html — HTML form for input
- ✅ TeacherServlet.java — Servlet for insert + display

- ✅ web.xml — Servlet configuration
- ✅ MySQL teacher table — You can create this manually in your database

---

🛢️ 1. teacher table (MySQL)

sql

CopyEdit

```
CREATE TABLE teacher (
   id INT PRIMARY KEY,
   name VARCHAR(50),
   subject VARCHAR(50),
   salary DOUBLE
);
```

🌐 2. index.html (form to accept teacher data)

html

CopyEdit

```
<!DOCTYPE html>
<html>
<head>
   <title>Teacher Form</title>
</head>
<body>
   <h2>Enter Teacher Details</h2>
   <form action="TeacherServlet" method="post">
      ID: <input type="text" name="id"><br><br>
```

```
    Name: <input type="text" name="name"><br><br>
    Subject: <input type="text" name="subject"><br><br>
    Salary: <input type="text" name="salary"><br><br>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

🫙 3. TeacherServlet.java (insert and display logic)

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

public class TeacherServlet extends HttpServlet {

  public void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();

    String id = req.getParameter("id");
    String name = req.getParameter("name");
    String subject = req.getParameter("subject");
```

```
        String salary = req.getParameter("salary");

        try {
            // Load JDBC Driver
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Connect to DB
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/yourDBname",
"yourUsername", "yourPassword");

            // Insert teacher
            PreparedStatement ps = con.prepareStatement("INSERT INTO teacher VALUES (?, ?, ?, ?)");
            ps.setInt(1, Integer.parseInt(id));
            ps.setString(2, name);
            ps.setString(3, subject);
            ps.setDouble(4, Double.parseDouble(salary));
            ps.executeUpdate();

            // Display all records
            Statement st = con.createStatement();
            ResultSet rs = st.executeQuery("SELECT * FROM teacher");

            out.println("<h2>All Teacher Records</h2>");
            out.println("<table border='1'>");
            out.println("<tr><th>ID</th><th>Name</th><th>Subject</th><th>Salary</th></tr>");
```

```
            while (rs.next()) {
                out.println("<tr>");
                out.println("<td>" + rs.getInt(1) + "</td>");
                out.println("<td>" + rs.getString(2) + "</td>");
                out.println("<td>" + rs.getString(3) + "</td>");
                out.println("<td>" + rs.getDouble(4) + "</td>");
                out.println("</tr>");
            }
            out.println("</table>");

            con.close();

        } catch (Exception e) {
            out.println("Error: " + e.getMessage());
        }
    }
}
```

🛠 4. web.xml (deployment descriptor)

xml

CopyEdit

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee" version="3.0">
    <servlet>
        <servlet-name>TeacherServlet</servlet-name>
        <servlet-class>TeacherServlet</servlet-class>
```

```
    </servlet>

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

⚙️ **JDBC Setup Reminder:**

- Add **MySQL JDBC driver** (like mysql-connector-java.jar) to WEB-INF/lib or classpath.
- Update "yourDBname", "yourUsername", "yourPassword" in the servlet code.

21) Write jsp application to insert record into teacher table and display record in tabular format assume suitable structure [write HTML and jsp]

Here's a **complete JSP application** that:

✅ Accepts teacher details using an HTML form
✅ Inserts the data into a teacher table using JDBC
✅ Displays all teacher records in a table format

🗃️ **Database Table: teacher**

You can create this in MySQL:

sql

CopyEdit

```
CREATE TABLE teacher (
    id INT PRIMARY KEY,
```

```
    name VARCHAR(50),
    subject VARCHAR(50),
    salary DOUBLE
);
```

📝 1. teacher_form.html – HTML Form

```
html
CopyEdit
<!DOCTYPE html>
<html>
<head>
    <title>Teacher Form</title>
</head>
<body>
    <h2>Enter Teacher Details</h2>
    <form action="teacher.jsp" method="post">
        ID: <input type="text" name="id"><br><br>
        Name: <input type="text" name="name"><br><br>
        Subject: <input type="text" name="subject"><br><br>
        Salary: <input type="text" name="salary"><br><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

💻 2. teacher.jsp – Insert and Display Logic

jsp

CopyEdit

```
<%@ page import="java.sql.*" %>
<%@ page import="java.util.*" %>

<%
   String id = request.getParameter("id");
   String name = request.getParameter("name");
   String subject = request.getParameter("subject");
   String salary = request.getParameter("salary");

   Connection con = null;
   PreparedStatement pst = null;
   Statement st = null;
   ResultSet rs = null;

   try {
      Class.forName("com.mysql.cj.jdbc.Driver");
      con = DriverManager.getConnection("jdbc:mysql://localhost:3306/yourDBname", "yourUsername",
"yourPassword");

      if (id != null && name != null && subject != null && salary != null) {
         pst = con.prepareStatement("INSERT INTO teacher VALUES (?, ?, ?, ?)");
         pst.setInt(1, Integer.parseInt(id));
         pst.setString(2, name);
```

```
        pst.setString(3, subject);
        pst.setDouble(4, Double.parseDouble(salary));
        pst.executeUpdate();
    }

    out.println("<h2>Teacher Records</h2>");
    out.println("<table border='1'>");
    out.println("<tr><th>ID</th><th>Name</th><th>Subject</th><th>Salary</th></tr>");

    st = con.createStatement();
    rs = st.executeQuery("SELECT * FROM teacher");

    while (rs.next()) {
        out.println("<tr>");
        out.println("<td>" + rs.getInt("id") + "</td>");
        out.println("<td>" + rs.getString("name") + "</td>");
        out.println("<td>" + rs.getString("subject") + "</td>");
        out.println("<td>" + rs.getDouble("salary") + "</td>");
        out.println("</tr>");
    }

    out.println("</table>");

} catch (Exception e) {
    out.println("Error: " + e.getMessage());
```

```
    } finally {
        try { if (rs != null) rs.close(); } catch (Exception e) {}
        try { if (pst != null) pst.close(); } catch (Exception e) {}
        try { if (st != null) st.close(); } catch (Exception e) {}
        try { if (con != null) con.close(); } catch (Exception e) {}
    }
%>
```

✅ **What You Need to Do:**

- Place teacher_form.html in the root of your web project.
- Place teacher.jsp in the same directory or appropriate path.
- Add MySQL JDBC Driver (like mysql-connector-java.jar) to WEB-INF/lib.
- Update the yourDBname, yourUsername, yourPassword in the JSP file.

22): Define a class Student (name, roll_no, class and marks of 6 subjects). Create an array of 5 Student objects. Calculate the percentage of each student using a method per().

Here's a **complete Java program** that:

✅ Defines a Student class with name, roll_no, className, and marks[] (for 6 subjects)
✅ Creates an array of 5 student objects
✅ Calculates and displays the percentage of each student using a method per()

✅ **Java Program**

java

CopyEdit

```
import java.util.Scanner;
```

```
class Student {
    String name;
    int roll_no;
    String className;
    int[] marks = new int[6];

    // Constructor to initialize student
    Student(String name, int roll_no, String className, int[] marks) {
        this.name = name;
        this.roll_no = roll_no;
        this.className = className;
        this.marks = marks;
    }

    // Method to calculate percentage
    double per() {
        int total = 0;
        for (int i = 0; i < 6; i++) {
            total += marks[i];
        }
        return (total / 6.0);
    }

    // Display student details
```

```
    void display() {
        System.out.println("Name: " + name);
        System.out.println("Roll No: " + roll_no);
        System.out.println("Class: " + className);
        System.out.print("Marks: ");
        for (int m : marks) {
            System.out.print(m + " ");
        }
        System.out.println("\nPercentage: " + per() + "%");
        System.out.println("------------------------------");
    }
}

public class StudentApp {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Student[] students = new Student[5];

        System.out.println("Enter details of 5 students:");

        for (int i = 0; i < 5; i++) {
            System.out.println("\nStudent " + (i + 1) + ":");
            System.out.print("Enter name: ");
            String name = sc.nextLine();
```

```
            System.out.print("Enter roll number: ");
            int roll_no = Integer.parseInt(sc.nextLine());

            System.out.print("Enter class: ");
            String className = sc.nextLine();

            int[] marks = new int[6];
            System.out.println("Enter marks for 6 subjects:");
            for (int j = 0; j < 6; j++) {
                System.out.print("Subject " + (j + 1) + ": ");
                marks[j] = Integer.parseInt(sc.nextLine());
            }

            students[i] = new Student(name, roll_no, className, marks);
        }

        System.out.println("\n--- Student Details ---");
        for (int i = 0; i < 5; i++) {
            students[i].display();
        }

        sc.close();
    }
}
```

💡 **How to Run:**

1. Save as StudentApp.java
2. Compile:

bash

CopyEdit

javac StudentApp.java

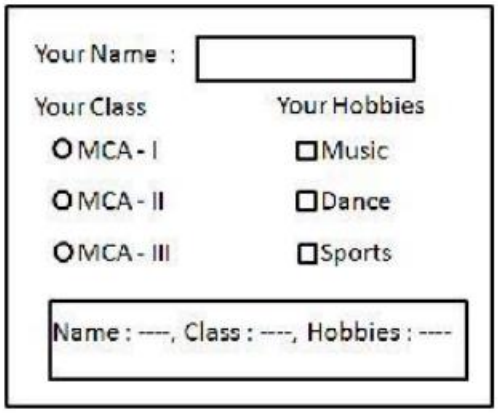
3. Run:

bash

CopyEdit

java StudentApp

23): Write a java program to create the following GUI screen using appropriate layout managers using AWT components



Here's a Java program using AWT (Abstract Window Toolkit) to create the GUI layout shown in your image. The program uses FlowLayout, GridLayout, and Panels to organize the components.

✅ **Java AWT GUI Program**

java

CopyEdit

import java.awt.*;

```
import java.awt.event.*;

public class StudentInfoForm extends Frame implements ActionListener {
   TextField nameField;
   Checkbox music, dance, sports;
   CheckboxGroup classGroup;
   Label result;

   public StudentInfoForm() {
      setTitle("Student Info");
      setSize(400, 300);
      setLayout(new FlowLayout());

      // Name section
      Panel namePanel = new Panel(new FlowLayout(FlowLayout.LEFT));
      namePanel.add(new Label("Your Name :"));
      nameField = new TextField(20);
      namePanel.add(nameField);

      // Class radio buttons
      Panel classPanel = new Panel(new GridLayout(3, 1));
      classPanel.add(new Label("Your Class"));
      classGroup = new CheckboxGroup();
      classPanel.add(new Checkbox("MCA - I", classGroup, false));
      classPanel.add(new Checkbox("MCA - II", classGroup, false));
```

```
classPanel.add(new Checkbox("MCA - III", classGroup, false));

// Hobbies checkboxes
Panel hobbiesPanel = new Panel(new GridLayout(3, 1));
hobbiesPanel.add(new Label("Your Hobbies"));
music = new Checkbox("Music");
dance = new Checkbox("Dance");
sports = new Checkbox("Sports");
hobbiesPanel.add(music);
hobbiesPanel.add(dance);
hobbiesPanel.add(sports);

// Combine class and hobbies panels side by side
Panel centerPanel = new Panel(new GridLayout(1, 2));
centerPanel.add(classPanel);
centerPanel.add(hobbiesPanel);

// Result label
result = new Label("Name: ----, Class: ----, Hobbies: ----");

// Button to display result
Button showBtn = new Button("Show");
showBtn.addActionListener(this);

// Add all panels to frame
```

```
        add(namePanel);
        add(centerPanel);
        add(showBtn);
        add(result);

        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        String name = nameField.getText();
        String cls = classGroup.getSelectedCheckbox() != null ?
classGroup.getSelectedCheckbox().getLabel() : "----";

        String hobbies = "";
        if (music.getState()) hobbies += "Music ";
        if (dance.getState()) hobbies += "Dance ";
        if (sports.getState()) hobbies += "Sports ";
        if (hobbies.equals("")) hobbies = "----";

        result.setText("Name: " + name + ", Class: " + cls + ", Hobbies: " + hobbies);
    }

    public static void main(String[] args) {
        new StudentInfoForm();
    }
}
```

How to Run

1. **Save the file as: StudentInfoForm.java**
2. **Compile: javac StudentInfoForm.java**
3. **Run: java StudentInfoForm**

⚠ **Note: This program uses AWT, so it may need permission to open a window if you're using it on an IDE like VS Code or from the terminal.**

24) Write JSP code to accept registration details for placement portal from candidate and insert the details in database (assume suitable table structure).

Here's a **complete JSP application** to:

✅ Accept **registration details** from a candidate for a **placement portal**
✅ Insert those details into a MySQL **database** using JDBC
✅ Display a confirmation message after successful registration

---

Database Table: placement_candidates

sql

CopyEdit

```
CREATE TABLE placement_candidates (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    email VARCHAR(100),
    phone VARCHAR(15),
    qualification VARCHAR(50),
    passing_year INT
);
```

📝 1. register.html – HTML Form to Accept Candidate Details

html

CopyEdit

```
<!DOCTYPE html>
<html>
<head>
  <title>Placement Registration</title>
</head>
<body>
  <h2>Candidate Registration Form</h2>
  <form action="register.jsp" method="post">
    Name: <input type="text" name="name" required><br><br>
    Email: <input type="email" name="email" required><br><br>
    Phone: <input type="text" name="phone" required><br><br>
    Qualification: <input type="text" name="qualification" required><br><br>
    Passing Year: <input type="number" name="year" required><br><br>
    <input type="submit" value="Register">
  </form>
</body>
</html>
```

💻 2. register.jsp – JSP Page to Insert Data in DB

jsp

CopyEdit

```
<%@ page import="java.sql.*" %>

<%
    String name = request.getParameter("name");
    String email = request.getParameter("email");
    String phone = request.getParameter("phone");
    String qualification = request.getParameter("qualification");
    String year = request.getParameter("year");

    Connection con = null;
    PreparedStatement pst = null;

    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/yourDBname", "yourUsername",
"yourPassword");

        pst = con.prepareStatement("INSERT INTO placement_candidates (name, email, phone,
qualification, passing_year) VALUES (?, ?, ?, ?, ?)");
        pst.setString(1, name);
        pst.setString(2, email);
        pst.setString(3, phone);
        pst.setString(4, qualification);
        pst.setInt(5, Integer.parseInt(year));

        int rows = pst.executeUpdate();
```

```
        if (rows > 0) {
            out.println("<h3>Registration Successful!</h3>");
            out.println("<p>Thank you, " + name + ". Your details have been saved.</p>");
        } else {
            out.println("<h3>Registration Failed!</h3>");
        }
    } catch (Exception e) {
        out.println("Error: " + e.getMessage());
    } finally {
        try { if (pst != null) pst.close(); } catch (Exception e) {}
        try { if (con != null) con.close(); } catch (Exception e) {}
    }
%>
```

✅ **Setup Notes:**

- Place register.html in your root folder (or as your homepage).
- Place register.jsp in the same or /jsp/ folder.
- Replace yourDBname, yourUsername, yourPassword with your MySQL credentials.
- Add MySQL JDBC driver (mysql-connector-java.jar) to your WEB-INF/lib.

25) Execution of two or more threads occurs in a random order. The keyword 'synchronized' in Java is used to control the execution of thread in a strict sequence. In the following, the program is expected to print some numbers. Do the necessary use of 'synchronized' keyword, so that, the program prints the output in the following order: ----------OUTPUT---------- 5 10 15 20 25 100 200 300 400 500

We will use a **shared synchronized method** so threads are executed one after another in a strict sequence.

✅ **Final Java Code**

java

CopyEdit

```
class NumberPrinter {
    synchronized void printTable(int num) {
        for (int i = 1; i <= 5; i++) {
            System.out.print(num * i + " ");
            try {
                Thread.sleep(100); // Small delay to simulate real-time thread behavior
            } catch (Exception e) {
                System.out.println(e);
            }
        }
        System.out.println();
    }
}

class Thread1 extends Thread {
    NumberPrinter p;

    Thread1(NumberPrinter p) {
        this.p = p;
    }
```

```
    public void run() {
        p.printTable(5);
    }
}

class Thread2 extends Thread {
    NumberPrinter p;

    Thread2(NumberPrinter p) {
        this.p = p;
    }

    public void run() {
        p.printTable(100);
    }
}

public class SyncThreadDemo {
    public static void main(String[] args) {
        NumberPrinter obj = new NumberPrinter();

        Thread1 t1 = new Thread1(obj);
        Thread2 t2 = new Thread2(obj);
```

```
        t1.start();
        try {
            t1.join(); // Ensures Thread1 completes before starting Thread2
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        t2.start();
    }
}
```

🔍 **Explanation:**

- printTable(int num) is synchronized, so only **one thread at a time** can execute it.
- join() ensures that Thread 1 finishes **before** Thread 2 starts.
- Output is always:

CopyEdit

5 10 15 20 25

100 200 300 400 500

26): Write JSP application to insert record into student table & display record in tabular format assume suitable table structure. write JDBC application to register students for convocation in MCA (Use PreparedStatement, assume suitable table structure).

✅ **Part 1: Insert and Display Student Records (via JSP)**

Table Name: student
Assumed Columns: id, name, email, course, year

□ 1. Create MySQL Table

sql

CopyEdit

```
CREATE TABLE student (
   id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(100),
   email VARCHAR(100),
   course VARCHAR(50),
   year INT
);
```

📝 2. HTML Form – student_form.html

html

CopyEdit

```
<!DOCTYPE html>
<html>
<head><title>Student Registration</title></head>
<body>
   <h2>Register Student</h2>
   <form action="insert_student.jsp" method="post">
      Name: <input type="text" name="name"><br><br>
      Email: <input type="email" name="email"><br><br>
      Course: <input type="text" name="course"><br><br>
      Year: <input type="number" name="year"><br><br>
```

```
    <input type="submit" value="Register">
  </form>
  <br>
  <a href="display_students.jsp">View All Students</a>
</body>
</html>
```

💻 3. JSP File – insert_student.jsp

jsp

CopyEdit

```
<%@ page import="java.sql.*" %>
<%
  String name = request.getParameter("name");
  String email = request.getParameter("email");
  String course = request.getParameter("course");
  int year = Integer.parseInt(request.getParameter("year"));

  Connection con = null;
  PreparedStatement pst = null;

  try {
    Class.forName("com.mysql.cj.jdbc.Driver");
    con = DriverManager.getConnection("jdbc:mysql://localhost:3306/yourDB", "yourUser", "yourPass");
    pst = con.prepareStatement("INSERT INTO student (name, email, course, year) VALUES (?, ?, ?, ?)");
```

```
        pst.setString(1, name);
        pst.setString(2, email);
        pst.setString(3, course);
        pst.setInt(4, year);
        int rows = pst.executeUpdate();

        if (rows > 0) {
            out.println("<h3>Student Registered Successfully!</h3>");
        }
    } catch (Exception e) {
        out.println("Error: " + e.getMessage());
    } finally {
        if (pst != null) pst.close();
        if (con != null) con.close();
    }
%>
```

📋 4. JSP File – display_students.jsp

jsp

CopyEdit

```
<%@ page import="java.sql.*" %>
<%
    Connection con = null;
    Statement stmt = null;
    ResultSet rs = null;
```

```
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/yourDB", "yourUser",
"yourPass");
        stmt = con.createStatement();
        rs = stmt.executeQuery("SELECT * FROM student");


        out.println("<h2>Student Records</h2>");
        out.println("<table
border='1'><tr><th>ID</th><th>Name</th><th>Email</th><th>Course</th><th>Year</th></tr>");
        while (rs.next()) {
            out.println("<tr><td>" + rs.getInt("id") + "</td><td>" +
                    rs.getString("name") + "</td><td>" +
                    rs.getString("email") + "</td><td>" +
                    rs.getString("course") + "</td><td>" +
                    rs.getInt("year") + "</td></tr>");
        }
        out.println("</table>");
    } catch (Exception e) {
        out.println("Error: " + e.getMessage());
    } finally {
        if (rs != null) rs.close();
        if (stmt != null) stmt.close();
        if (con != null) con.close();
    }
```

%>

✅ Part 2: JDBC Application to Register Students for MCA Convocation

Table: mca_convocation
Fields: roll_no, name, email, mobile, graduation_year

□ 1. MySQL Table

sql

CopyEdit

```
CREATE TABLE mca_convocation (
   roll_no VARCHAR(20) PRIMARY KEY,
   name VARCHAR(100),
   email VARCHAR(100),
   mobile VARCHAR(15),
   graduation_year INT
);
```

💻 2. Java Program – ConvocationRegistration.java

java

CopyEdit

```
import java.sql.*;
import java.util.Scanner;

public class ConvocationRegistration {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
```

```
        System.out.println("---- MCA Convocation Registration ----");
        System.out.print("Enter Roll No: ");
        String roll_no = sc.nextLine();

        System.out.print("Enter Name: ");
        String name = sc.nextLine();

        System.out.print("Enter Email: ");
        String email = sc.nextLine();

        System.out.print("Enter Mobile: ");
        String mobile = sc.nextLine();

        System.out.print("Enter Graduation Year: ");
        int gradYear = sc.nextInt();

        Connection con = null;
        PreparedStatement pst = null;

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/yourDB", "yourUser",
"yourPass");
```

```
        String query = "INSERT INTO mca_convocation (roll_no, name, email, mobile, graduation_year)
VALUES (?, ?, ?, ?, ?)";

        pst = con.prepareStatement(query);



        pst.setString(1, roll_no);

        pst.setString(2, name);

        pst.setString(3, email);

        pst.setString(4, mobile);

        pst.setInt(5, gradYear);



        int rows = pst.executeUpdate();



        if (rows > 0) {

           System.out.println("✅ Registration Successful!");

        }

      } catch (Exception e) {

        System.out.println("❌ Error: " + e.getMessage());

      } finally {

        try { if (pst != null) pst.close(); } catch (Exception e) {}

        try { if (con != null) con.close(); } catch (Exception e) {}

        sc.close();

      }

   }

}
```

27) : Define Employee class (name, designation, salary). Define a default and parameterized constructor. Override the toString method. Keep a count of objects created. Create objects using parameterized constructor and display the object count after each object is created. (Use static member and method). Also display the contents of each object.

✅ **Features Covered:**

- Employee class with name, designation, salary
- Default and parameterized constructors
- Overridden toString() method
- Static counter to track number of objects
- Static method to display object count
- Display content of each object after creation

💻 **Full Java Code:**

java

CopyEdit

```
class Employee {
    String name, designation;
    double salary;

    static int count = 0;

    // Default constructor
    Employee() {
        name = "Unknown";
        designation = "Not Assigned";
        salary = 0.0;
```

```
        count++;
    }

    // Parameterized constructor
    Employee(String name, String designation, double salary) {
        this.name = name;
        this.designation = designation;
        this.salary = salary;
        count++;
    }

    // Static method to show object count
    static void showCount() {
        System.out.println("Total Employees created: " + count);
    }

    // Overriding toString method
    public String toString() {
        return "Name: " + name + ", Designation: " + designation + ", Salary: ₹" + salary;
    }
}

public class Main {
    public static void main(String[] args) {
        Employee e1 = new Employee("Harry", "Manager", 55000);
```

```
        System.out.println(e1);
        Employee.showCount();
        System.out.println();

        Employee e2 = new Employee("Hermione", "Developer", 48000);
        System.out.println(e2);
        Employee.showCount();
        System.out.println();

        Employee e3 = new Employee("Ron", "Tester", 42000);
        System.out.println(e3);
        Employee.showCount();
    }
}
```

□ Sample Output:

yaml

CopyEdit

Name: Harry, Designation: Manager, Salary: ₹55000.0

Total Employees created: 1

Name: Hermione, Designation: Developer, Salary: ₹48000.0

Total Employees created: 2

Name: Ron, Designation: Tester, Salary: ₹42000.0

Total Employees created: 3

28): Write a JSP program to accept patient details from HTML and display patient details in proper format to update current details.

Here's a **complete JSP + HTML program** to:

✅ Accept **Patient Details** via HTML
✅ Display submitted data in **proper format**
✅ Allow user to **update current details**

🏥 **Assumed Patient Fields:**

- Patient ID
- Name
- Age
- Gender
- Disease
- Contact Number

📄 **1. patient_form.html – HTML Form**

html

CopyEdit

```
<!DOCTYPE html>
<html>
<head>
  <title>Patient Registration</title>
</head>
```

```
<body>
  <h2>Enter Patient Details</h2>
  <form action="display_patient.jsp" method="post">
    Patient ID: <input type="text" name="id"><br><br>
    Name: <input type="text" name="name"><br><br>
    Age: <input type="number" name="age"><br><br>
    Gender:
    <select name="gender">
      <option>Male</option>
      <option>Female</option>
      <option>Other</option>
    </select><br><br>
    Disease: <input type="text" name="disease"><br><br>
    Contact Number: <input type="text" name="contact"><br><br>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

💻 2. display_patient.jsp – JSP to Display & Update

```
jsp
CopyEdit
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%
  String id = request.getParameter("id");
```

```
    String name = request.getParameter("name");
    String age = request.getParameter("age");
    String gender = request.getParameter("gender");
    String disease = request.getParameter("disease");
    String contact = request.getParameter("contact");
%>
<!DOCTYPE html>
<html>
<head>
    <title>Patient Details</title>
</head>
<body>
    <h2>Patient Details</h2>
    <table border="1" cellpadding="8">
        <tr><th>Field</th><th>Value</th></tr>
        <tr><td>Patient ID</td><td><%= id %></td></tr>
        <tr><td>Name</td><td><%= name %></td></tr>
        <tr><td>Age</td><td><%= age %></td></tr>
        <tr><td>Gender</td><td><%= gender %></td></tr>
        <tr><td>Disease</td><td><%= disease %></td></tr>
        <tr><td>Contact</td><td><%= contact %></td></tr>
    </table>

    <h3>Update Patient Details</h3>
    <form action="display_patient.jsp" method="post">
```

```
    <input type="hidden" name="id" value="<%= id %>">
    Name: <input type="text" name="name" value="<%= name %>"><br><br>
    Age: <input type="number" name="age" value="<%= age %>"><br><br>
    Gender:
    <select name="gender">
      <option <%= gender.equals("Male") ? "selected" : "" %>>Male</option>
      <option <%= gender.equals("Female") ? "selected" : "" %>>Female</option>
      <option <%= gender.equals("Other") ? "selected" : "" %>>Other</option>
    </select><br><br>
    Disease: <input type="text" name="disease" value="<%= disease %>"><br><br>
    Contact Number: <input type="text" name="contact" value="<%= contact %>"><br><br>
    <input type="submit" value="Update Details">
  </form>
</body>
</html>
```

✅ **Features:**

- User can **submit new details**
- **Table** displays patient info
- **Form below** lets user update existing info (re-posts to same page)

29) Create a class called sports_accessories with attributes Accessory_id, description, quantity, rate, used_in_game. Accept details of 10 accessories from user (5 records), store it in array of objects. Display details of all accessories used in game cricket. Define a class Staff with members' id, name, DOB, joining_date and salary. . Define class TeachingStaff with subjects[], experience and extends

Staff. Using array of objects store details of teaching staff. Find senior staff member from the teaching staff (hint. Use joining_date).

Here's a **complete Java program** that includes:

✅ **Part 1:**

- sports_accessories class
- Accepts 5 records (for 10, you can increase later)
- Stores in an array of objects
- Displays only those **used in "cricket"**

✅ **Part 2:**

- Staff base class
- TeachingStaff class extends Staff
- Accepts details using array of objects
- Finds and displays **senior most staff** (by joining date)

💻 **Complete Java Code (Run this in terminal)**

java

CopyEdit

```
import java.util.*;


class SportsAccessory {
    int accessoryId;
    String description;
    int quantity;
    double rate;
    String usedInGame;
```

```
    SportsAccessory(int accessoryId, String description, int quantity, double rate, String usedInGame) {
        this.accessoryId = accessoryId;
        this.description = description;
        this.quantity = quantity;
        this.rate = rate;
        this.usedInGame = usedInGame;
    }

    void display() {
        System.out.println("ID: " + accessoryId + ", Desc: " + description + ", Qty: " + quantity +
                ", Rate: " + rate + ", Used In: " + usedInGame);
    }
}

class Staff {
    int id;
    String name;
    String dob;
    String joiningDate;
    double salary;

    Staff(int id, String name, String dob, String joiningDate, double salary) {
        this.id = id;
        this.name = name;
        this.dob = dob;
```

```
        this.joiningDate = joiningDate;
        this.salary = salary;
    }
}

class TeachingStaff extends Staff {
    String[] subjects;
    int experience;

    TeachingStaff(int id, String name, String dob, String joiningDate, double salary, String[] subjects, int experience) {
        super(id, name, dob, joiningDate, salary);
        this.subjects = subjects;
        this.experience = experience;
    }

    void display() {
        System.out.print("ID: " + id + ", Name: " + name + ", Joining: " + joiningDate + ", Salary: " + salary + ", Subjects: ");
        for (String sub : subjects)
            System.out.print(sub + " ");
        System.out.println(", Exp: " + experience + " yrs");
    }
}

public class Main {
```

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    // Part 1: Sports Accessories
    SportsAccessory[] accessories = new SportsAccessory[5];
    System.out.println("\nEnter 5 Sports Accessories Details:");
    for (int i = 0; i < 5; i++) {
        System.out.println("Accessory " + (i + 1));
        System.out.print("ID: "); int id = sc.nextInt(); sc.nextLine();
        System.out.print("Description: "); String desc = sc.nextLine();
        System.out.print("Quantity: "); int qty = sc.nextInt();
        System.out.print("Rate: "); double rate = sc.nextDouble(); sc.nextLine();
        System.out.print("Used in game: "); String game = sc.nextLine();

        accessories[i] = new SportsAccessory(id, desc, qty, rate, game);
    }

    System.out.println("\nAccessories used in game 'cricket':");
    for (SportsAccessory a : accessories) {
        if (a.usedInGame.equalsIgnoreCase("cricket")) {
            a.display();
        }
    }

    // Part 2: Teaching Staff
```

```
TeachingStaff[] staffList = new TeachingStaff[3];
System.out.println("\nEnter 3 Teaching Staff Details:");
for (int i = 0; i < 3; i++) {
   System.out.println("Staff " + (i + 1));
   System.out.print("ID: "); int id = sc.nextInt(); sc.nextLine();
   System.out.print("Name: "); String name = sc.nextLine();
   System.out.print("DOB (dd-mm-yyyy): "); String dob = sc.nextLine();
   System.out.print("Joining Date (yyyy-mm-dd): "); String join = sc.nextLine();
   System.out.print("Salary: "); double salary = sc.nextDouble(); sc.nextLine();
   System.out.print("Experience (in years): "); int exp = sc.nextInt(); sc.nextLine();
   System.out.print("Enter subjects (comma separated): ");
   String[] subs = sc.nextLine().split(",");

   staffList[i] = new TeachingStaff(id, name, dob, join, salary, subs, exp);
}

// Finding senior-most staff (earliest joining date)
TeachingStaff senior = staffList[0];
for (int i = 1; i < staffList.length; i++) {
   if (staffList[i].joiningDate.compareTo(senior.joiningDate) < 0) {
      senior = staffList[i];
   }
}

System.out.println("\nSenior Most Teaching Staff:");
```

```
        senior.display();
    }
}
```

✅ **How to Compile and Run:**

bash

CopyEdit

javac Main.java

java Main

30) Write a program to accept senior citizens name and age from command prompt. If age is below 60, throw "InvalidAgeException" exception.

Here's a simple Java program that accepts **senior citizen details (name and age)** from the **command line** and throws a custom InvalidAgeException if the age is below 60.

✅ **Java Program: SeniorCitizenCheck.java**

java

CopyEdit

```
// Custom Exception
class InvalidAgeException extends Exception {
    InvalidAgeException(String message) {
        super(message);
    }
}
```

```
public class SeniorCitizenCheck {
    public static void main(String[] args) {
        try {
            // Accept name and age from command-line
            if (args.length < 2) {
                System.out.println("Please provide name and age as command line arguments.");
                return;
            }

            String name = args[0];
            int age = Integer.parseInt(args[1]);

            if (age < 60) {
                throw new InvalidAgeException("Age is below 60. Not a senior citizen.");
            }

            System.out.println("Senior Citizen Name: " + name);
            System.out.println("Age: " + age + " (Valid)");

        } catch (InvalidAgeException e) {
            System.out.println("Exception: " + e.getMessage());
        } catch (NumberFormatException e) {
            System.out.println("Please enter a valid age (number).");
        }
    }
```

}

🔁 **How to Compile & Run from Terminal:**

bash

CopyEdit

javac SeniorCitizenCheck.java

java SeniorCitizenCheck Hermione 58

□ Sample Output:

bash

CopyEdit

Exception: Age is below 60. Not a senior citizen.

bash

CopyEdit

java SeniorCitizenCheck Aslan 70

Senior Citizen Name: Aslan

Age: 70 (Valid)

31) Write a java program to demonstrate at least 4 parameterized constructors for Employee class. It should use this() constructor and write appropriate methods to display the details of employee object.

Here's a complete Java program that demonstrates **at least 4 parameterized constructors** in the Employee class using **constructor overloading** and **this()** to avoid repetition. It also includes a method to display employee details.

✅ **Java Program: EmployeeDemo.java**

java

CopyEdit

```
class Employee {
    String name;
    int id;
    String designation;
    double salary;

    // Constructor 1: Only name
    Employee(String name) {
        this(name, 0, "Not Assigned", 0.0);
    }

    // Constructor 2: name and id
    Employee(String name, int id) {
        this(name, id, "Not Assigned", 0.0);
    }

    // Constructor 3: name, id, designation
    Employee(String name, int id, String designation) {
        this(name, id, designation, 0.0);
    }

    // Constructor 4: All parameters
    Employee(String name, int id, String designation, double salary) {
        this.name = name;
```

```
        this.id = id;
        this.designation = designation;
        this.salary = salary;
    }

    // Method to display employee details
    void display() {
        System.out.println("Name: " + name);
        System.out.println("ID: " + id);
        System.out.println("Designation: " + designation);
        System.out.println("Salary: ₹" + salary);
        System.out.println("--------------------------");
    }
}

public class EmployeeDemo {
    public static void main(String[] args) {
        // Creating Employee objects with different constructors
        Employee e1 = new Employee("Harry");
        Employee e2 = new Employee("Hermione", 101);
        Employee e3 = new Employee("Ron", 102, "Developer");
        Employee e4 = new Employee("Draco", 103, "Manager", 75000);

        // Displaying details
        System.out.println("Employee Details:\n");
```

```
        e1.display();
        e2.display();
        e3.display();
        e4.display();
    }
}
```

□ Sample Output:

bash

CopyEdit

Employee Details:

Name: Harry

ID: 0

Designation: Not Assigned

Salary: ₹0.0

Name: Hermione

ID: 101

Designation: Not Assigned

Salary: ₹0.0

Name: Ron

ID: 102

Designation: Developer

Salary: ₹0.0

Name: Draco

ID: 103

Designation: Manager

Salary: ₹75000.0

---

32): Write a program to demonstrate the HashMap, ArrayList class using collection framework. Use at least 5 methods for each class.

Here's a complete and **runnable Java program** that demonstrates both **HashMap** and **ArrayList** using the **Java Collection Framework**, and uses **at least 5 methods** for each class.

✅ **Java Program: CollectionDemo.java**

java

CopyEdit

```
import java.util.*;


public class CollectionDemo {

    public static void main(String[] args) {


        // ---------------------------

        // 1. ArrayList Demonstration

        // ---------------------------

        System.out.println("🔷 ArrayList Demo:");
```

```
ArrayList<String> list = new ArrayList<>();


// 5+ Useful Methods
list.add("Harry");              // add()
list.add("Hermione");
list.add("Ron");
list.add("Draco");


list.add(2, "Luna");            // add(index, element)
System.out.println("List: " + list);


list.remove("Draco");           // remove(Object)
System.out.println("After removing Draco: " + list);


System.out.println("Element at index 1: " + list.get(1));  // get(index)
System.out.println("List contains Ron? " + list.contains("Ron")); // contains()
System.out.println("List size: " + list.size());          // size()


System.out.println("\nIterating with for-each:");
for (String name : list) {
   System.out.println(name);
}


// ---------------------------
// 2. HashMap Demonstration
```

```
        // ---------------------------
        System.out.println("\n🔷 HashMap Demo:");
        HashMap<Integer, String> map = new HashMap<>();

        // 5+ Useful Methods
        map.put(1, "Aslan");            // put(key, value)
        map.put(2, "Susan");
        map.put(3, "Lucy");
        map.put(4, "Caspian");

        System.out.println("Map: " + map);

        map.remove(4);                  // remove(key)
        System.out.println("After removing key 4: " + map);

        System.out.println("Get value for key 2: " + map.get(2)); // get(key)
        System.out.println("Contains key 3? " + map.containsKey(3)); // containsKey()
        System.out.println("Contains value 'Lucy'? " + map.containsValue("Lucy")); // containsValue()

        System.out.println("\nIterating with entrySet:");
        for (Map.Entry<Integer, String> entry : map.entrySet()) {
            System.out.println("Key: " + entry.getKey() + ", Value: " + entry.getValue());
        }
    }
}
```

□ Sample Output:

bash

CopyEdit

◈ ArrayList Demo:

List: [Harry, Hermione, Luna, Ron, Draco]

After removing Draco: [Harry, Hermione, Luna, Ron]

Element at index 1: Hermione

List contains Ron? true

List size: 4

Iterating with for-each:

Harry

Hermione

Luna

Ron

◈ HashMap Demo:

Map: {1=Aslan, 2=Susan, 3=Lucy, 4=Caspian}

After removing key 4: {1=Aslan, 2=Susan, 3=Lucy}

Get value for key 2: Susan

Contains key 3? true

Contains value 'Lucy'? true

Iterating with entrySet:

Key: 1, Value: Aslan

Key: 2, Value: Susan

Key: 3, Value: Lucy

33) Write a program to demonstrate the Set, Vector class using collection framework. Use at least 5 methods for each class.

Here's a complete Java program that demonstrates both **Set** (using HashSet) and **Vector** classes from the **Java Collection Framework**, using **at least 5 methods** for each class. It's a runnable program with output shown as well.

✅ **Java Program: SetVectorDemo.java**

java

CopyEdit

```
import java.util.*;


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

        // ----------------------------
        // 1. Set Demonstration (HashSet)
        // ----------------------------
        System.out.println("🔷 HashSet Demo:");
        Set<String> games = new HashSet<>();

        // 5+ Useful Methods
```

```
games.add("Cricket");              // add()
games.add("Football");
games.add("Hockey");
games.add("Basketball");
games.add("Chess");

System.out.println("Games: " + games);
games.remove("Chess");             // remove()
System.out.println("After removing Chess: " + games);

System.out.println("Set contains Football? " + games.contains("Football")); // contains()
System.out.println("Set size: " + games.size());                        // size()
System.out.println("Is set empty? " + games.isEmpty());                 // isEmpty()

System.out.println("Iterating using for-each:");
for (String game : games) {
   System.out.println(game);
}

// ----------------------------
// 2. Vector Demonstration
// ----------------------------
System.out.println("\n Vector Demo:");
Vector<String> heroes = new Vector<>();
```

```
        // 5+ Useful Methods
        heroes.add("Harry");               // add()
        heroes.add("Hermione");
        heroes.add("Ron");
        heroes.add("Draco");

        heroes.insertElementAt("Aslan", 0);  // insertElementAt()
        heroes.removeElement("Draco");       // removeElement()
        heroes.set(2, "Caspian");            // set(index, element)

        System.out.println("Heroes: " + heroes);
        System.out.println("First Element: " + heroes.firstElement()); // firstElement()
        System.out.println("Last Element: " + heroes.lastElement());   // lastElement()

        System.out.println("Iterating using Enumeration:");
        Enumeration<String> e = heroes.elements();
        while (e.hasMoreElements()) {
           System.out.println(e.nextElement());
        }
    }
}
```

□ Sample Output:

bash

CopyEdit

🔷 HashSet Demo:

Games: [Cricket, Football, Basketball, Hockey, Chess]

After removing Chess: [Cricket, Football, Basketball, Hockey]

Set contains Football? true

Set size: 4

Is set empty? false

Iterating using for-each:

Cricket

Football

Basketball

Hockey

🔷 Vector Demo:

Heroes: [Aslan, Harry, Caspian, Ron]

First Element: Aslan

Last Element: Ron

Iterating using Enumeration:

Aslan

Harry

Caspian

Ron

34) Write an application to define a user defined exception “Insufficient Fund Exception”. Read the amount from console and if amount is available in your account, then draw the amount. If amount is not available, throw “Insufficient Fund Exception” and display amount available for withdrawal.

Here's a **complete Java application** that demonstrates a **user-defined exception** called InsufficientFundException. It reads the **amount from the console**, checks the balance, and either allows withdrawal or throws the exception if funds are insufficient.

✅ **Java Code: BankApp.java**

java

CopyEdit

```
import java.util.Scanner;


// Custom Exception
class InsufficientFundException extends Exception {
    private double availableAmount;


    public InsufficientFundException(double amount) {
        super("Insufficient Funds. Available amount: " + amount);
        this.availableAmount = amount;
    }


    public double getAvailableAmount() {
        return availableAmount;
    }
}


// Main Application
public class BankApp {
    public static void main(String[] args) {
```

```
        Scanner sc = new Scanner(System.in);
        double balance = 5000.00; // Assume current account balance

        System.out.print("Enter amount to withdraw: ");
        double amount = sc.nextDouble();

        try {
            if (amount <= balance) {
                balance -= amount;
                System.out.println("✅ Withdrawal successful!");
                System.out.println("Remaining Balance: ₹" + balance);
            } else {
                throw new InsufficientFundException(balance);
            }
        } catch (InsufficientFundException e) {
            System.out.println("❌ " + e.getMessage());
        }

        sc.close();
    }
}
```

🔹 Sample Output 1 (Sufficient Funds):

yaml

CopyEdit

Enter amount to withdraw: 2000

✅ Withdrawal successful!

Remaining Balance: ₹3000.0

□ Sample Output 2 (Insufficient Funds):

yaml

CopyEdit

Enter amount to withdraw: 8000

❌ Insufficient Funds. Available amount: 5000.0

35) Write a program to create a class MyThread, in this class a constructor calls the base class constructor, using super() constructor and starts the thread. The run method of the class starts after this. It can be observed that both main thread and created child thread are executed concurrently

Here's a complete Java program that demonstrates the use of super() to call the base class constructor and then starts a **thread** using a custom MyThread class. It also shows how the **main thread and the child thread run concurrently.**

✅ **Java Code: ThreadDemo.java**

java

CopyEdit

```
// Base class
class BaseThread extends Thread {
    public BaseThread() {
        super();  // Calls Thread class constructor
        System.out.println("🔷 BaseThread Constructor called");
    }
}
```

```
// Derived class
class MyThread extends BaseThread {
    public MyThread() {
        super(); // Calling base class constructor
        System.out.println("🔷 MyThread Constructor called");
        start(); // Start the thread
    }

    @Override
    public void run() {
        // Code to be executed by the new thread
        for (int i = 1; i <= 5; i++) {
            System.out.println("👶 Child Thread: " + i);
            try {
                Thread.sleep(500); // Just to see concurrent effect
            } catch (InterruptedException e) {
                System.out.println("Child thread interrupted");
            }
        }
    }
}

// Main class
public class ThreadDemo {
```

```
    public static void main(String[] args) {
        System.out.println("◈ Main Thread Started");

        // Creating MyThread object automatically starts the child thread
        MyThread t1 = new MyThread();

        // Main thread work
        for (int i = 1; i <= 5; i++) {
            System.out.println("□ Main Thread: " + i);
            try {
                Thread.sleep(500); // To simulate concurrency
            } catch (InterruptedException e) {
                System.out.println("Main thread interrupted");
            }
        }

        System.out.println("◈ Main Thread Ends");
    }
}
```

□ Sample Output (will vary due to thread scheduling):

mathematica

CopyEdit

◈ Main Thread Started

◈ BaseThread Constructor called

🔷 MyThread Constructor called

👶 Child Thread: 1

🧠 Main Thread: 1

👶 Child Thread: 2

🧠 Main Thread: 2

👶 Child Thread: 3

🧠 Main Thread: 3

👶 Child Thread: 4

🧠 Main Thread: 4

👶 Child Thread: 5

🧠 Main Thread: 5

🔷 Main Thread Ends

36) Write JDBC application using html and jsp that will accept author name & list the books of given author. Assume suitable table structure.

Here's a **complete working JDBC application** using **HTML** + **JSP** to accept an **author's name** and display all the books written by that author.

✅ **Assumed Table Structure: books**

Column Name	Data Type
book_id	INT (PK)
title	VARCHAR(100)
author	VARCHAR(100)

Column Name	Data Type
price	DOUBLE

---

1. HTML Form (file: authorForm.html)

```
html
CopyEdit
<!DOCTYPE html>
<html>
<head>
  <title>Author Book Lookup</title>
</head>
<body>
  <h2>Enter Author Name</h2>
  <form action="listBooks.jsp" method="post">
    Author Name: <input type="text" name="author" required />
    <br><br>
    <input type="submit" value="Show Books">
  </form>
</body>
</html>
```

---

2. JSP Page (file: listBooks.jsp)

```
jsp
CopyEdit
<%@ page import="java.sql.*" %>
```

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
   <title>Books by Author</title>
</head>
<body>
   <h2>Books by Author</h2>
   <%
      String author = request.getParameter("author");
      Connection conn = null;
      PreparedStatement ps = null;
      ResultSet rs = null;

      try {
         Class.forName("com.mysql.cj.jdbc.Driver");
         conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/your_database_name",
"root", "your_password");

         String sql = "SELECT * FROM books WHERE author = ?";
         ps = conn.prepareStatement(sql);
         ps.setString(1, author);
         rs = ps.executeQuery();

         boolean found = false;
   %>
         <table border="1" cellpadding="10">
```

```
            <tr>
                <th>Book ID</th>
                <th>Title</th>
                <th>Author</th>
                <th>Price</th>
            </tr>
<%
        while (rs.next()) {
            found = true;
%>
            <tr>
                <td><%= rs.getInt("book_id") %></td>
                <td><%= rs.getString("title") %></td>
                <td><%= rs.getString("author") %></td>
                <td><%= rs.getDouble("price") %></td>
            </tr>
<%
        }

        if (!found) {
%>
            <tr><td colspan="4">No books found for author: <%= author %></td></tr>
<%
        }
    } catch (Exception e) {
```

```
            out.println("Error: " + e.getMessage());
        } finally {
            try { if (rs != null) rs.close(); } catch (Exception e) {}
            try { if (ps != null) ps.close(); } catch (Exception e) {}
            try { if (conn != null) conn.close(); } catch (Exception e) {}
        }
    %>
            </table>
</body>
</html>
```

⚙️ **Make Sure:**

- Your MySQL DB has a database like your_database_name.
- Inside that DB, there is a books table with the mentioned structure.
- You change the DB credentials ("root" and "your_password").
- Add the MySQL JDBC driver to your project.

37) Write a Java program using HashMap to store and display employee IDs (as keys) and their names (as values). Then update the name of an employee given the ID. And Create a HashMap to count the frequency of each word in a given string. Print the word along with its frequency.

Here's a complete Java program that performs **both tasks** using **HashMap**:

✅ **Java Program: HashMapExample.java**

java

CopyEdit

```
import java.util.HashMap;
import java.util.Scanner;

public class HashMapExample {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // -------------------------
        // 1. Employee ID & Name Map
        // -------------------------
        HashMap<Integer, String> employeeMap = new HashMap<>();

        // Add some employees
        employeeMap.put(101, "Harry");
        employeeMap.put(102, "Hermione");
        employeeMap.put(103, "Ron");
        employeeMap.put(104, "Draco");

        System.out.println("🔷 Initial Employee List:");
        for (Integer id : employeeMap.keySet()) {
            System.out.println("ID: " + id + ", Name: " + employeeMap.get(id));
        }

        // Update employee name
```

```
System.out.print("\nEnter employee ID to update name: ");
int updateId = sc.nextInt();
sc.nextLine(); // clear buffer

if (employeeMap.containsKey(updateId)) {
    System.out.print("Enter new name: ");
    String newName = sc.nextLine();
    employeeMap.put(updateId, newName);
    System.out.println("✅ Name updated successfully!");
} else {
    System.out.println("❌ Employee ID not found.");
}

// Display updated employee list
System.out.println("\n🔷 Updated Employee List:");
for (Integer id : employeeMap.keySet()) {
    System.out.println("ID: " + id + ", Name: " + employeeMap.get(id));
}

// -------------------------
// 2. Word Frequency Counter
// -------------------------
System.out.print("\nEnter a sentence: ");
String sentence = sc.nextLine();
String[] words = sentence.toLowerCase().split("\\s+"); // split by spaces
```

```
        HashMap<String, Integer> wordCount = new HashMap<>();

        for (String word : words) {
            word = word.replaceAll("[^a-z]", ""); // remove punctuation
            if (word.isEmpty()) continue;
            wordCount.put(word, wordCount.getOrDefault(word, 0) + 1);
        }

        System.out.println("\n🔷 Word Frequency:");
        for (String word : wordCount.keySet()) {
            System.out.println(word + " → " + wordCount.get(word));
        }

        sc.close();
    }
}
```

□ Sample Output:

yaml

CopyEdit

🔷 Initial Employee List:

ID: 101, Name: Harry

ID: 102, Name: Hermione

ID: 103, Name: Ron

ID: 104, Name: Draco

Enter employee ID to update name: 103

Enter new name: Edmund

✅ Name updated successfully!

🔷 Updated Employee List:

ID: 101, Name: Harry

ID: 102, Name: Hermione

ID: 103, Name: Edmund

ID: 104, Name: Draco

Enter a sentence: Harry and Hermione went to Hogwarts with Ron and Draco.

🔷 Word Frequency:

harry → 1

and → 2

hermione → 1

went → 1

to → 1

hogwarts → 1

with → 1

ron → 1

draco → 1

38): Develop a Java program using threads to simulate a banking system, where: • One thread deposits money into an account • Another thread withdraws money from the account Use synchronization to prevent race conditions.

Here's a **complete Java program** that simulates a simple **banking system** using **threads** for deposit and withdrawal. It uses **synchronization** to prevent race conditions.

✅ **Java Program: BankSystem.java**

java

CopyEdit

```
class BankAccount {
    private int balance = 1000; // Initial balance

    // Synchronized deposit method
    public synchronized void deposit(int amount) {
        System.out.println("Depositing ₹" + amount + "...");
        balance += amount;
        System.out.println("✅ Deposit complete. Current Balance: ₹" + balance);
    }

    // Synchronized withdraw method
    public synchronized void withdraw(int amount) {
        System.out.println("Attempting to withdraw ₹" + amount + "...");
        if (balance >= amount) {
            balance -= amount;
            System.out.println("✅ Withdrawal successful. Remaining Balance: ₹" + balance);
```

```
        } else {
            System.out.println("❌ Insufficient funds! Current Balance: ₹" + balance);
        }
    }

    public int getBalance() {
        return balance;
    }
}

class DepositThread extends Thread {
    private BankAccount account;
    private int amount;

    public DepositThread(BankAccount acc, int amount) {
        this.account = acc;
        this.amount = amount;
    }

    public void run() {
        account.deposit(amount);
    }
}

class WithdrawThread extends Thread {
```

```
    private BankAccount account;
    private int amount;

    public WithdrawThread(BankAccount acc, int amount) {
        this.account = acc;
        this.amount = amount;
    }

    public void run() {
        account.withdraw(amount);
    }
}

public class BankSystem {
    public static void main(String[] args) {
        BankAccount account = new BankAccount();

        // Create deposit and withdraw threads
        DepositThread t1 = new DepositThread(account, 500);
        WithdrawThread t2 = new WithdrawThread(account, 800);
        WithdrawThread t3 = new WithdrawThread(account, 900);
        DepositThread t4 = new DepositThread(account, 1000);

        // Start all threads
        t1.start();
```

```
        t2.start();
        t3.start();
        t4.start();
    }
}
```

□ Sample Output (Order may vary):

pgsql

CopyEdit

Depositing ₹500...

✅ Deposit complete. Current Balance: ₹1500

Attempting to withdraw ₹800...

✅ Withdrawal successful. Remaining Balance: ₹700

Attempting to withdraw ₹900...

❌ Insufficient funds! Current Balance: ₹700

Depositing ₹1000...

✅ Deposit complete. Current Balance: ₹1700

💡 **Highlights:**

- synchronized ensures that **only one thread** accesses deposit/withdraw at a time.
- Prevents race condition where balance can become inconsistent.
- You can modify the amount or add more threads to test concurrency.

39) Create a class Person with aadhar, name, address as properties. Create at least 3 constructors. Create a subclass called Customer with cust_code, username, password, mobile, email. Demonstrate super() constructor and super keyword to call superclass's methods.

Here's a **complete Java program** demonstrating:

- **Superclass Person** with multiple constructors
- **Subclass Customer** with extra properties
- Use of super() to call superclass constructors
- Use of super.method() to access superclass methods

✅ **Java Code: CustomerApp.java**

java

CopyEdit

```
// Superclass
class Person {
    String aadhar;
    String name;
    String address;

    // Default constructor
    Person() {
        this.aadhar = "Unknown";
        this.name = "Unknown";
        this.address = "Unknown";
    }

    // Parameterized constructor (2 fields)
```

```
    Person(String aadhar, String name) {
        this.aadhar = aadhar;
        this.name = name;
        this.address = "Not Provided";
    }

    // Parameterized constructor (all fields)
    Person(String aadhar, String name, String address) {
        this.aadhar = aadhar;
        this.name = name;
        this.address = address;
    }

    void displayPerson() {
        System.out.println("Aadhar: " + aadhar);
        System.out.println("Name: " + name);
        System.out.println("Address: " + address);
    }
}

// Subclass
class Customer extends Person {
    String cust_code;
    String username;
    String password;
```

```
    String mobile;
    String email;

    // Constructor for Customer (calling Person constructor using super)
    Customer(String aadhar, String name, String address, String cust_code, String username, String password, String mobile, String email) {
        super(aadhar, name, address);  // Call superclass constructor
        this.cust_code = cust_code;
        this.username = username;
        this.password = password;
        this.mobile = mobile;
        this.email = email;
    }

    void displayCustomer() {
        System.out.println("Customer Code: " + cust_code);
        System.out.println("Username: " + username);
        System.out.println("Password: " + password);
        System.out.println("Mobile: " + mobile);
        System.out.println("Email: " + email);

        // Calling superclass method using super
        System.out.println("--- Person Info ---");
        super.displayPerson();
    }
}
```

```
// Main class
public class CustomerApp {
    public static void main(String[] args) {
        Customer c1 = new Customer("1234-5678-9012", "Lucy", "Narnia", "C001", "lucy123", "pass@123", "9876543210", "lucy@example.com");
        c1.displayCustomer();
    }
}
```

□ Output:

```
yaml
CopyEdit
Customer Code: C001
Username: lucy123
Password: pass@123
Mobile: 9876543210
Email: lucy@example.com
--- Person Info ---
Aadhar: 1234-5678-9012
Name: Lucy
Address: Narnia
```

✅ **Concepts Covered:**

- Multiple constructors in superclass
- super() for constructor chaining

- super.method() to call methods from superclass

40) Write a program to implement user defined exception “Not Prime Number”. Read number from command line and check whether number is prime. If it is prime, display message “Number is Prime” else throw “Not Prime Number” exception.

Here is a **complete Java program** that:

- Reads a number from the command line
- Checks if it's a **prime number**
- Throws a **user-defined exception** NotPrimeException if it's **not** prime
- Displays appropriate messages

✅ **Java Code: PrimeCheck.java**

```
// Custom Exception Class
class NotPrimeException extends Exception {
    public NotPrimeException(String message) {
        super(message);
    }
}

// Main Class
public class PrimeCheck {

    // Method to check if number is prime
    public static boolean isPrime(int num) {
        if (num <= 1) return false;
```

```
    for (int i = 2; i <= Math.sqrt(num); i++) {
        if (num % i == 0) return false;
    }
    return true;
}

public static void main(String[] args) {
    try {
        if (args.length < 1) {
            System.out.println("Please provide a number as a command-line argument.");
            return;
        }

        int number = Integer.parseInt(args[0]);

        if (isPrime(number)) {
            System.out.println("Number is Prime");
        } else {
            throw new NotPrimeException("Not Prime Number");
        }

    } catch (NotPrimeException e) {
        System.out.println("Exception Caught: " + e.getMessage());
    } catch (NumberFormatException e) {
        System.out.println("Invalid Input. Please enter a valid integer.");
```

```
    }
  }
}
```

🔧 **How to Compile and Run in Terminal:**

javac PrimeCheck.java

java PrimeCheck 7

✅ Sample Output:

> java PrimeCheck 7

Number is Prime

> java PrimeCheck 10

Exception Caught: Not Prime Number

---

41): Write a JSP program to accept patient id from user in HTML and display patient details in proper format. If details not found display error message

Here's a complete **JSP + HTML + JDBC** application that:

- Accepts **Patient ID** via an HTML form
- Connects to the database using **JDBC**
- Displays **patient details** if found
- Displays an **error message** if not found

🗃️ **Assumed Table: patient**

sql

CopyEdit

```
CREATE TABLE patient (
   pid INT PRIMARY KEY,
   name VARCHAR(100),
   age INT,
   gender VARCHAR(10),
   disease VARCHAR(100)
);
```

✅ **Step 1: patientForm.html**

```
html
CopyEdit
<!DOCTYPE html>
<html>
<head>
   <title>Search Patient</title>
</head>
<body>
   <h2>Enter Patient ID</h2>
   <form action="searchPatient.jsp" method="post">
      Patient ID: <input type="number" name="pid" required>
      <input type="submit" value="Search">
   </form>
</body>
</html>
```

✅ **Step 2: searchPatient.jsp**

jsp

CopyEdit

```
<%@ page import="java.sql.*" %>
<%@ page import="javax.sql.*" %>
<%
   String pidStr = request.getParameter("pid");
   int pid = Integer.parseInt(pidStr);

   Connection con = null;
   PreparedStatement ps = null;
   ResultSet rs = null;

   String url = "jdbc:mysql://localhost:3306/your_database";
   String user = "your_username";
   String pass = "your_password";

   try {
      Class.forName("com.mysql.cj.jdbc.Driver");
      con = DriverManager.getConnection(url, user, pass);

      String sql = "SELECT * FROM patient WHERE pid = ?";
      ps = con.prepareStatement(sql);
      ps.setInt(1, pid);
      rs = ps.executeQuery();
```

```
        if (rs.next()) {
%>
            <h2>Patient Details</h2>
            <table border="1">
                <tr><td>Patient ID</td><td><%= rs.getInt("pid") %></td></tr>
                <tr><td>Name</td><td><%= rs.getString("name") %></td></tr>
                <tr><td>Age</td><td><%= rs.getInt("age") %></td></tr>
                <tr><td>Gender</td><td><%= rs.getString("gender") %></td></tr>
                <tr><td>Disease</td><td><%= rs.getString("disease") %></td></tr>
            </table>
<%
        } else {
%>
            <h2 style="color:red;">Patient Not Found!</h2>
<%
        }
    } catch (Exception e) {
%>
        <h3 style="color:red;">Error: <%= e.getMessage() %></h3>
<%
    } finally {
        try { if (rs != null) rs.close(); } catch (Exception e) {}
        try { if (ps != null) ps.close(); } catch (Exception e) {}
        try { if (con != null) con.close(); } catch (Exception e) {}
```

```
    }
%>
```

💡 **Notes:**

- Replace your_database, your_username, your_password with actual values.
- Add JDBC driver .jar to your project (like mysql-connector-j.jar).

42): Create a multithreaded Java program where: • One thread prints "Good Morning" every 1 second • Another prints "Hello" every 2 seconds • A third prints "Welcome" every 3 seconds. Use sleep() method to delay the output.

Here's a complete **Java program** that creates **three threads**, each printing a different message at different intervals using sleep().

✅ **Multithreaded Message Printer**

java

CopyEdit

```
class GoodMorningThread extends Thread {
    public void run() {
        try {
            while (true) {
                System.out.println("Good Morning");
                Thread.sleep(1000); // 1 second
            }
        } catch (InterruptedException e) {
            System.out.println("GoodMorningThread Interrupted");
```

```
        }
    }
}

class HelloThread extends Thread {
    public void run() {
        try {
            while (true) {
                System.out.println("Hello");
                Thread.sleep(2000); // 2 seconds
            }
        } catch (InterruptedException e) {
            System.out.println("HelloThread Interrupted");
        }
    }
}

class WelcomeThread extends Thread {
    public void run() {
        try {
            while (true) {
                System.out.println("Welcome");
                Thread.sleep(3000); // 3 seconds
            }
        } catch (InterruptedException e) {
```

```
            System.out.println("WelcomeThread Interrupted");
        }
    }
}

public class MultithreadedMessages {
    public static void main(String[] args) {
        GoodMorningThread t1 = new GoodMorningThread();
        HelloThread t2 = new HelloThread();
        WelcomeThread t3 = new WelcomeThread();

        t1.start();
        t2.start();
        t3.start();
    }
}
```

💡 **How to Run:**

1. Save this as MultithreadedMessages.java
2. Compile using:

```
bash
CopyEdit
javac MultithreadedMessages.java
```

3. Run using:

```
bash
```

CopyEdit

java MultithreadedMessages

This will continuously print:

python-repl

CopyEdit

Good Morning

Hello

Good Morning

Welcome

Good Morning

Hello

...

43): Write a Java program that creates two threads: • One thread prints even numbers from 1 to 100 • Another thread prints odd numbers from 1 to 100. Use Thread class or implement Runnable interface.

Here's a **complete Java program** that creates two threads using the **Runnable interface**. One thread prints **even numbers** from 1 to 100, and the other prints **odd numbers**.

✅ **Java Program: Print Even and Odd Numbers using Threads**

java

CopyEdit

```
class EvenRunnable implements Runnable {
    public void run() {
        for (int i = 2; i <= 100; i += 2) {
```

```
            System.out.println("Even: " + i);
            try {
                Thread.sleep(100); // optional, slows down the output for clarity
            } catch (InterruptedException e) {
                System.out.println("Even thread interrupted");
            }
        }
    }
}

class OddRunnable implements Runnable {
    public void run() {
        for (int i = 1; i <= 100; i += 2) {
            System.out.println("Odd: " + i);
            try {
                Thread.sleep(100); // optional
            } catch (InterruptedException e) {
                System.out.println("Odd thread interrupted");
            }
        }
    }
}

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

```
        Runnable even = new EvenRunnable();
        Runnable odd = new OddRunnable();

        Thread evenThread = new Thread(even);
        Thread oddThread = new Thread(odd);

        evenThread.start();
        oddThread.start();
    }
}
```

How to Run This:

1. Save this code in a file named: EvenOddThreads.java
2. Open terminal and compile:

bash

CopyEdit

javac EvenOddThreads.java

3. Run it:

bash

CopyEdit

java EvenOddThreads

Output Example (interleaved):

makefile

CopyEdit

Odd: 1

Even: 2

Odd: 3

Even: 4

...

Odd: 99

Even: 100

Due to thread scheduling, the order may vary a bit — that's normal in multithreaded programs.

44) Write a Java program to remove duplicate elements from an ArrayList using a HashSet. Create a TreeSet to store and display student names in sorted order.

Here's a **Java program** that demonstrates both tasks:

1. **Removing duplicates from an ArrayList using HashSet**
2. **Storing and displaying student names in sorted order using TreeSet**

✅ **Java Program: HashSet & TreeSet**

java

CopyEdit

```
import java.util.*;


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


        // 1. Remove duplicates using HashSet
        ArrayList<String> accessories = new ArrayList<>();
```

```
accessories.add("Bat");
accessories.add("Ball");
accessories.add("Gloves");
accessories.add("Bat");
accessories.add("Helmet");
accessories.add("Ball");

System.out.println("Original ArrayList with duplicates:");
System.out.println(accessories);

HashSet<String> uniqueAccessories = new HashSet<>(accessories);

System.out.println("\nAfter removing duplicates using HashSet:");
System.out.println(uniqueAccessories);

// 2. Store and display student names in sorted order using TreeSet
TreeSet<String> studentNames = new TreeSet<>();
studentNames.add("Draco");
studentNames.add("Harry");
studentNames.add("Ron");
studentNames.add("Hermione");
studentNames.add("Lucy");
studentNames.add("Harry"); // Duplicate

System.out.println("\nStudent names in sorted order using TreeSet:");
```

```
        for (String name : studentNames) {
            System.out.println(name);
        }
    }
}
```

□ How to Run:

1. Save as: CollectionExample.java
2. Compile:

bash

CopyEdit

javac CollectionExample.java

3. Run:

bash

CopyEdit

java CollectionExample

🖨 **Sample Output:**

csharp

CopyEdit

Original ArrayList with duplicates:

[Bat, Ball, Gloves, Bat, Helmet, Ball]

After removing duplicates using HashSet:

[Gloves, Bat, Ball, Helmet]

Student names in sorted order using TreeSet:

Draco

Harry

Hermione

Lucy

Ron

45) Create a TreeSet to store and display student names in sorted order.

Here's a simple **Java program** that creates a TreeSet to store and display **student names in sorted order**.

✅ **Java Program: TreeSet Example**

java

CopyEdit

```
import java.util.TreeSet;


public class StudentTreeSet {
    public static void main(String[] args) {
        // Create a TreeSet of student names
        TreeSet<String> studentNames = new TreeSet<>();


        // Adding names (duplicates will be automatically removed)
        studentNames.add("Harry");
        studentNames.add("Hermione");
```

```
        studentNames.add("Ron");
        studentNames.add("Draco");
        studentNames.add("Lucy");
        studentNames.add("Aslan");
        studentNames.add("Caspian");
        studentNames.add("Peter");
        studentNames.add("Susan");
        studentNames.add("Edmund");

        // Displaying sorted student names
        System.out.println("Student names in sorted order:");
        for (String name : studentNames) {
            System.out.println(name);
        }
    }
}
```

□ How to Run:

1. Save the file as: StudentTreeSet.java
2. Open terminal and compile:

bash

CopyEdit

javac StudentTreeSet.java

3. Run it:

bash

CopyEdit

java StudentTreeSet

🖨️ **Output (Sorted Alphabetically):**

nginx

CopyEdit

Student names in sorted order:

Aslan

Caspian

Draco

Edmund

Harry

Hermione

Lucy

Peter

Ron

Susan

46) Write a Java program to find the intersection of two HashSets.

Here's a simple **Java program** to find the **intersection** of two HashSet collections:

✅ **Java Program: Intersection of Two HashSets**

java

CopyEdit

import java.util.HashSet;

```
public class HashSetIntersection {
    public static void main(String[] args) {
        // First HashSet
        HashSet<String> set1 = new HashSet<>();
        set1.add("Harry");
        set1.add("Hermione");
        set1.add("Ron");
        set1.add("Draco");
        set1.add("Lucy");

        // Second HashSet
        HashSet<String> set2 = new HashSet<>();
        set2.add("Hermione");
        set2.add("Ron");
        set2.add("Caspian");
        set2.add("Edmund");
        set2.add("Lucy");

        // Print original sets
        System.out.println("Set 1: " + set1);
        System.out.println("Set 2: " + set2);

        // Find intersection
        HashSet<String> intersection = new HashSet<>(set1);
```

```
        intersection.retainAll(set2);

        // Print intersection
        System.out.println("\nIntersection of Set 1 and Set 2:");
        System.out.println(intersection);
    }
}
```

How to Run:

1. Save as: HashSetIntersection.java
2. Compile:

bash

CopyEdit

javac HashSetIntersection.java

3. Run:

bash

CopyEdit

java HashSetIntersection

Sample Output:

sql

CopyEdit

Set 1: [Harry, Hermione, Ron, Draco, Lucy]

Set 2: [Hermione, Ron, Caspian, Edmund, Lucy]

Intersection of Set 1 and Set 2:

[Hermione, Ron, Lucy]

48) Implement a multithreading program where multiple threads print multiplication tables (e.g., table of 2, 3, 4...) concurrently using the Runnable interface.

Here's a **Java program** that demonstrates **multithreading** using the Runnable interface, where **each thread prints the multiplication table** of a different number **concurrently**:

✅ **Java Program: Multiplication Tables with Runnable Threads**

java

CopyEdit

```
class MultiplicationTable implements Runnable {
    private int number;

    public MultiplicationTable(int number) {
        this.number = number;
    }

    @Override
    public void run() {
        System.out.println("\nTable of " + number + " by " + Thread.currentThread().getName() + ":");
        for (int i = 1; i <= 10; i++) {
            System.out.println(number + " x " + i + " = " + (number * i));
            try {
                Thread.sleep(200); // Delay to simulate concurrent printing
```

```
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

public class MultiplicationThreadDemo {
    public static void main(String[] args) {
        // Create threads for tables 2, 3, 4
        Thread t1 = new Thread(new MultiplicationTable(2), "Thread-2");
        Thread t2 = new Thread(new MultiplicationTable(3), "Thread-3");
        Thread t3 = new Thread(new MultiplicationTable(4), "Thread-4");

        // Start all threads
        t1.start();
        t2.start();
        t3.start();
    }
}
```

How to Run:

1. Save as: MultiplicationThreadDemo.java
2. Compile:

bash

CopyEdit

javac MultiplicationThreadDemo.java

3. Run:

bash

CopyEdit

java MultiplicationThreadDemo

🖨️ **Sample Output (Output may vary due to concurrent execution):**

python-repl

CopyEdit

Table of 2 by Thread-2:

2 x 1 = 2

2 x 2 = 4

...

Table of 3 by Thread-3:

3 x 1 = 3

3 x 2 = 6

...

Table of 4 by Thread-4:

4 x 1 = 4

4 x 2 = 8

...

Output may interleave as threads run **concurrently**.

50) Write a Java program to: • Add elements to an ArrayList • Modify an element • Delete an element • Search for an element

Here's a **Java program** that shows how to:

- ✅ Add elements to an ArrayList
- 🖉 Modify an element
- ❌ Delete an element
- 🔍 Search for an element

✅ **Java Program: ArrayList Operations**

java

CopyEdit

```
import java.util.ArrayList;


public class ArrayListOperations {
    public static void main(String[] args) {
        // Step 1: Create an ArrayList of Strings
        ArrayList<String> names = new ArrayList<>();


        // Step 2: Add elements to the ArrayList
        names.add("Harry");
        names.add("Hermione");
        names.add("Ron");
        names.add("Draco");
        System.out.println("Initial List: " + names);
```

```
        // Step 3: Modify an element (change Ron to Edmund)
        names.set(2, "Edmund"); // index 2 was "Ron"
        System.out.println("After Modification: " + names);

        // Step 4: Delete an element (remove "Draco")
        names.remove("Draco");
        System.out.println("After Deletion: " + names);

        // Step 5: Search for an element
        String searchName = "Hermione";
        if (names.contains(searchName)) {
            System.out.println(searchName + " found at index " + names.indexOf(searchName));
        } else {
            System.out.println(searchName + " not found in the list.");
        }
    }
}
```

How to Run:

1. Save the file as ArrayListOperations.java
2. Compile it:

```
bash
CopyEdit
javac ArrayListOperations.java
```

3. Run it:

bash

CopyEdit

java ArrayListOperations

🖨️ **Sample Output:**

pgsql

CopyEdit

Initial List: [Harry, Hermione, Ron, Draco]

After Modification: [Harry, Hermione, Edmund, Draco]

After Deletion: [Harry, Hermione, Edmund]

Hermione found at index 1

51) Write a program to copy all elements from one ArrayList to another.

Here's a simple **Java program** that copies all elements from one ArrayList to another using the addAll() method.

✅ **Java Program: Copy Elements from One ArrayList to Another**

java

CopyEdit

```
import java.util.ArrayList;


public class CopyArrayList {
    public static void main(String[] args) {
        // Original ArrayList
```

```
        ArrayList<String> originalList = new ArrayList<>();
        originalList.add("Harry");
        originalList.add("Hermione");
        originalList.add("Ron");
        originalList.add("Draco");

        // Display original list
        System.out.println("Original ArrayList: " + originalList);

        // New ArrayList to copy elements
        ArrayList<String> copiedList = new ArrayList<>();
        copiedList.addAll(originalList); // Copying elements

        // Display copied list
        System.out.println("Copied ArrayList: " + copiedList);
    }
}
```

How to Run:

1. Save the file as CopyArrayList.java
2. Compile the code:

```
bash
CopyEdit
javac CopyArrayList.java
```

3. Run the program:

bash

CopyEdit

java CopyArrayList

🖨️ **Sample Output:**

nginx

CopyEdit

Original ArrayList: [Harry, Hermione, Ron, Draco]

Copied ArrayList: [Harry, Hermione, Ron, Draco]

53) Write a Java program using HashMap to store and display employee IDs (as keys) and their names (as values). Then update the name of an employee given the ID. And Create a HashMap to count the frequency of each word in a given string. Print the word along with its frequency.

Sure! Here's a **complete Java program** that does both tasks:

✅ **Task 1: HashMap for Employee Records**

✅ **Task 2: Count Word Frequency in a String**

💻 **Java Code:**

java

CopyEdit

import java.util.HashMap;

import java.util.Scanner;

public class HashMapExample {

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    // ---------- Task 1: Employee ID - Name HashMap ----------
    HashMap<Integer, String> employees = new HashMap<>();

    // Add employees
    employees.put(101, "Harry");
    employees.put(102, "Hermione");
    employees.put(103, "Ron");
    employees.put(104, "Draco");

    System.out.println("Initial Employee List:");
    for (Integer id : employees.keySet()) {
        System.out.println("ID: " + id + " -> Name: " + employees.get(id));
    }

    // Update employee name by ID
    System.out.print("\nEnter employee ID to update: ");
    int updateId = sc.nextInt();
    sc.nextLine(); // clear buffer

    if (employees.containsKey(updateId)) {
        System.out.print("Enter new name: ");
        String newName = sc.nextLine();
```

```
    employees.put(updateId, newName);
    System.out.println("Updated Employee List:");
    for (Integer id : employees.keySet()) {
        System.out.println("ID: " + id + " -> Name: " + employees.get(id));
    }
} else {
    System.out.println("Employee ID not found.");
}

// ---------- Task 2: Word Frequency Counter ----------
System.out.print("\nEnter a sentence: ");
String sentence = sc.nextLine();

String[] words = sentence.split("\\s+");
HashMap<String, Integer> frequency = new HashMap<>();

for (String word : words) {
    word = word.toLowerCase().replaceAll("[^a-z]", ""); // Clean word
    if (!word.isEmpty()) {
        frequency.put(word, frequency.getOrDefault(word, 0) + 1);
    }
}

System.out.println("\nWord Frequencies:");
for (String word : frequency.keySet()) {
```

```
            System.out.println(word + ": " + frequency.get(word));
        }

        sc.close();
    }
}
```

How to Run:

1. Save the file as HashMapExample.java
2. Compile the code:

nginx

CopyEdit

javac HashMapExample.java

3. Run the program:

nginx

CopyEdit

java HashMapExample

Sample Output:

yaml

CopyEdit

Initial Employee List:

ID: 101 -> Name: Harry

ID: 102 -> Name: Hermione

ID: 103 -> Name: Ron

ID: 104 -> Name: Draco

Enter employee ID to update: 103

Enter new name: Edmund

Updated Employee List:

ID: 101 -> Name: Harry

ID: 102 -> Name: Hermione

ID: 103 -> Name: Edmund

ID: 104 -> Name: Draco

Enter a sentence: Harry and Draco are good. Harry is brave.

Word Frequencies:

harry: 2

and: 1

draco: 1

are: 1

good: 1

is: 1

brave: 1

54) Write a Java program to find the intersection of two HashSets. Also write a Java program that merges two HashMaps containing student roll numbers and names.

Here's the **complete Java program** that does both:

✅ **Task 1: Find Intersection of Two HashSets**

✅ **Task 2: Merge Two HashMaps (Roll No → Name)**

💻 **Java Code:**

java

CopyEdit

```
import java.util.*;


public class SetMapOperations {


   public static void main(String[] args) {

      // ---------- Task 1: Intersection of Two HashSets ----------

      HashSet<String> set1 = new HashSet<>(Arrays.asList("Harry", "Hermione", "Ron", "Draco"));

      HashSet<String> set2 = new HashSet<>(Arrays.asList("Draco", "Ron", "Peter", "Susan"));


      System.out.println("Set1: " + set1);

      System.out.println("Set2: " + set2);


      HashSet<String> intersection = new HashSet<>(set1);

      intersection.retainAll(set2);


      System.out.println("Intersection: " + intersection);


      // ---------- Task 2: Merging Two HashMaps ----------
```

```
        HashMap<Integer, String> map1 = new HashMap<>();
        map1.put(101, "Lucy");
        map1.put(102, "Edmund");

        HashMap<Integer, String> map2 = new HashMap<>();
        map2.put(103, "Aslan");
        map2.put(104, "Caspian");
        map2.put(102, "Peter"); // Overlapping key with different value

        System.out.println("\nMap1: " + map1);
        System.out.println("Map2: " + map2);

        // Merging map2 into map1 (map2 values will overwrite map1 for duplicate keys)
        map1.putAll(map2);

        System.out.println("Merged Map: " + map1);
    }
}
```

How to Run:

1. Save the file as SetMapOperations.java
2. Compile it:

```
nginx
CopyEdit
javac SetMapOperations.java
```

3. Run it:

nginx

CopyEdit

java SetMapOperations

□ Sample Output:

yaml

CopyEdit

Set1: [Ron, Hermione, Draco, Harry]

Set2: [Draco, Susan, Ron, Peter]

Intersection: [Ron, Draco]

Map1: {101=Lucy, 102=Edmund}

Map2: {103=Aslan, 104=Caspian, 102=Peter}

Merged Map: {101=Lucy, 102=Peter, 103=Aslan, 104=Caspian}

55) Write a Java program that performs the following using a Vector: • Add elements • Insert at a specific index • Remove an element • Iterate using Enumeration

Here's a **complete Java program** that demonstrates how to use a Vector in Java to:

- ✅ Add elements
- ✅ Insert at a specific index
- ✅ Remove an element
- ✅ Iterate using Enumeration

💻 **Java Code: VectorDemo.java**

java

CopyEdit

```
import java.util.*;


public class VectorDemo {
    public static void main(String[] args) {
        // Create a Vector of student names
        Vector<String> students = new Vector<>();

        // ✅ Add elements
        students.add("Harry");
        students.add("Hermione");
        students.add("Ron");
        students.add("Draco");

        System.out.println("Initial Vector: " + students);

        // ✅ Insert element at specific index
        students.insertElementAt("Luna", 2);
        System.out.println("After inserting 'Luna' at index 2: " + students);

        // ✅ Remove an element
        students.remove("Draco");
        System.out.println("After removing 'Draco': " + students);
```

```
        // ✅ Iterate using Enumeration
        System.out.println("Iterating using Enumeration:");
        Enumeration<String> e = students.elements();
        while (e.hasMoreElements()) {
            System.out.println(e.nextElement());
        }
    }
}
```

🔧 How to Run:

1. Save the file as VectorDemo.java
2. Compile it:

```
nginx
CopyEdit
javac VectorDemo.java
```

3. Run it:

```
nginx
CopyEdit
java VectorDemo
```

📌 Sample Output:

```
nginx
CopyEdit
Initial Vector: [Harry, Hermione, Ron, Draco]
```

After inserting 'Luna' at index 2: [Harry, Hermione, Luna, Ron, Draco]

After removing 'Draco': [Harry, Hermione, Luna, Ron]

Iterating using Enumeration:

Harry

Hermione

Luna

Ron

56) Create a Vector of integers and calculate the sum of all elements.

Here's a simple **Java program** that creates a Vector of integers, adds elements, and calculates the **sum of all elements**.

💻 **Java Code: VectorSum.java**

java

CopyEdit

```
import java.util.*;


public class VectorSum {
    public static void main(String[] args) {
        // Create a Vector of integers
        Vector<Integer> numbers = new Vector<>();

        // Add elements to the Vector
        numbers.add(10);
        numbers.add(20);
```

```
        numbers.add(30);
        numbers.add(40);
        numbers.add(50);

        // Display the Vector
        System.out.println("Vector elements: " + numbers);

        // Calculate sum
        int sum = 0;
        for (int num : numbers) {
            sum += num;
        }

        // Display the sum
        System.out.println("Sum of elements: " + sum);
    }
}
```

How to Run:

1. Save the file as VectorSum.java
2. Open terminal and compile it:

```
nginx
CopyEdit
javac VectorSum.java
```

3. Then run:

nginx

CopyEdit

java VectorSum

□ Sample Output:

yaml

CopyEdit

Vector elements: [10, 20, 30, 40, 50]

Sum of elements: 150

57) Create a Java program where three threads run concurrently: • One counts from 1 to 10 • Another displays alphabets from A to J • The third prints special characters (! to *). Use Runnable interface and thread priority.

Here's a **Java program** where **three threads** run **concurrently** using the Runnable interface and set **thread priorities**:

💻 Java Program: MultiThreadDemo.java

java

CopyEdit

```
public class MultiThreadDemo {
    public static void main(String[] args) {
        // Thread 1: Counts from 1 to 10
        Runnable numberThread = () -> {
            for (int i = 1; i <= 10; i++) {
```

```
        System.out.println("Number: " + i);
        try {
            Thread.sleep(100); // just to slow down output
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
};

// Thread 2: Displays alphabets A to J
Runnable alphabetThread = () -> {
    for (char c = 'A'; c <= 'J'; c++) {
        System.out.println("Alphabet: " + c);
        try {
            Thread.sleep(100);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
};

// Thread 3: Displays special characters ! to *
Runnable specialCharThread = () -> {
    for (char c = '!'; c <= '*'; c++) {
        System.out.println("Special Char: " + c);
```

```
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    };

    // Create Threads
    Thread t1 = new Thread(numberThread);
    Thread t2 = new Thread(alphabetThread);
    Thread t3 = new Thread(specialCharThread);

    // Set Priorities
    t1.setPriority(Thread.MIN_PRIORITY);     // Priority 1
    t2.setPriority(Thread.NORM_PRIORITY);    // Priority 5
    t3.setPriority(Thread.MAX_PRIORITY);     // Priority 10

    // Start Threads
    t1.start();
    t2.start();
    t3.start();
  }
}
```

□ How to Run

1. Save this as MultiThreadDemo.java
2. Compile:

nginx

CopyEdit

javac MultiThreadDemo.java

3. Run:

nginx

CopyEdit

java MultiThreadDemo

□ Sample Output (Order may vary due to concurrency):

yaml

CopyEdit

Number: 1

Alphabet: A

Special Char: !

Special Char: "

Number: 2

Alphabet: B

...

58) Create a Vector of integers and calculate the sum of all elements. Accept 5 names from the user and store them in an ArrayList. Then sort the list alphabetically and display the result

Here's a **complete Java program** that:

1. Creates a Vector<Integer>, adds elements, and calculates their **sum**.
2. Accepts **5 names from user**, stores them in an ArrayList<String>, **sorts** them alphabetically, and **displays** them.

✅ **Java Program: VectorArrayListDemo.java**

java

CopyEdit

```
import java.util.*;


public class VectorArrayListDemo {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // ---------- Part 1: Vector of Integers and Sum ----------
        Vector<Integer> numbers = new Vector<>();
        System.out.println("Enter 5 integers:");

        for (int i = 0; i < 5; i++) {
            System.out.print("Enter number " + (i + 1) + ": ");
            int num = sc.nextInt();
            numbers.add(num);
        }

        int sum = 0;
        for (int num : numbers) {
            sum += num;
```

```
    }

    System.out.println("\nVector elements: " + numbers);
    System.out.println("Sum of elements: " + sum);

    // ---------- Part 2: ArrayList of Names and Sorting ----------
    ArrayList<String> names = new ArrayList<>();
    sc.nextLine(); // Consume leftover newline

    System.out.println("\nEnter 5 names:");
    for (int i = 0; i < 5; i++) {
        System.out.print("Enter name " + (i + 1) + ": ");
        String name = sc.nextLine();
        names.add(name);
    }

    Collections.sort(names);

    System.out.println("\nSorted Names:");
    for (String name : names) {
        System.out.println(name);
    }

    sc.close();
}
```

}

How to Run

1. Save as: VectorArrayListDemo.java
2. Compile: javac VectorArrayListDemo.java
3. Run: java VectorArrayListDemo