PART A

1. Write a program to convert numbers into words using Enumerations with constructors, methods and instance variables. (INPUT RANGE-0 TO 99999) EX: 36 THIRTY SIX

```
import java.util.Scanner;
 public class NumberToWords{
 enum Units { ZERO(""), ONE("one"), TWO("two"), THREE("three"), FOUR("four"),
FIVE("five"),SIX("six"), SEVEN("seven"), EIGHT("eight"), NINE("nine"),
TEN("ten"),ELEVEN("eleven"),    TWELVE("twelve"),
THIRTEEN("thirteen"),FOURTEEN("fourteen"),FIFTEEN("fifteen"), SIXTEEN("sixteen"),
SEVENTEEN("seventeen"), EIGHTEEN("eighteen"),NINETEEN("nineteen");
 private final String word;
 Units(String word) {
 this.word = word;
 }
 public String getWord() {
 return word;
 }
 }
 enum Tens {ZERO(""), TEN(""), TWENTY("twenty"),
THIRTY("thirty"),FORTY("forty"),FIFTY("fifty"),SIXTY("sixty"), SEVENTY("seventy"),
EIGHTY("eighty"), NINETY("ninety");
 private final String word;
 Tens(String word) {
 this.word = word;
 }
 public String getWord() {
 return word;
 }
 }
 public static String converttowords(int number) {
 if (number == 0) {
 return Units.ZERO.getWord();
 }
 StringBuilder result = new StringBuilder();
```

```
int thousands = number / 1000;
int remaining = number % 1000;
if (thousands > 0) {
result.append(convert(thousands)).append(" thousand ");
}
result.append(convert(remaining));
return result.toString().trim();
}
private static String convert(int number) {
StringBuilder result = new StringBuilder();
int hundreds = number / 100;
int remaining = number % 100;
if (hundreds > 0) {
result.append(Units.values()[hundreds].getWord()).append(" hundred ");
}
if (remaining != 0) {
if (remaining < 20) {
result.append(Units.values()[remaining].getWord());
} else {
int tens = remaining / 10;
int units = remaining % 10;
result.append(Tens.values()[tens].getWord());
if (units > 0) {
result.append(" ").append(Units.values()[units].getWord());
}
}
}
return result.toString().trim();
}
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.print("Enter a number (0 to 99999): ");
int number = scanner.nextInt();
if (number < 0 || number > 99999) {
System.out.println("Number out of range.");
```

```
 } else {
 System.out.println(number + " in words: " + converttowords(number));
 }
 scanner.close();
 }
}
```

OUTPUT:

Enter a number (0 to 99999): 60000

60000 in words: sixty thousand

2. Find the second maximum and second minimum in a set of numbers using auto boxing and unboxing.

```
import java.util.*;
public class SecondMinMax {
 public static void main(String[] args) {
 Scanner scanner = new Scanner(System.in);
 System.out.print("Enter the number of elements: ");
 int n = scanner.nextInt();
 List numbers = new ArrayList();
 for (int i = 0; i < n; i++) {
 System.out.print("Enter element " + (i + 1) + ": ");
 int input = scanner.nextInt();
 numbers.add(input); // Autoboxing: primitive int to Integer object
 }
 Integer secondMin = findSecondMin(numbers);
 Integer secondMax = findSecondMax(numbers);
 if (secondMin == null || secondMax == null) {
 System.out.println("There are less than 2 elements.");
 } else {
 System.out.println("Second minimum: " + secondMin);
 System.out.println("Second maximum: " + secondMax);
 }
 scanner.close();
 }
 public static Integer findSecondMin(List numbers) {
 if (numbers.size() < 2)
 return null;
 int min = Integer.MAX_VALUE;
 int secondMin = Integer.MAX_VALUE;
 for (Object obj : numbers) {
 int num =(int) obj; // Unboxing: Object to primitive int
 if (num < min) {
 secondMin = min;
 min = num;
```

```
} else if (num < secondMin && num != min) {
secondMin = num;
}
}
return secondMin == Integer.MAX_VALUE ? null : secondMin;
}
public static Integer findSecondMax(List numbers) {
if (numbers.size() < 2)
return null;
int max = Integer.MIN_VALUE;
int secondMax = Integer.MIN_VALUE;
for (Object obj : numbers) {
int num =  (int)obj; // Unboxing: Object to primitive int
if (num > max) {
secondMax = max;
max = num;
} else if (num > secondMax && num != max) {
secondMax = num;
}
}
return secondMax == Integer.MIN_VALUE ? null : secondMax;
}
}
```

OUTPUT:

```
Enter the number of elements: 5
Enter element 1: 10
Enter element 2: 46
Enter element 3: 78
Enter element 4: 34
Enter element 5: 56
Second minimum: 34
Second maximum: 56
```

3. Write a menu driven program to create an Arraylist and perform the following operations

i) Adding elements

ii) Sorting elements

iii) Replace an element with another

iv)Removing an element

v) Displaying all the elements

vi)Adding an element between two elements

```
import java.util.*;
public class ArrayListOperations {
 private static Scanner scanner = new Scanner(System.in);
 private static List<Integer> arrayList = new ArrayList();
 public static void main(String[] args) {
 boolean exit = false;
 while (!exit) {
 System.out.println("\nArrayList Operations Menu:");
 System.out.println("1. Add elements");
 System.out.println("2. Sort elements");
 System.out.println("3. Replace an element with another");
 System.out.println("4. Remove an element");
 System.out.println("5. Display all elements");
 System.out.println("6. Add an element between two elements");
 System.out.println("7. Exit");
 System.out.print("Enter your choice: ");
 int choice = scanner.nextInt();
 scanner.nextLine(); // Consume newline
 switch (choice) {
 case 1:
 addElements();
 break;
 case 2:
 sortElements();
 break;
 case 3:
```

```
replaceElement();
break;
case 4:
removeElement();
break;
case 5:
displayElements();
break;
case 6:
addElementBetween();
break;
case 7:
exit = true;
break;
default:
System.out.println("Invalid choice. Please enter a number between 1 and 7.");
}
}
scanner.close();
}
private static void addElements() {
System.out.print("Enter the number of elements to add: ");
int n = scanner.nextInt();
scanner.nextLine(); // Consume newline
System.out.println("Enter the elements:");
for (int i = 0; i < n; i++) {
int element = scanner.nextInt();
arrayList.add(element);
}
System.out.println("Elements added successfully.");
}
private static void sortElements() {
Collections.sort(arrayList);
System.out.println("Elements sorted successfully.");
}
```

```
private static void replaceElement() {
System.out.print("Enter the index of the element to replace: ");
int index = scanner.nextInt();
System.out.print("Enter the new element: ");
int newElement = scanner.nextInt();
arrayList.set(index, newElement);
System.out.println("Element replaced successfully.");
}
private static void removeElement() {
System.out.print("Enter the index of the element to remove: ");
int index = scanner.nextInt();
arrayList.remove(index);
System.out.println("Element removed successfully.");
}
private static void displayElements() {
System.out.println("Elements in the ArrayList:");
for (Integer element : arrayList) {
System.out.print(element + " ");
}
System.out.println();
}
private static void addElementBetween() {
System.out.print("Enter the index after which you want to add the element: ");
int index = scanner.nextInt();
System.out.print("Enter the element to add: ");
int newElement = scanner.nextInt();
arrayList.add(index + 1, newElement);
System.out.println("Element added successfully.");
}
}
```

OUTPUT:

ArrayList Operations Menu:

1. Add elements
2. Sort elements
3. Replace an element with another
4. Remove an element
5. Display all elements
6. Add an element between two elements
7. Exit

Enter your choice: 1

Enter the number of elements to add: 5

Enter the elements:

10 40 60 20 30

Elements added successfully.

ArrayList Operations Menu:

1. Add elements
2. Sort elements
3. Replace an element with another
4. Remove an element
5. Display all elements
6. Add an element between two elements
7. Exit

Enter your choice: 2

Elements sorted successfully.

ArrayList Operations Menu:

1. Add elements
2. Sort elements
3. Replace an element with another
4. Remove an element
5. Display all elements
6. Add an element between two elements
7. Exit

Enter your choice: 5

Elements in the ArrayList:
10 20 30 40 60

ArrayList Operations Menu:
1. Add elements
2. Sort elements
3. Replace an element with another
4. Remove an element
5. Display all elements
6. Add an element between two elements
7. Exit
Enter your choice: 3
Enter the index of the element to replace: 3
Enter the new element: 67
Element replaced successfully.

ArrayList Operations Menu:
1. Add elements
2. Sort elements
3. Replace an element with another
4. Remove an element
5. Display all elements
6. Add an element between two elements
7. Exit
Enter your choice: 5
Elements in the ArrayList:
10 20 30 67 60

ArrayList Operations Menu:
1. Add elements
2. Sort elements
3. Replace an element with another
4. Remove an element
5. Display all elements
6. Add an element between two elements

7. Exit

Enter your choice: 4

Enter the index of the element to remove: 2

Element removed successfully.

ArrayList Operations Menu:

1. Add elements

2. Sort elements

3. Replace an element with another

4. Remove an element

5. Display all elements

6. Add an element between two elements

7. Exit

Enter your choice: 5

Elements in the ArrayList:

10 20 67 60

ArrayList Operations Menu:

1. Add elements

2. Sort elements

3. Replace an element with another

4. Remove an element

5. Display all elements

6. Add an element between two elements

7. Exit

Enter your choice: 6

Enter the index after which you want to add the element: 3

Enter the element to add: 45

Element added successfully.

ArrayList Operations Menu:

1. Add elements

2. Sort elements

3. Replace an element with another

4. Remove an element

5. Display all elements

6. Add an element between two elements

7. Exit

Enter your choice: 5

Elements in the ArrayList:

10 20 67 60 45

ArrayList Operations Menu:

1. Add elements

2. Sort elements

3. Replace an element with another

4. Remove an element

5. Display all elements

6. Add an element between two elements

7. Exit

Enter your choice: 7

4. Write a java program to tind words with even number ot characters mn a string, then swap the pair of characters in those words and also toggle the characters in a given string
EX:Good Morning everyone
Output: 0Gdo vereoyen
gOOD mORNING EVERYONE

```
import java.util.*;
public class StringManipulation {
 public static void main(String[] args) {
 Scanner scanner = new Scanner(System.in);
 System.out.println("Enter a string: ");
 String input = scanner.nextLine();
 String swappedString = swapEvenCharacters(input);
 System.out.println("Swapped string: " + swappedString);
 String toggledString = toggleCharacters(input);
 System.out.println("Toggled string: " + toggledString);
 }
 public static String swapEvenCharacters(String input) {
 StringBuilder result = new StringBuilder();
 String[] words = input.split("\\s+");
 for (String word : words) {
 if (word.length() % 2 == 0) {
 char[] chars = word.toCharArray();
 for (int i = 0; i < chars.length - 1; i += 2) {
 char temp = chars[i];
 chars[i] = chars[i + 1];
 chars[i + 1] = temp;
 }
 result.append(chars).append(" ");
 } else {
 result.append(word).append(" ");
 }
 }
 return result.toString().trim();
 }
```

```
public static String toggleCharacters(String input) {
StringBuilder result = new StringBuilder();
for (char c : input.toCharArray()) {
if (Character.isUpperCase(c)) {
result.append(Character.toLowerCase(c));
} else if (Character.isLowerCase(c)) {
result.append(Character.toUpperCase(c));
} else {
result.append(c);
}
}
return result.toString();
}
}
```

OUTPUT:

```
Enter a string:
Computer Science
Swapped string: oCpmture Science
Toggled string: cOMPUTER sCIENCE
```

5. Write a Servlet program that accepts the age and name and displays it the user 1s Eligible for voting or not.

Index.html

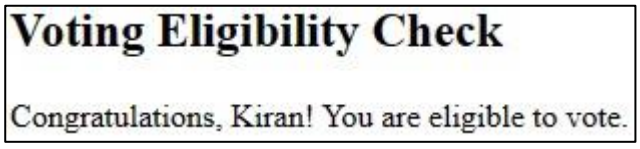
```
<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Voter App</title>
</head>
<body>
<form action="VoterCheck" method="post">
<fieldset style="width:20%; background-color:#80ffcc">
<table>
<tr><td>Name</td><td><input type="text" name="name"></td></tr>
<tr><td>Age</td><td><input type="text" name="age"></td></tr>
<tr><td></td><td><input type="submit" value="check voting eligibility"></td></tr>
</table>
</fieldset>
</form>
</body>
</html>
```

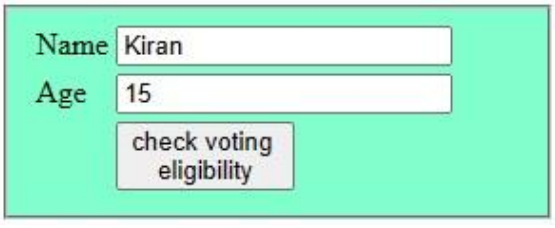
VoterCheck.java

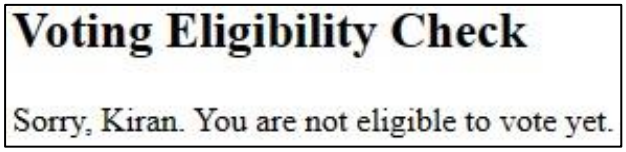
```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/VoterCheck")
public class VoterCheck extends HttpServlet {
private static final long serialVersionUID = 1L;
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html");
PrintWriter out = response.getWriter();
String name = request.getParameter("name");
String ageParam = request.getParameter("age");
try {
int age = Integer.parseInt(ageParam);
String message = "";
if (age >= 18) {
message = "Congratulations, " + name + "! You are eligible to vote.";
```

```
} else {
message = "Sorry, " + name + ". You are not eligible to vote yet.";
}
out.println("<html>");
out.println("<head>");
out.println("<title>Voting Eligibility Check</title>");
out.println("</head>");
out.println("<body>");
out.println("<h2>Voting Eligibility Check</h2>");
out.println("<p>" + message + "</p>");
out.println("</body>");
out.println("</html>");
} catch (NumberFormatException e) {
out.println("<html>");
out.println("<head>");
out.println("<title>Error</title>");
out.println("</head>");
out.println("<body>");
out.println("<h2>Error</h2>");
out.println("<p>Invalid age value: " + ageParam + "</p>");
out.println("</body>");
out.println("</html>");
}
}
}
```

OUTPUT:



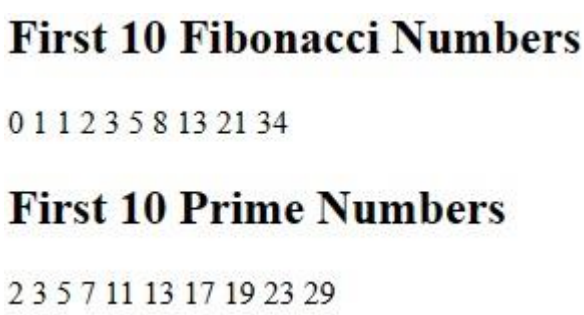




6. Write a JSP program to print first 10 Fibonacci and 10 prime numbers.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Fibonacci and Prime Numbers</title>
</head>
<body>
<h2>First 10 Fibonacci Numbers</h2>
<%
int n1 = 0, n2 = 1, n3, i, count = 10; out.println(n1 + " " + n2);
for (i = 2; i < count; ++i) { n3 = n1 + n2; out.println(" " + n3);
n1 = n2; n2 = n3;
}
%>
<h2>First 10 Prime Numbers</h2>
<%
int num = 1;
int countPrimes = 0;
while (countPrimes < 10) { boolean isPrime = true;
for (i = 2; i <= Math.sqrt(num); i++) { if (num % i == 0) {
isPrime = false; break;
}
}
if (isPrime && num > 1) { out.println(num + " "); countPrimes++;
}
num++;
}
%>
</body>
</html>
```

OUTPUT:



7. Write a JSP Program to design a shopping cart to add items, remove item and to display items from the cart using Sessions.

Shopping.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Shopping Cart</title>
</head>
<body>
<h2>Shopping Cart</h2>
<form action="CartController" method="post">
<label for="item">Add Item:</label>
<input type="text" id="item" name="item">
<input type="submit" value="Add to Cart">
<input type="hidden" name="action" value="add">
</form>
<form action="CartController" method="post">
<label for="removeItem">Remove Item:</label>
<select id="removeItem" name="removeItem">
<option value="">Select Item</option>
<%
String[] cartItems = (String[]) session.getAttribute("cartItems"); if (cartItems != null) {
for (String item : cartItems) {
out.println("<option value='" + item + "'>" + item + "</option>");
}
}
%>
</select>
<input type="submit" value="Remove from Cart">
<input type="hidden" name="action" value="remove">
</form>
<h3>Items in Cart:</h3>
<%
if (cartItems != null && cartItems.length > 0) { for (String item : cartItems) {
out.println("<p>" + item + "</p>");
}
} else {
out.println("<p>No items in the cart</p>");
}
%>
</body>
</html>
```

CartController.java

```
import java.io.IOException;
import javax.servlet.ServletException; import javax.servlet.annotation.WebServlet; import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;
```

```
import javax.servlet.http.HttpSession;

@WebServlet("/CartController")
public class CartController extends HttpServlet { private static final long serialVersionUID =
1L;
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException { String action = request.getParameter("action");
HttpSession session = request.getSession();
String[] cartItems = (String[]) session.getAttribute("cartItems");
if ("add".equals(action)) {
String newItem = request.getParameter("item"); if (cartItems == null) {
cartItems = new String[]{newItem};
} else {
String[] newCart = new String[cartItems.length + 1]; System.arraycopy(cartItems, 0, newCart,
0, cartItems.length); newCart[cartItems.length] = newItem;
cartItems = newCart;
}
session.setAttribute("cartItems", cartItems);
} else if ("remove".equals(action)) {
String itemToRemove = request.getParameter("removeItem"); if (cartItems != null) {
String[] newCart = new String[cartItems.length - 1]; int newIndex = 0;
for (String item : cartItems) {
if (!item.equals(itemToRemove)) { newCart[newIndex] = item; newIndex++;
}
}
cartItems = newCart; session.setAttribute("cartItems", cartItems);
}
}
response.sendRedirect("Shopping.jsp");
}
}
```

OUTPUT:

Shopping Cart

Add Item: Add to Cart
Remove Item: Select Item Remove from Cart

Items in Cart:

keyboard

mouse

After removing the item from the cart

Shopping Cart

Add Item: Add to Cart
Remove Item: Select Item Remove from Cart

Items in Cart:

keyboard

8. Write a java Servlet program to Download a file and display it on the screen(A link has to be provided in HTML, when the link is clicked corresponding file has to be displayed on screen).

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>File Display Example</title>
</head>
<body>
<h2>Click the link to view the file contents:</h2>
<a href="FileDownloadServlet?filename=file.txt">View File</a>
</body>
</html>
```

FileDownloadServlet.java

```
package com.mycompany.filedownload;
import java.io.*;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/FileDownloadServlet")
public class FileDownloadServlet extends HttpServlet {
private static final long serialVersionUID = 1L;
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
response.setContentType("text/html");
PrintWriter out = response.getWriter();

// Display HTML page with a link to download the file
out.println("<html><head><title>File Display Example</title></head><body>");
out.println("<h2>Click the link to view the file contents:</h2>");
out.println("<a href=\"FileDisplayServlet?filename=file.txt\">View File</a>");
out.println("</body></html>");

// Check if a filename is provided in the request
String fileName = request.getParameter("filename");
```

```
if (fileName != null && !fileName.isEmpty()) {
// Path to the file to be displayed
String filePath = "D:/JEE/" + fileName; //

// Display file contents
try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
out.println("<h2>File Contents:</h2>");
String line;
while ((line = reader.readLine()) != null) { out.println(line + "<br>");
}
} catch (FileNotFoundException e) { out.println("<p>File not found.</p>");
}
}
}
}
```

OUTPUT:

Click the link to view the file contents:

View File

Click the link to view the file contents:

View File

File Contents:

Hello How are you ?

Good Morning

PART B

1. Write a menu driven JDBC program to perform basic operations with Student Table.

MENU
1. Add new Student
2. Delete a specified students Record
3. Update Students Address specified students Record
4. Search for a particular Student
5. Exit

Database Name: StudentDatabase
Table Creation: Create Table Student(regno bigint,sname varchar(25),dob date,address varchar(25),class varchar(10),course varchar(10));

StudentMgt.java

```
package com.mycompany.studentmgt;
import java.util.Scanner;
import java.sql.*;

public class StudentMgt {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in); try {
Class.forName("org.apache.derby.jdbc.ClientDriver");
Connection
conn=DriverManager.getConnection("jdbc:derby://localhost:1527/StudentDatabase","vcp",
"vcp");
int choice;
do {
System.out.println("1. Add new Student");
System.out.println("2. Delete a specified student's Record");
System.out.println("3. Update Student's Address");
System.out.println("4. Search for a particular Student"); System.out.println("5. Exit");
System.out.print("Enter your choice: ");
choice = scanner.nextInt();
switch (choice) {
case 1:
        addNewStudent(conn, scanner);
        break;
case 2:
        deleteStudentRecord(conn, scanner);
        break;
case 3:
        updateStudentAddress(conn, scanner);
        break;
case 4:
        searchStudent(conn, scanner);
        break;
case 5:
        System.out.println("Exiting...");
        break;
default:
        System.out.println("Invalid choice. Please try again.");
}
```

```
} while (choice != 5);
conn.close(); scanner.close();
} catch (SQLException se) { se.printStackTrace();
} catch (Exception e) { e.printStackTrace();
}
}

static void addNewStudent(Connection conn, Scanner scanner) throws SQLException {
System.out.println("Enter Student Details:");
System.out.print("Registration Number: ");
int regNo = scanner.nextInt();
System.out.print("Name: ");
scanner.nextLine(); // Consume newline character
String sname = scanner.nextLine();
System.out.print("Date of Birth (YYYY-MM-DD): ");
String dob = scanner.nextLine();
System.out.print("Address: ");
String address = scanner.nextLine();
System.out.print("Class: ");
String class_ = scanner.nextLine();
System.out.print("Course: ");
String course = scanner.nextLine();
String sql = "INSERT INTO Student (regno, sname, dob, address, class, course) VALUES
(?, ?, ?,?, ?, ?)";
PreparedStatement preparedStatement = conn.prepareStatement(sql);
preparedStatement.setInt(1, regNo);
preparedStatement.setString(2, sname);
preparedStatement.setString(3, dob);
preparedStatement.setString(4, address);
preparedStatement.setString(5, class_);
preparedStatement.setString(6, course);
preparedStatement.executeUpdate();
System.out.println("Student added successfully!");
}

static void deleteStudentRecord(Connection conn, Scanner scanner) throws SQLException {
System.out.print("Enter Registration Number of the student to delete: ");
int regNo = scanner.nextInt();
String sql = "DELETE FROM Student WHERE regno = ?";
PreparedStatement preparedStatement = conn.prepareStatement(sql);
preparedStatement.setInt(1, regNo);
int rowsDeleted = preparedStatement.executeUpdate();
if (rowsDeleted > 0) {
System.out.println("Student record deleted successfully!");
} else {
System.out.println("No such student found with the given Registration Number.");
}
}

static void updateStudentAddress(Connection conn, Scanner scanner) throws SQLException
{
System.out.print("Enter Registration Number of the student to update: ");
```

```
int regNo = scanner.nextInt();
scanner.nextLine();
System.out.print("Enter new Address: ");
String newAddress = scanner.nextLine();
String sql = "UPDATE Student SET address = ? WHERE regno = ?";
PreparedStatement preparedStatement = conn.prepareStatement(sql);
preparedStatement.setString(1, newAddress);
preparedStatement.setInt(2, regNo);
int rowsUpdated = preparedStatement.executeUpdate();
if (rowsUpdated > 0) {
System.out.println("Student address updated successfully!");
} else {
System.out.println("No such student found with the given Registration Number.");
}
}

static void searchStudent(Connection conn, Scanner scanner) throws SQLException {
System.out.print("Enter Registration Number of the student to search: ");
int regNo = scanner.nextInt();
String sql = "SELECT * FROM Student WHERE regno = ?";
PreparedStatement preparedStatement = conn.prepareStatement(sql);
preparedStatement.setInt(1, regNo);
ResultSet resultSet = preparedStatement.executeQuery();
if (resultSet.next()) {
System.out.println("Student Details:");
System.out.println("Registration Number: " + resultSet.getInt("regno"));
System.out.println("Name: " + resultSet.getString("sname"));
System.out.println("Date of Birth: " + resultSet.getString("dob"));
System.out.println("Address: " + resultSet.getString("address"));
System.out.println("Class: " + resultSet.getString("class"));
System.out.println("Course: " + resultSet.getString("course"));
} else {
System.out.println("No student found with the given Registration Number.");
}
}
}
```

OUTPUT:

```
1. Add new Student
2. Delete a specified student's Record
3. Update Student's Address
4. Search for a particular Student
5. Exit
Enter your choice: 1
Enter Student Details:
Registration Number: 100
Name: Anushree
Date of Birth (YYYY-MM-DD): 2003-01-13
Address: Puttur
```

```
Class: 3rd
Course: BCA
Student added successfully!
1. Add new Student
2. Delete a specified student's Record
3. Update Student's Address
4. Search for a particular Student
5. Exit
Enter your choice: 1
Enter Student Details:
Registration Number: 200
Name: Karthik
Date of Birth (YYYY-MM-DD): 2003-05-18
Address: Sullia
Class: 3rd
Course: BCA
Student added successfully!
1. Add new Student
2. Delete a specified student's Record
3. Update Student's Address
4. Search for a particular Student
5. Exit
Enter your choice: 4
Enter Registration Number of the student to search: 200
Student Details:
Registration Number: 200
Name: Karthik
Date of Birth: 2003-05-18
Address: Sullia
Class: 3rd
Course: BCA
1. Add new Student
2. Delete a specified student's Record
3. Update Student's Address
4. Search for a particular Student
5. Exit
Enter your choice: 2
Enter Registration Number of the student to delete: 200
Student record deleted successfully!
1. Add new Student
2. Delete a specified student's Record
3. Update Student's Address
4. Search for a particular Student
5. Exit
Enter your choice: 4
Enter Registration Number of the student to search: 200
No student found with the given Registration Number.
1. Add new Student
```

2. Delete a specified student's Record
3. Update Student's Address
4. Search for a particular Student
5. Exit
Enter your choice: 3
Enter Registration Number of the student to update: 100
Enter new Address: Manglore
Student address updated successfully!
1. Add new Student
2. Delete a specified student's Record
3. Update Student's Address
4. Search for a particular Student
5. Exit
Enter your choice: 4
Enter Registration Number of the student to search: 100
Student Details:
Registration Number: 100
Name: Anushree
Date of Birth: 2003-03-13
Address: Mangalore
Class: 3rd
Course: BCA
1. Add new Student
2. Delete a specified student's Record
3. Update Student's Address
4. Search for a particular Student
5. Exit
Enter your choice: 5
Exiting...

2. Write a menu driven JDBC program to perform basic operations with Bank Table.

MENU
1. Add new Account Holder information.
2. Amount Deposit
3. Amount Withdrawal (Maintain minimum balance 500 Rs)
4. Display all information
5. Exit

Database Name: BankDB
Table Creation: Create Table Bank(accno bigint, acname varchar(25), address varchar(50), balance bigint);

BankMgt.java

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

public class BankMgt {
public static void main(String[] args) {
Connection conn = null;
Scanner scanner = new Scanner(System.in);
try {
Class.forName("org.apache.derby.jdbc.ClientDriver");
conn = DriverManager.getConnection("jdbc:derby://localhost:1527/BankDB",
"vcp","vcp");
int choice;
 do {
 System.out.println("MENU");
 System.out.println("1. Add new Account Holder information");
 System.out.println("2. Amount Deposit");
 System.out.println("3. Amount Withdrawal (Maintain minimum balance 500 Rs)");
 System.out.println("4. Display all information");
 System.out.println("5. Exit");
 System.out.print("Enter your choice: ");
 choice = scanner.nextInt();
 switch (choice) {
 case 1:
        addNewAccountHolder(conn, scanner);
        break;
 case 2:
        depositAmount(conn, scanner);
        break;
 case 3:
        withdrawAmount(conn, scanner);
        break;
 case 4:
        displayAllInformation(conn);
        break;
```

```
 case 5:
        System.out.println("Exiting...");
        break;
 default:
        System.out.println("Invalid choice. Please try again.");
       }
 } while (choice != 5);
 conn.close();
 scanner.close();
 } catch (SQLException se) {
   se.printStackTrace();
 } catch (Exception e) {
    e.printStackTrace();
 }
 }

 static void addNewAccountHolder(Connection conn, Scanner scanner) {
 try {
 System.out.print("Enter Account Number: ");
 int accNo = scanner.nextInt();
 scanner.nextLine();
 System.out.print("Enter Account Holder Name: ");
 String accHolderName = scanner.nextLine();
 System.out.print("Enter Address: ");
 String address = scanner.nextLine();
 System.out.print("Enter Initial Balance: ");
 double balance = scanner.nextDouble();
 String sql = "INSERT INTO Bank (accno, acname, address, balance) VALUES (?, ?, ?,?)";
 PreparedStatement preparedStatement = conn.prepareStatement(sql);
preparedStatement.setInt(1, accNo);
 preparedStatement.setString(2, accHolderName);
 preparedStatement.setString(3, address);
 preparedStatement.setDouble(4, balance);
 int rowsInserted = preparedStatement.executeUpdate();
 if (rowsInserted > 0) {
 System.out.println("New account holder information added successfully!");
 } else {
 System.out.println("Failed to add new account holder information.");
  }
  } catch (SQLException se) {
        se.printStackTrace();
   }
   }

   static void depositAmount(Connection conn, Scanner scanner) {
   try {
   System.out.print("Enter Account Number: ");
   int accNo = scanner.nextInt();
```

```
    System.out.print("Enter Deposit Amount: ");
    double amount = scanner.nextDouble();
    String sql = "UPDATE Bank SET balance = balance + ? WHERE accno = ?";
    PreparedStatement preparedStatement = conn.prepareStatement(sql);
    preparedStatement.setDouble(1, amount);
    preparedStatement.setInt(2, accNo);
    int rowsUpdated = preparedStatement.executeUpdate();
    if (rowsUpdated > 0) {
    System.out.println("Amount deposited successfully!");
    } else {
    System.out.println("No account found with the given Account Number.");
    }
    } catch (SQLException se) {
        se.printStackTrace();
    }
    }

    static void withdrawAmount(Connection conn, Scanner scanner) {
    try {
    System.out.print("Enter Account Number: ");
    int accNo = scanner.nextInt();
    System.out.print("Enter Withdrawal Amount: ");
    double amount = scanner.nextDouble();
    String checkBalanceSql = "SELECT balance FROM Bank WHERE accno = ?";
    PreparedStatement checkBalanceStmt = conn.prepareStatement(checkBalanceSql);
    checkBalanceStmt.setInt(1, accNo);
    ResultSet resultSet = checkBalanceStmt.executeQuery();
    if (resultSet.next()) {
    double balance = resultSet.getDouble("balance");
    if (balance - amount >= 500) {
    String updateSql = "UPDATE Bank SET balance = balance - ? WHERE accno = ?";
    PreparedStatement updateStmt = conn.prepareStatement(updateSql);
    updateStmt.setDouble(1, amount);
    updateStmt.setInt(2, accNo);
    int rowsUpdated = updateStmt.executeUpdate();
    if (rowsUpdated > 0) {
    System.out.println("Amount withdrawn successfully!");
    } else {
    System.out.println("Failed to withdraw amount.");
    }
    } else {
    System.out.println("Insufficient balance! Minimum balance of 500 Rs should be
maintained.");
}
} else {
System.out.println("No account found with the given Account Number.");
}
} catch (SQLException se) {
```

```
        se.printStackTrace();
}
}

static void displayAllInformation(Connection conn) {
try {
Statement stmt = conn.createStatement();
String sql = "SELECT * FROM Bank";
ResultSet resultSet = stmt.executeQuery(sql);
while (resultSet.next()) {
System.out.println("Account Number: " + resultSet.getInt("accno"));
System.out.println("Account Name: " + resultSet.getString("acname"));
System.out.println("Address: " + resultSet.getString("address"));
System.out.println("Balance: " + resultSet.getDouble("balance"));
System.out.println("-----------------------------");
}
} catch (SQLException se) {
        se.printStackTrace();
}
}
}
```

OUTPUT:

```
MENU
1. Add new Account Holder information
2. Amount Deposit
3. Amount Withdrawal (Maintain minimum balance 500 Rs)
4. Display all information
5. Exit
Enter your choice: 1
Enter Account Number: 100
Enter Account Holder Name: Karthik
Enter Address: Puttur
Enter Initial Balance: 10000
New account holder information added successfully!
MENU
1. Add new Account Holder information
2. Amount Deposit
3. Amount Withdrawal (Maintain minimum balance 500 Rs)
4. Display all information
5. Exit
Enter your choice: 2
Enter Account Number: 100
Enter Deposit Amount: 5000
Amount deposited successfully!
MENU
1. Add new Account Holder information
```

2. Amount Deposit
3. Amount Withdrawal (Maintain minimum balance 500 Rs)
4. Display all information
5. Exit
Enter your choice: 4
Account Number: 100
Account Name: Karthik
Address: Puttur
Balance: 15000.0
MENU
1. Add new Account Holder information
2. Amount Deposit
3. Amount Withdrawal (Maintain minimum balance 500 Rs)
4. Display all information
5. Exit
Enter your choice: 3
Enter Account Number: 100
Enter Withdrawal Amount: 3000
Amount withdrawn successfully!
MENU
1. Add new Account Holder information
2. Amount Deposit
3. Amount Withdrawal (Maintain minimum balance 500 Rs)
4. Display all information
5. Exit
Enter your choice: 4
Account Number: 100
Account Name: Karthik
Address: Puttur
Balance: 12000.0
MENU
1. Add new Account Holder information
2. Amount Deposit
3. Amount Withdrawal (Maintain minimum balance 500 Rs)
4. Display all information
5. Exit
Enter your choice: 5
Exiting...

3. Write a Java class called lax with methods for calculating Income lax. Have thus class as a servant and create a server program and register in the rmiregistry. Write a client program to invoke these remote methods of the servant and do the calculations. Accept inputs interactively.

<₹ 3,00,000	No Tax
₹ 3,00,001 to ₹ 6,00,000	5%
₹ 6,00,001 to ₹ 9,00,000	10%
₹ 9,00,001 to ₹ 12,00,000	15%
₹ 12,00,001 to ₹ 15,00,000	20%
>₹ 15,00,000	30%

Tax.java

```
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface Tax extends Remote {
double calculateTax(double income) throws RemoteException;
}
```

TaxImpl.java

```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
public class TaxImpl extends UnicastRemoteObject implements Tax {
public TaxImpl() throws RemoteException {
super();
}
@Override
public double calculateTax(double income) throws RemoteException {
if (income <= 300000) {
return 0;
} else if (income <= 600000) { return (income - 300000) * 0.05;
} else if (income <= 900000) {
return (300000 * 0.05) + ((income - 600000) * 0.10);
} else if (income <= 1200000) {
return (300000 * 0.05) + (300000 * 0.10) + ((income - 900000) * 0.15);
```

```
} else if (income <= 1500000) {
return (300000 * 0.05) + (300000 * 0.10) + (300000 * 0.15) + ((income - 1200000) * 0.20);
} else {
return (300000 * 0.05) + (300000 * 0.10) + (300000 * 0.15) + (300000 * 0.20) + ((income -
1500000) * 0.30);
}
}
}
```

Taxserver.java

```
import java.rmi.Naming;
public class TaxServer {
public static void main(String[] args) {
try {
Tax tax = new TaxImpl();
Naming.rebind("TaxService", tax);
System.out.println("Tax service is ready...");
} catch (Exception e) {
System.err.println("Server exception: " + e.toString()); e.printStackTrace();
}
}
}
```

TaxClient.java

```
import java.rmi.Naming; import java.util.Scanner;
public class TaxClient {
public static void main(String[] args) { try {
Tax tax = (Tax) Naming.lookup("//localhost/TaxService");
Scanner scanner = new Scanner(System.in);
System.out.print("Enter income: ");
double income = scanner.nextDouble();
double taxAmount = tax.calculateTax(income); System.out.println("Tax amount: " +
taxAmount);
} catch (Exception e) {
System.err.println("Client exception: " + e.toString()); e.printStackTrace();
```

```
}
}
}
```

OUTPUT:

Environment variable: C:\Program Files\Java\jdk-11\bin

Command prompt:

```
C:\Users\Personal>d:
D:\>cd adarsh
D:\adarsh>cd rmi
D:\adarsh\rmi>rmiregistry
```

Command prompt:

```
C:\Users\Personal>d:
D:\>cd adarsh\rmi\
D:\adarsh\rmi>java TaxServer
Tax service is ready...
```

Command prompt:

```
C:\Users\Personal>d:
D:\>cd adarsh\rmi\
D:\adarsh\rmi>java TaxClient
Enter income: 900000
Tax amount: 45000.0
```

4. Write a Java class called SimpleInterest with methods for calculating simple interest. Have this class as a servant and create a server program and register in the rmiregistry. Write a client program to invoke these remote methods of the servant and do the calculations. Accept inputs at command prompt.

SimpleInterest.java

```
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface SimpleInterest extends Remote {
double calculateSimpleInterest(double principal, double rate, double time) throws RemoteException;
}
```

SimpleInterestImpl.java

```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
public class SimpleInterestImpl extends UnicastRemoteObject implements SimpleInterest {
protected SimpleInterestImpl() throws RemoteException {
super();
}
@Override
public double calculateSimpleInterest(double principal, double rate, double time) throws RemoteException {
return (principal * rate * time) / 100;
}
}
```

SimpleInterestServer.java

```
import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;
public class SimpleInterestServer {
public static void main(String[] args) {
try {
SimpleInterest simpleInterest = new SimpleInterestImpl();
LocateRegistry.createRegistry(1098); // Start RMI registry on port 1098
Naming.rebind("//localhost/SimpleInterestService", simpleInterest); // Bind the remote object'sstub in the registry
System.out.println("Server ready");
} catch (Exception e) {
System.err.println("Server exception: " + e.toString());
e.printStackTrace();
}
}
}
```

SimpleInterestClient.java

```
import java.rmi.Naming;
import java.util.Scanner;
public class SimpleInterestClient {
public static void main(String[] args) {
try {
SimpleInterest simpleInterest = (SimpleInterest)
Naming.lookup("//localhost/SimpleInterestService");
Scanner sc=new Scanner(System.in);
System.out.println("Enter Principal amount,Rate and Time:");
double principal = sc.nextDouble();
double rate = sc.nextDouble();
double time = sc.nextDouble();
double result = simpleInterest.calculateSimpleInterest(principal, rate, time);
System.out.println("Simple Interest: " + result);
} catch (Exception e) {
System.err.println("Client exception: " + e.toString());
e.printStackTrace();
}
}
}
```

OUTPUT:

Environment variable: **C:\Program Files\Java\jdk-11\bin**

Command prompt:

```
C:\Users\Personal>d:
D:\>cd adarsh
D:\adarsh>cd rmi
D:\adarsh\rmi>rmiregistry
```

Command prompt:

```
C:\Users\Personal>d:
D:\>cd adarsh\rmi\
```

D:\adarsh\rmi> java SimpleInterestServer

Server ready

Command prompt:

C:\Users\Personal>d:

D:\>cd adarsh\rmi\

D:\adarsh\rmi> java SimpleInterestClient

Enter Principal amount,Rate and Time:

100000

4

6

Simple Interest: 24000.0

5. Write a Servlet Program to perform Insert, update and View operations on Employee Table.

Database Name: EMPDB
Table Creation: Create table EMP(ID INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY(START WITH 1,INCREMENT BY 1),
ENAME VARCHAR(255),
PASSWORD VARCHAR(255),
EMAIL VARCHAR(255),
COUNTRY VARCHAR(255)
);

Index.html

```
<!DOCTYPE html>
<html>
<head>
<title>Start Page</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<h1>Add Employee</h1>
<form method="POST" action="AddEmployee">
<table>
<tr>
<td> Name </td>
<td> <input type="text" name="ename"></td>
</tr>
<tr>
<td> Password</td>
<td> <input type="password" name="password"></td>
</tr>
<tr>
<td> Email</td>
<td> <input type="email" name="email"></td>
</tr>
<tr>
<td> Country</td>
<td>
<select name="country">
<option> India</option>
<option> US</option>
<option>Germany </option>
</select>
</td>
</tr>
<tr>
<td colspan="2">
<input type="submit" name="submit" value="Save Employee">
</td>
```

```
</tr>
</table>
</form>
<a href="ViewEmployee"> View Employees</a>
</body>
</html>
```

AddEmployee.java

```
package com.mycompany.pb5;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(name = "AddEmployee", urlPatterns = {"/AddEmployee"})
public class AddEmployee extends HttpServlet {
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html;charset=UTF-8");
try (PrintWriter out = response.getWriter())
{
String ename = request.getParameter("ename");
String password = request.getParameter("password");
String email = request.getParameter("email");
String country = request.getParameter("country");
Class.forName("org.apache.derby.jdbc.ClientDriver");
out.println("Hello");
Connection con =
DriverManager.getConnection("jdbc:derby://localhost:1527/EMPDB","vcp","vcp");
Statement stmt =con.createStatement();
String sql="INSERT INTO EMP(ENAME, PASSWORD, EMAIL,COUNTRY) VALUES
('"+ename+"','"+password+"','"+email+"','"+country+"')";
int r=stmt.executeUpdate(sql);
stmt.close();
con.close();
out.println("<!DOCTYPE html>");
out.println("<html>");
out.println("<head>");
out.println("<title>Servlet AddEmployee</title>");
out.println("</head>");
out.println("<body>");
if(r==1)
out.println("<h1>Record Saved</h1>"); else
```

```
out.println("<h1>Error in Saving Record</h1>");
out.println("</body>");
out.println("</html>");
}catch(ClassNotFoundException ex) {
Logger.getLogger(AddEmployee.class.getName()).log(Level.SEVERE, null, ex);
}catch(SQLException ex) {
Logger.getLogger(AddEmployee.class.getName()).log(Level.SEVERE, null, ex);
}
}
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
processRequest(request, response);
}
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
processRequest(request, response);
}
@Override
public String getServletInfo() { return "Short description";
}
}
```

UpdateEmployee.java

```
package com.mycompany.pb5;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(name = "UpdateEmployee", urlPatterns = {"/UpdateEmployee"})
public class UpdateEmployee extends HttpServlet {
protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
response.setContentType("text/html;charset=UTF-8");
try (PrintWriter out = response.getWriter()) {
int id=Integer.parseInt(request.getParameter("id"));
Class.forName("org.apache.derby.jdbc.ClientDriver");

Connection con =
DriverManager.getConnection("jdbc:derby://localhost:1527/EMPDB","vcp","vcp");
```

```
Statement stmt =con.createStatement();
String sql="Select * from vcp.EMP where ID="+id;
ResultSet rs=stmt.executeQuery(sql);
rs.next();
out.println("<!DOCTYPE html>");
out.println("<html>"); out.println("<head>");
out.println("<title>Servlet UpdateEmployee</title>");
out.println("</head>");
out.println("<body>");
out.println("<h1>Update Employee</h1>");
out.println("<form method=\"POST\" action=\"SaveEmployee\">");
out.println("<input type=\"hidden\" name=\"id\" value=\"" + rs.getString("ID") +"\">");
out.println("<h1>Update Employee</h1>");
out.println("<form method=\"POST\" action=\"SaveEmployee\">");
out.println("<table>"); out.println("<tr>"); out.println("<td> Name </td>");
out.println("<td> <input type=\"text\" name=\"ename\" value=\"" + rs.getString("ENAME")
+ "\"></td>");
out.println("</tr>");
out.println("<tr>");
out.println("<td> Password</td>");
out.println("<td> <input type=\"password\" name=\"password\" value=\"" +
rs.getString("PASSWORD") + "\"></td>");
out.println("</tr>");
out.println("<tr>");
out.println("<td> Email</td>");
out.println("<td> <input type=\"email\" name=\"email\" value=\"" + rs.getString("EMAIL")
+ "\"></td>");
out.println("</tr>");
out.println("<tr>"); out.println("<td> Country</td>"); out.println("<td>");
out.println("<select name=\"country\">");
String s =rs.getString("COUNTRY").equals("India")?"Selected":"";
out.println("<option selected>" + rs.getString("COUNTRY") + "</option>");
out.println("<option "+s+">India</option>");
s =rs.getString("COUNTRY").equals("US")?"Selected":"";
out.println("<option "+s+">US</option>");
s =rs.getString("COUNTRY").equals("Germany")?"Selected":"";
out.println("<option "+s+">Germany</option>");
out.println("</select>");
out.println("</td>"); out.println("</tr>");

out.println("<tr>");
out.println("<td colspan=\"2\">");
out.println("<input type=\"submit\" name=\"submit\" value=\"Edit & Save Employee\">");
out.println("</td>");
out.println("</tr>");
out.println("</table>");
out.println(" </form>");
out.println("</body>");
out.println("</html>");
} catch (ClassNotFoundException ex) {
Logger.getLogger(UpdateEmployee.class.getName()).log(Level.SEVERE, null, ex);
} catch (SQLException ex) {
```

```
Logger.getLogger(UpdateEmployee.class.getName()).log(Level.SEVERE, null, ex);
}
}
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
processRequest(request, response);
}
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
processRequest(request, response);
}
@Override
public String getServletInfo() { return "Short description";
}
}
```

SaveEmployee.java

```
package com.mycompany.pb5;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(name = "SaveEmployee", urlPatterns = {"/SaveEmployee"})
public class SaveEmployee extends HttpServlet {
protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
response.setContentType("text/html;charset=UTF-8");
try (PrintWriter out = response.getWriter()) {
String id =request.getParameter("id");
String ename = request.getParameter("ename");
String password = request.getParameter("password");
String email = request.getParameter("email");
String country = request.getParameter("country");
Class.forName("org.apache.derby.jdbc.ClientDriver");
Connection con =
DriverManager.getConnection("jdbc:derby://localhost:1527/EMPDB","vcp","vcp");
Statement stmt =con.createStatement();
String sql="Update vcp.EMP Set ENAME='"+ename+"', PASSWORD='"+password+"' , EMAIL='"+email+"' , COUNTRY='"+country+"' Where ID="+id;
int r=stmt.executeUpdate(sql);
```

```
out.println("<!DOCTYPE html>");
out.println("<html>");
out.println("<head>");
out.println("<title>Servlet SaveEmployee</title>");
out.println("</head>");
out.println("<body>");
if(r==1)
out.println("<h1>Record Saved</h1>"); else
out.println("<h1>Error in Saving Record</h1>");
out.println("<a href=\"index.html\"> Home</a>");
out.println("</body>");
out.println("</html>");
} catch (ClassNotFoundException ex) {
Logger.getLogger(SaveEmployee.class.getName()).log(Level.SEVERE, null, ex);
} catch (SQLException ex) {
Logger.getLogger(SaveEmployee.class.getName()).log(Level.SEVERE, null, ex);
}
}
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
processRequest(request, response);
}
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
processRequest(request, response);
}
@Override
public String getServletInfo() { return "Short description";
}
}
```

ViewEmployee.java

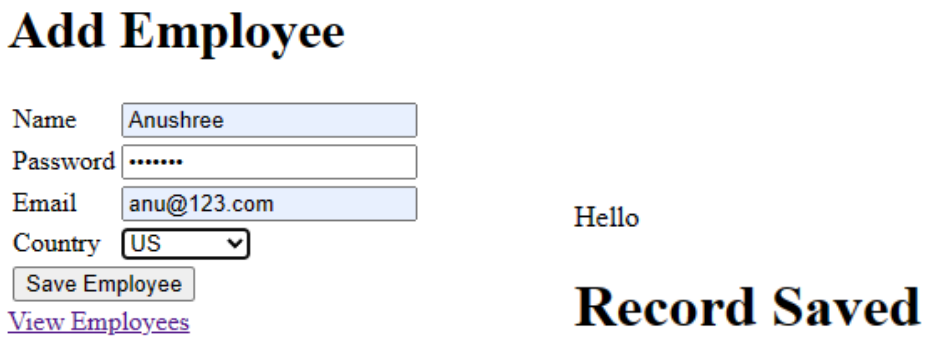
```
package com.mycompany.pb5;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(name = "ViewEmployee", urlPatterns = {"/ViewEmployee"})
public class ViewEmployee extends HttpServlet {
```

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException, ClassNotFoundException, SQLException {
response.setContentType("text/html;charset=UTF-8");
try (PrintWriter out = response.getWriter()) {
Class.forName("org.apache.derby.jdbc.ClientDriver");
Connection con
=DriverManager.getConnection("jdbc:derby://localhost:1527/EMPDB","vcp","vcp");
Statement stmt =con.createStatement();
String sql="Select * from vcp.EMP";
ResultSet rs=stmt.executeQuery(sql);
out.println("<!DOCTYPE html>");
out.println("<html>");
out.println("<head>");
out.println("<title>Servlet ViewEmployee</title>");
out.println("</head>");
out.println("<body>");
out.println("<h1>Employee List</h1>");
out.println("<table border=\"2\">");
out.println("<tr>");
out.println("<th>ID</th><th>Name</th><th>Password</th><th>email</th><th>country</t
h>");
out.println("</tr>");
while(rs.next())
{
out.println("<tr>");
out.println("<td>"+rs.getString("ID")+ "</td>");
out.println("<td>"+rs.getString("ENAME")+ "</td>");
out.println("<td>"+rs.getString("PASSWORD")+ "</td>");
out.println("<td>"+rs.getString("EMAIL")+ "</td>");
out.println("<td>"+rs.getString("COUNTRY")+ "</td>");
out.println("<td> <a href=\"UpdateEmployee?id="+rs.getString("ID")+"\">Edit</a></td>");
out.println("</tr>");
}
out.println("</table>");
out.println("</body>");
out.println("</html>");
stmt.close();
con.close();
}
}
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
try {
processRequest(request, response);
} catch (ClassNotFoundException ex) {
Logger.getLogger(ViewEmployee.class.getName()).log(Level.SEVERE, null, ex);
} catch (SQLException ex) {
Logger.getLogger(ViewEmployee.class.getName()).log(Level.SEVERE, null, ex);
}
}
@Override
```

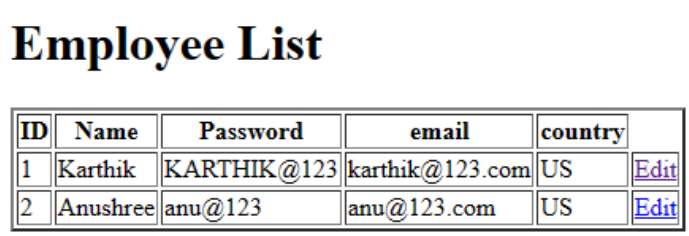
```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
try {
processRequest(request, response);
} catch (ClassNotFoundException ex) {
Logger.getLogger(ViewEmployee.class.getName()).log(Level.SEVERE, null, ex);
} catch (SQLException ex) {
Logger.getLogger(ViewEmployee.class.getName()).log(Level.SEVERE, null, ex);
}
}
@Override
public String getServletInfo() { return "Short description";
}
}
```

OUTPUT:

AddEmployee:



ViewEmployee:



UpdateEmployee:

6. Write a java JSP program to get student information through a HIML and create a JAVA Bean Class, populate Bean and Display the same information through another JSP.

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Student Information Form</title>
</head>
<body>
<h2>Student Information Form</h2>
<form action="Process_student.jsp" method="post">
<table>
<tr>
<td><label for="firstName">First Name:</label></td>
<td><input type="text" id="firstName" name="firstName"></td>
</tr>
<tr>
<td><label for="surname">Last Name:</label></td>
<td><input type="text" id="surname" name="surname"></td>
</tr>
<tr>
<td><label for="usn">USN:</label></td>
<td><input type="text" id="usn" name="usn"></td>
</tr>
<tr>
<td><label for="course">Course:</label></td>
<td><input type="text" id="course" name="course"></td>
</tr>
<tr>
<td><label for="sem">Semester:</label></td>
<td><input type="text" id="sem" name="sem"></td>
</tr>
<tr>
<td><label for="age">Age:</label></td>
<td><input type="text" id="age" name="age"></td>
</tr>
<tr>
<td><label for="address">Address:</label></td>
<td><input type="text" id="address" name="address"></td>
</tr>
</table>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Process_student.jsp

```
<%@ page import="student.Student" %>
```

```
<%@ page import="java.io.*" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%
// Create a new instance of Student Bean
Student student = new Student();

// Populate the bean with form data
student.setFirstName(request.getParameter("firstName"));
student.setSurname(request.getParameter("surname"));
student.setUsn(request.getParameter("usn"));
student.setCourse(request.getParameter("course"));
student.setSem(request.getParameter("sem"));
student.setAge(request.getParameter("age"));
student.setAddress(request.getParameter("address"));

// Store the bean in session
session.setAttribute("student", student);
%>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Student Information Processed</title>
</head>
<body>
<h2>Student Information Processed</h2>
<p>Student information has been stored.</p>
<a href="display_student.jsp">View Student Information</a>
</body>
</html>
```

display_student.jsp

```
<%@ page import="student.Student" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Student Information</title>
</head>
<body>
<h2>Student Information</h2>
<%
 // Retrieve the stored student bean from session
Student student = (Student) session.getAttribute("student");
%>
```

```
<p>First Name: <%= student.getFirstName() %></p>
<p>Last Name: <%= student.getSurname() %></p>
<p>USN: <%= student.getUsn() %></p>
<p>Course: <%= student.getCourse() %></p>
<p>Semester: <%= student.getSem() %></p>
<p>Age: <%= student.getAge() %></p>
<p>Address: <%= student.getAddress() %></p>
</body>
</html>
```

Student.java(Create package named student , under student package add java class named Student)

```
package student;
import java.io.Serializable;
public class Student implements Serializable {
private String firstName;
private String surname;
private String usn;
private String course;
private String sem;
private String age;
private String address;

// Getters and Setters
public String getFirstName() {
return firstName;
}
public void setFirstName(String firstName) {
this.firstName = firstName;
}

public String getSurname() {
return surname;
}
public void setSurname(String surname) {
this.surname = surname;
}

public String getUsn() {
return usn;
}
public void setUsn(String usn) {
this.usn = usn;
}

public String getCourse() {
return course;
}
```

```
public void setCourse(String course) {
this.course = course;
}

public String getSem() {
return sem;
}
public void setSem(String sem) {
this.sem = sem;
}

public String getAge() {
return age;
}
public void setAge(String age) {
this.age = age;
}

public String getAddress() {
return address;
}
public void setAddress(String address) {
this.address = address;
}
}
```

OUTPUT:

Student Information Form

First Name: Anushree
Last Name: K
USN: 1001
Course: BCA
Semester: 6
Age: 20
Address: Puttur
Submit

Student Information Processed

Student information has been stored.

View Student Information

Display:

Student Information

First Name: Anushree

Last Name: K

USN: 1001

Course: BCA

Semester: 6

Age: 20

Address: Puttur

7. Write a menu driven program to create a linked list and perform the following operations.

f. To Insert some Elements at the Specified Position

g. Swap two elements in a linked list

h. To Iterate a LinkedList in Reverse Order

i. To Compare Two LinkedList

j. To Convert a LinkedList to ArrayList

LList.java

```
import java.util.*;
public class LList {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
LinkedList<Integer> linkedList = new LinkedList<>();
int choice;
do {
System.out.println("\nMenu:");
System.out.println("1. Create linked list");
System.out.println("2. Display linked list");
System.out.println("3. Insert elements at specified position");
System.out.println("4. Swap two elements");
System.out.println("5. Iterate the linked list in reverse order");
System.out.println("6. Compare two linked lists");
System.out.println("7. Convert linked list to array list");
System.out.println("8. Exit");
System.out.print("Enter your choice: ");
choice = scanner.nextInt();
switch (choice) {
case 1:
createLinkedList(linkedList, scanner);
break;
case 2:
displayLinkedList(linkedList);
break;
case 3:
insertElementAtPosition(linkedList, scanner);
break;
case 4:
swapElements(linkedList, scanner);
break;
case 5:
iterateReverseOrder(linkedList);
break;
case 6:
compareLinkedLists(linkedList, scanner);
break;
case 7:
```

```
convertToArrayList(linkedList);
break;
case 8:
System.out.println("Exiting program...");
break;
default:
System.out.println("Invalid choice! Please enter a valid option.");
}
} while (choice != 8);
scanner.close();
}
private static void createLinkedList(LinkedList<Integer> linkedList, Scanner scanner) {
linkedList.clear();
System.out.print("Enter the number of elements: ");
int n = scanner.nextInt();
System.out.println("Enter the elements:");
for (int i = 0; i < n; i++) {
linkedList.add(scanner.nextInt());
}
System.out.println("Linked list created successfully!");
}
private static void displayLinkedList(LinkedList<Integer> linkedList) {
if (linkedList.isEmpty()) {
System.out.println("Linked list is empty.");
} else {
System.out.println("Linked list:");
for (Integer element : linkedList) {
System.out.print(element + " ");
}
System.out.println();
}
}
private static void insertElementAtPosition(LinkedList<Integer> linkedList, Scanner scanner)
{
if (linkedList.isEmpty()) {
System.out.println("Linked list is empty. Please create the linked list first.");
return;
}
System.out.print("Enter the element to insert: ");
int element = scanner.nextInt();
System.out.print("Enter the position to insert at: ");
int position = scanner.nextInt();
linkedList.add(position, element);
System.out.println("Element inserted successfully!");
}

private static void swapElements(LinkedList<Integer> linkedList, Scanner scanner) {
if (linkedList.isEmpty()) {
```

```
System.out.println("Linked list is empty. Please create the linked list first.");
return;
}
System.out.print("Enter the first position to swap: ");
int firstIndex = scanner.nextInt();
System.out.print("Enter the second position to swap: ");
int secondIndex = scanner.nextInt();
Collections.swap(linkedList, firstIndex, secondIndex);
System.out.println("Elements swapped successfully!");
}
private static void iterateReverseOrder(LinkedList<Integer> linkedList) {
        if (linkedList.isEmpty()) {
System.out.println("Linked list is empty. Please create the linked list first.");
return;
}
System.out.println("Linked list in reverse order:");
ListIterator<Integer> iterator = linkedList.listIterator(linkedList.size());
while (iterator.hasPrevious()) {
System.out.print(iterator.previous() + " ");
}
System.out.println();
}
private static void compareLinkedLists(LinkedList<Integer> linkedList, Scanner scanner)
{
if (linkedList.isEmpty()) {
System.out.println("Linked list is empty. Please create the linked list first."); return;
}
LinkedList<Integer> secondList = new LinkedList<>(linkedList);
System.out.println("Enter number of elements for second linked list:");
int count = scanner.nextInt();
System.out.println("Enter elements :");
for (int i = 0; i < count; i++) {
secondList.add(scanner.nextInt());
```

```
}
boolean isEqual = linkedList.equals(secondList);
if (isEqual) {
System.out.println("The two linked lists are equal.");
} else {
System.out.println("The two linked lists are not equal.");
}
}
private static void convertToArrayList(LinkedList<Integer> linkedList) {
if (linkedList.isEmpty()) {
System.out.println("Linked list is empty. Please create the linked list first.");
return;
}
ArrayList<Integer> arrayList = new ArrayList<>(linkedList);
System.out.println("Linked list converted to array list: " + arrayList);
}
}
```

OUTPUT:

```
Menu:
1. Create linked list
2. Display linked list
3. Insert elements at specified position
4. Swap two elements
5. Iterate the linked list in reverse order
6. Compare two linked lists
7. Convert linked list to array list
8. Exit
Enter your choice: 1
Enter the number of elements: 4
Enter the elements:
10 20 30 40
```

Linked list created successfully!

Menu:

1. Create linked list

2. Display linked list

3. Insert elements at specified position

4. Swap two elements

5. Iterate the linked list in reverse order

6. Compare two linked lists

7. Convert linked list to array list

8. Exit

Enter your choice: 2

Linked list:

10 20 30 40

Menu:

1. Create linked list

2. Display linked list

3. Insert elements at specified position

4. Swap two elements

5. Iterate the linked list in reverse order

6. Compare two linked lists

7. Convert linked list to array list

8. Exit

Enter your choice: 3

Enter the element to insert: 45

Enter the position to insert at: 4

Element inserted successfully!

Menu:

1. Create linked list

2. Display linked list

3. Insert elements at specified position

4. Swap two elements

5. Iterate the linked list in reverse order

6. Compare two linked lists

7. Convert linked list to array list

8. Exit

Enter your choice: 2

Linked list:

10 20 30 40 45

Menu:

1. Create linked list

2. Display linked list

3. Insert elements at specified position

4. Swap two elements

5. Iterate the linked list in reverse order

6. Compare two linked lists

7. Convert linked list to array list

8. Exit

Enter your choice: 4

Enter the first position to swap: 3

Enter the second position to swap: 2

Elements swapped successfully!

Menu:

1. Create linked list

2. Display linked list

3. Insert elements at specified position

4. Swap two elements

5. Iterate the linked list in reverse order

6. Compare two linked lists

7. Convert linked list to array list

8. Exit

Enter your choice: 2

Linked list:

10 20 40 30 45

Menu:

1. Create linked list

2. Display linked list

3. Insert elements at specified position

4. Swap two elements

5. Iterate the linked list in reverse order

6. Compare two linked lists

7. Convert linked list to array list

8. Exit

Enter your choice: 5

Linked list in reverse order:

45 30 40 20 10

Menu:

1. Create linked list

2. Display linked list

3. Insert elements at specified position

4. Swap two elements

5. Iterate the linked list in reverse order

6. Compare two linked lists

7. Convert linked list to array list

8. Exit

Enter your choice: 6

Enter number of elements for second linked list:

5

Enter elements :

10 20 30 40 50

The two linked lists are not equal.

Menu:

1. Create linked list

2. Display linked list

3. Insert elements at specified position

4. Swap two elements

5. Iterate the linked list in reverse order

6. Compare two linked lists

7. Convert linked list to array list

8. Exit

Enter your choice: 7

Linked list converted to array list: [10, 20, 40, 30, 45]

Menu:

1. Create linked list

2. Display linked list

3. Insert elements at specified position

4. Swap two elements

5. Iterate the linked list in reverse order

6. Compare two linked lists

7. Convert linked list to array list

8. Exit

Enter your choice: 8

Exiting program...

Press any key to continue . . .

8. Implement a java application based on the MVC design pattern.

Input student Rollno, name ,marks in three subject calculate result and grade and display the result in neat format.

Percentage of Marks	Grade
Above 90%	A
80% to 90%	B
70% to 80%	C
60% to 70%	D
Below 60%	E

Student.java

```
package MVC;
class Student {
private String rollNo;
private String name;
private int[] marks;
public Student(String rollNo, String name, int[] marks) {
this.rollNo = rollNo;
this.name = name;
this.marks = marks;
}
public double calculatePercentage() {
int totalMarks = 0;
for (int mark : marks) {
totalMarks += mark;
}
return (double) totalMarks / marks.length;
}
public char calculateGrade() {
double percentage = calculatePercentage();
if (percentage > 90) {
return 'A';
} else if (percentage >= 80) {
return 'B';
} else if (percentage >= 70) {
return 'C';
```

```
} else if (percentage >= 60) {
return 'D';
} else {
return 'E';
}
}
public String getRollNo() {
return rollNo;
}
public String getName() {
return name;
}
 public int[] getMarks() {
return marks;
}
}
```

StudentView.java

```
package MVC;
class StudentView {
public void displayStudentDetails(Student student) {
System.out.println("Student Details:");
System.out.println("Roll No: " + student.getRollNo());
System.out.println("Name: " + student.getName());
System.out.println("Marks:");
int[] marks = student.getMarks();
System.out.println("Subject 1: " + marks[0]);
System.out.println("Subject 2: " + marks[1]);
System.out.println("Subject 3: " + marks[2]);
System.out.println("Percentage: " + student.calculatePercentage());
System.out.println("Grade: " + student.calculateGrade());
}
}
```

StudentController.java

```
package MVC;
```

```
class StudentController {
private Student model;
private StudentView view;
public StudentController(Student model, StudentView view) {
this.model = model;
this.view = view;
}
public void updateView() {
view.displayStudentDetails(model);
}
}
```

MVCDemo.java

```
package MVC;
public class MVCDemo {
public static void main(String[] args) {
// Create a student object
int[] marks = {85, 75, 90};
Student student = new Student("123456", "Vivek", marks);
// Create view to display student details
StudentView view = new StudentView();
// Create controller
StudentController controller = new StudentController(student, view);
// Display student details
controller.updateView();
}
}
```

OUTPUT:

Student Details:

Roll No: 123456

Name: Vivek

Marks:

Subject 1: 85

Subject 2: 75

Subject 3: 90

Percentage: 83.33333333333333

Grade: B