Slip-1

1. Write a Java program to display all the alphabets between ‘A’ to ‘Z’ after every 2 seconds

→

```
public class slip1_a {
    public static void main(String[] args) {
        char start = 'A';
        char end = 'Z';

        for (char c = start; c <= end; c++) {
            System.out.print(c + " ");
            try {
                Thread.sleep(2000); // Pauses for 2
seconds
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

2. Write a Java program to accept the details of Employee (Eno, EName, Designation, Salary) from a user and store it into the database. (Use Swing)

```
-->import javax.swing.*;

import java.awt.*;

import java.awt.event.*;

import java.sql.*;


public class slip1_b extends JFrame {

    private JTextField txtEno, txtEName, txtDesignation, txtSalary;

    private JButton btnSave;
```

```
public slip1_b() {
    setTitle("Employee Details Form");
    setSize(400, 200);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JPanel panel = new JPanel(new GridLayout(5, 2));

    panel.add(new JLabel("Employee No:"));
    txtEno = new JTextField();
    panel.add(txtEno);

    panel.add(new JLabel("Employee Name:"));
    txtEName = new JTextField();
    panel.add(txtEName);

    panel.add(new JLabel("Designation:"));
    txtDesignation = new JTextField();
    panel.add(txtDesignation);

    panel.add(new JLabel("Salary:"));
    txtSalary = new JTextField();
    panel.add(txtSalary);

    btnSave = new JButton("Save");
    btnSave.addActionListener(new SaveButtonListener());
    panel.add(btnSave);
```

```
        add(panel);
    }

    private class SaveButtonListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            String eno = txtEno.getText();
            String ename = txtEName.getText();
            String designation = txtDesignation.getText();
            String salary = txtSalary.getText();

            try {
                // Connect to your PostgreSQL database
                Connection conn = DriverManager.getConnection(
"jdbc:postgresql://localhost:5432/your_database_name",
                        "your_username",
                        "your_password");

                Statement stmt = conn.createStatement();
                String sql = "INSERT INTO Employee(Eno, EName,
Designation, Salary) " +
                             "VALUES('" + eno + "', '" + ename + "',
'" + designation + "', '" + salary + "')";
                stmt.executeUpdate(sql);
                JOptionPane.showMessageDialog(slip1_b.this, "Employee
details saved successfully!");
                stmt.close();
```

```
                conn.close();
            } catch (SQLException ex) {
                JOptionPane.showMessageDialog(slip1_b.this, "Error
saving employee details: " + ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
            }
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            slip1_b form = new slip1_b();
            form.setVisible(true);
        });
    }
}
```

Slip-2

1. Write a java program to read ‘N’ names of your friends, store it into HashSet and display them in ascending order

→

```
import java.util.HashSet;
import java.util.Scanner;
import java.util.TreeSet;

public class FriendNamesAscending {

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

```
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of friends: ");
        int n = scanner.nextInt();
        scanner.nextLine(); // Consume newline character

        HashSet<String> friendSet = new HashSet<>();

        for (int i = 0; i < n; i++) {
            System.out.print("Enter name of friend " + (i + 1) + ":
");
            String name = scanner.nextLine();
            friendSet.add(name);
        }

        TreeSet<String> sortedFriends = new TreeSet<>(friendSet);

        System.out.println("\nList of Friends in Ascending Order:");
        for (String friend : sortedFriends) {
            System.out.println(friend);
        }

        scanner.close();
    }
}
```

Slip-3

1.Write a JSP program to display the details of Patient (PNo, PName, Address, age, disease) in tabular form on browser.

```
→<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
```

```
    <title>Patient Details</title>
    <style>
        table {
            width: 50%;
            border-collapse: collapse;
            margin: 20px;
        }
        th, td {
            border: 1px solid black;
            padding: 8px;
            text-align: left;
        }
        th {
            background-color: #f2f2f2;
        }
    </style>
</head>
<body>
    <h2>Patient Details</h2>
    <table>
        <tr>
            <th>Patient Number</th>
            <th>Name</th>
            <th>Address</th>
            <th>Age</th>
            <th>Disease</th>
        </tr>
        <%
            // Sample Patient data (replace with your actual data)
            String[][] patients = {
                {"P001", "Alice", "123 Main St, City", "35", "Fever"},
                {"P002", "Bob", "456 Elm St, Town", "45", "Headache"},
                {"P003", "Charlie", "789 Oak St, Village", "28",
"Allergy"}
            };

            // Loop through each patient and display details in table
rows
            for (String[] patient : patients) {
        %>
        <tr>
            <td><%= patient[0] %></td>
            <td><%= patient[1] %></td>
            <td><%= patient[2] %></td>
            <td><%= patient[3] %></td>
            <td><%= patient[4] %></td>
```

```
        </tr>
        <% } %>
    </table>
</body>
</html>
```

2. Write a Java program to create LinkedList of String objects and perform the following:

i. Add element at the end of the list

ii. Delete first element of the list

iii. Display the contents of list in reverse order

```
→import java.util.LinkedList;
import java.util.ListIterator;

public class StringLinkedListOperations {
    public static void main(String[] args) {
        LinkedList<String> linkedList = new LinkedList<>();
        linkedList.add("Apple");
        linkedList.add("Banana");
        linkedList.add("Cherry");

        System.out.println("LinkedList after adding elements:");
        System.out.println(linkedList);

        if (!linkedList.isEmpty()) {
            linkedList.removeFirst();
            System.out.println("\nLinkedList after removing the first element:");
            System.out.println(linkedList);
        } else {
            System.out.println("\nLinkedList is empty. Cannot remove the first element.");
        }

        System.out.println("\nContents of the list in reverse order:");
        ListIterator<String> iterator = linkedList.listIterator(linkedList.size());
        while (iterator.hasPrevious()) {
            System.out.println(iterator.previous());
        }
    }
}
```

Slip-4

1. Write a Java program using Runnable interface to blink Text on the frame. [15 M]

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

public class BlinkingText implements Runnable {
    private JLabel label;
    private boolean isBlinking;

    public BlinkingText() {
        label = new JLabel("Blinking Text");
        label.setFont(new Font("Arial", Font.BOLD, 20));
        isBlinking = true;
    }

    @Override
    public void run() {
        while (isBlinking) {
            try {
                label.setVisible(!label.isVisible());
                Thread.sleep(500);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    public void stopBlinking() {
        isBlinking = false;
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            JFrame frame = new JFrame("Blinking Text");
            BlinkingText blinkingText = new BlinkingText();
            Thread thread = new Thread(blinkingText);

            JPanel panel = new JPanel();
            panel.setLayout(new FlowLayout());
            panel.add(blinkingText.label);

            JButton stopButton = new JButton("Stop Blinking");
            stopButton.addActionListener(e -> {
                blinkingText.stopBlinking();
                stopButton.setEnabled(false);
            });
            panel.add(stopButton);

            frame.add(panel);
            frame.setSize(300, 100);
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setVisible(true);

            thread.start();
        });
    }
}
```

2. Write a Java program to store city names and their STD codes using an appropriate collection and perform following operations:

i. Add a new city and its code (No duplicates

) ii. Remove a city from the collection

iii. Search for a city name and display the code

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

public class BlinkingText implements Runnable {
    private JLabel label;
    private boolean isBlinking;

    public BlinkingText() {
        label = new JLabel("Blinking Text");
        label.setFont(new Font("Arial", Font.BOLD, 20));
        isBlinking = true;
    }

    @Override
    public void run() {
        while (isBlinking) {
            try {
                label.setVisible(!label.isVisible());
                Thread.sleep(500);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    public void stopBlinking() {
        isBlinking = false;
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            JFrame frame = new JFrame("Blinking Text");
            BlinkingText blinkingText = new BlinkingText();
            Thread thread = new Thread(blinkingText);

            JPanel panel = new JPanel();
            panel.setLayout(new FlowLayout());
            panel.add(blinkingText.label);

            JButton stopButton = new JButton("Stop Blinking");
            stopButton.addActionListener(e -> {
                blinkingText.stopBlinking();
                stopButton.setEnabled(false);
            });
            panel.add(stopButton);

            frame.add(panel);
            frame.setSize(300, 100);
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setVisible(true);

            thread.start();
        });
    }
}
```

SLIP-5

1. Write a Java Program to create the hash table that will maintain the mobile number and student name. Display the details of student using Enumeration interface

→

```java
import java.util.Hashtable;
import java.util.Enumeration;

public class StudentHashTable {
    public static void main(String[] args) {
        Hashtable<String, String> studentDetails = new Hashtable<>();


        studentDetails.put("1234567890", "Alice");
        studentDetails.put("9876543210", "Bob");
        studentDetails.put("5678901234", "Charlie");


        System.out.println("Details of Students:");
        System.out.println("====================");
        Enumeration<String> mobileNumbers = studentDetails.keys();
        while (mobileNumbers.hasMoreElements()) {
            String mobileNumber = mobileNumbers.nextElement();
            String studentName = studentDetails.get(mobileNumber);
            System.out.println("Mobile Number: " + mobileNumber + ", Student Name: " + studentName);
        }
    }
}
```

2. Create a JSP page for an online multiple choice test. The questions are randomly selected from a database and displayed on the screen. The choices are displayed using radio buttons. When the user clicks on next, the next question is displayed. When the user clicks on submit, display the total score on the screen

→

Slip-6

1. Write a Java program to accept ‘n’ integers from the user and store them in a collection. Display them in the sorted order. The collection should not accept duplicate elements. (Use a suitable collection). Search for a particular element using predefined search method in the Collection framework.

→

```
import java.util.*;

public class IntegerCollection {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        TreeSet<Integer> numberSet = new TreeSet<>();

        System.out.print("Enter the number of integers (n): ");
        int n = scanner.nextInt();

        System.out.println("Enter " + n + " integers:");

        for (int i = 0; i < n; i++) {
            int num = scanner.nextInt();
            numberSet.add(num);
        }


        System.out.println("Integers in sorted order:");
        for (int num : numberSet) {
            System.out.print(num + " ");
        }
        System.out.println();


        System.out.print("Enter the element to search: ");
        int searchNum = scanner.nextInt();

        if (numberSet.contains(searchNum)) {
            System.out.println("Element " + searchNum + " found in the collection.");
        } else {
            System.out.println("Element " + searchNum + " not found in the collection.");
        }

        scanner.close();
    }
}
```

2. Write a java program to simulate traffic signal using threads. [15 M]

→

```
public class slip6_b {
    private static final int RED_TIME = 5000;     // 5 seconds
    private static final int YELLOW_TIME = 2000;  // 2 seconds
    private static final int GREEN_TIME = 5000;   // 5 seconds

    public static void main(String[] args) {
        Thread redThread = new Thread(new RedSignal());
        Thread yellowThread = new Thread(new YellowSignal());
```

```
        Thread greenThread = new Thread(new GreenSignal());

        redThread.start();
        yellowThread.start();
        greenThread.start();
    }

    static class RedSignal implements Runnable {
        public void run() {
            while (true) {
                try {
                    System.out.println("Red Signal: Stop");
                    Thread.sleep(RED_TIME);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    static class YellowSignal implements Runnable {
        public void run() {
            while (true) {
                try {
                    System.out.println("Yellow Signal: Prepare to move");
                    Thread.sleep(YELLOW_TIME);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    static class GreenSignal implements Runnable {
        public void run() {
            while (true) {
                try {
                    System.out.println("Green Signal: Go");
                    Thread.sleep(GREEN_TIME);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

Slip-7

1. Write a java program that implements a multi-thread application that has three threads. First thread generates random integer number after every one second, if the number is even; second thread computes the square of that number and print it. If the number is odd, the third thread computes the of cube of that number and print it. [15 M]

→

```java
import java.util.Random;

public class NumberProcessor {
    public static void main(String[] args) {
        NumberGenerator numberGenerator = new NumberGenerator();
        NumberSquareCalculator squareCalculator = new NumberSquareCalculator();
        NumberCubeCalculator cubeCalculator = new NumberCubeCalculator();

        Thread generatorThread = new Thread(numberGenerator);
        Thread squareThread = new Thread(squareCalculator);
        Thread cubeThread = new Thread(cubeCalculator);

        generatorThread.start();
        squareThread.start();
        cubeThread.start();
    }

    static class NumberGenerator implements Runnable {
        public void run() {
            Random random = new Random();
            while (true) {
                try {
                    int randomNumber = random.nextInt(100); // Generate random number between 0 and 99
                    System.out.println("Generated number: " + randomNumber);

                    if (randomNumber % 2 == 0) {
                        synchronized (NumberSquareCalculator.lock) {
                            NumberSquareCalculator.number = randomNumber;
                            NumberSquareCalculator.lock.notify();
                        }
                    } else {
                        synchronized (NumberCubeCalculator.lock) {
                            NumberCubeCalculator.number = randomNumber;
                            NumberCubeCalculator.lock.notify();
                        }
                    }

                    Thread.sleep(1000); // Sleep for 1 second
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    static class NumberSquareCalculator implements Runnable {
        static int number;
        static final Object lock = new Object();

        public void run() {
            while (true) {
                synchronized (lock) {
                    try {
                        lock.wait();
                        int square = number * number;
```

```
                    System.out.println("Square of " + number + ": " + square);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    static class NumberCubeCalculator implements Runnable {
        static int number;
        static final Object lock = new Object();

        public void run() {
            while (true) {
                synchronized (lock) {
                    try {
                        lock.wait();
                        int cube = number * number * number;
                        System.out.println("Cube of " + number + ": " + cube);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        }
    }
}
```

2. Write a java program for the following

: i. To create a Product(Pid, Pname, Price) table

. ii. Insert at least five records into the table

. iii. Display all the records from a table

```
import java.sql.*;

public class ProductTableDemo {

    static final String DB_URL = "jdbc:postgresql://localhost:5432/your_database";
    static final String USER = "username";
    static final String PASS = "password";

    public static void main(String[] args) {
        try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASS)) {
            if (conn != null) {
                System.out.println("Connected to the PostgreSQL database!");


                insertRecords(conn);


                displayRecords(conn);
            }
        } catch (SQLException e) {
```

```
            e.printStackTrace();
        }
    }

    private static void insertRecords(Connection conn) throws SQLException {
        String[] products = {
            "('101', 'Product A', 50.00)",
            "('102', 'Product B', 75.50)",
            "('103', 'Product C', 100.25)",
            "('104', 'Product D', 120.75)",
            "('105', 'Product E', 90.99)"
        };

        String sql = "INSERT INTO Product (Pid, Pname, Price) VALUES ";

        try (Statement stmt = conn.createStatement()) {
            for (String product : products) {
                stmt.addBatch(sql + product);
            }
            stmt.executeBatch();
            System.out.println("Records inserted into Product table!");
        }
    }

    private static void displayRecords(Connection conn) throws SQLException {
        String sql = "SELECT * FROM Product";

        try (Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {

            System.out.println("Product Table:");
            System.out.println("===============");
            System.out.println("Pid\tPname\tPrice");
            while (rs.next()) {
                int pid = rs.getInt("Pid");
                String pname = rs.getString("Pname");
                double price = rs.getDouble("Price");
                System.out.println(pid + "\t" + pname + "\t" + price);
            }
        }
    }
}
```

SQL file

```
CREATE TABLE IF NOT EXISTS Product (
    Pid SERIAL PRIMARY KEY,
    Pname VARCHAR(255) NOT NULL,
    Price DECIMAL(10, 2) NOT NULL
);
```

1. Write a java program to define a thread for printing text on output screen for ‘n’ number of times. Create 3 threads and run them. Pass the text ‘n’ parameters to the thread constructor. Example:

i. First thread prints “COVID19” 10 times.

ii. Second thread prints “LOCKDOWN2020” 20 times

iii. Third thread prints “VACCINATED2021” 30 times [15 M]

→

```
public class A1 extends Thread {
    String str;
    int n;

    A1(String str, int n) {
        this.str = str;
        this.n = n;
    }

    public void run() {
        try {
            for (int i = 0; i < n; i++) {
                System.out.println(getName() + " : " + str);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public static void main(String[] args) {
        A1 t1 = new A1("COVID19", 10);
        A1 t2 = new A1("LOCKDOWN2020", 20);
        A1 t3 = new A1("VACCINATED", 30);

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

    }
}
```

2.Write a JSP program to check whether a given number is prime or not. Display the result in red color.

```
→<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Prime Number Checker</title>
</head>
<body>
    <h1>Prime Number Checker</h1>


    <% String numberStr = request.getParameter("number"); %>
```

```
    <% if (numberStr != null && !numberStr.isEmpty()) { %>
        <%-- Convert the string number to an integer --%>
        <% int number = Integer.parseInt(numberStr); %>

        <% boolean isPrime = true; %>
        <% if (number <= 1) { %>
            <% isPrime = false; %>
        <% } else { %>
            <% for (int i = 2; i <= Math.sqrt(number); i++) { %>
                <% if (number % i == 0) { %>
                    <% isPrime = false; %>
                    <% break; %>
                <% } %>
            <% } %>
        <% } %>


        <p><font color="red"><%= number %> is <%= isPrime ? "prime" : "not prime" %></font></p>
    <% } else { %>

        <p>Please provide a number to check if it is prime.</p>
    <% } %>

    <form method="post" action="PrimeChecker.jsp">
        <label>Enter a number:</label>
        <input type="text" name="number">
        <input type="submit" value="Check">
    </form>
</body>
</html>
```

Slip-9→

1. Write a Java program to create a thread for moving a ball inside a panel vertically. The ball should be created when the user clicks on the start button. [15 M]

```
→import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class BallMovementGUI {
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            new BallMovementFrame();
        });
    }

    static class BallMovementFrame extends JFrame {
        private JPanel ballPanel;
        private JButton startButton;
        private Thread ballThread;

        public BallMovementFrame() {
            setTitle("Ball Movement");
            setSize(400, 400);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            ballPanel = new JPanel() {
                private int ballY = 180;

                @Override
                protected void paintComponent(Graphics g) {
                    super.paintComponent(g);
                    g.setColor(Color.RED);
                    g.fillOval(175, ballY, 50, 50);
                }

                public void moveBall() {
                    ballY -= 5;
                    if (ballY < 0) {
                        ballY = 180;
                    }
                    repaint();
                }
            };

            startButton = new JButton("Start");
            startButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    if (ballThread == null || !ballThread.isAlive()) {
                        ballThread = new Thread(() -> {
                            while (true) {
                                ballPanel.moveBall();
                                try {
                                    Thread.sleep(50);
                                } catch (InterruptedException ex) {
                                    ex.printStackTrace();
                                }
                            }
                        });
                        ballThread.start();
                    }
                }
            });

            setLayout(new BorderLayout());
```

```
            add(ballPanel, BorderLayout.CENTER);
            add(startButton, BorderLayout.SOUTH);

            setVisible(true);
        }
    }
}
```

Slip 9 And 10 are not solved because spring framework not explain in practicals

Slip-11

1. Design an HTML page which passes customer number to a search servlet. The servlet searches for the customer number in a database (customer table) and returns customer details if found the number otherwise display error message.

→

main.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Search Customer</title>
</head>
<body>
    <h2>Search Customer</h2>
    <form action="SearchCustomerServlet" method="get">
        <label for="custNumber">Customer Number:</label>
        <input type="text" id="custNumber" name="custNumber" required>
        <br><br>
        <input type="submit" value="Search">
    </form>
</body>
</html>
```

SearchCustomer.java→

```
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.*;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class SearchCustomerServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
```

```
        String custNumber = request.getParameter("custNumber");


        String url = "jdbc:postgresql://localhost:5432/your_database";
        String username = "username";
        String password = "password";

        try {

            Connection conn = DriverManager.getConnection(url, username, password);


            Statement stmt = conn.createStatement();


            String query = "SELECT * FROM customer WHERE custNumber = '" + custNumber + "'";
            ResultSet rs = stmt.executeQuery(query);

            if (rs.next()) {

                out.println("<h2>Customer Details</h2>");
                out.println("<p>Customer Number: " + rs.getString("custNumber") + "</p>");
                out.println("<p>Name: " + rs.getString("name") + "</p>");
                out.println("<p>Email: " + rs.getString("email") + "</p>");
                out.println("<p>Phone: " + rs.getString("phone") + "</p>");
            } else {

                out.println("<h2>Error</h2>");
                out.println("<p>Customer with number " + custNumber + " not found.</p>");
            }


            rs.close();
            stmt.close();
            conn.close();

        } catch (SQLException e) {
            out.println("<h2>Error</h2>");
            out.println("<p>Database Error: " + e.getMessage() + "</p>");
        }

        out.close();
    }
}
```

2. Write a Java program to display information about all columns in the DONAR table using ResultSetMetaData.

```
/**
 * @author Jadhav Rohan
 *
 * C].Write a program to display information about all coumns in the DONOR table using ResultSetMetaData.
 *
 *for database material use here slip11_b.sql file
**/

import java.sql.*;

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

```
        try {

            Class.forName("org.postgresql.Driver");


            Connection conn = DriverManager.getConnection("jdbc:postgresql://localhost/postgres",
"postgres", "dsk");

            Statement stmt = null;
            stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("select * from donor");

            ResultSetMetaData rsmd = rs.getMetaData();
            System.out.println("\t-------------------------------------------------");

            int count = rsmd.getColumnCount();
            System.out.println("\t No. of Columns: " + rsmd.getColumnCount());
            System.out.println("\t-------------------------------------------------");
            for (int i = 1; i <= count; i++)
            {
                System.out.println("\t\tColumn No : " + i);
                System.out.println("\t\tColumn Name : " + rsmd.getColumnName(i));
                System.out.println("\t\tColumn Type : " + rsmd.getColumnTypeName(i));
                System.out.println("\t\tColumn Display Size : " + rsmd.getColumnDisplaySize(i));
                System.out.println();
            }
            System.out.println("\t-------------------------------------------------");

            rs.close();
            stmt.close();
            conn.close();
        }
        catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Sql file

→create table donor(did int, dname char(22),daddr varchar(22));

insert into donor VALUES(1,'AAA','zzz');

insert into donor VALUES(2,'BBB','yyy');

insert into donor VALUES(3,'CCC','xxx');

insert into donor VALUES(4,'DDD','www');

slip-12

1. Write a JSP program to check whether given number is Perfect or not. (Use Include directive).

→

```
<%@ page language="java" %>
<!DOCTYPE html>
<html>
<head>
    <title>Perfect Number Checker</title>
</head>
<body>
    <h2>Perfect Number Checker</h2>

    <%
        boolean isPerfect(int num) {
            int sum = 1;
            for (int i = 2; i * i <= num; i++) {
                if (num % i == 0) {
                    sum += i;
                    if (i * i != num) {
                        sum += num / i;
                    }
                }
            }
            return sum == num && num != 1;
        }

        String numberStr = request.getParameter("number");
        if (numberStr != null && !numberStr.isEmpty()) {
            int number = Integer.parseInt(numberStr);
            if (isPerfect(number)) {
                out.println("<p>" + number + " is a Perfect Number.</p>");
            } else {
                out.println("<p>" + number + " is not a Perfect Number.</p>");
            }
        }
    %>

    <form method="post">
        Enter a number: <input type="text" name="number">
        <input type="submit" value="Check">
    </form>
</body>
</html>
```

2. Write a Java Program to create a PROJECT table with field's project_id, Project_name, Project_description, Project_Status. Insert values in the table. Display all the details of the PROJECT table in a tabular format on the screen.(using swing)

→

Skip this bcoz there are 360+ line in the code

Slip-13

1. Write a Java program to display information about the database and list all the tables in the database. (Use DatabaseMetaData).

```
/**
 * @author : Jadhav Rohan
 *
 * b) Write a program to display information about the database and list all the tables in the database.
(Use DatabaseMetaData).
 */

import java.sql.*;

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

                        Class.forName("org.postgresql.Driver");


                        Connection conn =
DriverManager.getConnection("jdbc:postgresql://localhost/postgres", "postgres", "dsk");
                        DatabaseMetaData dbmd = conn.getMetaData();
                        System.out.println("\t-------------------------------------------");
                        System.out.println("\t\tDriver Name : " + dbmd.getDriverName());
                        System.out.println("\t\tDriver Version : " + dbmd.getDriverVersion());
                        System.out.println("\t\tUserName : " + dbmd.getUserName());
                        System.out.println("\t\tDatabase Product Name : " +
dbmd.getDatabaseProductName());
                        System.out.println("\t\tDatabase Product Version : " +
dbmd.getDatabaseProductVersion());
                        System.out.println("\t-----------------------------");

                        String table[] = { "TABLE" };
                        ResultSet rs = dbmd.getTables(null, null, null, table);
                        System.out.println("\t\tTable Names:");

                        while (rs.next()) {
                                System.out.println(rs.getString("TABLE_NAME"));
                        }
                        rs.close();
                        conn.close();
                }
                catch (Exception e) {
                        System.out.println(e);
                }
        }/
}
```

2. Write a Java program to show lifecycle (creation, sleep, and dead) of a thread. Program should print randomly the name of thread and value of sleep time. The name of the thread should be hard coded through constructor. The sleep time of a thread will be a random integer in the range 0 to 4999.

→

```
import java.util.Random;

public class ThreadLifecycleDemo {
    public static void main(String[] args) {
        Thread thread1 = new MyThread("Thread 1");
        Thread thread2 = new MyThread("Thread 2");

        thread1.start();
        thread2.start();
    }

    static class MyThread extends Thread {
        private String threadName;
        private Random random = new Random();

        public MyThread(String name) {
            this.threadName = name;
        }

        @Override
        public void run() {
            System.out.println("Thread " + threadName + " created.");

            try {
                int sleepTime = random.nextInt(5000);
                System.out.println("Thread " + threadName + " will sleep for " + sleepTime + "
milliseconds.");
                Thread.sleep(sleepTime);
            } catch (InterruptedException e) {
                System.out.println("Thread " + threadName + " interrupted.");
            }

            System.out.println("Thread " + threadName + " is dead.");
        }
    }
}
```

Slip-14

1. Write a Java program for a simple search engine. Accept a string to be searched. Search the string in all text files in the current folder. Use a separate thread for each file. The result should display the filename and line number where the string is found.

→

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class SimpleSearchEngine {
    private static final String SEARCH_STRING = "search";
    private static final String FILE_EXTENSION = ".txt";

    public static void main(String[] args) {
        File folder = new File(".");
        File[] files = folder.listFiles();

        List<SearchThread> searchThreads = new ArrayList<>();

        if (files != null) {
            for (File file : files) {
                if (file.isFile() && file.getName().toLowerCase().endsWith(FILE_EXTENSION)) {
                    SearchThread searchThread = new SearchThread(file, SEARCH_STRING);
                    searchThreads.add(searchThread);
                    searchThread.start();
                }
            }
        }

        for (SearchThread thread : searchThreads) {
            try {
                thread.join();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    static class SearchThread extends Thread {
        private File file;
        private String searchString;

        public SearchThread(File file, String searchString) {
            this.file = file;
            this.searchString = searchString;
        }

        @Override
        public void run() {
            try (BufferedReader reader = new BufferedReader(new FileReader(file))) {
                String line;
                int lineNumber = 1;

                while ((line = reader.readLine()) != null) {
                    if (line.contains(searchString)) {
                        System.out.println("Found in file: " + file.getName() + ", Line: " + lineNumber +
": " + line);
                    }
                    lineNumber++;
```

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

TXt file

--:> This is a sample text file for testing.

It contains some lines of text with the word "search" in it.

This line does not have the search term.

Another line with the search term.

Here's a line without it.

And another line with the search term.

The end. e

2. Write a JSP program to calculate sum of first and last digit of a given number. Display sum in Red Color with font size 18.

```
<%@ page language="java" %>
<!DOCTYPE html>
<html>
<head>
    <title>Sum of First and Last Digit</title>
</head>
<body>
    <h2>Calculate Sum of First and Last Digit</h2>

    <%

        String numberStr = request.getParameter("number");
```

```
        if (numberStr != null && !numberStr.isEmpty()) {
            try {

                int number = Integer.parseInt(numberStr);


                int lastDigit = number % 10;
                int firstDigit = 0;


                while (number != 0) {
                    firstDigit = number % 10;
                    number /= 10;
                }


                int sum = firstDigit + lastDigit;


    %>
                <p style="color: red; font-size: 18px;">
                    Sum of First and Last Digit of <%= number %> is: <%= sum %>
                </p>
    <%
            } catch (NumberFormatException e) {

    %>
                <p style="color: red; font-size: 18px;">
                    Invalid input. Please enter a valid number.
                </p>
    <%
            }
        }
    %>

    <form method="post">
        Enter a number: <input type="text" name="number">
        <input type="submit" value="Calculate">
    </form>
</body>
</html>
```

Slip-15

→1. Write a java program to display name and priority of a Thread.

```
package practical;

public class Main {
    public static void main(String[] args) {
        Thread thread = Thread.currentThread();
        System.out.println("Current Thread: " + thread.getName());
        System.out.println("Thread Priority: " + thread.getPriority());
    }
}
```

2. Write a SERVLET program which counts how many times a user has visited a web page. If user is visiting the page for the first time, display a welcome message. If the user is revisiting the page, display the number of times visited. (Use Cookie)

→

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

@WebServlet("/VisitCounterServlet")
public class VisitCounterServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();


        Cookie[] cookies = request.getCookies();
        Cookie visitCookie = null;
        int visitCount = 0;


        if (cookies != null) {
            for (Cookie cookie : cookies) {
                if (cookie.getName().equals("visitCount")) {
                    visitCookie = cookie;
                    break;
                }
```

```
            }
        }


        if (visitCookie != null) {
            visitCount = Integer.parseInt(visitCookie.getValue());
        }


        visitCount++;


        visitCookie = new Cookie("visitCount", String.valueOf(visitCount));
        response.addCookie(visitCookie);

        out.println("<html><head><title>Visit Counter Servlet</title></head><body>");
        if (visitCount == 1) {
            out.println("<h2>Welcome! This is your first visit.</h2>");
        } else {
            out.println("<h2>You have visited this page " + visitCount + " times.</h2>");
        }
        out.println("</body></html>");
    }
}
```

Slip-16

1. Write a java program to create a TreeSet, add some colors (String) and print out the content of TreeSet in ascending order.

→
```java
import java.util.TreeSet;

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

        TreeSet<String> colors = new TreeSet<>();


        colors.add("Red");
        colors.add("Green");
        colors.add("Blue");
        colors.add("Yellow");
        colors.add("Orange");


        System.out.println("Colors in ascending order:");
        for (String color : colors) {
            System.out.println(color);
        }
    }
}
```

Slip-17

1. Write a java program to accept 'N' integers from a user. Store and display integers in sorted order having proper collection class. The collection should not accept duplicate elements.

→

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

public class SortedIntegers {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        TreeSet<Integer> numbers = new TreeSet<>();

        System.out.print("Enter the number of integers (N): ");
        int n = scanner.nextInt();

        System.out.println("Enter " + n + " integers:");
        for (int i = 0; i < n; i++) {
            int num = scanner.nextInt();
            numbers.add(num);
        }

        System.out.println("Integers in sorted order without duplicates:");
        for (int number : numbers) {
            System.out.println(number);
        }

        scanner.close();
    }
}
```

Slip-18

1. Write a java program to display name and priority of a Thread.

```
→public class ThreadInfo {
    public static void main(String[] args) {
        Thread thread = Thread.currentThread();
        System.out.println("Current Thread Name: " + thread.getName());
        System.out.println("Current Thread Priority: " + thread.getPriority());
    }
}
```

Slip-19

→

1. Write a java program to accept ‘N’ Integers from a user store them into LinkedList Collection and display only negative integers.

→

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

public class NegativeIntegers {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        LinkedList<Integer> numbers = new LinkedList<>();

        System.out.print("Enter the number of integers (N): ");
        int n = scanner.nextInt();

        System.out.println("Enter " + n + " integers:");
        for (int i = 0; i < n; i++) {
            int num = scanner.nextInt();
            numbers.add(num);
        }

        System.out.println("Negative Integers:");
        for (int number : numbers) {
            if (number < 0) {
                System.out.println(number);
            }
        }

        scanner.close();
    }
}
```

Slip-20

1. Create a JSP page to accept a number from a user and display it in words: Example: 123 – One Two Three. The output should be in red color.

```
→<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Number to Words Converter</title>
</head>
<body>
    <h1>Number to Words Converter</h1>

    <form method="post">
        Enter a number: <input type="text" name="number">
        <input type="submit" value="Convert">
    </form>

    <%-- Convert number to words --%>
    <%@ page import="java.util.Scanner" %>
    <%
        String numberStr = request.getParameter("number");

        if (numberStr != null && !numberStr.isEmpty()) {
            int number = Integer.parseInt(numberStr);
            String words = convertNumberToWords(number);
    %>

    <h2 style="color: red;">Number in Words: <%= words %></h2>

    <% } %>

    <%!

        public String convertNumberToWords(int number) {
            String[] units = {"", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine",
"Ten",
                    "Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen", "Sixteen", "Seventeen",
"Eighteen", "Nineteen"};
            String[] tens = {"", "", "Twenty", "Thirty", "Forty", "Fifty", "Sixty", "Seventy", "Eighty",
"Ninety"};
            String[] thousands = {"", "Thousand", "Million", "Billion"};

            String words = "";

            int i = 0;
            while (number > 0) {
                if (number % 1000 != 0) {
                    words = convertLessThanThousand(number % 1000, units, tens) + thousands[i] + " " +
words;
                }
                number /= 1000;
                i++;
            }

            return words.trim();
        }

        public String convertLessThanThousand(int number, String[] units, String[] tens) {
            String current;

            if (number % 100 < 20) {
```

```
                current = units[number % 100];
                number /= 100;
            } else {
                current = units[number % 10];
                number /= 10;

                current = tens[number % 10] + " " + current;
                number /= 10;
            }

            if (number == 0) {
                return current;
            }

            return units[number] + " Hundred " + current;
        }
    %>
</body>
</html>
```

2. Write a java program to blink image on the JFrame continuously

→

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class BlinkingImage extends JFrame {
    private ImageIcon[] images;
    private JLabel imageLabel;
    private int currentIndex;
    private Timer timer;

    public BlinkingImage() {
        setTitle("Blinking Image");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 300);


        images = new ImageIcon[2];
        images[0] = new ImageIcon("image1.png");
        images[1] = new ImageIcon("image2.png");

        imageLabel = new JLabel(images[0]);

        add(imageLabel, BorderLayout.CENTER);


        timer = new Timer(500, new TimerListener());
        timer.start();
    }

    private class TimerListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            currentIndex = (currentIndex + 1) % images.length;
            imageLabel.setIcon(images[currentIndex]);
        }
    }
```

```
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            BlinkingImage blinkingImage = new BlinkingImage();
            blinkingImage.setVisible(true);
        });
    }
}
```

Slip-21

1. Write a java program to accept ‘N’ Subject Names from a user store them into LinkedList Collection and Display them by using Iterator interface.

→

```
import java.util.LinkedList;
import java.util.Iterator;
import java.util.Scanner;

public class SubjectNames {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        LinkedList<String> subjects = new LinkedList<>();

        System.out.print("Enter the number of subjects (N): ");
        int n = scanner.nextInt();

        scanner.nextLine();

        System.out.println("Enter " + n + " subject names:");
        for (int i = 0; i < n; i++) {
            String subject = scanner.nextLine();
            subjects.add(subject);
        }

        System.out.println("Subject Names:");
        Iterator<String> iterator = subjects.iterator();
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }

        scanner.close();
    }
}
```

Slip-22

1. Write a Menu Driven program in Java for the following: Assume Employee table with attributes (ENo, EName, Salary) is already created. 1. Insert 2. Update 3. Display 4. Exit.

→

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

public class EmployeeManagement {
    private static final String DB_URL = "jdbc:postgresql://localhost:5432/your_database_name";
    private static final String DB_USER = "your_username";
    private static final String DB_PASSWORD = "your_password";

    public static void main(String[] args) {
        try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD)) {
            System.out.println("Connected to database.");

            createEmployeeTable(conn);

            Scanner scanner = new Scanner(System.in);
            int choice;
            do {
                System.out.println("\nMenu:");
                System.out.println("1. Insert");
                System.out.println("2. Update");
                System.out.println("3. Display");
                System.out.println("4. Exit");
                System.out.print("Enter your choice: ");
                choice = scanner.nextInt();
                scanner.nextLine(); // Consume newline

                switch (choice) {
                    case 1:
                        insertEmployee(conn, scanner);
                        break;
                    case 2:
                        updateEmployee(conn, scanner);
                        break;
                    case 3:
                        displayEmployees(conn);
                        break;
                    case 4:
                        System.out.println("Exiting...");
                        break;
                    default:
                        System.out.println("Invalid choice. Please try again.");
                }
            } while (choice != 4);

            scanner.close();
        } catch (SQLException e) {
            System.err.println("Error: " + e.getMessage());
        }
    }

    private static void createEmployeeTable(Connection conn) throws SQLException {
        try (Statement stmt = conn.createStatement()) {
            String createTableSQL = "CREATE TABLE IF NOT EXISTS Employee (" +
                                    "ENo SERIAL PRIMARY KEY," +
                                    "EName VARCHAR(255) NOT NULL," +
                                    "Salary FLOAT NOT NULL)";
            stmt.executeUpdate(createTableSQL);
            System.out.println("Employee table created or already exists.");
```

```
        }
    }

    private static void insertEmployee(Connection conn, Scanner scanner) throws SQLException {
        System.out.print("Enter Employee Name: ");
        String eName = scanner.nextLine();

        System.out.print("Enter Salary: ");
        float salary = scanner.nextFloat();
        scanner.nextLine(); // Consume newline

        String insertSQL = "INSERT INTO Employee (EName, Salary) VALUES (?, ?)";
        try (PreparedStatement pstmt = conn.prepareStatement(insertSQL)) {
            pstmt.setString(1, eName);
            pstmt.setFloat(2, salary);
            pstmt.executeUpdate();
            System.out.println("Employee inserted successfully.");
        }
    }

    private static void updateEmployee(Connection conn, Scanner scanner) throws SQLException {
        System.out.print("Enter Employee Number to update: ");
        int eNo = scanner.nextInt();
        scanner.nextLine(); // Consume newline

        System.out.print("Enter new Employee Name: ");
        String eName = scanner.nextLine();

        System.out.print("Enter new Salary: ");
        float salary = scanner.nextFloat();
        scanner.nextLine(); // Consume newline

        String updateSQL = "UPDATE Employee SET EName = ?, Salary = ? WHERE ENo = ?";
        try (PreparedStatement pstmt = conn.prepareStatement(updateSQL)) {
            pstmt.setString(1, eName);
            pstmt.setFloat(2, salary);
            pstmt.setInt(3, eNo);
            int rowsUpdated = pstmt.executeUpdate();
            if (rowsUpdated > 0) {
                System.out.println("Employee updated successfully.");
            } else {
                System.out.println("Employee with ENo " + eNo + " not found.");
            }
        }
    }

    private static void displayEmployees(Connection conn) throws SQLException {
        String selectSQL = "SELECT * FROM Employee";
        try (Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(selectSQL)) {
            System.out.println("Employee Table:");
            System.out.println("ENo\tEName\tSalary");
            while (rs.next()) {
                int eNo = rs.getInt("ENo");
                String eName = rs.getString("EName");
                float salary = rs.getFloat("Salary");
                System.out.println(eNo + "\t" + eName + "\t" + salary);
            }
        }
    }
}
```

2. Write a JSP program which accepts UserName in a TextBox and greets the user according to the time on server machine.

→

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Greeting Page</title>
</head>
<body>
    <h1>Greeting Page</h1>

    <form method="post">
        Enter your name: <input type="text" name="username">
        <input type="submit" value="Greet">
    </form>

    <%-- Java code to process the form submission --%>
    <%@ page import="java.util.Date" %>
    <%@ page import="java.text.SimpleDateFormat" %>
    <%@ page import="java.util.Calendar" %>
    <%@ page import="java.io.PrintWriter" %>

    <%-- Get the current time on the server --%>
    <% Calendar cal = Calendar.getInstance();
       SimpleDateFormat sdf = new SimpleDateFormat("HH");
       int hour = Integer.parseInt(sdf.format(cal.getTime()));
    %>

    <%-- Get the username from the form submission --%>
    <% String username = request.getParameter("username");
       if (username != null && !username.isEmpty()) {
           String greeting = "";
           if (hour >= 0 && hour < 12) {
               greeting = "Good Morning";
           } else if (hour >= 12 && hour < 18) {
               greeting = "Good Afternoon";
           } else {
               greeting = "Good Evening";
           }
    %>

    <h2>
        <% out.println(greeting + ", " + username + "!"); %>
    </h2>

    <% } %>
</body>
</html>
```

Slip-23

1. Write a java program to accept a String from a user and display each vowel from a String after every 3 seconds.

→

```java
import java.util.Scanner;

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

        System.out.print("Enter a string: ");
        String input = scanner.nextLine();

        scanner.close();

        char[] characters = input.toCharArray();


        for (char ch : characters) {

            if (isVowel(ch)) {
                System.out.println(ch);
                try {
                    Thread.sleep(3000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }


    private static boolean isVowel(char ch) {
        ch = Character.toUpperCase(ch);
        return ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U';
    }
}
```

2. Write a java program to accept ‘N’ student names **through command line**, store them into the appropriate Collection and display them by using Iterator and ListIterator interface

→

Use command line

```
→import java.util.ArrayList;
import java.util.Iterator;
import java.util.ListIterator;

public class StudentNames {
    public static void main(String[] args) {
        if (args.length < 1) {
            System.out.println("Usage: java StudentNames <name1> <name2> <name3> ...");
            return;
        }

        ArrayList<String> studentNames = new ArrayList<>();


        for (String arg : args) {
            studentNames.add(arg);
        }


        System.out.println("Student Names (Using Iterator):");
        Iterator<String> iterator = studentNames.iterator();
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }


        System.out.println("\nStudent Names (Using ListIterator in Reverse):");
        ListIterator<String> listIterator = studentNames.listIterator(studentNames.size());
        while (listIterator.hasPrevious()) {
            System.out.println(listIterator.previous());
        }
    }
}
```

Slip-24

1. Write a java program to scroll the text from left to right continuously.

```
→public class TextScrolling {
    public static void main(String[] args) throws InterruptedException {
        String text = "This is a scrolling text. ";
        int width = 20;

        while (true) {
            for (int i = 0; i < width; i++) {
                clearScreen();
                System.out.print(text.substring(i) + text.substring(0, i));
                Thread.sleep(200); /
            }
        }
    }


    private static void clearScreen() {
        System.out.print("\033[H\033[2J");
        System.out.flush();
    }
}
```

2. Write a JSP script to accept username and password from user, if they are same then display “Login Successfully” message in Login.html file, otherwise display “Login Failed” Message in Error.html file.

Main.html

→

```
<!DOCTYPE html>
<html>
<head>
    <title>Login Form</title>
</head>
<body>
    <h1>Login Form</h1>
    <form action="slip24_b.jsp" method="post">
        Username: <input type="text" name="username"><br>
        Password: <input type="password" name="password"><br>
        <input type="submit" value="Login">
    </form>
</body>
</html>
```

Slip24_b.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ page import="java.io.*" %>

<%
    // Retrieve username and password from the request
    String username = request.getParameter("username");
    String password = request.getParameter("password");

    // Check if username and password are the same
```

```
    if (username != null && password != null && username.equals(password)) {
        response.sendRedirect("LoginSuccess.html"); // Redirect to LoginSuccess.html
    } else {
        response.sendRedirect("LoginError.html"); // Redirect to LoginError.html
    }
%>
```

Slip-25

1. Write a JSP program to accept Name and Age of Voter and check whether he is eligible for voting or not.

→

Main.html

```
→<!DOCTYPE html>
<html>
<head>
    <title>Voter Eligibility Check</title>
</head>
<body>
    <h1>Voter Eligibility Check</h1>

    <form action="slip25_a.jsp" method="post">
        Enter your name: <input type="text" name="name"><br>
        Enter your age: <input type="text" name="age"><br>
        <input type="submit" value="Check Eligibility">
    </form>
</body>
</html>
```

Slip25 a.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ page import="java.io.*" %>

<!DOCTYPE html>
<html>
<head>
    <title>Voter Eligibility Result</title>
</head>
<body>
    <h1>Voter Eligibility Result</h1>

    <%

        String name = request.getParameter("name");
```

```
        int age = Integer.parseInt(request.getParameter("age"));


        boolean isEligible = (age >= 18);


        out.println("<p>Name: " + name + "</p>");
        out.println("<p>Age: " + age + "</p>");

        if (isEligible) {
            out.println("<p>Congratulations! You are eligible to vote.</p>");
        } else {
            out.println("<p>Sorry! You are not eligible to vote.</p>");
        }
    %>
</body>
</html>
```

Slip-26

1. Write a Java program to delete the details of given employee (ENo EName Salary). Accept employee ID through command line. (Use PreparedStatement Interface)

→
JSP file

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ page import="java.io.*" %>

<!DOCTYPE html>
<html>
<head>
    <title>Voter Eligibility Result</title>
</head>
<body>
    <h1>Voter Eligibility Result</h1>

    <%

        String name = request.getParameter("name");
        int age = Integer.parseInt(request.getParameter("age"));


        boolean isEligible = (age >= 18);


        out.println("<p>Name: " + name + "</p>");
        out.println("<p>Age: " + age + "</p>");

        if (isEligible) {
            out.println("<p>Congratulations! You are eligible to vote.</p>");
        } else {
            out.println("<p>Sorry! You are not eligible to vote.</p>");
        }
    %>
</body>
</html>
```

Javafile

→

```java
import java.sql.*;

public class DeleteEmployee {
    public static void main(String[] args) {
        if (args.length < 1) {
            System.out.println("Usage: java DeleteEmployee <EmployeeID>");
            return;
        }

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

        String url = "jdbc:postgresql://localhost:5432/your_database_name";
        String username = "your_username";
        String password = "your_password";

        Connection conn = null;
        PreparedStatement pstmt = null;

        try {
            conn = DriverManager.getConnection(url, username, password);

            String sql = "DELETE FROM Employee WHERE ENo = ?";
            pstmt = conn.prepareStatement(sql);
            pstmt.setInt(1, employeeId);

            int rowsAffected = pstmt.executeUpdate();

            if (rowsAffected > 0) {
                System.out.println("Employee with ID " + employeeId + " deleted successfully.");
            } else {
                System.out.println("Employee with ID " + employeeId + " not found.");
            }

        } catch (SQLException e) {
            System.out.println("Error: " + e.getMessage());
        } finally {
            try {
                if (pstmt != null) {
                    pstmt.close();
                }
                if (conn != null) {
                    conn.close();
                }
            } catch (SQLException e) {
                System.out.println("Error closing resources: " + e.getMessage());
            }
        }
    }
}
```

2. Write a JSP program to calculate sum of first and last digit of a given number. Display sum in Red Color with font size 18.

```
→<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!DOCTYPE html>
<html>
<head>
    <title>Sum of First and Last Digit</title>
    <style>
        .red {
            color: red;
            font-size: 18px;
        }
    </style>
</head>
<body>
    <h1>Sum of First and Last Digit</h1>

    <form method="post">
        Enter a number: <input type="text" name="number">
        <input type="submit" value="Calculate">
    </form>

    <%
        if (request.getMethod().equals("POST")) {
            // Retrieve the number from the form
            String numberStr = request.getParameter("number");

            if (numberStr != null && !numberStr.isEmpty()) {
                // Parse the input number as an integer
                int number = Integer.parseInt(numberStr);

                // Calculate the first and last digits
                int firstDigit = 0;
                int lastDigit = number % 10;

                while (number != 0) {
                    firstDigit = number % 10;
                    number /= 10;
                }

                // Calculate the sum of the first and last digits
                int sum = firstDigit + lastDigit;

                // Display the result in red color with font size 18
    %>
                <p>Number: <%= numberStr %></p>
                <p class="red">Sum of First and Last Digit: <%= sum %></p>
    <%
            } else {
    %>
                <p class="red">Please enter a valid number.</p>
    <%
            }
        }
    %>
</body>
</html>
```

Slip-27

→

1. Write a Java Program to display the details of College (CID, CName, address, Year) on JTable.

→

```java
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;

public class CollegeDetailsTable extends JFrame {

    public CollegeDetailsTable() {
        setTitle("College Details Table");
        setSize(600, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


        Object[][] data = {
                {"1", "ABC College", "123 Main St, City1", "2020"},
                {"2", "XYZ College", "456 Elm St, City2", "2018"},
                {"3", "PQR College", "789 Oak St, City3", "2019"},
                {"4", "LMN College", "321 Pine St, City4", "2021"},
                {"5", "EFG College", "654 Maple St, City5", "2017"}
        };


        String[] columnNames = {"CID", "CName", "Address", "Year"};


        DefaultTableModel model = new DefaultTableModel(data, columnNames);


        JTable table = new JTable(model);


        table.setPreferredScrollableViewportSize(new Dimension(500, 300));
        table.setFillsViewportHeight(true);


        JScrollPane scrollPane = new JScrollPane(table);


        add(scrollPane, BorderLayout.CENTER);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            CollegeDetailsTable frame = new CollegeDetailsTable();
            frame.setVisible(true);
        });
    }
}
```

2. Write a SERVLET program to change inactive time interval of session

→

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

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


        HttpSession session = request.getSession();


        int currentTimeout = session.getMaxInactiveInterval();


        int newTimeout = 300;
        session.setMaxInactiveInterval(newTimeout);

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

        out.println("<html><head><title>Session Timeout Changed</title></head><body>");
        out.println("<h1>Session Timeout Changed</h1>");
        out.println("<p>Previous Timeout: " + currentTimeout + " seconds</p>");
        out.println("<p>New Timeout: " + newTimeout + " seconds</p>");
        out.println("<p>Session Timeout has been changed successfully.</p>");
        out.println("</body></html>");
    }
}
```

Slip-28

1. Write a JSP script to accept a String from a user and display it in reverse order

→

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!DOCTYPE html>
<html>
<head>
    <title>Reverse String</title>
</head>
<body>
    <h1>Reverse String</h1>

    <form method="post">
        Enter a string: <input type="text" name="inputString">
        <input type="submit" value="Reverse">
    </form>

    <%
        if (request.getMethod().equals("POST")) {
            // Retrieve the input string from the form
            String inputString = request.getParameter("inputString");

            // Check if the input string is not null
            if (inputString != null && !inputString.isEmpty()) {
                // Reverse the input string
                String reversedString = new StringBuilder(inputString).reverse().toString();

                // Display the reversed string
    %>
                <p>Original String: <%= inputString %></p>
                <p>Reversed String: <%= reversedString %></p>
    <%
            } else {
    %>
                <p>Please enter a valid string.</p>
    <%
            }
        }
    %>
</body>
</html>
```

2. Write a java program to display name of currently executing Thread in multithreading.

→

```java
public class CurrentThreadName {
    public static void main(String[] args) {
        Thread thread1 = new Thread(new MyRunnable(), "Thread-1");
        Thread thread2 = new Thread(new MyRunnable(), "Thread-2");
        thread1.start();
        thread2.start();
        String mainThreadName = Thread.currentThread().getName();
        System.out.println("Main thread name: " + mainThreadName);
    }
    static class MyRunnable implements Runnable {
        @Override
        public void run() {
            String currentThreadName = Thread.currentThread().getName();
            System.out.println("Current thread name: " + currentThreadName);
        }
    }
}
```

Slip-29

→

1. Write a Java program to display information about all columns in the DONAR table using ResultSetMetaData.

→

```java
/**
 * @author Jadhav Rohan
 *
 * C].Write a program to display information about all coumns in the DONOR table using ResultSetMetaData.
 *
 *for database material use here Slip29_a.pgsql file
**/

import java.sql.*;

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

            Class.forName("org.postgresql.Driver");


            Connection conn = DriverManager.getConnection("jdbc:postgresql://localhost/postgres",
"postgres", "dsk");

            Statement stmt = null;
            stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("select * from donor");

            ResultSetMetaData rsmd = rs.getMetaData();
            System.out.println("\t-------------------------------------------------");

            int count = rsmd.getColumnCount();
            System.out.println("\t No. of Columns: " + rsmd.getColumnCount());
            System.out.println("\t-------------------------------------------------");
            for (int i = 1; i <= count; i++)
            {
                System.out.println("\t\tColumn No : " + i);
```

```
                System.out.println("\t\tColumn Name : " + rsmd.getColumnName(i));
                System.out.println("\t\tColumn Type : " + rsmd.getColumnTypeName(i));
                System.out.println("\t\tColumn Display Size : " + rsmd.getColumnDisplaySize(i));
                System.out.println();
            } // for
            System.out.println("\t-------------------------------------------------");

            rs.close();
            stmt.close();
            conn.close();
        } // try
        catch (Exception e) {
            System.out.println(e);
        } // catch
    }
}
```

Sql file

```
→-- create table donor(did int, dname char(22),daddr varchar(22));

-- insert into donor VALUES(1,'AAA','zzz');
-- insert into donor VALUES(2,'BBB','yyy');
-- insert into donor VALUES(3,'CCC','xxx');
-- insert into donor VALUES(4,'DDD','www');

SELECT * from donor;
```

2. Write a Java program to create LinkedList of integer objects and perform the following:

i. Add element at first position

ii. Delete last element

iii. Display the size of link list

→

```
import java.util.LinkedList;

public class LinkedListOperations {

    public static void main(String[] args) {

        LinkedList<Integer> linkedList = new LinkedList<>();


        linkedList.addFirst(10);
        linkedList.addFirst(20);
        linkedList.addFirst(30);


        System.out.println("LinkedList after adding elements at first position:");
        displayLinkedList(linkedList);
```

```
        if (!linkedList.isEmpty()) {
            linkedList.removeLast();
        }


        System.out.println("\nLinkedList after deleting last element:");
        displayLinkedList(linkedList);


        System.out.println("\nSize of the LinkedList: " + linkedList.size());
    }


    private static void displayLinkedList(LinkedList<Integer> list) {
        for (Integer element : list) {
            System.out.print(element + " ");
        }
        System.out.println();
    }
}
```

Slip-30)

1. Write a java program for the implementation of synchronization.

→

```java
public class SynchronizationExample {

    public static void main(String[] args) {
        Counter counter = new Counter();


        Thread thread1 = new Thread(new IncrementTask(counter));
        Thread thread2 = new Thread(new IncrementTask(counter));


        thread1.start();
        thread2.start();

        try {

            thread1.join();
            thread2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }


        System.out.println("Final Count: " + counter.getCount());
    }

    static class Counter {
        private int count;

        public Counter() {
            this.count = 0;
        }

        public void increment() {
            synchronized (this) {

                count++;
            }
        }

        public int getCount() {
            return count;
        }
    }

    static class IncrementTask implements Runnable {
        private final Counter counter;

        public IncrementTask(Counter counter) {
            this.counter = counter;
        }

        @Override
        public void run() {
            for (int i = 0; i < 10000; i++) {
                counter.increment();
            }
        }
    }
}
```

2. Write a Java Program for the implementation of scrollable ResultSet. Assume Teacher table with attributes (TID, TName, Salary) is already created.

→

```
import java.sql.*;

public class ScrollableResultSetExample {

    public static void main(String[] args) {
        try {

            String url = "jdbc:postgresql://localhost:5432/your_database";
            String username = "your_username";
            String password = "your_password";

            Connection conn = DriverManager.getConnection(url, username, password);


            Statement stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);


            String query = "SELECT * FROM Teacher";
            ResultSet rs = stmt.executeQuery(query);


            if (!rs.isBeforeFirst()) {
                System.out.println("ResultSet is not scrollable.");
            } else {

                rs.last();


                System.out.println("Teacher Details (in reverse order):");
                System.out.println("==================================");
                do {
                    int tid = rs.getInt("TID");
                    String tname = rs.getString("TName");
                    double salary = rs.getDouble("Salary");

                    System.out.println("TID: " + tid + ", TName: " + tname + ", Salary: " + salary);
                } while (rs.previous());
            }


            rs.close();
            stmt.close();
            conn.close();

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