JAVA-2 SLIPS

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

```
public class AlphabetDisplay {

    public static void main(String[] args) {
        char currentChar = 'A';

        try {
            while (currentChar <= 'Z') {
                System.out.print(currentChar + " ");
                currentChar++;
                Thread.sleep(2000); // Sleep 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.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class EmployeeDetailsForm extends JFrame {
    private JTextField txtEno, txtEName, txtDesignation, txtSalary;

    public EmployeeDetailsForm() {
        setTitle("Employee Details Form");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new GridLayout(5, 2));

        JLabel lblEno = new JLabel("Employee Number:");
        txtEno = new JTextField();
```

```
        JLabel lblEName = new JLabel("Employee Name:");
        txtEName = new JTextField();
        JLabel lblDesignation = new JLabel("Designation:");
        txtDesignation = new JTextField();
        JLabel lblSalary = new JLabel("Salary:");
        txtSalary = new JTextField();

        JButton btnSave = new JButton("Save");
        btnSave.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                saveEmployeeDetails();
            }
        });

        add(lblEno);
        add(txtEno);
        add(lblEName);
        add(txtEName);
        add(lblDesignation);
        add(txtDesignation);
        add(lblSalary);
        add(txtSalary);
        add(new JLabel()); // Empty label for spacing
        add(btnSave);

        setLocationRelativeTo(null);
        setVisible(true);
    }

    private void saveEmployeeDetails() {
        try {
            // JDBC connection
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/your_database", "your_username",
"your_password");

            // SQL query to insert data into the table
            String query = "INSERT INTO employee (eno, ename, designation, salary) VALUES (?,
?, ?, ?)";
            PreparedStatement preparedStatement = connection.prepareStatement(query);

            // Set values from the text fields
```

```
            preparedStatement.setInt(1, Integer.parseInt(txtEno.getText()));
            preparedStatement.setString(2, txtEName.getText());
            preparedStatement.setString(3, txtDesignation.getText());
            preparedStatement.setDouble(4, Double.parseDouble(txtSalary.getText()));

            // Execute the query
            preparedStatement.executeUpdate();

            // Close resources
            preparedStatement.close();
            connection.close();

            JOptionPane.showMessageDialog(this, "Employee details saved successfully!");
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "Error saving employee details.");
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new EmployeeDetailsForm();
            }
        });
    }
}
```

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.Iterator;
import java.util.Scanner;
import java.util.TreeSet;

public class FriendNames {

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

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

        // Create a HashSet to store names
        HashSet<String> friendSet = new HashSet<>();

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

        // Create a TreeSet to store names in ascending order
        TreeSet<String> sortedFriendSet = new TreeSet<>(friendSet);

        // Display names in ascending order
        System.out.println("\nFriend Names in Ascending Order:");
        Iterator<String> iterator = sortedFriendSet.iterator();
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }

        scanner.close();
    }
}
```

2. Design a servlet that provides information about a HTTP request from a client, such as IP-Address and browser type. The servlet also provides information about the server on which the servlet is running, such as the operating system type, and the names of currently loaded servlets.

```
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Enumeration;
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("/RequestInfoServlet")
public class RequestInfoServlet 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();

        out.println("<html>");
        out.println("<head>");
        out.println("<title>Request and Server Information</title>");
        out.println("</head>");
        out.println("<body>");

        // Client Information
        out.println("<h2>Client Information:</h2>");
        out.println("<p>IP Address: " + request.getRemoteAddr() + "</p>");
        out.println("<p>Browser Type: " + request.getHeader("User-Agent") + "</p>");

        // Server Information
        out.println("<h2>Server Information:</h2>");
        out.println("<p>Server OS: " + System.getProperty("os.name") + "</p>");

        // Loaded Servlets
        out.println("<h2>Loaded Servlets:</h2>");
        Enumeration<String> servletNames =
getServletConfig().getServletContext().getServletNames();
        while (servletNames.hasMoreElements()) {
            String servletName = servletNames.nextElement();
            out.println("<p>" + servletName + "</p>");
        }

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

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>
</head>
<body>

   <h2>Patient Details</h2>

   <table border="1">
      <thead>
         <tr>
            <th>Patient Number</th>
            <th>Patient Name</th>
            <th>Address</th>
            <th>Age</th>
            <th>Disease</th>
         </tr>
      </thead>
      <tbody>
         <%-- Sample data, replace it with actual data retrieval logic --%>
         <%
            // Sample data (you should replace this with actual data retrieval logic)
            String[][] patients = {
               {"P001", "John Doe", "123 Main St", "25", "Fever"},
               {"P002", "Jane Smith", "456 Oak St", "30", "Headache"},
               {"P003", "Bob Johnson", "789 Pine St", "40", "Cough"}
            };

            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>
         <%
            }
         %>
      </tbody>
   </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;

public class StringLinkedListExample {

    public static void main(String[] args) {
        // Create a LinkedList of String objects
        LinkedList<String> stringList = new LinkedList<>();

        // Add elements at the end of the list
        stringList.add("Apple");
        stringList.add("Banana");
        stringList.add("Orange");

        // Display the original list
        System.out.println("Original List: " + stringList);

        // Add element at the end of the list
        stringList.add("Grapes");
        System.out.println("List after adding 'Grapes': " + stringList);

        // Delete the first element of the list
        if (!stringList.isEmpty()) {
            stringList.removeFirst();
            System.out.println("List after deleting the first element: " + stringList);
        } else {
            System.out.println("List is empty. Cannot remove the first element.");
        }

        // Display the contents of the list in reverse order
        System.out.println("List in reverse order:");
        for (int i = stringList.size() - 1; i >= 0; i--) {
            System.out.println(stringList.get(i));
        }
    }
}
```

SLIP 4

1. Write a Java program using Runnable interface to blink Text on the frame

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

public class BlinkingTextFrame implements Runnable {

    private JFrame frame;
    private JLabel label;

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new BlinkingTextFrame());
    }

    @Override
    public void run() {
        frame = new JFrame("Blinking Text Frame");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 100);

        label = new JLabel("Blinking Text", JLabel.CENTER);
        frame.add(label);

        Timer timer = new Timer(500, new ActionListener() {
            private boolean isVisible = true;

            @Override
            public void actionPerformed(ActionEvent e) {
                isVisible = !isVisible;
                label.setVisible(isVisible);
            }
        });
        timer.start();

        frame.setVisible(true);
    }
}
```

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 java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class CityStdCodeCollection {

    private static Map<String, String> cityStdCodes = new HashMap<>();

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

        do {
            System.out.println("1. Add a new city and its code");
            System.out.println("2. Remove a city from the collection");
            System.out.println("3. Search for a city name and display the code");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    addCityAndCode(scanner);
                    break;
                case 2:
                    removeCity(scanner);
                    break;
                case 3:
                    searchCity(scanner);
                    break;
                case 4:
                    System.out.println("Exiting program.");
                    break;
                default:
                    System.out.println("Invalid choice. Please enter a valid option.");
                    break;
            }

        } while (choice != 4);

        scanner.close();
    }

    private static void addCityAndCode(Scanner scanner) {
```

```
        System.out.print("Enter city name: ");
        String cityName = scanner.next();

        if (!cityStdCodes.containsKey(cityName)) {
            System.out.print("Enter STD code for " + cityName + ": ");
            String stdCode = scanner.next();
            cityStdCodes.put(cityName, stdCode);
            System.out.println("City added successfully.");
        } else {
            System.out.println("City already exists. Cannot add duplicate entries.");
        }
    }

    private static void removeCity(Scanner scanner) {
        System.out.print("Enter city name to remove: ");
        String cityName = scanner.next();

        if (cityStdCodes.containsKey(cityName)) {
            cityStdCodes.remove(cityName);
            System.out.println("City removed successfully.");
        } else {
            System.out.println("City not found. Cannot remove.");
        }
    }

    private static void searchCity(Scanner scanner) {
        System.out.print("Enter city name to search: ");
        String cityName = scanner.next();

        if (cityStdCodes.containsKey(cityName)) {
            System.out.println("STD code for " + cityName + ": " + cityStdCodes.get(cityName));
        } else {
            System.out.println("City not found.");
        }
    }
}
```

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

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

```
import java.util.Hashtable;

public class StudentHashTable {

    public static void main(String[] args) {
        // Create a Hashtable to store mobile numbers and student names
        Hashtable<String, String> studentTable = new Hashtable<>();

        // Add entries to the hash table
        studentTable.put("1234567890", "John Doe");
        studentTable.put("9876543210", "Jane Smith");
        studentTable.put("8765432109", "Bob Johnson");

        // Display the details of students using Enumeration
        displayStudentDetails(studentTable);
    }

    private static void displayStudentDetails(Hashtable<String, String> studentTable) {
        System.out.println("Student Details:");

        Enumeration<String> mobileNumbers = studentTable.keys();
        while (mobileNumbers.hasMoreElements()) {
            String mobileNumber = mobileNumbers.nextElement();
            String studentName = studentTable.get(mobileNumber);

            System.out.println("Mobile Number: " + mobileNumber + ", Student Name: " +
studentName);
        }
    }
}
```

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.Scanner;
import java.util.Set;
import java.util.TreeSet;

public class SortedIntegerCollection {

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

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

        // Accept 'n' integers from the user
        System.out.print("Enter the number of integers (n): ");
        int n = scanner.nextInt();

        Set<Integer> integerSet = new TreeSet<>();

        for (int i = 0; i < n; i++) {
            System.out.print("Enter integer " + (i + 1) + ": ");
            int num = scanner.nextInt();
            integerSet.add(num);
        }

        // Display integers in sorted order
        System.out.println("Integers in sorted order: " + integerSet);

        // Search for a particular element
        System.out.print("Enter the element to search: ");
        int searchElement = scanner.nextInt();

        if (integerSet.contains(searchElement)) {
            System.out.println("Element " + searchElement + " is present in the collection.");
        } else {
            System.out.println("Element " + searchElement + " is not present in the collection.");
        }

        scanner.close();
    }
}
```

2. Write a java program to simulate traffic signal using threads.

```
public class TrafficSignalSimulation {

    public static void main(String[] args) {
        TrafficSignal signal = new TrafficSignal();

        Thread redThread = new Thread(new SignalLight(signal, LightColor.RED, 5000)); // 5
seconds for red light
```

```
        Thread yellowThread = new Thread(new SignalLight(signal, LightColor.YELLOW, 2000)); //
2 seconds for yellow light
        Thread greenThread = new Thread(new SignalLight(signal, LightColor.GREEN, 5000)); // 5
seconds for green light

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

enum LightColor {
    RED, YELLOW, GREEN
}

class TrafficSignal {
    private LightColor currentColor = LightColor.RED;

    public synchronized LightColor getCurrentColor() {
        return currentColor;
    }

    public synchronized void setCurrentColor(LightColor color) {
        this.currentColor = color;
    }
}

class SignalLight implements Runnable {
    private TrafficSignal signal;
    private LightColor lightColor;
    private int duration; // Duration in milliseconds

    public SignalLight(TrafficSignal signal, LightColor lightColor, int duration) {
        this.signal = signal;
        this.lightColor = lightColor;
        this.duration = duration;
    }

    @Override
    public void run() {
        while (true) {
            synchronized (signal) {
                signal.setCurrentColor(lightColor);
                System.out.println(lightColor + " Light is ON");
```

```
            try {
                signal.wait(duration);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            System.out.println(lightColor + " Light is OFF");
        }
    }
  }
}
```

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.

```
import java.util.Random;

public class MultiThreadApplication {

    public static void main(String[] args) {
        NumberGenerator generator = new NumberGenerator();
        SquareCalculator squareCalculator = new SquareCalculator(generator);
        CubeCalculator cubeCalculator = new CubeCalculator(generator);

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

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

class NumberGenerator implements Runnable {
    private Random random = new Random();

    @Override
    public void run() {
```

```
        try {
            while (true) {
                int randomNumber = getRandomNumber();
                System.out.println("Generated number: " + randomNumber);

                if (randomNumber % 2 == 0) {
                    SquareCalculator.setNumber(randomNumber);
                } else {
                    CubeCalculator.setNumber(randomNumber);
                }

                Thread.sleep(1000);
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    private int getRandomNumber() {
        return random.nextInt(100);
    }
}

class SquareCalculator implements Runnable {
    private static int number;
    private NumberGenerator generator;

    public SquareCalculator(NumberGenerator generator) {
        this.generator = generator;
    }

    public static void setNumber(int num) {
        number = num;
    }

    @Override
    public void run() {
        try {
            while (true) {
                synchronized (this) {
                    wait();

                    int square = number * number;
                    System.out.println("Square of " + number + ": " + square);
```

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

class CubeCalculator implements Runnable {
  private static int number;
  private NumberGenerator generator;

  public CubeCalculator(NumberGenerator generator) {
    this.generator = generator;
  }

  public static void setNumber(int num) {
    number = num;
  }

  @Override
  public void run() {
    try {
      while (true) {
        synchronized (this) {
          wait();

          int cube = number * number * number;
          System.out.println("Cube of " + number + ": " + cube);
        }
      }
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
  }
}
```

SLIP 8

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

```
class PrintThread extends Thread {
    private String text;
    private int repetitionCount;

    public PrintThread(String text, int repetitionCount) {
        this.text = text;
        this.repetitionCount = repetitionCount;
    }

    @Override
    public void run() {
        for (int i = 0; i < repetitionCount; i++) {
            System.out.println(text);
        }
    }
}

public class TextPrintingExample {

    public static void main(String[] args) {
        // Create three threads with different text and repetition count
        PrintThread thread1 = new PrintThread("COVID19", 10);
        PrintThread thread2 = new PrintThread("LOCKDOWN2020", 20);
        PrintThread thread3 = new PrintThread("VACCINATED2021", 30);

        // Start the threads
        thread1.start();
        thread2.start();
        thread3.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>
```

```
    <h2>Prime Number Checker</h2>

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

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

        if (numberStr != null && !numberStr.isEmpty()) {
            try {
                int number = Integer.parseInt(numberStr);

                boolean isPrime = checkPrime(number);

                // Display the result in red color
                String resultColor = isPrime ? "red" : "black";
    %>
                <p style="color: <%= resultColor %>;">
                    <%= number %> is <%= isPrime ? "prime" : "not prime" %>.
                </p>
    <%
            } catch (NumberFormatException e) {
    %>
                <p style="color: red;">
                    Please enter a valid integer.
                </p>
    <%
            }
        }
    %>

</body>
</html>

<%!
    boolean checkPrime(int num) {
        if (num <= 1) {
            return false;
        }

        for (int i = 2; i <= Math.sqrt(num); i++) {
```

```
            if (num % i == 0) {
                return false;
            }
        }

        return true;
    }
%>
```

SLIP 9

2). Write a Java program using Spring to display the message “If you can't explain it simply, you don't understand it well enough”.

```
public class MessageService {
    private String message;

    public void setMessage(String message) {
        this.message = message;
    }

    public String getMessage() {
        return message;
    }
}

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class AppConfig {

    @Bean
    public MessageService messageService() {
        MessageService messageService = new MessageService();
        messageService.setMessage("If you can't explain it simply, you don't understand it well
enough.");
        return messageService;
    }
}

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
```

```
public class MainApp {

   public static void main(String[] args) {
      ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);

      MessageService messageService = context.getBean(MessageService.class);
      String message = messageService.getMessage();

      System.out.println(message);
   }
}
```

SLIP 10

1. Write a Java program to display the Current Date using spring.

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainApp {

   public static void main(String[] args) {
      ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);

      DateService dateService = context.getBean(DateService.class);
      String currentDate = dateService.getCurrentDate();

      System.out.println("Current Date: " + currentDate);
   }
}

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import java.text.SimpleDateFormat;
import java.util.Date;

@Configuration
public class AppConfig {

   @Bean
   public DateService dateService() {
      return new DateService();
   }
}
```

```
import java.text.SimpleDateFormat;
import java.util.Date;

public class DateService {

    public String getCurrentDate() {
        Date currentDate = new Date();
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        return dateFormat.format(currentDate);
    }
}
```

SLIP 11

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

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;

public class DisplayTableColumns {

    private static final String DATABASE_URL = "jdbc:mysql://localhost:3306/your_database"; // Replace with your database URL
    private static final String USERNAME = "your_username"; // Replace with your database username
    private static final String PASSWORD = "your_password"; // Replace with your database password

    public static void main(String[] args) {
        try (Connection connection = DriverManager.getConnection(DATABASE_URL, USERNAME, PASSWORD)) {
            String selectQuery = "SELECT * FROM DONAR"; // Assuming DONAR is the table name
            try (PreparedStatement preparedStatement = connection.prepareStatement(selectQuery);
                 ResultSet resultSet = preparedStatement.executeQuery()) {
```

```
            // Get metadata
            ResultSetMetaData metaData = resultSet.getMetaData();
            int columnCount = metaData.getColumnCount();

            System.out.println("Information about columns in the DONAR table:");

            for (int i = 1; i <= columnCount; i++) {
                System.out.println("Column " + i + ":");
                System.out.println("   Name: " + metaData.getColumnName(i));
                System.out.println("   Type: " + metaData.getColumnTypeName(i));
                System.out.println("   Size: " + metaData.getColumnDisplaySize(i));
                System.out.println("   Nullable: " + (metaData.isNullable(i) ==
ResultSetMetaData.columnNoNulls ? "No" : "Yes"));
                System.out.println("   Auto Increment: " + metaData.isAutoIncrement(i));
                System.out.println("---------------------------");
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
  }
}
```

SLIP 12

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).

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

public class ProjectTableApp extends JFrame {

   private static final String DATABASE_URL = "jdbc:mysql://localhost:3306/your_database"; //
Replace with your database URL
   private static final String USERNAME = "your_username"; // Replace with your database
username
   private static final String PASSWORD = "your_password"; // Replace with your database
password
```

```
    private DefaultTableModel tableModel;

    public ProjectTableApp() {
        super("Project Table");

        // Create table model with columns
        tableModel = new DefaultTableModel();
        tableModel.addColumn("Project ID");
        tableModel.addColumn("Project Name");
        tableModel.addColumn("Project Description");
        tableModel.addColumn("Project Status");

        // Create table with the table model
        JTable table = new JTable(tableModel);

        // Create scroll pane and add the table to it
        JScrollPane scrollPane = new JScrollPane(table);

        // Create button to insert values into the table
        JButton insertButton = new JButton("Insert Values");
        insertButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                insertValuesIntoTable();
            }
        });

        // Set layout and add components to the frame
        setLayout(new BorderLayout());
        add(scrollPane, BorderLayout.CENTER);
        add(insertButton, BorderLayout.SOUTH);

        // Set frame properties
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(600, 400);
        setLocationRelativeTo(null); // Center the frame on the screen
    }

    private void insertValuesIntoTable() {
        try (Connection connection = DriverManager.getConnection(DATABASE_URL,
USERNAME, PASSWORD)) {
            // Insert values into the PROJECT table
```

```
        String insertQuery = "INSERT INTO PROJECT (project_id, Project_name,
Project_description, Project_Status) VALUES (?, ?, ?, ?)";
        try (PreparedStatement preparedStatement =
connection.prepareStatement(insertQuery)) {
            // Replace these values with the actual values you want to insert
            preparedStatement.setInt(1, 1);
            preparedStatement.setString(2, "Project 1");
            preparedStatement.setString(3, "Description 1");
            preparedStatement.setString(4, "Active");

            int rowsAffected = preparedStatement.executeUpdate();
            if (rowsAffected > 0) {
                System.out.println("Values inserted successfully.");
            }
        }

        // Display all details of the PROJECT table
        String selectQuery = "SELECT * FROM PROJECT";
        try (Statement statement = connection.createStatement();
             ResultSet resultSet = statement.executeQuery(selectQuery)) {
            tableModel.setRowCount(0); // Clear existing rows in the table
            while (resultSet.next()) {
                Object[] rowData = {
                        resultSet.getInt("project_id"),
                        resultSet.getString("Project_name"),
                        resultSet.getString("Project_description"),
                        resultSet.getString("Project_Status")
                };
                tableModel.addRow(rowData);
            }
        }

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

public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            new ProjectTableApp().setVisible(true);
        }
    });
```

```
    }
}
```

SLIP 13

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

```
import java.sql.Connection;
import java.sql.DatabaseMetaData;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;

public class DatabaseInfo {

    private static final String DATABASE_URL = "jdbc:mysql://localhost:3306/your_database"; // Replace with your database URL
    private static final String USERNAME = "your_username"; // Replace with your database username
    private static final String PASSWORD = "your_password"; // Replace with your database password

    public static void main(String[] args) {
        try (Connection connection = DriverManager.getConnection(DATABASE_URL, USERNAME, PASSWORD)) {
            DatabaseMetaData metaData = connection.getMetaData();

            // Display database information
            System.out.println("Database Information:");
            System.out.println("Database Name: " + metaData.getDatabaseProductName());
            System.out.println("Database Version: " + metaData.getDatabaseProductVersion());
            System.out.println("Driver Name: " + metaData.getDriverName());
            System.out.println("Driver Version: " + metaData.getDriverVersion());
            System.out.println();

            // List all tables in the database
            System.out.println("Tables in the Database:");
            ResultSet resultSet = metaData.getTables(null, null, null, new String[]{"TABLE"});
            while (resultSet.next()) {
                String tableName = resultSet.getString("TABLE_NAME");
                System.out.println(tableName);
            }
        } catch (SQLException e) {
            e.printStackTrace();
```

```
        }
    }
}
```

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;

class MyThread extends Thread {

    public MyThread(String name) {
        super(name);
    }

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

        // Generate a random sleep time between 0 and 4999 milliseconds
        Random random = new Random();
        int sleepTime = random.nextInt(5000);

        System.out.println("Thread " + getName() + " will sleep for " + sleepTime + "
milliseconds.");

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

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

public class ThreadLifecycleDemo {

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

        thread1.start();
```

```
        thread2.start();
    }
}
```

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;

public class SimpleSearchEngine {

    private static final String SEARCH_STRING = "your_search_string"; // Replace with your search string

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

        if (files != null) {
            for (File file : files) {
                if (file.isFile() && file.getName().endsWith(".txt")) {
                    Thread searchThread = new Thread(() -> searchInFile(file));
                    searchThread.start();
                }
            }
        }
    }

    private static void searchInFile(File file) {
        try (BufferedReader reader = new BufferedReader(new FileReader(file))) {
            String line;
            int lineNumber = 0;

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

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

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" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Sum of First and Last Digit</title>
</head>
<body>

<%
    // Get the number from the request parameter
    String numberStr = request.getParameter("number");

    // Validate if the number is not null and not empty
    if (numberStr != null && !numberStr.isEmpty()) {
        try {
            // Parse the number from the string
            int number = Integer.parseInt(numberStr);

            // Calculate the sum of the first and last digits
            int lastDigit = number % 10;
            int firstDigit = Character.getNumericValue(Integer.toString(number).charAt(0));
            int sum = firstDigit + lastDigit;
%>
            <!-- Display the result in red color with font size 18 -->
            <p style="color: red; font-size: 18px;">
                Sum of the first and last digit of <%= number %> is <%= sum %>.
            </p>
<%
        } catch (NumberFormatException e) {
%>
            <!-- Display an error message if the input is not a valid integer -->
            <p style="color: red; font-size: 18px;">
                Please enter a valid integer.
            </p>
```

```
<%
        }
    }
%>

<!-- Input form for entering the number -->
<form action="" method="post">
    Enter a number: <input type="text" name="number">
    <input type="submit" value="Calculate Sum">
</form>

</body>
</html>
```

SLIP 15

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

```
public class ThreadInfoExample {

    public static void main(String[] args) {
        // Create and start a thread
        Thread myThread = new Thread(() -> {
            // Thread's logic goes here
            for (int i = 0; i < 5; i++) {
                System.out.println(Thread.currentThread().getName() + ": " + i);
            }
        });

        // Set thread name
        myThread.setName("MyCustomThread");

        // Set thread priority (1 to 10, where 1 is the lowest and 10 is the highest)
        myThread.setPriority(Thread.NORM_PRIORITY); // Default priority

        // Start the thread
        myThread.start();

        // Display thread information
        displayThreadInfo(myThread);
    }

    private static void displayThreadInfo(Thread thread) {
        System.out.println("Thread Name: " + thread.getName());
        System.out.println("Thread Priority: " + thread.getPriority());
    }
```

```
}
```

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.

```
import java.util.TreeSet;

public class TreeSetExample {

    public static void main(String[] args) {
        // Create a TreeSet to store colors in ascending order
        TreeSet<String> colorSet = new TreeSet<>();

        // Add colors to the TreeSet
        colorSet.add("Red");
        colorSet.add("Green");
        colorSet.add("Blue");
        colorSet.add("Yellow");
        colorSet.add("Purple");

        // Print the content of TreeSet in ascending order
        System.out.println("Colors in ascending order:");
        for (String color : colorSet) {
            System.out.println(color);
        }
    }
}
```

2. Write a Java program to accept the details of Teacher (TNo, TName, Subject). Insert at least 5 Records into Teacher Table and display the details of Teacher who is teaching “JAVA” Subject. (Use PreparedStatement Interface)

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class TeacherDetailsExample {

    private static final String DATABASE_URL = "jdbc:mysql://localhost:3306/your_database"; // Replace with your database URL
    private static final String USERNAME = "your_username"; // Replace with your database username
```

```
    private static final String PASSWORD = "your_password"; // Replace with your database
password

    public static void main(String[] args) {
        try (Connection connection = DriverManager.getConnection(DATABASE_URL,
USERNAME, PASSWORD)) {
            // Insert at least 5 records into the Teacher table
            insertTeacherRecords(connection);

            // Display details of teachers who are teaching "JAVA"
            displayJavaTeachers(connection);

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

    private static void insertTeacherRecords(Connection connection) throws SQLException {
        // SQL query to insert records into Teacher table
        String insertQuery = "INSERT INTO Teacher (TNo, TName, Subject) VALUES (?, ?, ?)";
        try (PreparedStatement preparedStatement = connection.prepareStatement(insertQuery)) {
            // Inserting at least 5 records
            for (int i = 1; i <= 5; i++) {
                preparedStatement.setInt(1, i);
                preparedStatement.setString(2, "Teacher" + i);
                preparedStatement.setString(3, "Subject" + i);

                int rowsAffected = preparedStatement.executeUpdate();
                if (rowsAffected > 0) {
                    System.out.println("Record inserted successfully.");
                }
            }
        }
    }

    private static void displayJavaTeachers(Connection connection) throws SQLException {
        // SQL query to select teachers who are teaching "JAVA" subject
        String selectQuery = "SELECT * FROM Teacher WHERE Subject = ?";
        try (PreparedStatement preparedStatement = connection.prepareStatement(selectQuery))
{
            // Set the subject to "JAVA"
            preparedStatement.setString(1, "JAVA");

            try (ResultSet resultSet = preparedStatement.executeQuery()) {
```

```
            System.out.println("Details of teachers teaching \"JAVA\" subject:");
            while (resultSet.next()) {
                System.out.println("TNo: " + resultSet.getInt("TNo"));
                System.out.println("TName: " + resultSet.getString("TName"));
                System.out.println("Subject: " + resultSet.getString("Subject"));
                System.out.println("--------------------------");
            }
        }
    }
}
```

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.

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

public class SortedIntegerCollection {

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

        // Accept 'N' integers from the user
        System.out.print("Enter the value of N: ");
        int n = scanner.nextInt();

        // Create a TreeSet to store integers in sorted order without duplicates
        TreeSet<Integer> integerSet = new TreeSet<>();

        // Accept integers from the user and add them to the TreeSet
        for (int i = 0; i < n; i++) {
            System.out.print("Enter integer " + (i + 1) + ": ");
            int inputInt = scanner.nextInt();
            integerSet.add(inputInt);
        }

        // Display the integers in sorted order
        System.out.println("Integers in sorted order:");
        for (int num : integerSet) {
            System.out.println(num);
        }

        scanner.close();
```

```
    }
}
```

SLIP 18

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

```
public class ThreadInfoExample {

    public static void main(String[] args) {
        // Create a thread
        Thread myThread = new Thread(() -> {
            // Thread's logic goes here
            for (int i = 0; i < 5; i++) {
                System.out.println(Thread.currentThread().getName() + ": " + i);
            }
        });

        // Set thread name
        myThread.setName("MyCustomThread");

        // Set thread priority (1 to 10, where 1 is the lowest and 10 is the highest)
        myThread.setPriority(Thread.NORM_PRIORITY); // Default priority

        // Display thread information before starting the thread
        displayThreadInfo(myThread);

        // Start the thread
        myThread.start();

        // Display thread information after starting the thread
        displayThreadInfo(myThread);
    }

    private static void displayThreadInfo(Thread thread) {
        System.out.println("Thread Name: " + thread.getName());
        System.out.println("Thread Priority: " + thread.getPriority());
        System.out.println("--------------------------");
    }
}
```

2. Write a SERVLET program in java to accept details of student (SeatNo, Stud_Name, Class, Total_Marks). Calculate percentage and grade obtained and display details on page.

```
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("/StudentDetailsServlet")
public class StudentDetailsServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

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

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

        // Get student details from request parameters
        int seatNo = Integer.parseInt(request.getParameter("seatNo"));
        String studName = request.getParameter("studName");
        String className = request.getParameter("className");
        int totalMarks = Integer.parseInt(request.getParameter("totalMarks"));

        // Calculate percentage and grade
        double percentage = (double) totalMarks / 500 * 100;
        String grade = calculateGrade(percentage);

        // Display student details, percentage, and grade on the web page
        out.println("<html><body>");
        out.println("<h2>Student Details:</h2>");
        out.println("<p>Seat No: " + seatNo + "</p>");
        out.println("<p>Student Name: " + studName + "</p>");
        out.println("<p>Class: " + className + "</p>");
        out.println("<p>Total Marks: " + totalMarks + "</p>");
        out.println("<h2>Result:</h2>");
        out.println("<p>Percentage: " + percentage + "%</p>");
        out.println("<p>Grade: " + grade + "</p>");
        out.println("</body></html>");

        out.close();
    }

    private String calculateGrade(double percentage) {
        if (percentage >= 90) {
```

```
            return "A+";
        } else if (percentage >= 80) {
            return "A";
        } else if (percentage >= 70) {
            return "B";
        } else if (percentage >= 60) {
            return "C";
        } else if (percentage >= 50) {
            return "D";
        } else {
            return "Fail";
        }
    }
}
```

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 NegativeIntegersLinkedList {

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

        // Accept 'N' integers from the user
        System.out.print("Enter the value of N: ");
        int n = scanner.nextInt();

        // Create a LinkedList to store integers
        LinkedList<Integer> integerList = new LinkedList<>();

        // Accept integers from the user and add them to the LinkedList
        for (int i = 0; i < n; i++) {
            System.out.print("Enter integer " + (i + 1) + ": ");
            int inputInt = scanner.nextInt();
            integerList.add(inputInt);
        }

        // Display only negative integers from the LinkedList
        System.out.println("Negative Integers:");
        for (int num : integerList) {
```

```
            if (num < 0) {
                System.out.println(num);
            }
        }

        scanner.close();
    }
}
```

2. Write a SERVLET application to accept username and password, search them into database, if found then display appropriate message on the browser otherwise display error message.

```
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("/LoginServlet")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

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

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

        // Get username and password from request parameters
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        // Simulate database check (Replace this with actual database interaction)
        if (isValidUser(username, password)) {
            out.println("<html><body>");
            out.println("<h2>Login Successful</h2>");
            out.println("<p>Welcome, " + username + "!</p>");
            out.println("</body></html>");
        } else {
            out.println("<html><body>");
            out.println("<h2>Login Failed</h2>");
            out.println("<p>Invalid username or password. Please try again.</p>");
```

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

        out.close();
    }

    private boolean isValidUser(String username, String password) {
        // Simulate database check (Replace this with actual database interaction)
        // Here, we assume a valid username: "user123" and password: "password123"
        return username.equals("user123") && password.equals("password123");
    }
}
```

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=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Number to Words</title>
</head>
<body>

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

    <%
        // Process the form submission
        if (request.getMethod().equalsIgnoreCase("POST")) {
            // Get the number from the form
            String numberStr = request.getParameter("number");

            // Display the number in words in red color
            if (numberStr != null && !numberStr.isEmpty()) {
                int number = Integer.parseInt(numberStr);
                String numberInWords = convertToWords(number);

                // Display the result in red color
    %>
```

```
            <p style="color: red;"><strong><%= numberInWords %></strong></p>
    <%
            }
        }

        // Method to convert number to words
        String convertToWords(int number) {
            // Implement your logic to convert the number to words here
            // For simplicity, a basic example is provided
            // You may need to implement a more sophisticated logic based on your requirements

            // Basic example mapping digits to words
            String[] words = {"Zero", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight",
"Nine"};
            StringBuilder result = new StringBuilder();

            while (number > 0) {
                int digit = number % 10;
                result.insert(0, words[digit] + " ");
                number /= 10;
            }

            return result.toString().trim();
        }
    %>

</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 imageIcon;
    private JLabel imageLabel;
    private Timer timer;

    public BlinkingImage() {
        // Load the image icon
        imageIcon = new ImageIcon("path/to/your/image.png");
```

```
        // Create a JLabel with the image icon
        imageLabel = new JLabel(imageIcon);

        // Set up the JFrame
        setTitle("Blinking Image");
        setSize(300, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Add the image label to the JFrame
        add(imageLabel);

        // Create a Timer to toggle the visibility of the image
        timer = new Timer(500, new ActionListener() {
            private boolean visible = true;

            @Override
            public void actionPerformed(ActionEvent e) {
                // Toggle the visibility of the image
                imageLabel.setVisible(visible);
                visible = !visible;
            }
        });

        // Start the timer to blink the image
        timer.start();
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            new 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 SubjectNamesLinkedList {

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

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

        // Accept 'N' subject names from the user
        System.out.print("Enter the value of N: ");
        int n = scanner.nextInt();

        // Create a LinkedList to store subject names
        LinkedList<String> subjectList = new LinkedList<>();

        // Accept subject names from the user and add them to the LinkedList
        for (int i = 0; i < n; i++) {
            System.out.print("Enter subject name " + (i + 1) + ": ");
            String subjectName = scanner.next();
            subjectList.add(subjectName);
        }

        // Display subject names using Iterator
        System.out.println("Subject Names:");
        Iterator<String> iterator = subjectList.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.

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

class Employee {
    private int ENo;
    private String EName;
    private double Salary;

    public Employee(int ENo, String EName, double Salary) {
        this.ENo = ENo;
        this.EName = EName;
        this.Salary = Salary;
    }
```

```
    public int getENo() {
        return ENo;
    }

    public String getEName() {
        return EName;
    }

    public double getSalary() {
        return Salary;
    }

    public void setSalary(double newSalary) {
        this.Salary = newSalary;
    }

    @Override
    public String toString() {
        return "ENo: " + ENo + ", EName: " + EName + ", Salary: " + Salary;
    }
}

public class EmployeeManagementSystem {

    private static ArrayList<Employee> employeeList = new ArrayList<>();
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        int choice;

        do {
            System.out.println("Menu:");
            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();

            switch (choice) {
                case 1:
                    insertEmployee();
                    break;
```

```
            case 2:
                updateEmployee();
                break;
            case 3:
                displayEmployees();
                break;
            case 4:
                System.out.println("Exiting the program. Goodbye!");
                break;
            default:
                System.out.println("Invalid choice. Please enter a valid option.");
        }
    } while (choice != 4);

    scanner.close();
}

private static void insertEmployee() {
    System.out.print("Enter Employee Number (ENo): ");
    int ENo = scanner.nextInt();

    System.out.print("Enter Employee Name (EName): ");
    String EName = scanner.next();

    System.out.print("Enter Employee Salary: ");
    double Salary = scanner.nextDouble();

    Employee newEmployee = new Employee(ENo, EName, Salary);
    employeeList.add(newEmployee);

    System.out.println("Employee added successfully!");
}

private static void updateEmployee() {
    System.out.print("Enter Employee Number (ENo) to update salary: ");
    int ENo = scanner.nextInt();

    boolean found = false;
    for (Employee employee : employeeList) {
        if (employee.getENo() == ENo) {
            System.out.print("Enter new salary for Employee " + ENo + ": ");
            double newSalary = scanner.nextDouble();
            employee.setSalary(newSalary);
            System.out.println("Salary updated successfully!");
```

```
                found = true;
                break;
            }
        }

        if (!found) {
            System.out.println("Employee with Employee Number " + ENo + " not found.");
        }
    }

    private static void displayEmployees() {
        if (employeeList.isEmpty()) {
            System.out.println("No employees to display.");
        } else {
            System.out.println("Employee List:");
            for (Employee employee : employeeList) {
                System.out.println(employee);
            }
        }
    }
}
```

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

```
<%@ page import="java.util.Calendar" %>
<%@ page import="java.text.SimpleDateFormat" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Greeting Page</title>
</head>
<body>

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

    <%
        // Get the username from the request parameters
        String username = request.getParameter("username");

        // Get the current time on the server machine
```

```
        Calendar calendar = Calendar.getInstance();
        int hourOfDay = calendar.get(Calendar.HOUR_OF_DAY);

        // Determine the appropriate greeting based on the time
        String greeting;
        if (hourOfDay >= 0 && hourOfDay < 12) {
            greeting = "Good morning";
        } else if (hourOfDay >= 12 && hourOfDay < 18) {
            greeting = "Good afternoon";
        } else {
            greeting = "Good evening";
        }
    %>

    <%-- Display the greeting message --%>
    <h2><%= 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.

```
import java.util.Scanner;

public class DisplayVowelsWithDelay {

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

        // Accept a string from the user
        System.out.print("Enter a string: ");
        String inputString = scanner.nextLine();

        // Display each vowel after every 3 seconds
        displayVowelsWithDelay(inputString);

        scanner.close();
    }

    private static void displayVowelsWithDelay(String inputString) {
        char[] vowels = {'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'};

        for (char ch : inputString.toCharArray()) {
```

```
            if (isVowel(ch)) {
                System.out.println("Vowel: " + ch);

                try {
                    // Pause execution for 3 seconds
                    Thread.sleep(3000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    private static boolean isVowel(char ch) {
        return "aeiouAEIOU".indexOf(ch) != -1;
    }
}
```

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.

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

public class StudentNamesCollection {

    public static void main(String[] args) {
        // Check if at least one student name is provided as a command line argument
        if (args.length < 1) {
            System.out.println("Please provide at least one student name.");
            return;
        }

        // Create an ArrayList to store student names
        List<String> studentNamesList = new ArrayList<>();

        // Add student names from command line arguments to the ArrayList
        for (String studentName : args) {
            studentNamesList.add(studentName);
        }

        // Display student names using Iterator
        displayStudentNamesWithIterator(studentNamesList);
```

```
        // Display student names using ListIterator
        displayStudentNamesWithListIterator(studentNamesList);
    }

    private static void displayStudentNamesWithIterator(List<String> studentNamesList) {
        System.out.println("Student Names using Iterator:");
        Iterator<String> iterator = studentNamesList.iterator();
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
        System.out.println();
    }

    private static void displayStudentNamesWithListIterator(List<String> studentNamesList) {
        System.out.println("Student Names using ListIterator (in reverse order):");
        ListIterator<String> listIterator = studentNamesList.listIterator(studentNamesList.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

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

public class TextScrolling extends JFrame {

    private final String textToScroll = "This is a scrolling text! ";
    private int scrollPosition = 0;

    public TextScrolling() {
        setTitle("Text Scrolling");
        setSize(400, 100);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        Timer timer = new Timer(100, new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                scrollPosition++;
```

```
            repaint();
        }
    });

    timer.start();
  }

  @Override
  public void paint(Graphics g) {
    super.paint(g);
    g.drawString(textToScroll, getWidth() - scrollPosition, getHeight() / 2);
    if (scrollPosition >= getWidth()) {
      scrollPosition = 0;
    }
  }

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

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.

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

<%
  // Get username and password from request parameters
  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)) {
    // Login successful
    response.sendRedirect("Login.html");
  } else {
    // Login failed
    response.sendRedirect("Error.html");
  }
%>
```

```
<!DOCTYPE html>
<html>
<head>
   <title>Login Successfully</title>
</head>
<body>
   <h2>Login Successfully</h2>
   <p>Welcome, <%= request.getParameter("username") %>!</p>
</body>
</html>

<!DOCTYPE html>
<html>
<head>
   <title>Login Failed</title>
</head>
<body>
   <h2>Login Failed</h2>
   <p>Invalid username or password. Please try again.</p>
</body>
</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.

```
<!DOCTYPE html>
<html>
<head>
   <title>Voter Eligibility Checker</title>
</head>
<body>
   <h2>Voter Eligibility Checker</h2>

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

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

```
<%
   // Get name and age from request parameters
   String name = request.getParameter("name");
   String ageStr = request.getParameter("age");

   // Check if age is a valid number
   try {
      int age = Integer.parseInt(ageStr);

      // Check eligibility based on age criterion (e.g., 18 years or older)
      if (age >= 18) {
         // Eligible for voting
         out.println("<h2>Hello " + name + "!</h2>");
         out.println("<p>Congratulations! You are eligible for voting.</p>");
      } else {
         // Not eligible for voting
         out.println("<h2>Hello " + name + "!</h2>");
         out.println("<p>Sorry! You are not eligible for voting.</p>");
      }
   } catch (NumberFormatException e) {
      // Handle the case where age is not a valid number
      out.println("<h2>Error!</h2>");
      out.println("<p>Please enter a valid age.</p>");
   }
%>
```

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)

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class DeleteEmployeeDetails {

   public static void main(String[] args) {
      // Check if the correct number of command line arguments are provided
      if (args.length != 1) {
         System.out.println("Usage: java DeleteEmployeeDetails <employeeID>");
         return;
      }
```

```
        // Extract employee ID from command line argument
        int employeeID = Integer.parseInt(args[0]);

        // JDBC URL, username, and password of MySQL server
        String jdbcURL = "jdbc:mysql://localhost:3306/your_database";
        String username = "your_username";
        String password = "your_password";

        try (Connection connection = DriverManager.getConnection(jdbcURL, username,
password)) {
            // SQL query to delete employee details based on employee ID
            String sqlQuery = "DELETE FROM Employee WHERE ENo = ?";

            // Create a PreparedStatement
            try (PreparedStatement preparedStatement = connection.prepareStatement(sqlQuery)) {
                // Set the parameter (employee ID) in the PreparedStatement
                preparedStatement.setInt(1, employeeID);

                // Execute the query
                int rowsAffected = preparedStatement.executeUpdate();

                // Check if any rows were affected
                if (rowsAffected > 0) {
                    System.out.println("Employee details deleted successfully.");
                } else {
                    System.out.println("Employee not found with ID: " + employeeID);
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

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" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Sum of First and Last Digit</title>
</head>
<body>
```

```
    <h2>Sum of First and Last Digit Calculator</h2>

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

    <%
        // Retrieve the number from the request parameter
        String numberStr = request.getParameter("number");

        if (numberStr != null && !numberStr.isEmpty()) {
            try {
                // Parse the number
                int number = Integer.parseInt(numberStr);

                // Calculate the sum of first and last digit
                int lastDigit = number % 10;
                int firstDigit = Character.getNumericValue(Integer.toString(number).charAt(0));
                int sum = firstDigit + lastDigit;

                // Display the result in red color with font size 18
    %>
                <p style="color: red; font-size: 18px;">
                    Sum of the first and last digit of <%= number %> is: <%= sum %>
                </p>
    <%
            } catch (NumberFormatException e) {
                // Handle the case where the input is not a valid number
    %>
                <p style="color: red; font-size: 18px;">
                    Invalid input! 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.

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

public class CollegeDetailsJTable extends JFrame {

    public CollegeDetailsJTable() {
        // Sample data for College details
        Object[][] data = {
                {1, "ABC College", "123 Main St", 2020},
                {2, "XYZ College", "456 Broad St", 2019},
                {3, "PQR College", "789 Oak St", 2021}
        };

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

        // Create a DefaultTableModel
        DefaultTableModel model = new DefaultTableModel(data, columnNames);

        // Create a JTable with the DefaultTableModel
        JTable jTable = new JTable(model);

        // Create a JScrollPane and add the JTable to it
        JScrollPane jScrollPane = new JScrollPane(jTable);

        // Set up the JFrame
        setTitle("College Details");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Add the JScrollPane to the JFrame
        add(jScrollPane, BorderLayout.CENTER);
    }

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

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

```
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("/ChangeInactiveIntervalServlet")
public class ChangeInactiveIntervalServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

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

        // Get the session
        HttpSession session = request.getSession();

        // Set the inactive time interval to 180 seconds (3 minutes)
        session.setMaxInactiveInterval(180);

        response.getWriter().println("Inactive time interval changed to 180 seconds.");
    }
}
```

SLIP 28

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

```
<!DOCTYPE html>
<html>
<head>
    <title>Reverse String</title>
</head>
<body>

    <h2>Reverse String</h2>

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

    <%
```

```
        // Retrieve the input string from the request parameters
        String inputString = request.getParameter("inputString");

        // Check if the form has been submitted
        if (inputString != null && !inputString.isEmpty()) {
            // Reverse the input string
            String reversedString = new StringBuilder(inputString).reverse().toString();
    %>
            <p>Original String: <%= inputString %></p>
            <p>Reversed String: <%= reversedString %></p>
    <%
        }
    %>

</body>
</html>
```

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

```
public class CurrentThreadExample {

    public static void main(String[] args) {
        // Create and start two threads
        Thread thread1 = new Thread(new MyRunnable(), "Thread 1");
        Thread thread2 = new Thread(new MyRunnable(), "Thread 2");

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

        // Display the name of the main thread
        System.out.println("Main thread name: " + Thread.currentThread().getName());
    }

    static class MyRunnable implements Runnable {
        @Override
        public void run() {
            // Display the name of the currently executing thread
            System.out.println("Thread name: " + Thread.currentThread().getName());
        }
    }
}
```

SLIP 29

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

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;

public class ColumnInfoExample {

    public static void main(String[] args) {
        // JDBC URL, username, and password of MySQL server
        String jdbcURL = "jdbc:mysql://localhost:3306/your_database";
        String username = "your_username";
        String password = "your_password";

        try (Connection connection = DriverManager.getConnection(jdbcURL, username,
password)) {
            // SQL query to retrieve all columns from the DONAR table
            String sqlQuery = "SELECT * FROM DONAR WHERE 1 = 0"; // Adding WHERE
condition to fetch no records

            try (PreparedStatement preparedStatement = connection.prepareStatement(sqlQuery);
                 ResultSet resultSet = preparedStatement.executeQuery()) {

                // Get ResultSetMetaData to retrieve column details
                ResultSetMetaData metaData = resultSet.getMetaData();

                // Get the number of columns
                int columnCount = metaData.getColumnCount();

                System.out.println("Column Details for DONAR table:");

                // Iterate over columns and print details
                for (int i = 1; i <= columnCount; i++) {
                    System.out.println("Column " + i + ":");
                    System.out.println("  Name: " + metaData.getColumnName(i));
                    System.out.println("  Type: " + metaData.getColumnTypeName(i));
                    System.out.println("  Size: " + metaData.getColumnDisplaySize(i));
                    System.out.println("  Nullable: " + (metaData.isNullable(i) ==
ResultSetMetaData.columnNullable ? "Yes" : "No"));
                    System.out.println();
                }
            }
```

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

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 LinkedListExample {

    public static void main(String[] args) {
        // Create a LinkedList of integers
        LinkedList<Integer> linkedList = new LinkedList<>();

        // i. Add element at the first position
        addElementAtFirstPosition(linkedList, 10);
        addElementAtFirstPosition(linkedList, 20);
        addElementAtFirstPosition(linkedList, 30);

        // ii. Delete last element
        deleteLastElement(linkedList);

        // iii. Display the size of the linked list
        displaySize(linkedList);
    }

    // Add element at the first position
    private static void addElementAtFirstPosition(LinkedList<Integer> list, int element) {
        list.addFirst(element);
        System.out.println("Added " + element + " at the first position. List: " + list);
    }

    // Delete last element
    private static void deleteLastElement(LinkedList<Integer> list) {
        if (!list.isEmpty()) {
            Integer removedElement = list.removeLast();
            System.out.println("Removed last element " + removedElement + ". List: " + list);
        } else {
            System.out.println("The list is empty.");
        }
    }
```

```
    // Display the size of the linked list
    private static void displaySize(LinkedList<Integer> list) {
        System.out.println("Size of the linked list: " + list.size());
    }
}
```

SLIP 30

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.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class ScrollableResultSetExample {

    public static void main(String[] args) {
        // JDBC URL, username, and password of the database
        String jdbcURL = "jdbc:mysql://localhost:3306/your_database";
        String username = "your_username";
        String password = "your_password";

        try (Connection connection = DriverManager.getConnection(jdbcURL, username,
password)) {
            // Create a Statement with a scrollable ResultSet
            Statement statement =
connection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);

            // Execute a query to retrieve data from the Teacher table
            String query = "SELECT TID, TName, Salary FROM Teacher";
            ResultSet resultSet = statement.executeQuery(query);

            // Move to the last row
            resultSet.last();

            // Get the row count
            int rowCount = resultSet.getRow();

            System.out.println("Total rows in Teacher table: " + rowCount);

            // Move back to the first row
            resultSet.beforeFirst();
```

```
            // Iterate over the result set and print data
            System.out.println("Teacher table data:");
            while (resultSet.next()) {
                int teacherId = resultSet.getInt("TID");
                String teacherName = resultSet.getString("TName");
                double salary = resultSet.getDouble("Salary");

                System.out.println("TID: " + teacherId + ", TName: " + teacherName + ", Salary: " +
salary);
            }

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

1. Write a java program for the implementation of synchronization

```
class Counter {
    private int count = 0;

    // Synchronized method to increment the counter
    public synchronized void increment() {
        count++;
        System.out.println(Thread.currentThread().getName() + " incremented count to " + count);
    }

    // Synchronized method to decrement the counter
    public synchronized void decrement() {
        count--;
        System.out.println(Thread.currentThread().getName() + " decremented count to " + count);
    }

    // Method to get the current count value
    public int getCount() {
        return count;
    }
}

class IncrementThread extends Thread {
    private Counter counter;

    public IncrementThread(Counter counter) {
```

```
        this.counter = counter;
    }

    @Override
    public void run() {
        for (int i = 0; i < 5; i++) {
            counter.increment();
            try {
                Thread.sleep(100); // Simulate some work being done
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

class DecrementThread extends Thread {
    private Counter counter;

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

    @Override
    public void run() {
        for (int i = 0; i < 5; i++) {
            counter.decrement();
            try {
                Thread.sleep(150); // Simulate some work being done
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

public class SynchronizationExample {
    public static void main(String[] args) {
        // Create a shared Counter object
        Counter counter = new Counter();

        // Create two threads to increment and decrement the counter
        IncrementThread incrementThread = new IncrementThread(counter);
        DecrementThread decrementThread = new DecrementThread(counter);
```

```
        // Start the threads
        incrementThread.start();
        decrementThread.start();

        // Wait for threads to finish
        try {
            incrementThread.join();
            decrementThread.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        // Display the final count value
        System.out.println("Final count value: " + counter.getCount());
    }
}
```