Slip 1

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

→

```
public class AlphabetDisplay implements Runnable {
  @Override
  public void run() {
     // Loop through characters 'A' to 'Z'
     for (char c = 'A'; c <= 'Z'; c++) {
        // Display the current character
        System.out.print(c + " ");
        try {
           // Sleep for 2000 milliseconds (2 seconds)
           Thread.sleep(2000);
        } catch (InterruptedException e) {
           e.printStackTrace(); // Handle any interruption errors
        }
     }
  }
  public static void main(String[] args) {
     // Create an instance of AlphabetDisplay (which implements Runnable)
     AlphabetDisplay alphabetDisplay = new AlphabetDisplay();
     // Create a new thread, passing the Runnable instance
     Thread alphabetThread = new Thread(alphabetDisplay);
     // Start the thread
     alphabetThread.start();
  }
}
```

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)

→

```
package com.DPU;

import java.awt.GridLayout;
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;
```

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class SlipOneKaTwo {

	public static void main(String[] args) {
		// create the frame of form

		JFrame frame=new JFrame("Employee form");
		frame.setSize(400,300);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		// create a panel to hold the components

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

		// create the labels and text fields for employee details


		JLabel Leno=new JLabel("employee no");
		JLabel Lename=new JLabel("employee name");
		JLabel Ldesignation=new JLabel("employee designation");
		JLabel Lsalary=new JLabel("employee salary");

		JTextField Feno=new JTextField();
		JTextField Fename=new JTextField();
		JTextField Fdesignation=new JTextField();
		JTextField Fsalary=new JTextField();

		panel.add(Leno);
		panel.add(Feno);
		panel.add(Lename);
		panel.add(Fename);
		panel.add(Ldesignation);
		panel.add(Fdesignation);
		panel.add(Lsalary);
		panel.add(Fsalary);
```

```
		JButton saveButton=new JButton("save");
		saveButton.addActionListener(new ActionListener() {

			@Override
			public void actionPerformed(ActionEvent e) {
				// TODO Auto-generated method stub
				int eno=Integer.parseInt(Feno.getText());
				String ename=Fename.getText();
				String designation=Fdesignation.getText();
				double salary=Double.parseDouble(Fsalary.getText());

				saveEmployeeToDatabase(eno, ename, designation, salary);
			}
		}

				);

		// add savebutton to panel
		panel.add(saveButton);

		// add panel to frame

		frame.add(panel);
		frame.setVisible(true);

			}
	public static void saveEmployeeToDatabase(int eno,String ename,String
designation,double salary) {

		Connection conn=null;
		PreparedStatement pstmt=null;

		try {
			// make connection


conn=DriverManager.getConnection("jdbc:postgresql://localhost:5432/dpu","postgres","12345");

			// SQL query to insert employee details

			String sql="insert into
employee(eno,ename,designation,salary)values(?,?,?,?)";

			pstmt=conn.prepareStatement(sql);
```

```
                        pstmt.setInt(1, eno);
                        pstmt.setString(2, ename);
                        pstmt.setString(3, designation);
                        pstmt.setDouble(4, salary);

                        // execute the update

                        pstmt.executeUpdate();

                        JOptionPane.showMessageDialog(null,"employee detail enter
successfully");

                }
                catch(SQLException e) {
                        // handle database errors;
                        JOptionPane.showMessageDialog(null,"error saving employee
details:"+e.getMessage());
                        e.printStackTrace();
                }

                finally {
                        try {
                                if(pstmt!=null) {
                                        pstmt.close();
                                }
                                if(conn!=null) {
                                        conn.close();
                                }
                        }
                        catch(SQLException e) {
                                e.printStackTrace();
                        }
                }

        }

}
```

Slip 2

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

→

```
import java.util.*;

public class FriendNames {
 public static void main(String[] args) {
 // Scanner to read user input
 Scanner scanner = new Scanner(System.in);

 // Ask the user how many names they want to enter
 System.out.print("Enter the number of friends: ");
 int N = scanner.nextInt();
 scanner.nextLine(); // To consume the newline character left by nextInt()

 // Create a HashSet to store the names (HashSet removes duplicates)
 Set<String> friendsSet = new HashSet<>();

 // Loop to read N names from the user
 System.out.println("Enter the names of your friends:");
 for (int i = 0; i < N; i++) {
 String name = scanner.nextLine();
 friendsSet.add(name); // Add name to the HashSet
 }

 // Convert the HashSet to a List and sort it in ascending order
 List<String> sortedList = new ArrayList<>(friendsSet);
 Collections.sort(sortedList);

 // Display the names in ascending order
 System.out.println("\nFriends' names in ascending order:");
 for (String name : sortedList) {
 System.out.println(name);
 }

 // Close the scanner
 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.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class RequestInfoServlet extends HttpServlet {

        @Override
        protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
                // TODO Auto-generated method stub

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

      // Get client information
      String clientIP = request.getRemoteAddr();
      String userAgent = request.getHeader("User-Agent");

      // Get server information
      ServletContext context = getServletContext();
      String serverOS = System.getProperty("os.name");
      Enumeration<String> servletNames = context.getServletNames();

      // Output the information
```

```
        out.println("<html><head><title>Request & Server Info</title></head><body>");
        out.println("<h2>Client Information</h2>");
        out.println("<p><strong>IP Address:</strong> " + clientIP + "</p>");
        out.println("<p><strong>Browser:</strong> " + userAgent + "</p>");

        out.println("<h2>Server Information</h2>");
        out.println("<p><strong>Operating System:</strong> " + serverOS + "</p>");
        out.println("<p><strong>Loaded Servlets:</strong></p><ul>");

        while (servletNames.hasMoreElements()) {
            out.println("<li>" + servletNames.nextElement() + "</li>");
        }

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

          }
}
—----------------------------------------------------------------------------------
web.xml
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd" id="WebApp_ID" version="4.0">
  <display-name>ServleJSPAssignment</display-name>
  <servlet>
        <servlet-name>RequestInfoServlet</servlet-name>
        <servlet-class>com.assign1.RequestInfoServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>RequestInfoServlet</servlet-name>
        <url-pattern>/requestinfo</url-pattern>
    </servlet-mapping>

</web-app>
```

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=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
    <title>Patient Details</title>
    <style>
        table {
            width: 60%;
            border-collapse: collapse;
            margin: 20px 0;
        }
        th, td {
            border: 1px solid black;
            padding: 8px;
            text-align: center;
        }
        th {
            background-color: #f2f2f2;
        }
    </style>
</head>
<body>

    <h2>Patient Details</h2>

    <table>
        <tr>
            <th>PNo</th>
            <th>PName</th>
            <th>Address</th>
            <th>Age</th>
            <th>Disease</th>
        </tr>
```

```
    <%-- Simulating Patient Data using an Array --%>
    <%
        class Patient {
            int pNo;
            String pName;
            String address;
            int age;
            String disease;

            public Patient(int pNo, String pName, String address, int age, String disease) {
                this.pNo = pNo;
                this.pName = pName;
                this.address = address;
                this.age = age;
                this.disease = disease;
            }
        }

        // Sample Patient Data (Can be retrieved from a database in real scenarios)
        Patient[] patients = {
            new Patient(101, "John Doe", "New York", 35, "Fever"),
            new Patient(102, "Alice Smith", "Los Angeles", 28, "Cold"),
            new Patient(103, "Bob Johnson", "Chicago", 45, "Diabetes"),
            new Patient(104, "Emma Watson", "Houston", 50, "Hypertension")
        };

        // Loop through the array and display patient details in the table
        for (Patient p : patients) {
    %>
    <tr>
        <td><%= p.pNo %></td>
        <td><%= p.pName %></td>
        <td><%= p.address %></td>
        <td><%= p.age %></td>
        <td><%= p.disease %></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.*;

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

 // Step 1: Add elements at the end of the list
 list.add("Alice");
 list.add("Bob");
 list.add("Charlie");
 list.add("David");

 System.out.println("List after adding elements at the end: " + list);

 // Step 2: Delete the first element of the list
 if (!list.isEmpty()) {
 list.removeFirst(); // Removes the first element
 System.out.println("List after deleting the first
element: " + list);  } else {
 System.out.println("List is empty, cannot remove first
element.");  }

 // Step 3: Display the contents of the list in reverse order
 System.out.println("List in reverse order:");
 Iterator<String> iterator = list.descendingIterator(); // Iterator for
reverse order  while (iterator.hasNext()) {
 System.out.println(iterator.next());
 }
 }
```

}

Slip 4

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

→

```
import javax.swing.*;
import java.awt.*;
public class TextBlinking implements Runnable {
  private JLabel label;  // Label to display the blinking text
  private boolean isVisible;  // Flag to control the visibility of the text
  public TextBlinking(JLabel label) {
    this.label = label;
    this.isVisible = true;  // Start with text visible
  }
  @Override
  public void run() {
    while (true) {
      try {
        // Toggle the visibility of the text
        if (isVisible) {
          label.setText("Blinking Text");
          label.setForeground(Color.RED);  // Set text color to red
        } else {
          label.setText("");  // Hide text by setting it to an empty string
        }

        isVisible = !isVisible;  // Toggle the visibility flag
        Thread.sleep(500);  // Wait for 500 milliseconds (0.5 seconds)
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }
  }
  public static void main(String[] args) {
    // Create JFrame window
    JFrame frame = new JFrame("Text Blinking Example");
    frame.setSize(400, 200);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setLocationRelativeTo(null);
    // Create a label to display blinking text
    JLabel label = new JLabel("", SwingConstants.CENTER);
    label.setFont(new Font("Serif", Font.BOLD, 30));
    label.setPreferredSize(new Dimension(400, 100));

    // Add the label to the frame
```

```
        frame.add(label, BorderLayout.CENTER);
        // Create a new TextBlinking Runnable instance
        TextBlinking textBlinking = new TextBlinking(label);
        // Create a new thread to run the blinking text
        Thread blinkThread = new Thread(textBlinking);
        // Start the thread
        blinkThread.start();
        // Make the frame visible
        frame.setVisible(true);
    }
}
================================================================================
====
package com.test;
/*
*  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 MultiThreadExample {
    // Shared resource to store the generated number
    private static int generatedNumber = 0;
    public static void main(String[] args) {
        // Thread 1 - Generates a random number every second
        Thread generatorThread = new Thread(new Runnable() {
            @Override
            public void run() {
                Random random = new Random();
                while (true) {
                    try {
                        // Generate a random number between 1 and 100
                        generatedNumber = random.nextInt(100) + 1;
                        System.out.println("Generated Number: " + generatedNumber);
                        Thread.sleep(1000); // Wait for 1 second before generating a new number
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        });
        // Thread 2 - Computes the square if the number is even
        Thread squareThread = new Thread(new Runnable() {
            @Override
            public void run() {
                while (true) {
                    if (generatedNumber % 2 == 0) { // Check if the number is even
```

```
                int square = generatedNumber * generatedNumber;
                System.out.println("Square of " + generatedNumber + " is: " + square);
            }
            try {
                Thread.sleep(500); // Wait for 0.5 seconds before checking again
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
});
// Thread 3 - Computes the cube if the number is odd
Thread cubeThread = new Thread(new Runnable() {
    @Override
    public void run() {
        while (true) {
            if (generatedNumber % 2 != 0) { // Check if the number is odd
                int cube = generatedNumber * generatedNumber * generatedNumber;
                System.out.println("Cube of " + generatedNumber + " is: " + cube);
            }
            try {
                Thread.sleep(500); // Wait for 0.5 seconds before checking again
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
});
// Start all threads
generatorThread.start();
squareThread.start();
cubeThread.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 java.util.*;

public class CitySTDCode {
```

```
 public static void main(String[] args) {
 // Create a HashMap to store city names and their STD codes
 Map<String, String> citySTDMap = new HashMap<>();

 // Add some initial data
 citySTDMap.put("Mumbai", "022");
 citySTDMap.put("Delhi", "011");
 citySTDMap.put("Bangalore", "080");
 // Print the initial cities and their codes
 System.out.println("Initial City STD Codes: " + citySTDMap);

 // i. Add a new city and its code (No duplicates)
 addCity(citySTDMap, "Chennai", "044");
 addCity(citySTDMap, "Mumbai", "022"); // Duplicate city, will
not be added  System.out.println("\nAfter adding cities: " +
citySTDMap);

 // ii. Remove a city from the collection
 removeCity(citySTDMap, "Delhi");
 System.out.println("\nAfter removing Delhi: " + citySTDMap);

 // iii. Search for a city name and display the code
 searchCity(citySTDMap, "Bangalore");
 searchCity(citySTDMap, "Delhi");
 }

 // Method to add a city and its code (no duplicates)
 public static void addCity(Map<String, String> map, String city,
String code) {  if (!map.containsKey(city)) {
 map.put(city, code);
 System.out.println("Added: " + city + " with STD code
" + code);  } else {
 System.out.println("City " + city + " already exists with STD code " +
map.get(city));  }
 }

 // Method to remove a city from the collection
 public static void removeCity(Map<String, String>
map, String city) {  if (map.containsKey(city)) {
 map.remove(city);
```

```
 System.out.println("Removed: " + city);
 } else {
 System.out.println("City " + city + " not found in the
collection.");  }
 }

 // Method to search for a city and display its STD code
 public static void searchCity(Map<String, String>
map, String city) {  if (map.containsKey(city)) {
 System.out.println("STD code for " + city + " is: " +
map.get(city));  } else {
 System.out.println("City " + city + " not found in the
collection.");  }
 }
}
```

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 students using the Enumeration interface.

→

```
import java.util.*;

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

```
// Adding student names and mobile numbers
studentTable.put("Alice", "9876543210");
studentTable.put("Bob", "9123456789");
studentTable.put("Charlie", "9345678901");
studentTable.put("David", "9054321098");

// Display student details using Enumeration interface
System.out.println("Student Details (Name - Mobile Number):");

// Get the enumeration for keys (names)
Enumeration<String> names = studentTable.keys();

// Get the enumeration for values (mobile numbers)
Enumeration<String> numbers = studentTable.elements();

// Print the details by iterating over both the name and mobile number using
Enumeration  while (names.hasMoreElements()) {
String name = names.nextElement();
String number = numbers.nextElement();
System.out.println(name + " - " + number);
}
}
}
```

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.

→

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
    <title>Prime Number Checker</title>
</head>
```

```
<body>
    <h2>Enter a Number to Check if it's Prime</h2>
    <form method="post">
        <input type="number" name="num" required>
        <input type="submit" value="Check">
    </form>

    <%
        // Get the number from the form
        String numStr = request.getParameter("num");

        if (numStr != null) {
            int num = Integer.parseInt(numStr);
            boolean isPrime = (num > 1); // Assume prime for numbers > 1

            // Check for factors
            for (int i = 2; i <= Math.sqrt(num); i++) {
                if (num % i == 0) {
                    isPrime = false;
                    break;
                }
            }

            // Display result in red color
    %>
            <h3 style="color: red;">
                <%= num %> is <%= (isPrime ? "a Prime Number" : "not a Prime Number") %>.
            </h3>
    <%
        }
    %>
</body>
```

==========

/* 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 */

customerSearch.html

```
<!DOCTYPE html>
<html>
<head>
   <title>Customer Search</title>
</head>
<body>
   <h2>Search Customer Details</h2>
   <form action="CustomerSearchServlet" method="get">
      <label>Enter Customer Number:</label>
      <input type="number" name="cust_no" required>
      <input type="submit" value="Search">
   </form>
</body>
</html>
```

CustomerSearchServlet.java

```
public class CustomerSearchServlet extends HttpServlet {

        @Override
        protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
                // TODO Auto-generated method stub
                response.setContentType("text/html");
      PrintWriter out = response.getWriter();

      int custNo = Integer.parseInt(request.getParameter("cust_no"));

      try {
         // Database Connection
         Class.forName("com.mysql.cj.jdbc.Driver");
         Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/CustomerDB?useSSL=false&serverTi
mezone=UTC\r\n"
                        , "root", "mysqldb");

         // SQL Query
         String query = "SELECT * FROM customer WHERE cust_no = ?";
         PreparedStatement ps = con.prepareStatement(query);
         ps.setInt(1, custNo);
```

```
            ResultSet rs = ps.executeQuery();

            // Display Customer Details if Found
            if (rs.next()) {
                out.println("<h2>Customer Details:</h2>");
                out.println("<p><b>Customer Number:</b> " + rs.getInt("cust_no") + "</p>");
                out.println("<p><b>Name:</b> " + rs.getString("name") + "</p>");
                out.println("<p><b>Address:</b> " + rs.getString("address") + "</p>");
                out.println("<p><b>Phone:</b> " + rs.getString("phone") + "</p>");
            } else {
                out.println("<h2 style='color: red;'>Customer not found!</h2>");
            }

            con.close();
        } catch (Exception e) {
            out.println("<h3 style='color: red;'>Error: " + e.getMessage() + "</h3>");
        }

          }

}
```

---

web.xml

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

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

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 a predefined search method in the Collection framework.

→

```
import java.util.*;

public class SortedUniqueIntegers {
 public static void main(String[] args) {
 // Create a TreeSet to store integers (automatically sorted and no duplicates)
 Set<Integer> numberSet = new TreeSet<>();

 // Scanner to read user input
 Scanner scanner = new Scanner(System.in);

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

 // Input n integers and store them in the TreeSet
 System.out.println("Enter the integers:");
 for (int i = 0; i < n; i++) {
 int num = scanner.nextInt();
 numberSet.add(num); // TreeSet automatically handles duplicates
 }

 // Display the integers in sorted order
 System.out.println("\nSorted integers without duplicates:");
 for (Integer number : numberSet) {
 System.out.println(number);
 }

 // Search for a particular element
 System.out.print("\nEnter the integer to search for: ");
 int searchNum = scanner.nextInt();
 if (numberSet.contains(searchNum)) {
 System.out.println(searchNum + " is present in the collection.");
 } else {
 System.out.println(searchNum + " is not present in the collection.");
```

```
}

// Close the scanner
scanner.close();
}
}
```

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

→

```
public class ThreadInfoExample {
  public static void main(String[] args) {
    // Create a new thread using a lambda expression
    Thread myThread = new Thread(() -> {
      // Inside the thread, display the name and priority of the current thread
      System.out.println("Thread Name: " + Thread.currentThread().getName());
      System.out.println("Thread Priority: " + Thread.currentThread().getPriority());
    });
    // Set a custom name and priority for the thread
    myThread.setName("MyCustomThread");
    myThread.setPriority(Thread.MAX_PRIORITY); // Highest priority (10)
    // Display the main thread's name and priority
    System.out.println("Main Thread Name: " + Thread.currentThread().getName());
    System.out.println("Main Thread Priority: " + Thread.currentThread().getPriority());
    // Start the custom thread
    myThread.start();
  }
}
```

Slip 7

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

→

```
import java.util.Random;
public class MultiThreadExample {
  // Shared resource to store the generated number
  private static int generatedNumber = 0;
  public static void main(String[] args) {
```

```
// Thread 1 - Generates a random number every second
Thread generatorThread = new Thread(new Runnable() {
    @Override
    public void run() {
        Random random = new Random();
        while (true) {
            try {
                // Generate a random number between 1 and 100
                generatedNumber = random.nextInt(100) + 1;
                System.out.println("Generated Number: " + generatedNumber);
                Thread.sleep(1000); // Wait for 1 second before generating a new number
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
});
// Thread 2 - Computes the square if the number is even
Thread squareThread = new Thread(new Runnable() {
    @Override
    public void run() {
        while (true) {
            if (generatedNumber % 2 == 0) { // Check if the number is even
                int square = generatedNumber * generatedNumber;
                System.out.println("Square of " + generatedNumber + " is: " + square);
            }
            try {
                Thread.sleep(500); // Wait for 0.5 seconds before checking again
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
});
// Thread 3 - Computes the cube if the number is odd
Thread cubeThread = new Thread(new Runnable() {
    @Override
    public void run() {
        while (true) {
            if (generatedNumber % 2 != 0) { // Check if the number is odd
                int cube = generatedNumber * generatedNumber * generatedNumber;
                System.out.println("Cube of " + generatedNumber + " is: " + cube);
            }
            try {
                Thread.sleep(500); // Wait for 0.5 seconds before checking again
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
```

```
        }
    });
    // Start all threads
    generatorThread.start();
    squareThread.start();
    cubeThread.start();
  }
}
```

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.
→

```
package com.DPU;
import java.sql.*;
import java.util.*;

public class Slip7ka2 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Creating product table");
        createProductTable();

        System.out.println("Inserting product records");
        insertProductRecords(scanner);

        System.out.println("Displaying product records");
        displayAllProducts();

        scanner.close();
    }

    public static void createProductTable() {
        Connection conn = null;
        Statement stmt = null;
        try {
            conn = DriverManager.getConnection("jdbc:postgresql://localhost:5432/dpu", "postgres", "12345");
            stmt = conn.createStatement();

            String createTableSQL = "CREATE TABLE IF NOT EXISTS product(" +
                                    "pid INT PRIMARY KEY, " +
                                    "pname VARCHAR(20), " +
                                    "price DECIMAL(10,2))";
            stmt.executeUpdate(createTableSQL);
            System.out.println("Product table created successfully");
```

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

public static void insertProductRecords(Scanner scanner) {
    Connection conn = null;
    PreparedStatement pstmt = null;
    try {
        conn = DriverManager.getConnection("jdbc:postgresql://localhost:5432/dpu", "postgres", "12345");
        pstmt = conn.prepareStatement("INSERT INTO product VALUES (?, ?, ?)");

        for (int i = 0; i < 3; i++) { // Fix loop condition
            System.out.println("Enter the details for product " + (i + 1) + ":");

            System.out.print("Product ID: ");
            int pid = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            System.out.print("Product Name: ");
            String pname = scanner.nextLine();

            System.out.print("Product Price: ");
            double price = scanner.nextDouble();

            pstmt.setInt(1, pid);
            pstmt.setString(2, pname);
            pstmt.setDouble(3, price);
            pstmt.executeUpdate();
        }

        System.out.println("Product records inserted successfully");

    } catch (SQLException e) {
        System.out.println("Error inserting records: " + 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());
        }
```

```
        }
    }

    public static void displayAllProducts() {
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;
        try {
            conn = DriverManager.getConnection("jdbc:postgresql://localhost:5432/dpu", "postgres", "12345");
            stmt = conn.createStatement();
            rs = stmt.executeQuery("SELECT * FROM product");

            System.out.println("Product Records:");
            while (rs.next()) {
                int pid = rs.getInt("pid"); // Fix case issue
                String pname = rs.getString("pname");
                double price = rs.getDouble("price");

                System.out.println("PID: " + pid + ", Name: " + pname + ", Price: " + price);
            }
        } catch (SQLException e) {
            System.out.println("Error fetching records: " + e.getMessage());
        } finally {
            try {
                if (rs != null) rs.close();
                if (stmt != null) stmt.close();
                if (conn != null) conn.close();
            } catch (SQLException e) {
                System.out.println("Error closing resources: " + e.getMessage());
            }
        }
    }
}
```

Slip 8

1. Write a java program to define a thread for printing text on the output screen for ‘n’ number of times. Create 3 threads and run them. Pass the text ‘n’ parameters to the thread constructor. Example: i. ii. iii. First thread prints “COVID19” 10 times. Second thread prints “LOCKDOWN 2020” 20 times Third thread prints “VACCINATED 2021” 30 times

→

```
class PrintTextThread extends Thread {
  private String text;
  private int n;
  // Constructor to accept the text and the number of times to print
  public PrintTextThread(String text, int n) {
    this.text = text;
    this.n = n;
  }
  // The run method will print the text 'n' times
  @Override
  public void run() {
    for (int i = 0; i < n; i++) {
      System.out.println(text);
      try {
        Thread.sleep(100); // Adding a small delay to avoid overwhelming the output
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }
  }
}
public class MultiThreadTextPrinting {
  public static void main(String[] args) {
    // Creating three threads with different text and repetition counts
    Thread thread1 = new PrintTextThread("COVID19", 10);
    Thread thread2 = new PrintTextThread("LOCKDOWN2020", 20);
    Thread thread3 = new PrintTextThread("VACCINATED2021", 30);
    // Starting all threads
    thread1.start();
    thread2.start();
    thread3.start();
  }
}
==========================================================================
==
package com.test;
/*
* Write a Java program to create a thread for moving a ball inside a panel vertically.
* T ball should be created when the user clicks on the start button.
* */
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class BallMovement extends JFrame {
  private int ballX = 100; // Starting X position of the ball
  private int ballY = 100; // Starting Y position of the ball
```

```
private final int ballDiameter = 30; // Diameter of the ball
private final int panelHeight = 400; // Height of the panel for boundary check
private boolean moveDown = true; // Direction of ball movement (down or up)
// JPanel to draw the ball
private JPanel panel;
public BallMovement() {
    // Set up the JFrame
    setTitle("Ball Movement");
    setSize(400, 450); // Set frame size
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLocationRelativeTo(null);
    // Create the panel
    panel = new JPanel() {
        @Override
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            g.setColor(Color.RED);
            g.fillOval(ballX, ballY, ballDiameter, ballDiameter); // Draw the ball
        }
    };
    panel.setBackground(Color.WHITE);
    add(panel, BorderLayout.CENTER);
    // Create the start button
    JButton startButton = new JButton("Start");
    add(startButton, BorderLayout.SOUTH);
    // Add action listener to the button to start the ball movement
    startButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            // Start the ball movement in a new thread
            startBallMovement();
        }
    });
}
// Method to start the ball movement in a new thread
private void startBallMovement() {
    Thread ballThread = new Thread(new Runnable() {
        @Override
        public void run() {
            while (true) {
                try {
                    // Move the ball vertically
                    if (moveDown) {
                        ballY += 5; // Move the ball down
                    } else {
                        ballY -= 5; // Move the ball up
                    }
                    // If the ball hits the bottom or top of the panel, change direction
                    if (ballY >= panelHeight - ballDiameter) {
```

```
                    moveDown = false; // Change direction to upwards
                } else if (ballY <= 0) {
                    moveDown = true; // Change direction to downwards
                }
                // Repaint the panel to reflect the new position of the ball
                panel.repaint();
                // Sleep for a short time to control the speed of the movement
                Thread.sleep(20);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    });
    ballThread.start(); // Start the movement thread
  }
  public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
      @Override
      public void run() {
        new BallMovement().setVisible(true); // Create and show the GUI
      }
    });
  }
}
```

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=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
    <title>Prime Number Checker</title>
</head>
<body>
    <h2>Enter a Number to Check if it's Prime</h2>
    <form method="post">
        <input type="number" name="num" required>
        <input type="submit" value="Check">
    </form>
```

```
    <%
        // Get the number from the form
        String numStr = request.getParameter("num");

        if (numStr != null) {
            int num = Integer.parseInt(numStr);
            boolean isPrime = (num > 1); // Assume prime for numbers > 1

            // Check for factors
            for (int i = 2; i <= Math.sqrt(num); i++) {
                if (num % i == 0) {
                    isPrime = false;
                    break;
                }
            }

            // Display result in red color
    %>
            <h3 style="color: red;">
                <%= num %> is <%= (isPrime ? "a Prime Number" : "not a Prime Number") %>.
            </h3>
    <%
        }
    %>
</body>
```

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.

→

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

```
import java.awt.event.ActionListener;
public class BallMovement extends JFrame {
    private int ballX = 100; // Starting X position of the ball
    private int ballY = 100; // Starting Y position of the ball
    private final int ballDiameter = 30; // Diameter of the ball
    private final int panelHeight = 400; // Height of the panel for boundary check
    private boolean moveDown = true; // Direction of ball movement (down or up)
    // JPanel to draw the ball
    private JPanel panel;
    public BallMovement() {
        // Set up the JFrame
        setTitle("Ball Movement");
        setSize(400, 450); // Set frame size
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        // Create the panel
        panel = new JPanel() {
            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                g.setColor(Color.RED);
                g.fillOval(ballX, ballY, ballDiameter, ballDiameter); // Draw the ball
            }
        };
        panel.setBackground(Color.WHITE);
        add(panel, BorderLayout.CENTER);
        // Create the start button
        JButton startButton = new JButton("Start");
        add(startButton, BorderLayout.SOUTH);
        // Add action listener to the button to start the ball movement
        startButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Start the ball movement in a new thread
                startBallMovement();
            }
        });
    }
    // Method to start the ball movement in a new thread
    private void startBallMovement() {
        Thread ballThread = new Thread(new Runnable() {
            @Override
            public void run() {
                while (true) {
                    try {
                        // Move the ball vertically
                        if (moveDown) {
                            ballY += 5; // Move the ball down
                        } else {
```

```
                    ballY -= 5; // Move the ball up
                }
                // If the ball hits the bottom or top of the panel, change direction
                if (ballY >= panelHeight - ballDiameter) {
                    moveDown = false; // Change direction to upwards
                } else if (ballY <= 0) {
                    moveDown = true; // Change direction to downwards
                }
                // Repaint the panel to reflect the new position of the ball
                panel.repaint();
                // Sleep for a short time to control the speed of the movement
                Thread.sleep(20);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
});
ballThread.start(); // Start the movement thread
}
public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            new BallMovement().setVisible(true); // Create and show the GUI
        }
    });
}
}
```

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”.

→

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.*;
```

```
@SpringBootApplication
@RestController
public class QuoteApp {

   @GetMapping("/")
   public String showMessage() {
      return "If you can't explain it simply, you don't understand it well enough";
   }

   public static void main(String[] args) {
      SpringApplication.run(QuoteApp.class, args);
   }
}
```

Slip 10

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

→

2. Write a Java program to display the first record from the student table (RNo, SName, Per) onto the TextFields by clicking on the button. (Assume the Student table is already created).

→

```
package com.DPU;

import java.awt.GridLayout;
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.ResultSet;
import java.sql.SQLException;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
```

```
import javax.swing.JTextField;

public class Slip10ka2 {

    public static void main(String[] args) {
        // Create the frame for the form
        JFrame frame = new JFrame("Student Record Display");
        frame.setSize(400, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Create a panel to hold components
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(4, 2));

        // Create labels and text fields for student details
        JLabel rnoLabel = new JLabel("Roll No:");
        JLabel snameLabel = new JLabel("Student Name:");
        JLabel perLabel = new JLabel("Percentage:");

        JTextField rnoField = new JTextField();
        JTextField snameField = new JTextField();
        JTextField perField = new JTextField();

        panel.add(rnoLabel);
        panel.add(rnoField);
        panel.add(snameLabel);
        panel.add(snameField);
        panel.add(perLabel);
        panel.add(perField);

        JButton displayButton = new JButton("Display First Record");
        displayButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                displayFirstStudentRecord(rnoField, snameField, perField);
            }
        });

        // Add button to panel
        panel.add(new JLabel()); // Empty label for spacing
        panel.add(displayButton);

        // Add panel to frame
        frame.add(panel);
        frame.setVisible(true);
    }

    public static void displayFirstStudentRecord(JTextField rnoField, JTextField snameField, JTextField perField) {
        Connection conn = null;
```

```
        PreparedStatement pstmt = null;
        ResultSet rs = null;

        try {
            // Establish database connection
            conn = DriverManager.getConnection("jdbc:postgresql://localhost:5432/dpu", "postgres", "postgres");

            // SQL query to fetch first student record
            String query = "SELECT * FROM Student LIMIT 1";
            pstmt = conn.prepareStatement(query);
            rs = pstmt.executeQuery();

            if (rs.next()) {
                int rno = rs.getInt("RNo");
                String sname = rs.getString("SName");
                double per = rs.getDouble("Per");

                // Set values in text fields
                rnoField.setText(String.valueOf(rno));
                snameField.setText(sname);
                perField.setText(String.valueOf(per));
            } else {
                JOptionPane.showMessageDialog(null, "No records found in the Student table.");
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Error fetching record: " + e.getMessage());
            e.printStackTrace();
        } finally {
            try {
                if (rs != null) rs.close();
                if (pstmt != null) pstmt.close();
                if (conn != null) conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}


CREATE TABLE Student (
    RNo INT PRIMARY KEY,
    SName VARCHAR(100) NOT NULL,
    Per DECIMAL(5,2) NOT NULL
);
```

Slip 11

1. Design an HTML page which passes customer numbers 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 displays an error message.
→

customerSearch.html

```
<!DOCTYPE html>
<html>
<head>
   <title>Customer Search</title>
</head>
<body>
   <h2>Search Customer Details</h2>
   <form action="CustomerSearchServlet" method="get">
      <label>Enter Customer Number:</label>
      <input type="number" name="cust_no" required>
      <input type="submit" value="Search">
   </form>
</body>
</html>
```

CustomerSearchServlet.java

```
public class CustomerSearchServlet extends HttpServlet {

        @Override
        protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
                // TODO Auto-generated method stub
                response.setContentType("text/html");
      PrintWriter out = response.getWriter();

      int custNo = Integer.parseInt(request.getParameter("cust_no"));

      try {
         // Database Connection
```

```
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/CustomerDB?useSSL=false&serverTi
mezone=UTC\r\n"
                                , "root", "mysqldb");

            // SQL Query
            String query = "SELECT * FROM customer WHERE cust_no = ?";
            PreparedStatement ps = con.prepareStatement(query);
            ps.setInt(1, custNo);
            ResultSet rs = ps.executeQuery();

            // Display Customer Details if Found
            if (rs.next()) {
                out.println("<h2>Customer Details:</h2>");
                out.println("<p><b>Customer Number:</b> " + rs.getInt("cust_no") + "</p>");
                out.println("<p><b>Name:</b> " + rs.getString("name") + "</p>");
                out.println("<p><b>Address:</b> " + rs.getString("address") + "</p>");
                out.println("<p><b>Phone:</b> " + rs.getString("phone") + "</p>");
            } else {
                out.println("<h2 style='color: red;'>Customer not found!</h2>");
            }

            con.close();
        } catch (Exception e) {
            out.println("<h3 style='color: red;'>Error: " + e.getMessage() + "</h3>");
        }

          }

}
```

---

web.xml

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

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

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

→

```
package com.sqltest;
import java.sql.*;
public class DonarTableMetaData {
   // JDBC URL, username, and password
   static final String DB_URL = "jdbc:postgresql://localhost:5432/dpu";
   static final String USER = "postgres";
   static final String PASS = "postgres";
   public static void main(String[] args) {
      Connection conn = null;
      Statement stmt = null;
      ResultSet rs = null;
      ResultSetMetaData rsMetaData = null;
      try {
         // Establish connection
         conn = DriverManager.getConnection(DB_URL, USER, PASS);
         // Create a statement
         stmt = conn.createStatement();
         // Execute the query to select all data from the DONAR table
         rs = stmt.executeQuery("SELECT * FROM DONAR");
         // Get ResultSetMetaData object
         rsMetaData = rs.getMetaData();
         // Get the number of columns in the DONAR table
         int columnCount = rsMetaData.getColumnCount();
         // Display column information
         System.out.println("Column Information for the DONAR Table:");
         for (int i = 1; i <= columnCount; i++) {
            String columnName = rsMetaData.getColumnName(i);
            int columnType = rsMetaData.getColumnType(i);
```

```
            String columnTypeName = rsMetaData.getColumnTypeName(i);
            int columnSize = rsMetaData.getColumnDisplaySize(i);
            String nullable = rsMetaData.isNullable(i) == ResultSetMetaData.columnNullable ?
"Nullable" : "Not Nullable";
            // Display each column's details
            System.out.println("Column " + i + ":");
            System.out.println("  Name: " + columnName);
            System.out.println("  Type: " + columnTypeName + " (" + columnType + ")");
            System.out.println("  Size: " + columnSize);
            System.out.println("  Nullable: " + nullable);
            System.out.println("---------------------------------");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (rs != null) rs.close();
            if (stmt != null) stmt.close();
            if (conn != null) conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
  }
}
```

Slip 12

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

→

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<!DOCTYPE html><meta charset="ISO-8859-1">
<html>
<head>
   <title>Perfect Number Check</title>
</head>
<body>
   <h2>Check if a Number is Perfect</h2>
   <form method="post">
```

```
    Enter a Number: <input type="text" name="num">
    <input type="submit" value="Check">
  </form>

  <%
    String numStr = request.getParameter("num");
    if (numStr != null) {
        int num = Integer.parseInt(numStr);
        int sum = 0;

        for (int i = 1; i < num; i++) {
            if (num % i == 0) {
                sum += i;
            }
        }

        if (sum == num) {
            out.println("<h3 style='color: green;'>Yes! " + num + " is a Perfect Number.</h3>");
        } else {
            out.println("<h3 style='color: red;'>No! " + num + " is NOT a Perfect Number.</h3>");
        }
    }
  %>
</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).

→

```
package com.sqltest;

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

public class ProjectTableApp {

  // JDBC URL, username, and password
```

```
static final String DB_URL = "jdbc:postgresql://localhost:5432/dpu";
static final String USER = "postgres";
static final String PASS = "postgres";

public static void main(String[] args) {
    // Create the JFrame for the GUI
    JFrame frame = new JFrame("PROJECT Table Viewer");
    frame.setSize(600, 400);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    // Create a panel for buttons and table display
    JPanel panel = new JPanel();
    panel.setLayout(new BorderLayout());

    // Create a button to load data from the PROJECT table
    JButton loadButton = new JButton("Load Project Data");
    panel.add(loadButton, BorderLayout.NORTH);

    // Create JTable to display project records
    String[] columns = {"Project ID", "Project Name", "Project Description", "Project Status"};
    DefaultTableModel tableModel = new DefaultTableModel(columns, 0);
    JTable table = new JTable(tableModel);
    JScrollPane scrollPane = new JScrollPane(table);
    panel.add(scrollPane, BorderLayout.CENTER);

    // Add the panel to the frame
    frame.add(panel);
    frame.setVisible(true);

    // ActionListener for the Load button
    loadButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            // Load the project records when the button is clicked
            loadProjectData(tableModel);
        }
    });

    // Create the PROJECT table and insert sample data (if not already created)
    createAndInsertProjectData();
}

// Method to create the PROJECT table and insert sample data
public static void createAndInsertProjectData() {
    Connection conn = null;
    Statement stmt = null;

    try {
        // Establish connection to the database
```

```
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            stmt = conn.createStatement();

            // SQL query to create the PROJECT table
            String createTableSQL = "CREATE TABLE IF NOT EXISTS PROJECT (" +
                                    "project_id SERIAL PRIMARY KEY, " +
                                    "project_name VARCHAR(100), " +
                                    "project_description TEXT, " +
                                    "project_status VARCHAR(50))";
            stmt.executeUpdate(createTableSQL);

            // Insert sample records into the PROJECT table
            String insertSQL = "INSERT INTO PROJECT (project_name, project_description, project_status) VALUES "
+
                               "('Project A', 'This is the description for Project A', 'Ongoing')," +
                               "('Project B', 'This is the description for Project B', 'Completed')," +
                               "('Project C', 'This is the description for Project C', 'Not Started')";
            stmt.executeUpdate(insertSQL);

            System.out.println("PROJECT table created and sample data inserted successfully!");

        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                if (stmt != null) stmt.close();
                if (conn != null) conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    // Method to load data from the PROJECT table and display it in the JTable
    public static void loadProjectData(DefaultTableModel tableModel) {
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;

        try {
            // Establish connection to the database
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            stmt = conn.createStatement();

            // SQL query to retrieve all records from the PROJECT table
            String selectSQL = "SELECT * FROM PROJECT";
            rs = stmt.executeQuery(selectSQL);

            // Clear existing rows in the table model
```

```
        tableModel.setRowCount(0);

        // Iterate through the result set and add rows to the table model
        while (rs.next()) {
            int projectId = rs.getInt("project_id");
            String projectName = rs.getString("project_name");
            String projectDescription = rs.getString("project_description");
            String projectStatus = rs.getString("project_status");

            // Add row to the table model
            tableModel.addRow(new Object[]{projectId, projectName, projectDescription, projectStatus});
        }

    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (rs != null) rs.close();
            if (stmt != null) stmt.close();
            if (conn != null) conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }  } }
```

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.*;
public class DatabaseInfo {

    public static void main(String[] args) {
        // PostgreSQL database connection details
        String url = "jdbc:postgresql://localhost:5432/dpu";  // Your database name (dpu)
        String user = "postgres"; // Your PostgreSQL username
        String password = "postgres"; // Your PostgreSQL password

        Connection conn = null;
        DatabaseMetaData metaData = null;

        try {
```

```
            // Establish the database connection
            conn = DriverManager.getConnection(url, user, password);

            // Get DatabaseMetaData object
            metaData = conn.getMetaData();

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

            // Get the list of all tables in the database
            System.out.println("\nList of Tables:");
            ResultSet rs = metaData.getTables(null, null, "%", new String[] {"TABLE"});
            while (rs.next()) {
                String tableName = rs.getString("TABLE_NAME");
                System.out.println(tableName);
            }

            rs.close();  // Close the result set

        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                if (conn != null) {
                    conn.close();  // Close the connection
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

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

→

```
package com.dpu;
import java.util.Random;
```

```
class MyThread extends Thread {
    private Random rand = new Random(); // Random number generator

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

    @Override
    public void run() {
        System.out.println(getName() + " is Created and Running");

        int sleepTime = rand.nextInt(5000); // Random sleep time (0-4999 ms)
        System.out.println(getName() + " will sleep for " + sleepTime + " ms");

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

        System.out.println(getName() + " is now Dead");
    }
}

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

        MyThread t1 = new MyThread("Thread-1");
        MyThread t2 = new MyThread("Thread-2");
        MyThread t3 = new MyThread("Thread-3");

        // Starting the threads
        t1.start();
        t2.start();
        t3.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.*;
import java.nio.file.*;
import java.util.concurrent.*;

public class SimpleSearchEngine {

    public static void main(String[] args) throws InterruptedException, ExecutionException {
        // Accept the search string from the user
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Enter the string to search: ");
        String searchString = reader.readLine();

        // Get the current directory
        File currentDir = new File(System.getProperty("user.dir"));
        File[] textFiles = currentDir.listFiles((dir, name) -> name.toLowerCase().endsWith(".txt"));

        if (textFiles == null || textFiles.length == 0) {
            System.out.println("No text files found in the current directory.");
            return;
        }

        // Use ExecutorService to manage threads
        ExecutorService executorService = Executors.newFixedThreadPool(textFiles.length);

        // List to hold the future results
        List<Future<Void>> futures = new ArrayList<>();

        for (File file : textFiles) {
            // Submit a search task for each file
            futures.add(executorService.submit(new SearchTask(file, searchString)));
        }

        // Wait for all tasks to complete
        for (Future<Void> future : futures) {
            future.get();  // Wait for completion of each thread
        }

        // Shutdown the executor service
```

```
        executorService.shutdown();
    }
}

class SearchTask implements Callable<Void> {
    private File file;
    private String searchString;

    // Constructor to accept file and search string
    public SearchTask(File file, String searchString) {
        this.file = file;
        this.searchString = searchString;
    }

    @Override
    public Void call() {
        try (BufferedReader br = new BufferedReader(new FileReader(file))) {
            String line;
            int lineNumber = 1;
            boolean found = false;
            while ((line = br.readLine()) != null) {
                if (line.contains(searchString)) {
                    if (!found) {
                        System.out.println("Found in file: " + file.getName());
                        found = true;
                    }
                    System.out.println("Line " + lineNumber + ": " + line);
                }
                lineNumber++;
            }
            if (!found) {
                System.out.println("No match found in file: " + file.getName());
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

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

→

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
   <title>Sum of First and Last Digit</title>
</head>
<body>
   <h2>Calculate Sum of First and Last Digit</h2>
   <form method="post">
      Enter a Number: <input type="text" name="num">
      <input type="submit" value="Calculate">
   </form>

   <%
      String numStr = request.getParameter("num");
      if (numStr != null) {
         int num = Integer.parseInt(numStr);
         int lastDigit = num % 10;

         int firstDigit = num;
         while (firstDigit >= 10) {
            firstDigit /= 10;
         }

         int sum = firstDigit + lastDigit;
   %>
         <h3 style="color: red; font-size: 18px;">
            Sum of First and Last Digit of <%= num %> is: <%= sum %>
         </h3>
   <% } %>
</body>
</html>
```

Slip 15

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

→

```
public class ThreadInfoExample {
  public static void main(String[] args) {
     // Create a new thread using a lambda expression
     Thread myThread = new Thread(() -> {
        // Inside the thread, display the name and priority of the current thread
        System.out.println("Thread Name: " + Thread.currentThread().getName());
        System.out.println("Thread Priority: " + Thread.currentThread().getPriority());
     });
     // Set a custom name and priority for the thread
     myThread.setName("MyCustomThread");
     myThread.setPriority(Thread.MAX_PRIORITY); // Highest priority (10)
     // Display the main thread's name and priority
     System.out.println("Main Thread Name: " + Thread.currentThread().getName());
     System.out.println("Main Thread Priority: " + Thread.currentThread().getPriority());
     // Start the custom thread
     myThread.start();
  }
}
```

2. Write a SERVLET program which counts how many times a user has visited a web page. If a 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)

→

```
public class VisitCounterServlet extends HttpServlet  {

                private static final long serialVersionUID = 1L;

        @Override
        protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
                // TODO Auto-generated method stub
                response.setContentType("text/html");
      PrintWriter out = response.getWriter();

      int visitCount = 0;
      boolean newUser = true;
```

```
        // Get all cookies from the request
        Cookie[] cookies = request.getCookies();

        if (cookies != null) {
            for (Cookie cookie : cookies) {
                if (cookie.getName().equals("visitCount")) {
                    visitCount = Integer.parseInt(cookie.getValue());
                    newUser = false;
                    break;
                }
            }
        }

        visitCount++; // Increment visit count

        // Set the updated visit count in a new cookie
        Cookie visitCookie = new Cookie("visitCount", String.valueOf(visitCount));
        visitCookie.setMaxAge(60 * 60 * 24 * 7); // Cookie valid for 7 days
        response.addCookie(visitCookie);

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

}
```

web.xml

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

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.*;

public class ColorTreeSet {
 public static void main(String[] args) {
 // Create a TreeSet to store color names (Strings)
 Set<String> colorSet = new TreeSet<>();

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

 // Print the content of the 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 the Teacher Table and display the details of the Teacher who is teaching “JAVA” Subject. (Use PreparedStatement Interface)

→

```
package com.DPU;
import java.sql.*;
import java.util.*;
public class Slip16Ka2 {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		Scanner scanner=new Scanner(System.in);

		for(int i=0;i<6;i++) {
			System.out.println("\n Enter details of teacher"+i+":");

			System.out.println("Enter the teacher no :");
			int tno=scanner.nextInt();
			scanner.nextLine();
			System.out.println("Enter the name");
			String name=scanner.nextLine();

			System.out.println("enter the subject");
			String subject=scanner.nextLine();

			// to save in database;

			saveTeacherToDatabase(tno,name,subject);


			}

		//display teacher teaching subject=jave

		displayJavaTeacher();
		scanner.close();
	}

	public static void saveTeacherToDatabase(int tno,String name, String subject) {

		Connection conn=null;
		PreparedStatement pstmt=null;

		try {
```

```
conn=DriverManager.getConnection("jdbc:postgresql://localhost:5432/dpu","postgres","12345");
                        String sql="insert into teacher values(?,?,?)";

                        pstmt=conn.prepareStatement(sql);
                        pstmt.setInt(1, tno);
                        pstmt.setString(2, name);
                        pstmt.setString(3, subject);

                        pstmt.executeUpdate();
                        System.out.println("teacher details entered successfully");
                }
                catch(SQLException e){
                        System.out.println("errors saving teacher details:"+e.getMessage());
                        e.printStackTrace();
                }
        }

        public static void displayJavaTeacher() {
                try {
                        Connection
conn=DriverManager.getConnection("jdbc:postgresql://localhost:5432/dpu","postgres","12345");
                        String sql="select * from teacher where subject=?";
                        PreparedStatement pstmt=conn.prepareStatement(sql);

                        pstmt.setString(1,"Java");
                        ResultSet rs=pstmt.executeQuery();

                        System.out.println("\n teacher teaching java");
                        while(rs.next()) {
                                int tno=rs.getInt("tno");
                                String name=rs.getString("name");
                                String subject=rs.getString("subject");
                                System.out.println("tno :"+tno+", Name :"+name+", subject
:"+subject);
                        }
                }
                catch(SQLException e) {
                        System.out.println("error fetching java teacher :"+e.getMessage());
                        e.printStackTrace();
                }
        }

}
```

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.*;

public class SortedUniqueIntegers {
 public static void main(String[] args) {
 // Create a TreeSet to store integers (automatically sorted and no duplicates)
 Set<Integer> numberSet = new TreeSet<>();

 // Scanner to read user input
 Scanner scanner = new Scanner(System.in);

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

 // Input n integers and store them in the TreeSet
 System.out.println("Enter the integers:");
 for (int i = 0; i < n; i++) {
 int num = scanner.nextInt();
 numberSet.add(num); // TreeSet automatically handles duplicates
 }

 // Display the integers in sorted order
 System.out.println("\nSorted integers without duplicates:");
 for (Integer number : numberSet) {
 System.out.println(number);
 }
```

```
// Search for a particular element
System.out.print("\nEnter the integer to search for: ");
int searchNum = scanner.nextInt();
if (numberSet.contains(searchNum)) {
System.out.println(searchNum + " is present in the collection.");
} else {
System.out.println(searchNum + " is not present in the collection.");
}

// Close the scanner
scanner.close();
}
}
```

2. Write a Multithreading program in java to display the numbers between 1 to 100 continuously in a TextField by clicking on the button. (Use Runnable Interface).

→

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class NumberDisplayApp extends JFrame {
  private JTextField textField;  // TextField to display numbers
  private JButton startButton;   // Button to start displaying numbers
  public NumberDisplayApp() {
     // Set up JFrame
     setTitle("Display Numbers");
     setSize(300, 150);
     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     setLocationRelativeTo(null);
     // Create components
     textField = new JTextField();
     textField.setEditable(false); // Make the text field non-editable
     startButton = new JButton("Start");
     // Layout to arrange components
     setLayout(new FlowLayout());
     add(textField);
     add(startButton);
     // Action for the button
     startButton.addActionListener(new ActionListener() {
        @Override
```

```
        public void actionPerformed(ActionEvent e) {
            // Start a new thread to display numbers
            new Thread(new NumberDisplayRunnable()).start();
        }
    });
  }
  // Runnable class to display numbers from 1 to 100
  private class NumberDisplayRunnable implements Runnable {
    @Override
    public void run() {
        for (int i = 1; i <= 100; i++) {
            textField.setText(String.valueOf(i)); // Update the text field with the number
            try {
                Thread.sleep(100); // Pause for 100 milliseconds
            } catch (InterruptedException e) {
                e.printStackTrace(); // Handle the error if the thread is interrupted
            }
        }
    }
  }
  public static void main(String[] args) {
    // Show the GUI
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            new NumberDisplayApp().setVisible(true);
        }
    });
  }
}
```

Slip 18

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

→

```
public class ThreadInfoExample {
  public static void main(String[] args) {
    // Create a new thread using a lambda expression
    Thread myThread = new Thread(() -> {
        // Inside the thread, display the name and priority of the current thread
        System.out.println("Thread Name: " + Thread.currentThread().getName());
        System.out.println("Thread Priority: " + Thread.currentThread().getPriority());
    });
```

```
        // Set a custom name and priority for the thread
        myThread.setName("MyCustomThread");
        myThread.setPriority(Thread.MAX_PRIORITY); // Highest priority (10)
        // Display the main thread's name and priority
        System.out.println("Main Thread Name: " + Thread.currentThread().getName());
        System.out.println("Main Thread Priority: " + Thread.currentThread().getPriority());
        // Start the custom thread
        myThread.start();
    }
}
```

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

→

StudentDetails.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Student Details</title>
</head>
<body>
    <h2>Enter Student Details</h2>
    <form action="StudentServlet" method="post">
        Seat No: <input type="text" name="seatNo" required><br><br>
        Name: <input type="text" name="studName" required><br><br>
        Class: <input type="text" name="studClass" required><br><br>
        Total Marks: <input type="number" name="totalMarks" required><br><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

StudentServlet.java

```
public class StudentServlet extends HttpServlet {

        @Override
        protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
                // TODO Auto-generated method stub
```

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

        // Retrieve form data
        String seatNo = request.getParameter("seatNo");
        String studName = request.getParameter("studName");
        String studClass = request.getParameter("studClass");
        int totalMarks = Integer.parseInt(request.getParameter("totalMarks"));
        int maxMarks = 500;  // Assuming total marks are out of 500

        // Calculate percentage
        double percentage = (totalMarks * 100.0) / maxMarks;

        // Determine grade
        String grade;
        if (percentage >= 90) {
            grade = "A+";
        } else if (percentage >= 80) {
            grade = "A";
        } else if (percentage >= 70) {
            grade = "B+";
        } else if (percentage >= 60) {
            grade = "B";
        } else if (percentage >= 50) {
            grade = "C";
        } else {
            grade = "Fail";
        }

        // Display student details
        out.println("<h2>Student Result</h2>");
        out.println("<table border='1' cellpadding='5'>");
        out.println("<tr><th>Seat No</th><td>" + seatNo + "</td></tr>");
        out.println("<tr><th>Name</th><td>" + studName + "</td></tr>");
        out.println("<tr><th>Class</th><td>" + studClass + "</td></tr>");
        out.println("<tr><th>Total Marks</th><td>" + totalMarks + "</td></tr>");
        out.println("<tr><th>Percentage</th><td>" + String.format("%.2f", percentage) +
"%</td></tr>");
        out.println("<tr><th>Grade</th><td><b>" + grade + "</b></td></tr>");
        out.println("</table>");
```

```
        }

}
```

web.xml

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

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.*;

public class NegativeIntegersLinkedList {
 public static void main(String[] args) {
 // Create a LinkedList to store integers
 LinkedList<Integer> integerList = new LinkedList<>();

 // Scanner to read user input
 Scanner scanner = new Scanner(System.in);

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

 // Input 'N' integers and store them in the LinkedList
 System.out.println("Enter the integers:");
 for (int i = 0; i < n; i++) {
```

```
int num = scanner.nextInt();
integerList.add(num); // Add the integer to the LinkedList
}

// Display only negative integers
System.out.println("\nNegative integers are:");
boolean foundNegative = false;
for (Integer number : integerList) {
if (number < 0) {
System.out.println(number);
foundNegative = true;
}
}

// If no negative integers are found
if (!foundNegative) {
System.out.println("No negative integers found.");
}

// Close the scanner
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.

→

Login.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Login Page</title>
</head>
<body>
    <h2>User Login</h2>
    <form action="LoginServlet" method="post">
```

```
        Username: <input type="text" name="username" required><br><br>
        Password: <input type="password" name="password" required><br><br>
        <input type="submit" value="Login">
    </form>
</body>
</html>
```

LoginServlet.java

```
public class LoginServlet extends HttpServlet{

        @Override
        protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
                // TODO Auto-generated method stub
                response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        // Retrieve user inputs
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        try {
            // Load JDBC Driver
           //Class.forName("com.mysql.cj.jdbc.Driver");
Class.forName(“org.postgresql.Driver”);
          Connection conn = DriverManager.getConnection("jdbc:postgresql://localhost:5432/dpu",
"postgres", "12345");
            // Prepare SQL Query
            String sql = "SELECT * FROM users WHERE username=? AND password=?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setString(1, username);
            stmt.setString(2, password);

            ResultSet rs = stmt.executeQuery();

            if (rs.next()) {
                out.println("<h2>Welcome, " + username + "! Login Successful.</h2>");
            } else {
                out.println("<h2 style='color:red;'>Invalid Username or Password.</h2>");
            }
```

```
            // Close resources
            rs.close();
            stmt.close();
            conn.close();

        } catch (Exception e) {
            out.println("<h2 style='color:red;'>Database Connection Failed: " + e.getMessage() +
"</h2>");
        }

          }
}
```

web.xml

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

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

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.

→

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

→

```
package com.dpu;

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

```
public class ImageBlinkingExample extends JFrame {
  /**
          *
          */
         private static final long serialVersionUID = 1L;
private JLabel label;
  private ImageIcon image;
  private boolean isVisible;
  public ImageBlinkingExample() {
     // Set JFrame properties
     setTitle("Image Blinking Example");
     setSize(400, 400);
     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     setLocationRelativeTo(null);
     // Load the image and set it on a JLabel
     image = new ImageIcon("select photo go on properties and select locatiom -->shift windows
and S"); // Use the path to your image
     label = new JLabel(image);
     add(label);
     // Start the thread to blink the image
     startBlinking();
  }
  // Method to start the blinking effect using Thread
  private void startBlinking() {
     isVisible = true;  // To track the visibility of the image
     Thread blinkingThread = new Thread(() -> {
        while (true) {
           try {
              // Toggle the image visibility
              if (isVisible) {
                 label.setVisible(false);
              } else {
                 label.setVisible(true);
              }
              // Toggle the visibility state
              isVisible = !isVisible;
              // Wait for 500 milliseconds before the next toggle
              Thread.sleep(500);
           } catch (InterruptedException e) {
              e.printStackTrace();
           }
        }
     });
     blinkingThread.start();  // Start the thread
```

```
  }
  public static void main(String[] args) {
      // Create and show the JFrame
      SwingUtilities.invokeLater(() -> {
          new ImageBlinkingExample().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.*;
public class SubjectNamesLinkedList {
 public static void main(String[] args) {
 // Create a LinkedList to store subject names
 LinkedList<String> subjectList = new LinkedList<>();
 // Scanner to read user input
 Scanner scanner = new Scanner(System.in);
 // Accept 'N' subject names from the user
 System.out.print("Enter the number of subjects you want to input: ");
 int n = scanner.nextInt();
 scanner.nextLine(); // Consume the newline character after integer input
 // Input 'N' subject names and store them in the LinkedList
 System.out.println("Enter the subject names:");
 for (int i = 0; i < n; i++) {
 String subject = scanner.nextLine();
 subjectList.add(subject); // Add the subject name to the LinkedList
 }
 // Create an Iterator to traverse through the LinkedList
 Iterator<String> iterator = subjectList.iterator();
 // Display the subject names using the Iterator interface
 System.out.println("\nSubjects in the list:");
 while (iterator.hasNext()) {
 System.out.println(iterator.next());
 }
 // Close the scanner
 scanner.close();
 }
}
```

2. Write a java program to solve producer consumer problems in which a producer produces a value and consumer consumes the value before producer generates the next value. (Hint: use thread synchronization)

→

```
package com.dpu;
class Shared {
    private int data;
    private boolean available = false;

    public synchronized void produce(int value) {
        while (available) {
            try { wait(); } catch (InterruptedException e) { e.printStackTrace(); }
        }
        data = value;
        available = true;
        System.out.println("Produced: " + value);
        notify();
    }

    public synchronized void consume() {
        while (!available) {
            try { wait(); } catch (InterruptedException e) { e.printStackTrace(); }
        }
        System.out.println("Consumed: " + data);
        available = false;
        notify();
    }
}

class ProducerThread extends Thread {
    private Shared shared;
    private int n;

    public ProducerThread(Shared shared, int n) {
        this.shared = shared;
        this.n = n;
    }

    @Override
    public void run() {
```

```
        for (int i = 1; i <= n; i++) {
            shared.produce(i);
            try { Thread.sleep(500); } catch (InterruptedException e) { e.printStackTrace(); }
        }
    }
}

class ConsumerThread extends Thread {
    private Shared shared;
    private int n;

    public ConsumerThread(Shared shared, int n) {
        this.shared = shared;
        this.n = n;
    }

    @Override
    public void run() {
        for (int i = 1; i <= n; i++) {
            shared.consume();
            try { Thread.sleep(500); } catch (InterruptedException e) { e.printStackTrace(); }
        }
    }
}

public class ProducerConsumerThreads {
    public static void main(String[] args) {
        Shared shared = new Shared();
        Thread producer = new ProducerThread(shared, 5);
        Thread consumer = new ConsumerThread(shared, 5);

        producer.start();
        consumer.start();
    }
}
```

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

public class EmployeeMenuApp {

    // JDBC URL, username, and password
    static final String DB_URL = "jdbc:postgresql://localhost:5432/dpu";
    static final String USER = "postgres";
    static final String PASS = "postgres";

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

        do {
            // Display menu options
            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 statement to handle the menu options
        switch (choice) {
            case 1:
                insertEmployee(scanner);
                break;
            case 2:
                updateEmployee(scanner);
                break;
            case 3:
                displayEmployees();
                break;
            case 4:
                System.out.println("Exiting...");
                break;
            default:
                System.out.println("Invalid choice! Please try again.");
        }
    } while (choice != 4);

    scanner.close();
}

// Method to insert an employee record into the Employee table
public static void insertEmployee(Scanner scanner) {
    System.out.print("Enter Employee Number (ENo): ");
    int eno = scanner.nextInt();
    scanner.nextLine();  // Consume newline left by nextInt
    System.out.print("Enter Employee Name (EName): ");
    String ename = scanner.nextLine();
    System.out.print("Enter Salary: ");
    double salary = scanner.nextDouble();

    Connection conn = null;
    PreparedStatement stmt = null;

    try {
        conn = DriverManager.getConnection(DB_URL, USER, PASS);

        // SQL query to insert data
        String insertSQL = "INSERT INTO Employee (ENo, EName, Salary) VALUES (?, ?, ?)";
        stmt = conn.prepareStatement(insertSQL);

        // Set parameters
```

```
            stmt.setInt(1, eno);
            stmt.setString(2, ename);
            stmt.setDouble(3, salary);

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

            if (rowsAffected > 0) {
                System.out.println("Employee record inserted successfully!");
            } else {
                System.out.println("Failed to insert employee record.");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                if (stmt != null) stmt.close();
                if (conn != null) conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    // Method to update an employee's salary in the Employee table
    public static void updateEmployee(Scanner scanner) {
        System.out.print("Enter Employee Number (ENo) to update: ");
        int eno = scanner.nextInt();
        System.out.print("Enter new Salary: ");
        double newSalary = scanner.nextDouble();

        Connection conn = null;
        PreparedStatement stmt = null;

        try {
            conn = DriverManager.getConnection(DB_URL, USER, PASS);

            // SQL query to update employee salary
            String updateSQL = "UPDATE Employee SET Salary = ? WHERE ENo = ?";
            stmt = conn.prepareStatement(updateSQL);

            // Set parameters
            stmt.setDouble(1, newSalary);
            stmt.setInt(2, eno);
```

```
        // Execute the update query
        int rowsAffected = stmt.executeUpdate();

        if (rowsAffected > 0) {
            System.out.println("Employee salary updated successfully!");
        } else {
            System.out.println("No employee found with the given ENo.");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (stmt != null) stmt.close();
            if (conn != null) conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

// Method to display all employee records from the Employee table
public static void displayEmployees() {
    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;

    try {
        conn = DriverManager.getConnection(DB_URL, USER, PASS);
        stmt = conn.createStatement();

        // SQL query to fetch all employee records
        String selectSQL = "SELECT * FROM Employee";
        rs = stmt.executeQuery(selectSQL);

        // Display the results in tabular format
        System.out.println("Employee Records:");
        System.out.println("ENo\tEName\t\tSalary");
        System.out.println("---------------------------------");

        while (rs.next()) {
            int eno = rs.getInt("ENo");
            String ename = rs.getString("EName");
            double salary = rs.getDouble("Salary");
```

```
            System.out.println(eno + "\t" + ename + "\t\t" + salary);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (rs != null) rs.close();
            if (stmt != null) stmt.close();
            if (conn != null) conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
  }
}
```

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

→

greet.jsp

```
<%@ page import="java.util.Calendar" %>
<!DOCTYPE html>
<html>
<head>
  <title>Greeting Page</title>
</head>
<body>
  <h2>
    <%
      // Retrieve user input
      String userName = request.getParameter("userName");

      // Get the server time
      Calendar calendar = Calendar.getInstance();
      int hour = calendar.get(Calendar.HOUR_OF_DAY);
      String greeting;

      // Determine greeting based on server time
      if (hour >= 5 && hour < 12) {
```

```
            greeting = "Good Morning";
        } else if (hour >= 12 && hour < 17) {
            greeting = "Good Afternoon";
        } else if (hour >= 17 && hour < 21) {
            greeting = "Good Evening";
        } else {
            greeting = "Good Night";
        }
    %>

    <%= greeting %>, <%= userName %>! Welcome to our website it is.<%=hour %>
  </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.

→

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.*;

public class StudentNames {
    public static void main(String[] args) {
        // If no command-line arguments are passed, ask the user to input the names
        List<String> students = new ArrayList<>();

        if (args.length == 0) {
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter the number of students: ");
            int n = scanner.nextInt();
            scanner.nextLine();  // Consume the newline character
```

```
            System.out.println("Enter the names of students:");
            for (int i = 0; i < n; i++) {
                students.add(scanner.nextLine());
            }
            scanner.close();
        } else {
            students = new ArrayList<>(Arrays.asList(args));
        }

        // Using Iterator (Forward Traversal)
        System.out.println("Student Names (Forward):");
        Iterator<String> itr = students.iterator();
        while (itr.hasNext()) {
            System.out.println(itr.next());
        }

        // Using ListIterator (Backward Traversal)
        System.out.println("Student Names (Backward):");
        ListIterator<String> listItr = students.listIterator(students.size());
        while (listItr.hasPrevious()) {
            System.out.println(listItr.previous());
        }
    }
}
```

Slip 24

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

→

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

→

Login.html

```
<!DOCTYPE html>
<html>
<head>
```

```
    <title>Login Page</title>
</head>
<body>
    <h2>User Login</h2>
    <form action="LoginServlet" method="post">
        Username: <input type="text" name="username" required><br><br>
        Password: <input type="password" name="password" required><br><br>
        <input type="submit" value="Login">
    </form>
</body>
</html>
```

LoginServlet.java

```
public class LoginServlet extends HttpServlet{

        @Override
        protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
                // TODO Auto-generated method stub
                response.setContentType("text/html");
      PrintWriter out = response.getWriter();

      // Retrieve user inputs
      String username = request.getParameter("username");
      String password = request.getParameter("password");

      try {
         // Load JDBC Driver
         Class.forName("com.mysql.cj.jdbc.Driver");
          Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/CustomerDB?useSSL=false&serverTi
mezone=UTC\r\n"
                        , "root", "mysqldb");

         // Prepare SQL Query
         String sql = "SELECT * FROM users WHERE username=? AND password=?";
         PreparedStatement stmt = conn.prepareStatement(sql);
         stmt.setString(1, username);
         stmt.setString(2, password);

         ResultSet rs = stmt.executeQuery();
```

```
            if (rs.next()) {
                out.println("<h2>Welcome, " + username + "! Login Successful.</h2>");
            } else {
                out.println("<h2 style='color:red;'>Invalid Username or Password.</h2>");
            }

            // Close resources
            rs.close();
            stmt.close();
            conn.close();

        } catch (Exception e) {
            out.println("<h2 style='color:red;'>Database Connection Failed: " + e.getMessage() +
"</h2>");
        }

          }
}
```

web.xml

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

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

Slip 25

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

→

2. Write a Java Program for the following: Assume database is already created.

Slip 26

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

```
package com.sqltest;

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

public class EmployeeForm {

    // JDBC URL, username, and password
    static final String DB_URL = "jdbc:postgresql://localhost:5432/dpu";
    static final String USER = "postgres";
    static final String PASS = "postgres";

    public static void main(String[] args) {
        if (args.length == 0) {
            System.out.println("Please provide the Employee ID (ENO) as a command-line
argument.");
            return;
        }

        // Get the Employee ID (ENO) from the command line
        int eno = Integer.parseInt(args[0]);

        // Delete employee details from the database
        deleteEmployeeFromDatabase(eno);
    }
```

```
    // Method to delete employee details from the database
    public static void deleteEmployeeFromDatabase(int eno) {
        Connection conn = null;
        PreparedStatement pstmt = null;

        try {
            // Establish connection to the database
            conn = DriverManager.getConnection(DB_URL, USER, PASS);

            // SQL query to delete employee data from the Employee table
            String sql = "DELETE FROM Employee WHERE Eno = ?";

            // Prepare the statement
            pstmt = conn.prepareStatement(sql);
            pstmt.setInt(1, eno);

            // Execute the update (delete operation)
            int rowsAffected = pstmt.executeUpdate();

            if (rowsAffected > 0) {
                // Show a message on successful deletion
                JOptionPane.showMessageDialog(null, "Employee details deleted successfully!");
            } else {
                // Show a message if no employee was found with the given ID
                JOptionPane.showMessageDialog(null, "Employee not found with ENo: " + eno);
            }

        } catch (SQLException e) {
            // Handle database errors
            JOptionPane.showMessageDialog(null, "Error deleting employee details: " +
e.getMessage());
            e.printStackTrace();
        } finally {
            try {
                if (pstmt != null) {
                    pstmt.close();
                }
                if (conn != null) {
                    conn.close();
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
```

```
    }
}
```

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

→

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
    <title>Sum of First and Last Digit</title>
</head>
<body>
    <h2>Calculate Sum of First and Last Digit</h2>
    <form method="post">
        Enter a Number: <input type="text" name="num">
        <input type="submit" value="Calculate">
    </form>

    <%
        String numStr = request.getParameter("num");
        if (numStr != null) {
            int num = Integer.parseInt(numStr);
            int lastDigit = num % 10;

            int firstDigit = num;
            while (firstDigit >= 10) {
                firstDigit /= 10;
            }

            int sum = firstDigit + lastDigit;
    %>
            <h3 style="color: red; font-size: 18px;">
                Sum of First and Last Digit of <%= num %> is: <%= sum %>
            </h3>
    <% } %>
</body>
</html>
```

Slip 27

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

→

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

Slip 28

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

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

Slip 29

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

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

Slip 30

1. Write a java program for the implementation of synchronization. [15 M] 2. Write a Java Program for the implementation of scrollable ResultSet. Assume Teacher table with attributes (TID, TName, Salary) is already created.