HNDIT4232 - Enterprise Architecture

Sri Lanka Institute of Advanced Technological Education

(SLIATE)

Department of Information Technology

GAM/IT/2022/F/0033- S.N.U.Sarathchandra

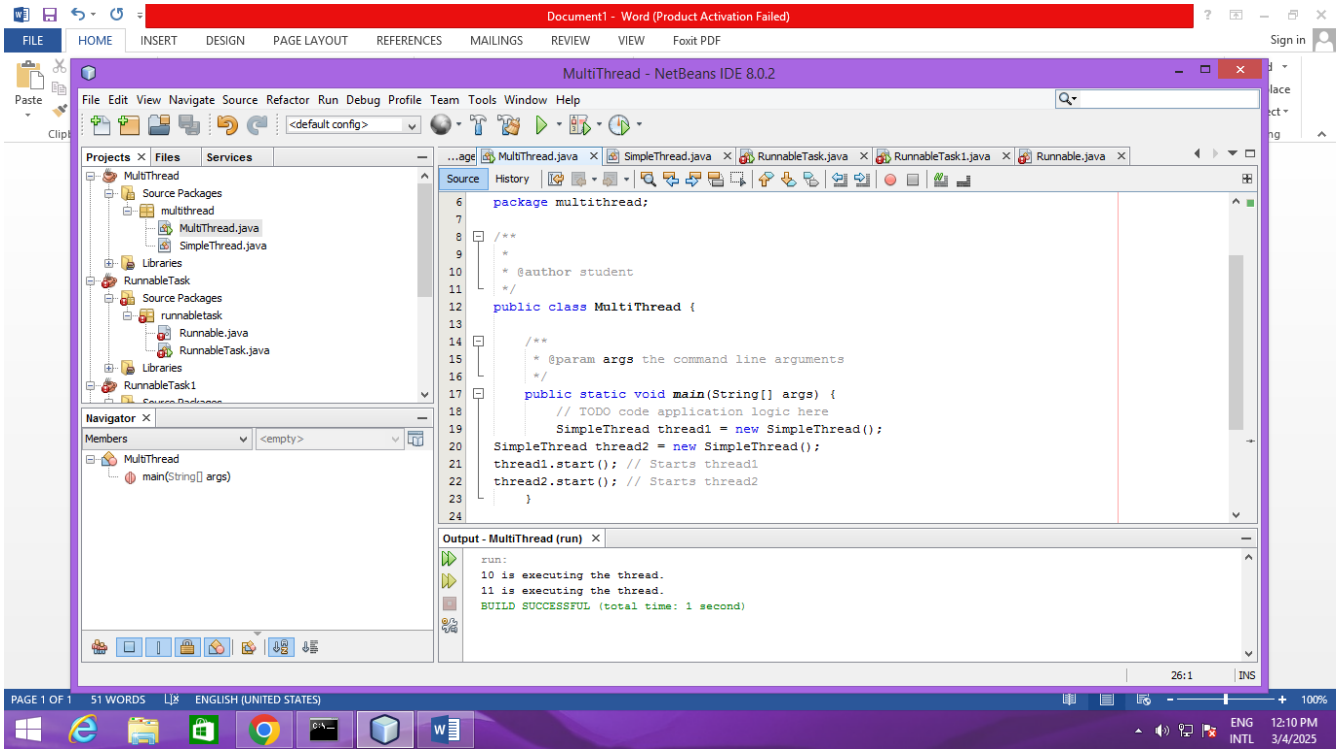
Submitted To: Ms. M.V.M. Jayathilaka

Java Thread

LAB2-TASK 01

```
public class SimpleThread extends Thread {


public void run() {
System.out.println(Thread.currentThread().getId() + " is executing the thread.");
}
}
public static void main(String[] args) {
SimpleThread thread1 = new SimpleThread();
SimpleThread thread2 = new SimpleThread();
thread1.start();
thread2.start();
  }
  }
```

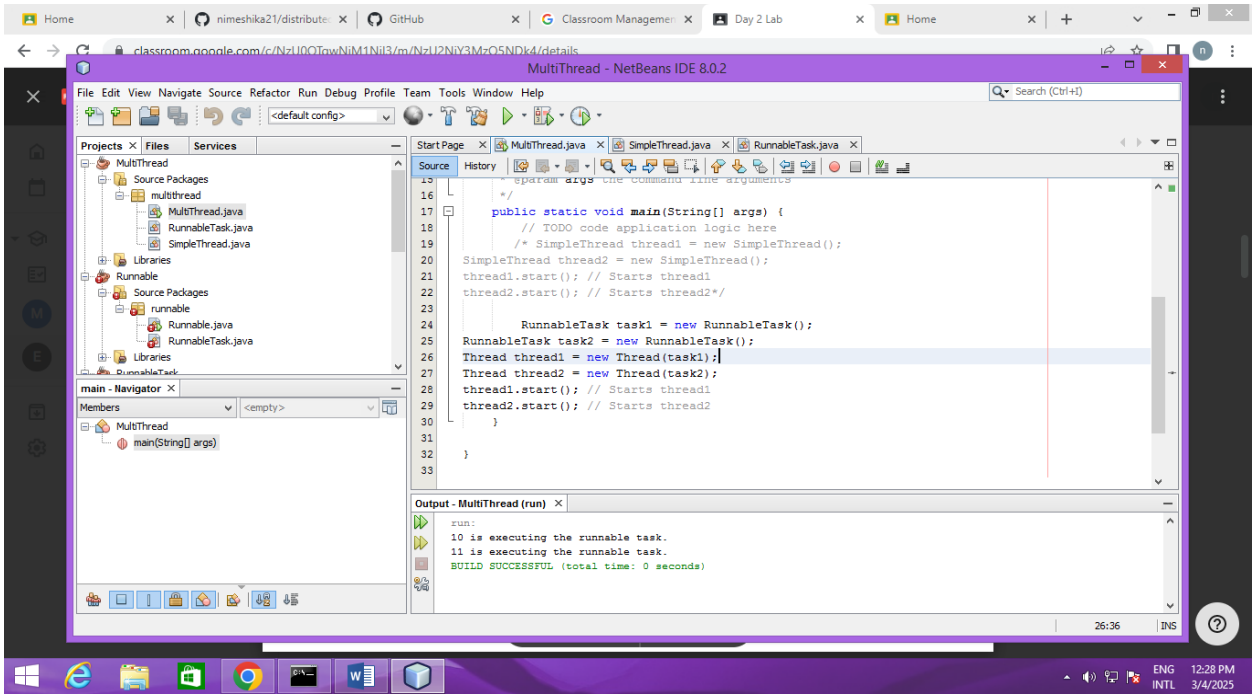
OUTPUT



2) Lab02- TASK 02 - creating runnable class

```
public class RunnableTask implements Runnable {
public void run() {
System.out.println(Thread.currentThread().getId() + " is executing the runnable task.");
  }
  }
public static void main(String[] args) {
RunnableTask task1 = new RunnableTask();
RunnableTask task2 = new RunnableTask();
Thread thread1 = new Thread(task1);
Thread thread2 = new Thread(task2);
thread1.start();
thread2.start();
  }
  }
```

OUTPUT

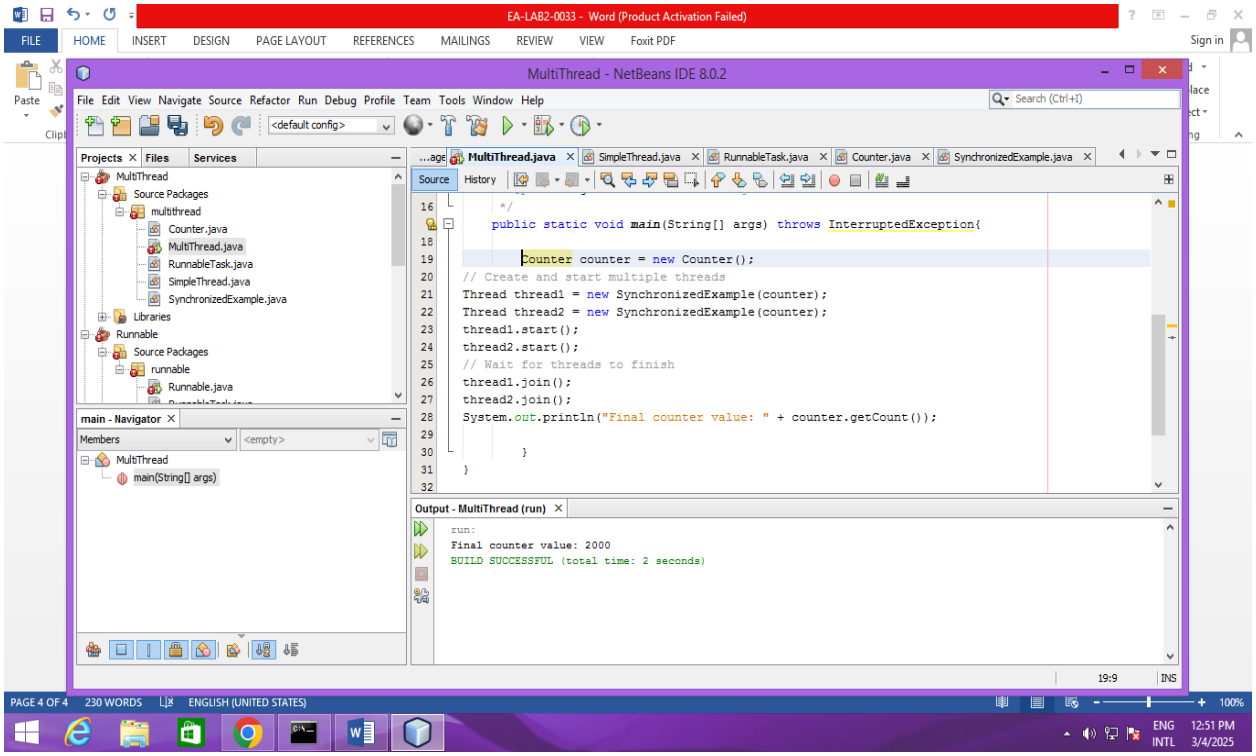


3) LAB03-TASK-03- synchronizing shared resources

```
public class Counter {
privateint count = 0;
// Synchronized method to ensure thread-safe access to the counter
public synchronized void increment() {
count++;
}
publicintgetCount() {
return count;
  }
public class SynchronizedExample extends Thread{
private Counter counter;
publicSynchronizedExample(Counter counter) {
this.counter = counter;  }
@Override
public void run() {
for (inti = 0; i< 1000; i++) {
counter.increment();
}}}
public static void main(String[] args) throws InterruptedException{
Counter counter = new Counter;
Thread thread1 = new SynchronizedExample(counter);
Thread thread2 = new SynchronizedExample(counter);
thread1.start();
thread2.start();

thread1.join();
thread2.join();
```

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

OUTPUT

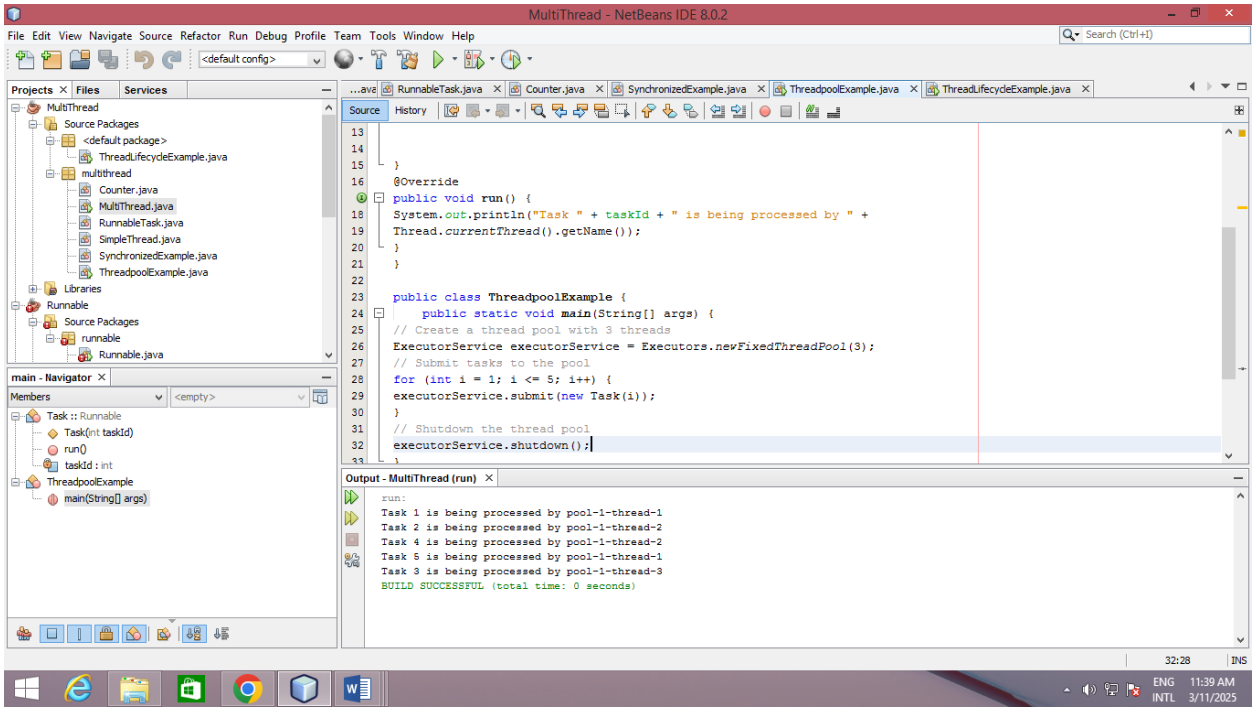


4) Lab02-task-04- Using Executor Service for Thread Pooling

```
package multithread;

importjava.util.concurrent.ExecutorService;

importjava.util.concurrent.Executors;


class Task implements Runnable {

privateinttaskId;

public Task(inttaskId) {

this.taskId = taskId;

}

@Override
```

```
public void run() {
System.out.println("Task " + taskId + " is being processed by " +
Thread.currentThread().getName());
}
}

public class ThreadpoolExample {
public static void main(String[] args) {
// Create a thread pool with 3 threads
ExecutorServiceexecutorService = Executors.newFixedThreadPool(3);
// Submit tasks to the pool
for (inti = 1; i<= 5; i++) {
executorService.submit(new Task(i));
}
// Shutdown the thread pool
executorService.shutdown();
}
}
```

OUTPUT

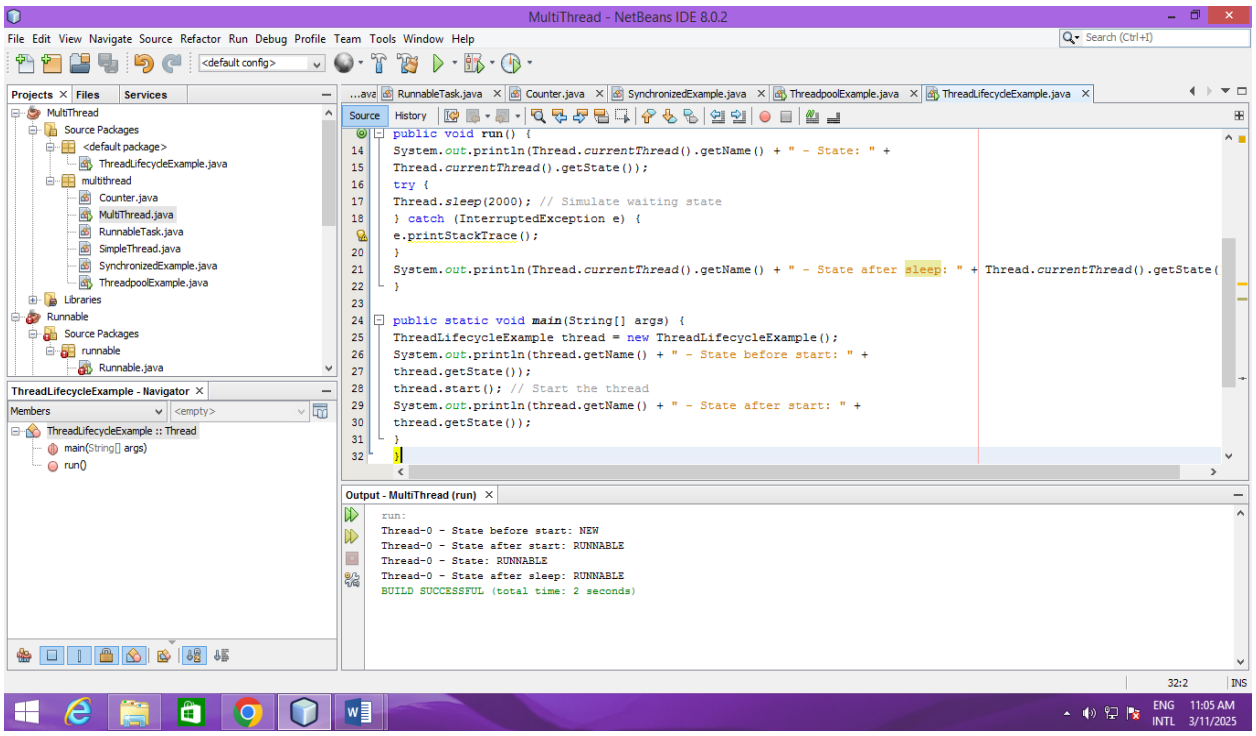


5) Lab02- TASK 05- Thread Lifecycle Example

```
public class ThreadLifecycleExample extends Thread{
   @Override
public void run() {
System.out.println(Thread.currentThread().getName() + " - State: " +
Thread.currentThread().getState());
try {
Thread.sleep(2000);
} catch (InterruptedException e) {
e.printStackTrace();
}
System.out.println(Thread.currentThread().getName() + " - State after sleep: " +
Thread.currentThread().getState());
}
```

```
public static void main(String[] args) {
ThreadLifecycleExample thread = new ThreadLifecycleExample();
System.out.println(thread.getName() + " - State before start: " +thread.getState());
thread.start();
System.out.println(thread.getName() + " - State after start: " +
thread.getState());
}
}
```

OUTPUT



JDBC

```
CREATE DATABASE employee_db;
USE employee_db;
CREATE TABLE employees (
id INT PRIMARY KEY AUTO_INCREMENT,
name VARCHAR(100),
position VARCHAR(100),
salary DECIMAL(10, 2)
);
-- Insert some sample data
INSERT INTO employees (name, position, salary) VALUES (‘John Doe’, ‘Software
Engineer’, 75000);
INSERT INTO employees (name, position, salary) VALUES (‘Jane Smith’, ‘HR
Manager’, 65000);
INSERT INTO employees (name, position, salary) VALUES (‘Steve Brown’, ‘Team
Lead’, 85000);
//Code for DatabaseConnection.java:
packagejdbcexample;
importjava.sql.Connection;
importjava.sql.DriverManager;
importjava.sql.SQLException;
/**
 *
 * @author student
 */
public class DatabaseConnection {
private static final String URL ="jdbc:mysql://localhost:3306/employee_db"; // Database URL
private static final String USER = "root";
```

```
private static final String PASSWORD = "";

public static Connection getConnection() throws SQLException {

try {

Class.forName("com.mysql.cj.jdbc.Driver");

returnDriverManager.getConnection(URL, USER, PASSWORD);
}
catch (ClassNotFoundException | SQLException e) {
System.out.println("Connection failed:" + e.getMessage());
throw new SQLException("Failed to establish connection.");
   }
 }
}
```

1. Open NetBeans IDE 8.2.

2. Create a new Java application:

- Go to File > New Project.
- Select Java as the project type, and choose Java Application.
- Name your project JDBCExample.
- 3. Add MySQL JDBC Driver to your project:
- Right-click on the project in the Projects pane.
- Select Properties.
- In the Libraries tab, click Add JAR/Folder.
- Navigate to the location of your mysql-connector-java-x.x.xx.jar file and add it.

//Code for EmployeeDAO.java:

```
packagejdbcexample;
importjava.sql.Connection;
importjava.sql.DriverManager;
importjava.sql.SQLException;
```

```
/**
 *
 * @author student
 */
public class DatabaseConnection {
private static final String URL ="jdbc:mysql://localhost:3306/employee_db"; // Database URL
private static final String USER = "root"; // Your MySQL username
private static final String PASSWORD = ""; // Your MySQL password
public static Connection getConnection() throws SQLException {
try {
Class.forName("com.mysql.cj.jdbc.Driver");
returnDriverManager.getConnection(URL, USER, PASSWORD);
}
catch (ClassNotFoundException | SQLException e) {
System.out.println("Connection failed:" + e.getMessage());
throw new SQLException("Failed to establish connection.");
   }
 }
}
```

//Code for EmployeeDAO.java:

```
packagejdbcexample;
importjava.sql.*;
importjava.util.ArrayList;
importjava.util.List;
/**
 *
 * @author student
 */
public class EmployeeDAO {
```

```
public static void addEmployee(String name, String position, double salary) {
String sql = "INSERT INTO employees (name, position, salary) VALUES(?, ?, ?)";
try (Connection conn = DatabaseConnection.getConnection();
PreparedStatementstmt = conn.prepareStatement(sql)) {
stmt.setString(1, name);
stmt.setString(2, position);
stmt.setDouble(3, salary);
introwsAffected = stmt.executeUpdate();
System.out.println("Employee added successfully. Rows affected:" + rowsAffected);
} catch (SQLException e) {
e.printStackTrace();
}
}
```

// Read all employees

```
public static List<Employee>getAllEmployees() {
List<Employee> employees = new ArrayList<>();
String sql = "SELECT * FROM employees";
try (Connection conn = DatabaseConnection.getConnection();
Statement stmt = conn.createStatement();
ResultSetrs = stmt.executeQuery(sql)) {
while (rs.next()) {
Employee employee = new Employee(

rs.getInt("id"),
rs.getString("name"),
rs.getString("position"),
rs.getDouble("salary")
);
employees.add(employee);
```

```
}
} catch (SQLException e) {
e.printStackTrace();
}
return employees;
}
// Update an employee  information
public static void updateEmployee(int id, String name, String position,
double salary) {
String sql = "UPDATE employees (name , position , salary)VALUES(?,?,?)";

try (Connection conn = DatabaseConnection.getConnection();
PreparedStatementstmt = conn.prepareStatement(sql)) {
stmt.setString(1, name);
stmt.setString(2, position);
stmt.setDouble(3, salary);
stmt.setInt(4, id);
introwsAffected = stmt.executeUpdate();
System.out.println("Employee updated successfully. Rows affected:" + rowsAffected);
} catch (SQLException e) {
e.printStackTrace();
}
}
// Delete an employee
public static void deleteEmployee(int id) {
String sql = "DELETE FROM employees WHERE id = ?";
try (Connection conn = DatabaseConnection.getConnection();
PreparedStatementstmt = conn.prepareStatement(sql)) {
stmt.setInt(1, id);
```

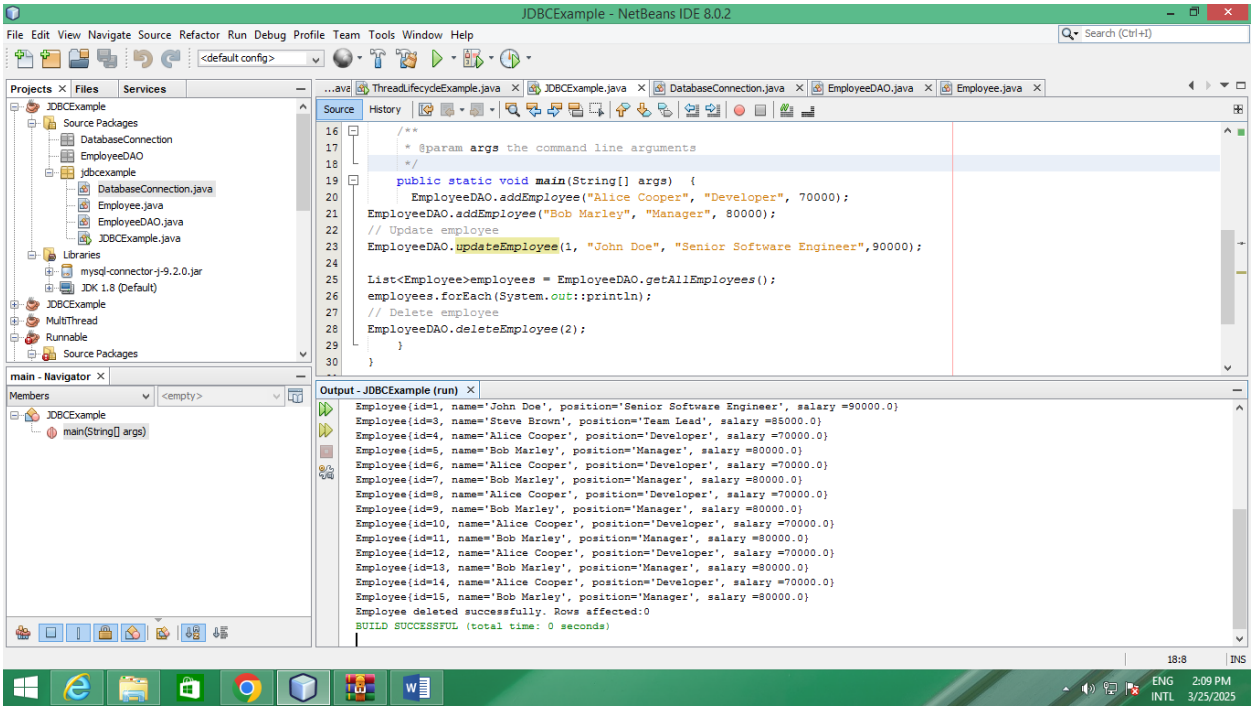
```
introwsAffected = stmt.executeUpdate();
System.out.println("Employee deleted successfully. Rows affected:" + rowsAffected);
} catch (SQLException e) {
e.printStackTrace();
}
}
}
```

//Code for Employee.java:

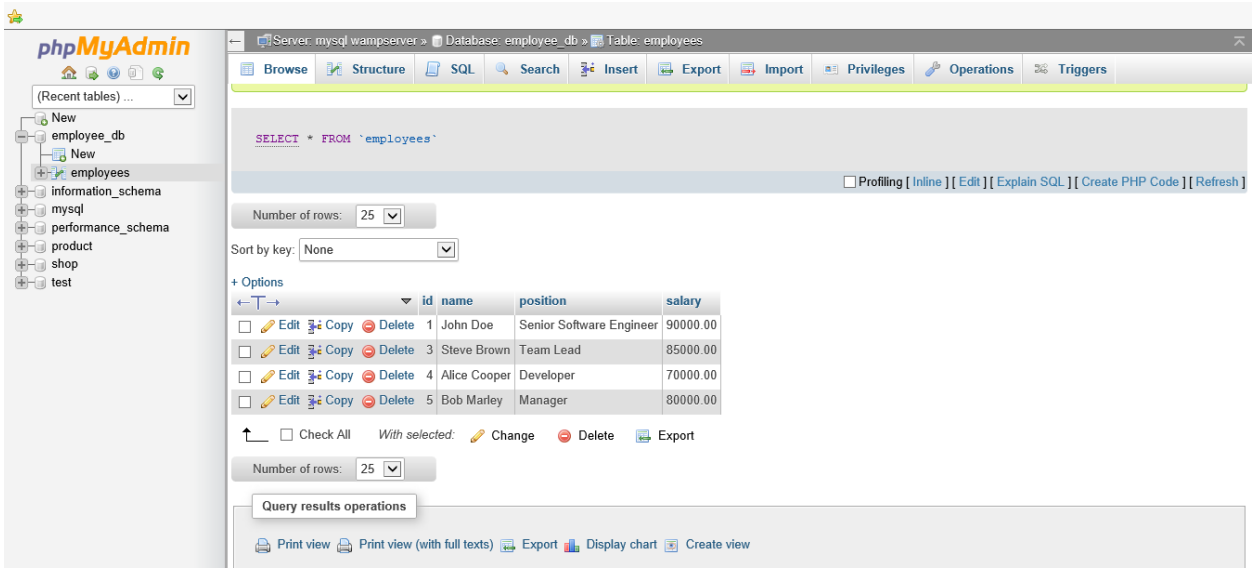
```
public class Employee {

privateint id;
private String name;
private String position;
private double salary;
public Employee(int id, String name, String position, double salary) {
this.id = id;
this.name = name;
this.position = position;
this.salary = salary;
}
// Getters and setters
publicintgetId() { return id; }
public void setId(int id) { this.id = id; }
public String getName() { return name; }
public void setName(String name) { this.name = name; }
public String getPosition() { return position; }
public void setPosition(String position) { this.position = position; }
public double getSalary() { return salary; }
public void setSalary(double salary) { this.salary = salary; }
```

```
@Override
public String toString() {
return "Employee{id=" + id + ", name='" + name + "', position='" +position + "', salary =" + salary + '}';
}
}
//Code for JDBCExample.java:
packagejdbcexample;

importjava.util.List;

/**
 *
 * @author student
 */
public class JDBCExample {

   /**
    * @paramargs the command line arguments
    */
public static void main(String[] args)  {
EmployeeDAO.addEmployee("Alice Cooper", "Developer", 70000);
EmployeeDAO.addEmployee("Bob Marley", "Manager", 80000);
EmployeeDAO.updateEmployee(1, "John Doe", "Senior Software Engineer",90000);

List<Employee>employees = EmployeeDAO.getAllEmployees();
employees.forEach(System.out::println);
EmployeeDAO.deleteEmployee(2);
   }}
```

OUT PUT



DATABASE UPDATE



XML

books.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<library>
 <book>
 <title>The Great Gatsby</title>
 <author>F. Scott Fitzgerald</author>
 <year>1925</year>
 <genre>Fiction</genre>
 </book>
 <book>
 <title>To Kill a Mockingbird</title>
 <author>Harper Lee</author>
 <year>1960</year>
<genre>Fiction</genre>
 </book>
 <book>
<title>1984</title>
 <author>George Orwell</author>
 <year>1949</year>
 <genre>Dystopian</genre>
 </book>
 </library>
```

Parsing XML in Java

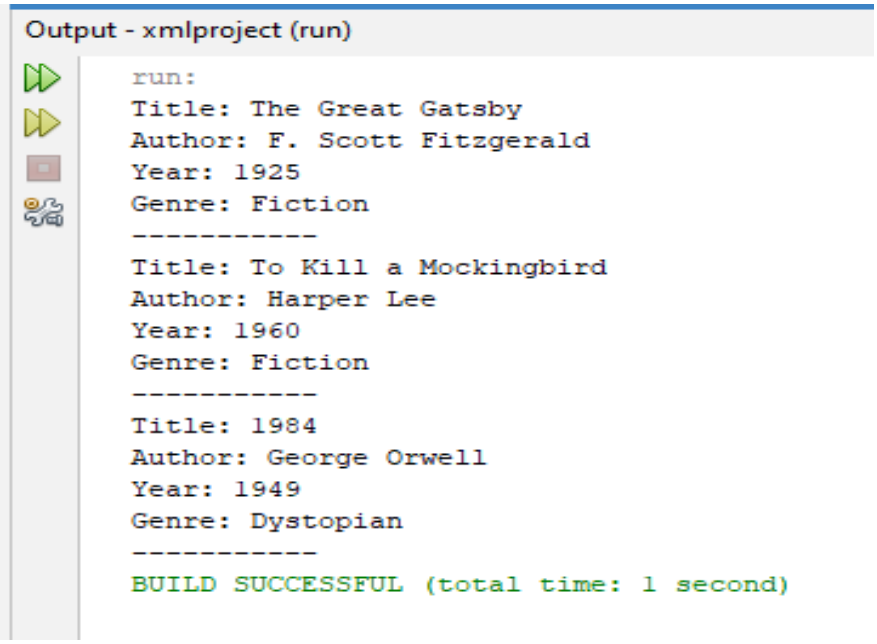
Create a Java Class for XML Parsing:

```
package javaapplication8;
import java.io.File;
import org.w3c.dom.*;
import javax.xml.parsers.*;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
/**
 *
 * @author student
 */
public class xmlparser {
```

```
   public static void main(String[] args) {
 try {
 // Create a new DocumentBuilderFactory and DocumentBuilder
DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
 DocumentBuilder builder = factory.newDocumentBuilder();

 // Parse the XML file

 Document document =
builder.parse("C:\\Users\\student\\Documents\\NetBeansProjects\\JavaApplication8\\src\\java
application8\\books.xml");

 // Normalize the document
 document.getDocumentElement().normalize();

 // Get the root element (library)
 NodeList nodeList = document.getElementsByTagName("book");

 // Loop through each book in the XML document
 for (int i = 0; i < nodeList.getLength(); i++) {
 Node node = nodeList.item(i);

 if (node.getNodeType() == Node.ELEMENT_NODE) {
 Element element = (Element) node;

 // Get and print the details of each book
String title =
element.getElementsByTagName("title").item(0).getTextContent();
 String author = element.getElementsByTagName("author").item(0).getTextContent();
 String year = element.getElementsByTagName("year").item(0).getTextContent();
 String genre = element.getElementsByTagName("genre").item(0).getTextContent();

 System.out.println("Title: " + title);
 System.out.println("Author: " + author);
System.out.println("Year: " + year);
 System.out.println("Genre: " + genre);
 System.out.println("-----------");
 }
 }
// Modify the year of the first book
 Element firstBook = (Element) nodeList.item(0);
 firstBook.getElementsByTagName("year").item(0).setTextContent("2023");
```

```
 // Save the modified document
 TransformerFactory transformerFactory =
TransformerFactory.newInstance();
 Transformer transformer = transformerFactory.newTransformer();
 DOMSource source = new DOMSource(document);
 StreamResult result = new StreamResult(new
File("C:\\Users\\student\\Documents\\NetBeansProjects\\JavaApplication8\\src\\javaapplicatio
n8\\updated_books.xml"));
 transformer.transform(source, result);
} catch (Exception e) {
 e.printStackTrace();
 }
 }
}
```

OUTPUT



Modifying XML Data


```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><library>
<book>
<title>The Great Gatsby</title>
<author>F. Scott Fitzgerald</author>
 <year>2023</year>
<genre>Fiction</genre>
 </book>
 <book>
<title>To Kill a Mockingbird</title>
<author>Harper Lee</author>
<year>1960</year>
 <genre>Fiction</genre>
 </book>
 <book>
 <title>1984</title>
<author>George Orwell</author>
<year>1949</year>
<genre>Dystopian</genre>
</book>
 </library>
```

OUTPUT

Output - xmlproject (run)

```
run:
Updated XML content:
<?xml version="1.0" encoding="UTF-8" standalone="no"?><library>
    <book>
        <title>The Great Gatsby</title>
        <author>F. Scott Fitzgerald</author>
        <year>2023</year>
        <genre>Fiction</genre>
    </book>
    <book>
        <title>To Kill a Mockingbird</title>
        <author>Harper Lee</author>
        <year>1960</year>
        <genre>Fiction</genre>
    </book>
    <book>
        <title>1984</title>
        <author>George Orwell</author>
        <year>1949</year>
        <genre>Dystopian</genre>
    </book>
</library>BUILD SUCCESSFUL (total time: 1 second)
```

Servlet

Create a servlet that receives user input from an HTML form and displays it back to the user.

Steps:

1. Create an HTML form to collect the user's name.

2. Create a Servlet (GetUserInputServlet) to handle the form submission and display the user's name.

HTML Form (index.html):

```
<html>
<head>
<title>TODO supply a title</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body bgcolor ="#ffbf00">

<div><h1><center>User input</center></h1></div>

<form action="getUserInput12" method="POST">
<table style = "width:70%">
<tr><th><h3> Name </h3></th><th><input type="text" name="un"</tr>
<tr><td colspan="2" align="center"><input type ="submit" value="save"</td></tr>
</form>
</table>
<tr>
<hr>
<h1>Calculate</h1>
<form action="CalculateSumServlet"method="post">
        First Number:<input type="number" name="num1" required><br>
```

```
                Second Number: <input type="number" name="num2" required><br>
<input type="submit" value="Calculate Sum">
</form>
</tr>
</form>

</body>
</html>
```

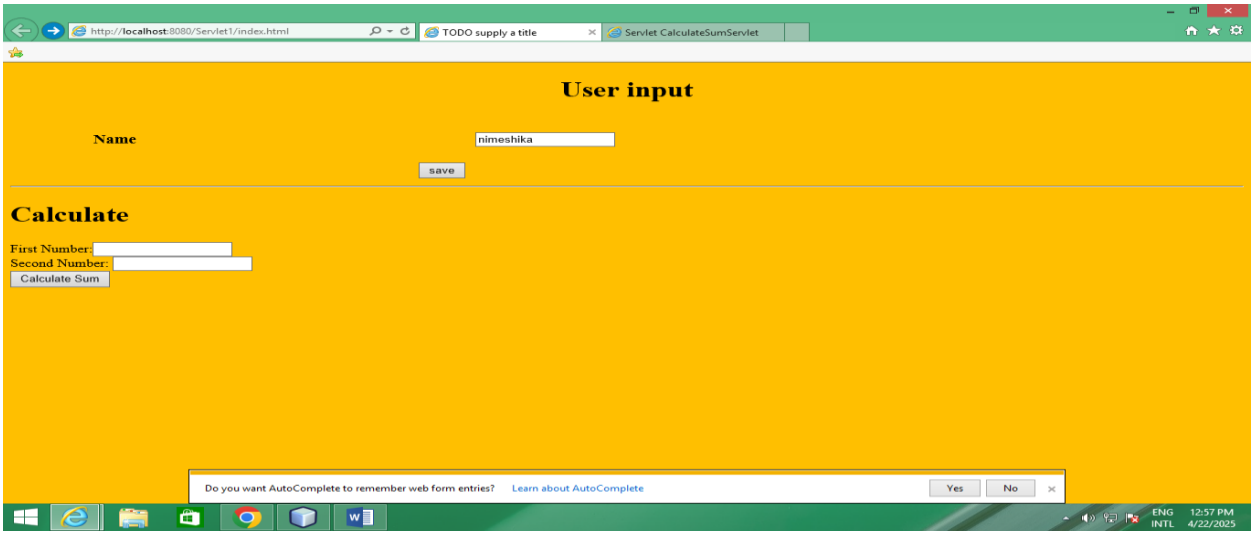
(getUserInput12.java)

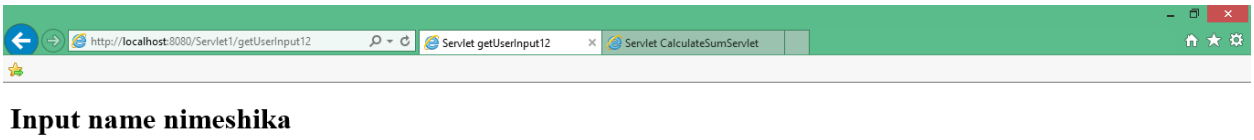
```
importjava.io.IOException;
importjava.io.PrintWriter;
importjavax.servlet.ServletException;
importjavax.servlet.annotation.WebServlet;
importjavax.servlet.http.HttpServlet;
importjavax.servlet.http.HttpServletRequest;
importjavax.servlet.http.HttpServletResponse;

/**
 *
 * @author student
 */
@WebServlet(urlPatterns = {"/getUserInput12"})
public class getUserInput12 extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
```

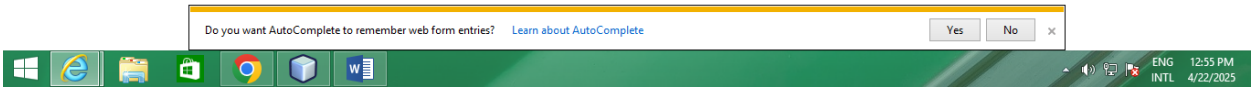
```
    * @param response servlet response
    * @throws ServletException if a servlet-specific error occurs
    * @throws IOException if an I/O error occurs
    */
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throwsServletException, IOException {
        String name = request.getParameter("un");
response.setContentType("text/html;charset=UTF-8");
try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
out.println("<!DOCTYPE html>");
out.println("<html>");
out.println("<head>");
out.println("<title>Servlet getUserInput12</title>");
out.println("</head>");
out.println("<body>");
out.println("<h1> Input name " + name + "</h1>");
out.println("</body>");
out.println("</html>");
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to
edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
```

```
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throwsServletException, IOException {
processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throwsServletException, IOException {
processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
```

```
public String getServletInfo() {

return "Short description";

   }// </editor-fold>


}}
```





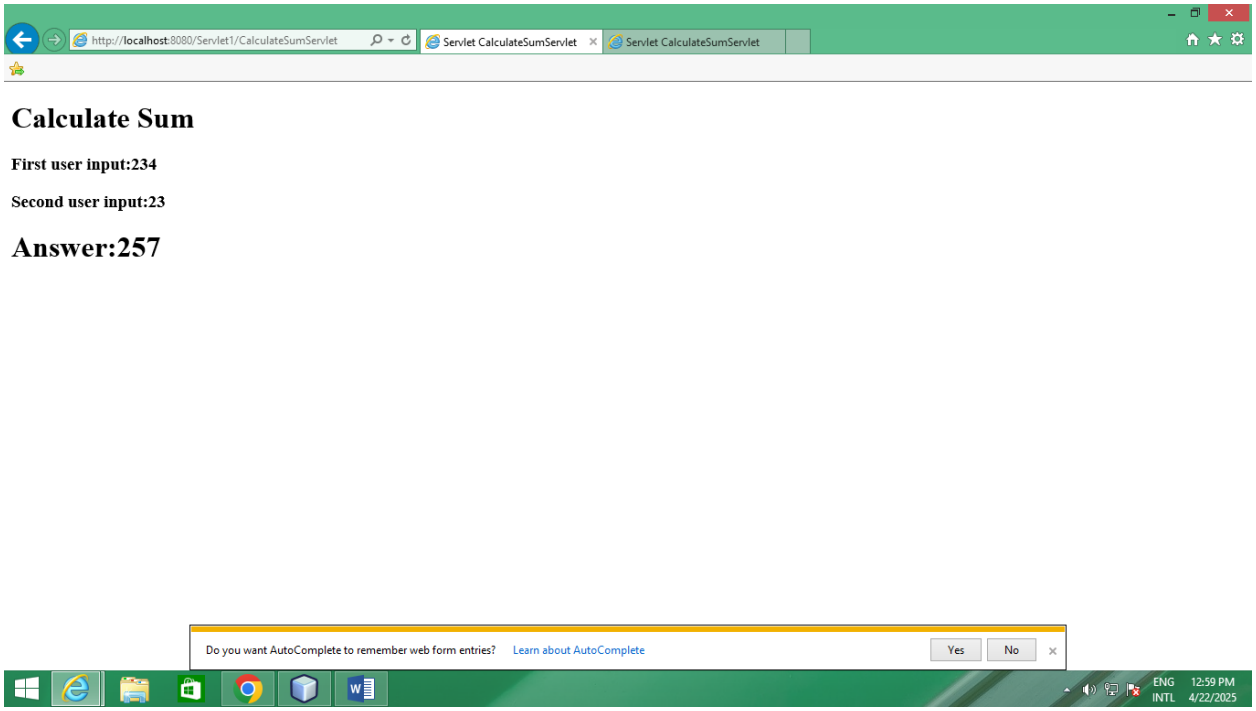


```
importjava.io.IOException;
importjava.io.PrintWriter;
importjavax.servlet.ServletException;
importjavax.servlet.annotation.WebServlet;
importjavax.servlet.http.HttpServlet;
importjavax.servlet.http.HttpServletRequest;
importjavax.servlet.http.HttpServletResponse;

/**
 *
 * @author student
 */
@WebServlet(urlPatterns = {"/CalculateSumServlet"})
public class CalculateSumServlet extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throwsServletException, IOException {
```

```
int num1 = Integer.parseInt(request.getParameter("num1"));
int num2 = Integer.parseInt(request.getParameter("num2"));
int sum = num1 + num2;

response.setContentType("text/html;charset=UTF-8");
try (PrintWriter out = response.getWriter()) {
        /* TODO output your page here. You may use following sample code. */
out.println("<!DOCTYPE html>");
out.println("<html>");
out.println("<head>");
out.println("<title>Servlet CalculateSumServlet</title>");
out.println("</head>");
out.println("<body>");
out.println("<h1> Calculate Sum</h1>");
out.println("<h3> First user input:"+num1+"</h1>");
out.println("<h3> Second  user input:"+num2+"</h1>");
out.println("<h1> Answer:"+sum+"</h1>");
        //out.println("<h1>The sum of " + num1 + " and " + num2 + " is: " + sum + "</h1>");
        //out.println("<h1>Servlet CalculateSumServlet at " + request.getContextPath() + "</h1>");
out.println("</body>");
out.println("</html>");
      }
   }

   // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to
edit the code.">
   /**
    * Handles the HTTP <code>GET</code> method.
    *
```

```
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throwsServletException, IOException {
processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throwsServletException, IOException {
processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
```

```
    */
    @Override
public String getServletInfo() {
return "Short description";
    }
```



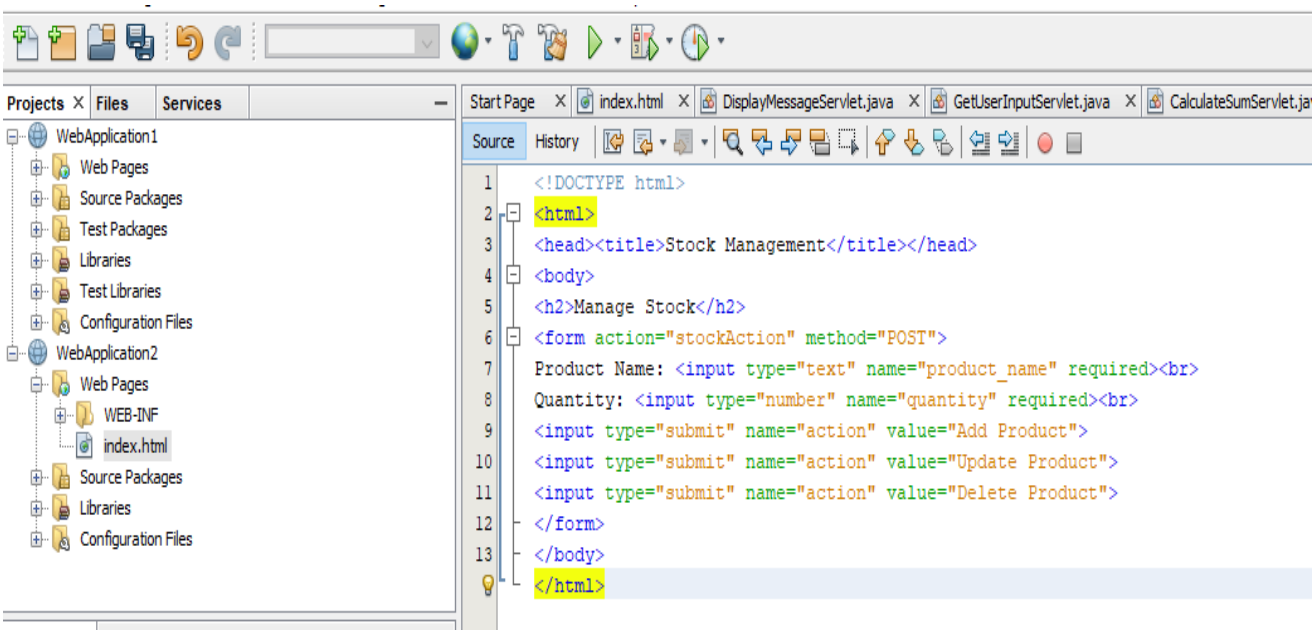
Lab Task 4: Java Servlet with Database CRUD Operations

<u>Database Setup :</u>

```sql
CREATE DATABASE stock_management;
USE stock_management;
CREATE TABLE stock (
id INT AUTO_INCREMENT PRIMARY KEY,
product_name VARCHAR(255),
quantity INT
);
```

Table	Action	Rows	Type	Collation	Size	Overhead
stock	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 KiB	-
1 table	Sum	0	InnoDB	utf8mb4_general_ci	16.0 KiB	0 B

HTML Form (stockForm.html)



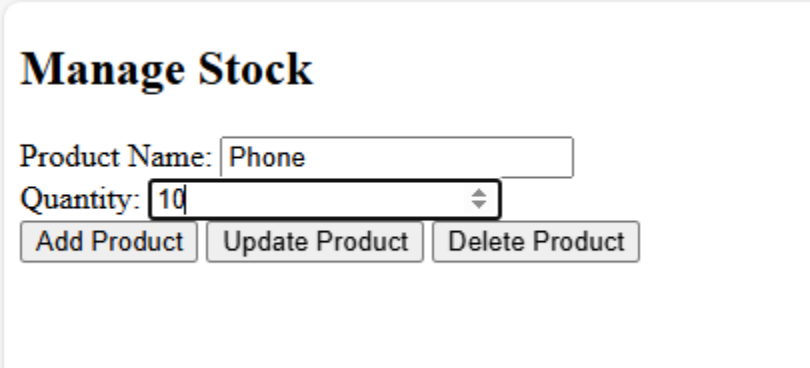
Servlet Code (StockManagementServlet.java):

```
package com.example;

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

@WebServlet("/stockAction")
public class StockManagementServlet extends HttpServlet {
```

```
    // Initialize driver when servlet loads
    @Override
    public void init() throws ServletException {
        try {
Class.forName("com.mysql.cj.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            throw new ServletException("MySQL JDBC Driver not found", e);
        }
    }

    private Connection getConnection() throws SQLException {
        String url =
"jdbc:mysql://localhost:3306/stock_management?useSSL=false&serverTimezone=UTC";
        String username = "root";
        String password = "316830059";
        return DriverManager.getConnection(url, username, password);
    }

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

        String action = request.getParameter("action");
        String productName = request.getParameter("product_name");
        int quantity = Integer.parseInt(request.getParameter("quantity"));

        try (Connection conn = getConnection()) {
            switch (action) {
                case "Add Product":
addProduct(conn, productName, quantity, out);
                    break;
                case "Update Product":
updateProduct(conn, productName, quantity, out);
                    break;
                case "Delete Product":
deleteProduct(conn, productName, out);
                    break;
                default:
out.println("<h1>Invalid Action</h1>");
            }
        } catch (SQLException e) {
out.println("<h1>Database Error: " + e.getMessage() + "</h1>");
e.printStackTrace();
        }

out.println("<br><a href='stockForm.html'>Back to Form</a>");
```

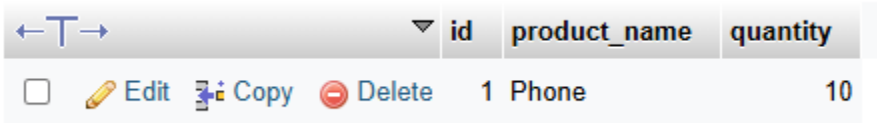
```
    }

    private void addProduct(Connection conn, String name, int quantity, PrintWriter out)
            throws SQLException {
        String sql = "INSERT INTO stock (product_name, quantity) VALUES (?, ?)";
        try (PreparedStatementstmt = conn.prepareStatement(sql)) {
stmt.setString(1, name);
stmt.setInt(2, quantity);
stmt.executeUpdate();
out.println("<h1>Product Added Successfully</h1>");
        }
    }

    private void updateProduct(Connection conn, String name, int quantity, PrintWriter out)
            throws SQLException {
        String sql = "UPDATE stock SET quantity = ? WHERE product_name= ?";
        try (PreparedStatementstmt = conn.prepareStatement(sql)) {
stmt.setInt(1, quantity);
stmt.setString(2, name);
            int rows = stmt.executeUpdate();
            if (rows > 0) {
out.println("<h1>Product Updated Successfully</h1>");
            } else {
out.println("<h1>Product Not Found</h1>");
            }
        }
    }

    private void deleteProduct(Connection conn, String name, PrintWriter out)
            throws SQLException {
        String sql = "DELETE FROM stock WHERE product_name= ?";
        try (PreparedStatementstmt = conn.prepareStatement(sql)) {
stmt.setString(1, name);
            int rows = stmt.executeUpdate();
            if (rows > 0) {
out.println("<h1>Product Deleted Successfully</h1>");
            } else {
out.println("<h1>Product Not Found</h1>");
            }
        }
    } }
```

Output



Product Added Successfully

Back to Form



Display Data from Database on Another Web Page

Servlet Code (DisplayProductsServlet.java):

```
package com.example;


import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
@WebServlet("/displayProducts")
public class DisplayProductsServlet extends HttpServlet {
    // Reuse your existing connection method
    private Connection getConnection() throws SQLException {
        String url =
"jdbc:mysql://localhost:3306/stock_management?useSSL=false&serverTimezone=UTC";
        String username = "root";
        String password = "316830059";
        return DriverManager.getConnection(url, username, password);
    }
```

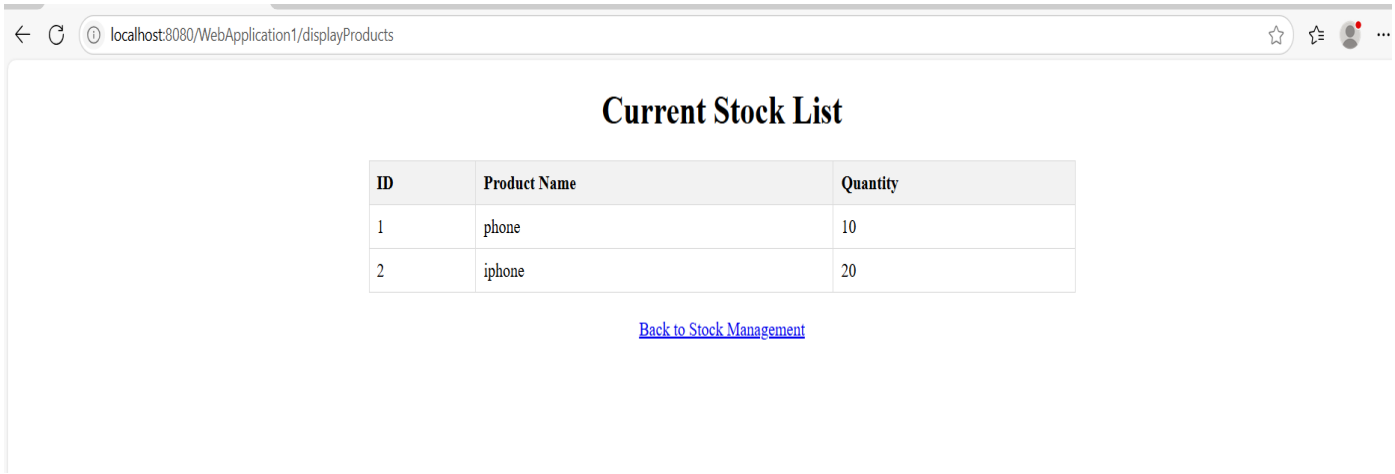
```
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<!DOCTYPE html>");
out.println("<html>");
out.println("<head>");
out.println("<title>Stock List</title>");
out.println("<style>");
out.println("table { border-collapse: collapse; width: 50%; margin: 20px auto; }");
out.println("th, td { border: 1px solid #ddd; padding: 8px; text-align: left; }");
out.println("th { background-color: #f2f2f2; }");
out.println("</style>");
out.println("</head>");
out.println("<body>");
out.println("<h1 style='text-align: center;'>Current Stock List</h1>");

try (Connection conn = getConnection();
            Statement stmt = conn.createStatement();
ResultSetrs = stmt.executeQuery("SELECT * FROM stock")) {
out.println("<table>");
out.println("<tr><th>ID</th><th>Product Name</th><th>Quantity</th></tr>");

            while (rs.next()) {
out.println("<tr>");
out.println("<td>" + rs.getInt("id") + "</td>");
out.println("<td>" + rs.getString("product_name") + "</td>");
out.println("<td>" + rs.getInt("quantity") + "</td>");
out.println("</tr>");
            }

out.println("</table>");
        } catch (SQLException e) {
out.println("<h2 style='color: red; text-align: center;'>Error retrieving stock: "
                    + e.getMessage() + "</h2>");
e.printStackTrace();      }

out.println("<div style='text-align: center; margin-top: 20px;'>");
out.println("<a href='stockForm.html'>Back to Stock Management</a>");
out.println("</div>");
out.println("</body>");
out.println("</html>");
    }
}
```

Updated stockForm.html

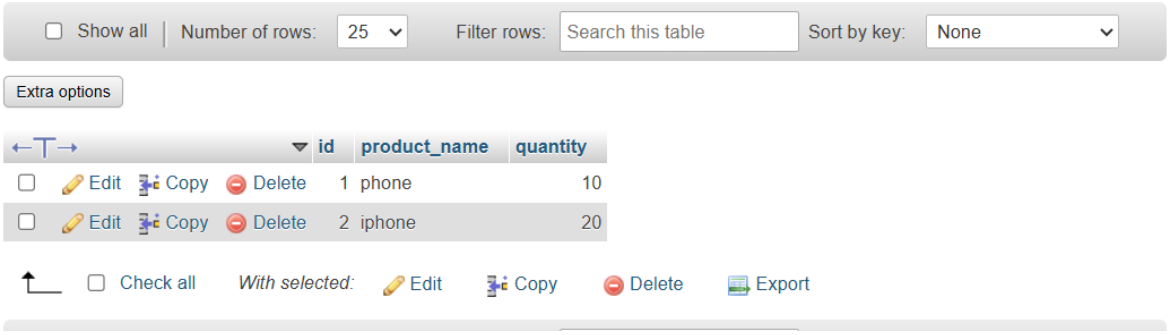
```
<!DOCTYPE html>
<html>
```

```
<head><title>Stock Management</title>
<style>    body { font-family: Arial, sans-serif; margin: 20px; }
            form { max-width: 500px; margin: 0 auto; padding: 20px; border: 1px solid #ddd; border-
radius: 5px; }
            input[type="text"], input[type="number"] { width: 100%; padding: 8px; margin: 5px 0 15px; }
              input[type="submit"] { padding: 8px 15px; margin-right: 10px; }
.view-link { display: block; text-align: center; margin-top: 20px; }
</style>
</head>
<body><h2 style="text-align: center;">   Manage Stock   </h2>
<form action="stockAction" method="POST">
           Product Name: <input type="text" name="product_name" required><br>
            Quantity: <input type="number" name="quantity" required><br>
<input type="submit" name="action" value="Add Product">
<input type="submit" name="action" value="Update Product">
<input type="submit" name="action" value="Delete Product">
</form>
<div class="view-link">
<a href="displayProducts"> View All Products   </a>
</div>
</body>
</html>
```

Output



Database


Show all
Number of rows:
25
Filter rows:
Search this table
Sort by key:
None
Extra options
id
product_name
quantity
Edit
Copy
Delete
1
phone
10
Edit
Copy
Delete
2
iphone
20
Check all
With selected:
Edit
Copy
Delete
Export