Advanced Technological Institute

Gampaha

Assignment 01- Enterprise Architecture FT 2025

adding code for as answer for each question and relevant Git Hub Commit Report

Submit by: A.A.A.T. Athauda

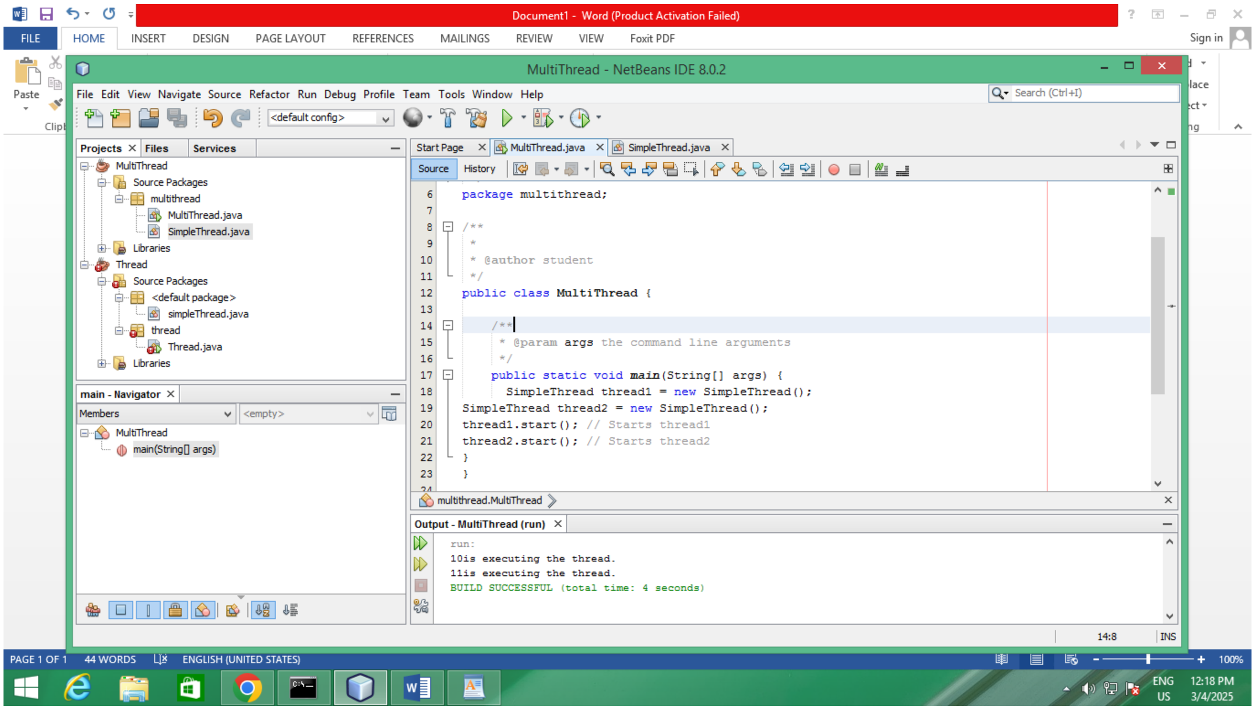
GAM/IT/2022/F/0073 S

Java Thread

Lab 02 – 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(); // Starts thread1
thread2.start(); // Starts thread2
}
}
```

Output



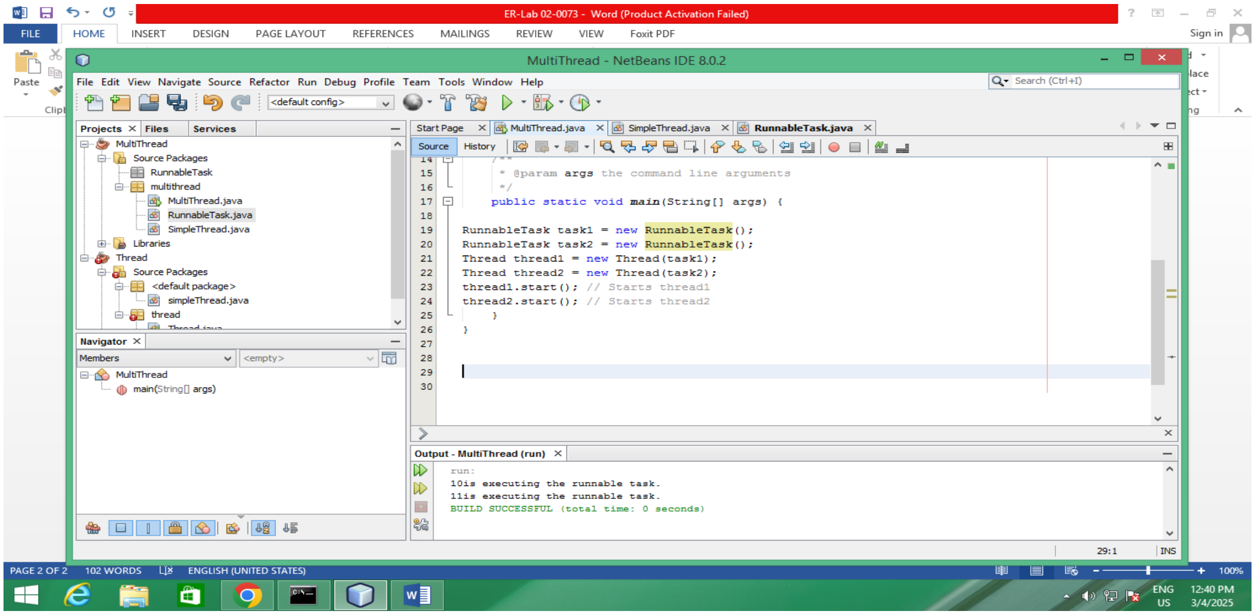
```
package multithread;

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

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
     SimpleThread thread1 = new SimpleThread();
SimpleThread thread2 = new SimpleThread();
thread1.start(); // Starts thread1
thread2.start(); // Starts thread2
}
}
```

```
run:
10is executing the thread.
11is executing the thread.
BUILD SUCCESSFUL (total time: 4 seconds)
```

Task 02

```
public class RunnableTask implements Runnable{
@Override
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(); // Starts thread1

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

OUTPUT



```
    /**
     * @param args the command line arguments
     */
    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(); // Starts thread1
thread2.start(); // Starts thread2
    }
}
```

Output - MultiThread (run)

```
run:
10is executing the runnable task.
11is executing the runnable task.
BUILD SUCCESSFUL (total time: 0 seconds)
```

Task 03

public class Counter {

private int count = 0;

// Synchronized method to ensure thread-safe access to the counter

public synchronized void increment() {

count++;

}

public int getCount() {

return count;

}

}

public class SynchronizedExample extends Thread {

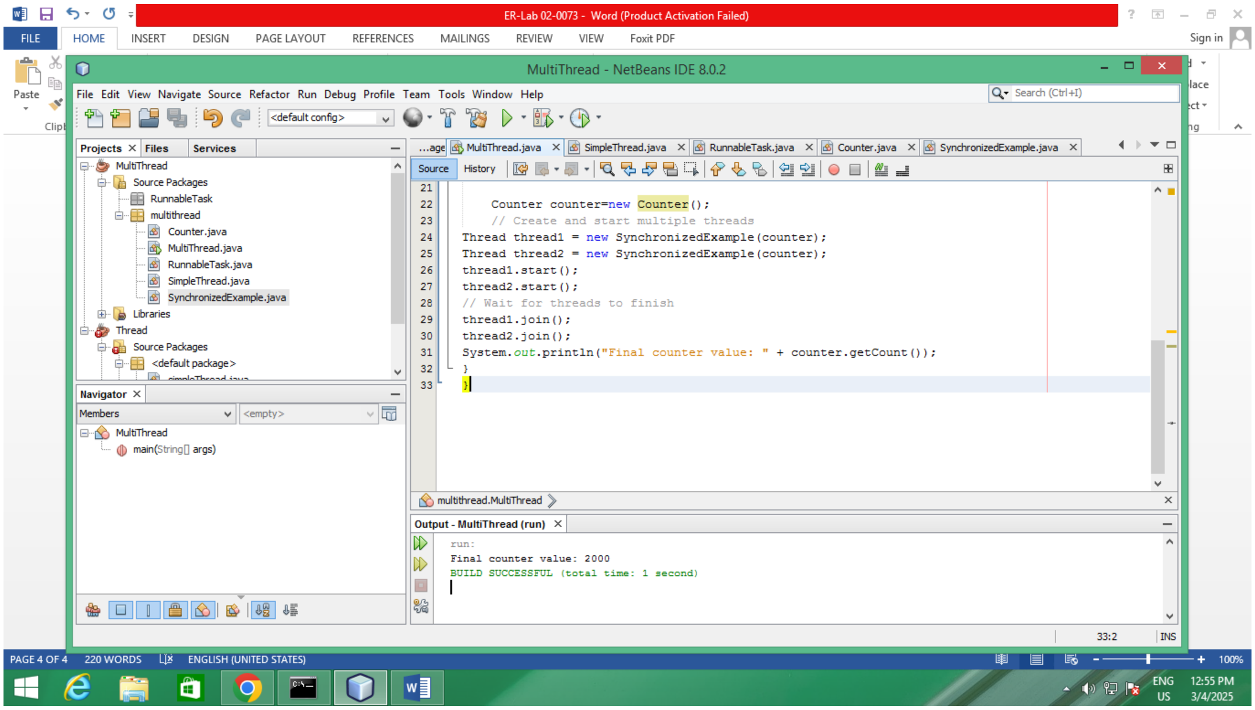
private Counter counter;

public SynchronizedExample(Counter counter) {

this.counter = counter;

```
}
@Override
public void run() {
for (int i = 0; i < 1000; i++) {
counter.increment();
}

}
}
   public static void main(String[] args) throws InterruptedException {
  Counter counter=new Counter();
   // Create and start multiple threads
Thread thread1 = new SynchronizedExample(counter);
Thread thread2 = new SynchronizedExample(counter);
thread1.start();
thread2.start();
// Wait for threads to finish
thread1.join();
thread2.join();
System.out.println("Final counter value: " + counter.getCount());
}
}
```

Output



```
    Counter counter=new Counter();
    // Create and start multiple threads
Thread thread1 = new SynchronizedExample(counter);
Thread thread2 = new SynchronizedExample(counter);
thread1.start();
thread2.start();
// Wait for threads to finish
thread1.join();
thread2.join();
System.out.println("Final counter value: " + counter.getCount());
}
}
```

```
run:
Final counter value: 2000
BUILD SUCCESSFUL (total time: 1 second)
```

Task04

package multithread1;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

class Task implements Runnable {

private int taskId;

public Task(int taskId) {

this.taskId = taskId;

}

@Override

public void run() {

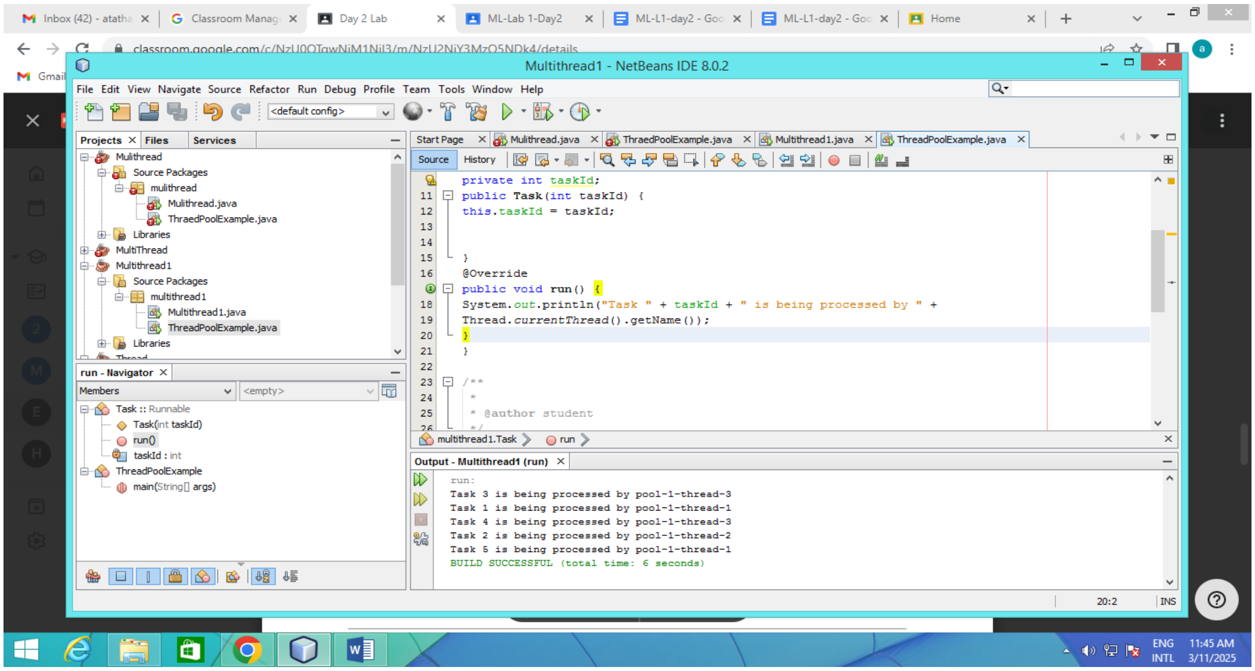
System.out.println("Task " + taskId + " is being processed by " +

Thread.currentThread().getName());

}

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

Output



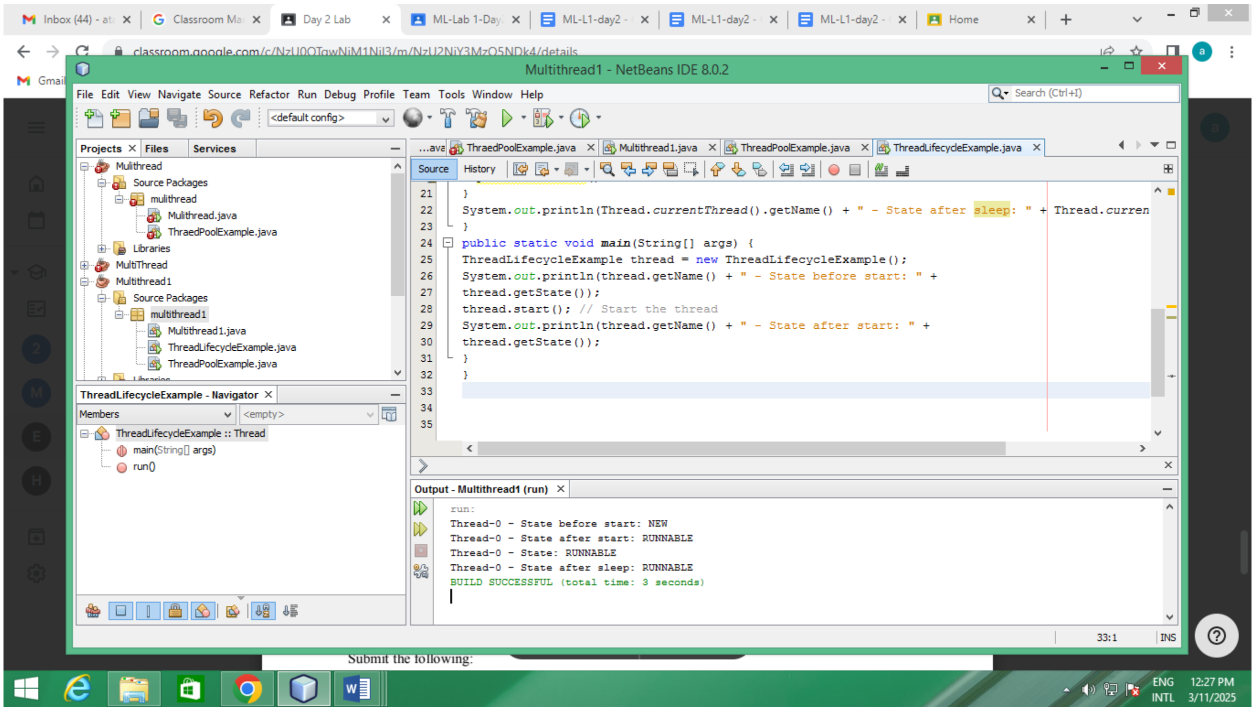
```
private int taskId;
public Task(int taskId) {
this.taskId = taskId;


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

/**
 *
 * @author student
```

```
run:
Task 3 is being processed by pool-1-thread-3
Task 1 is being processed by pool-1-thread-1
Task 4 is being processed by pool-1-thread-3
Task 2 is being processed by pool-1-thread-2
Task 5 is being processed by pool-1-thread-1
BUILD SUCCESSFUL (total time: 6 seconds)
```

Task 05

```
package multithread1;
/**
 *
 * @author student
 */
public class ThreadLifecycleExample extends Thread  {
   @Override
public void run() {
System.out.println(Thread.currentThread().getName() + " - State: " +
Thread.currentThread().getState());
try {
Thread.sleep(2000); // Simulate waiting state
} 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(); // Start the thread
System.out.println(thread.getName() + " - State after start: " +
thread.getState());
}
}
```

Output



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

Output - Multithread1 (run)

```
run:
Thread-0 - State before start: NEW
Thread-0 - State after start: RUNNABLE
Thread-0 - State: RUNNABLE
Thread-0 - State after sleep: RUNNABLE
BUILD SUCCESSFUL (total time: 3 seconds)
```

JDBC

Lab sheet -03

```
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:

```
package jdbcexample;
import java.sql.Connection;
import java.sql.DriverManager;
import java.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");

 return DriverManager.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 &gt; 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:

```
package jdbcexample;

import java.sql.Connection;

import java.sql.DriverManager;

import java.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");
 return DriverManager.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:

```
package jdbcexample;
import java.sql.*;
import java.util.ArrayList;
import java.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();
PreparedStatement stmt = conn.prepareStatement(sql)) {
```

```java
package jdbcexample;
import java.sql.*;
import java.util.ArrayList;
import java.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();
PreparedStatement stmt = conn.prepareStatement(sql)) {
stmt.setString(1, name);
stmt.setString(2, position);
stmt.setDouble(3, salary);
int rowsAffected = 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();
ResultSet rs = 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&#39;s 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();
PreparedStatement stmt = conn.prepareStatement(sql)) {
stmt.setString(1, name);
stmt.setString(2, position);
stmt.setDouble(3, salary);
stmt.setInt(4, id);
int rowsAffected = 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();
PreparedStatement stmt = conn.prepareStatement(sql)) {
stmt.setInt(1, id);
int rowsAffected = stmt.executeUpdate();
System.out.println("Employee deleted successfully. Rows affected:" + rowsAffected);
} catch (SQLException e) {
e.printStackTrace();
}
}
}
```

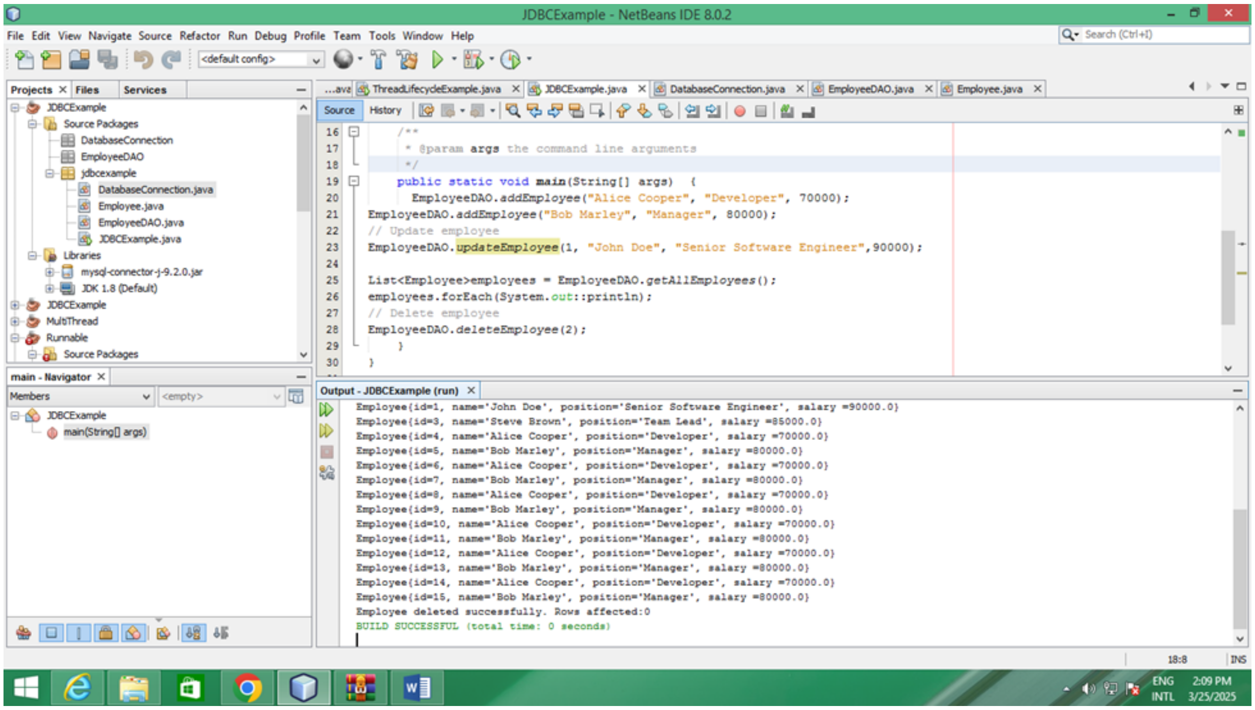
Code for Employee.java:

```
public class Employee {

    private int 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
public int getId() { 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:

```
package jdbcexample;
import java.util.List;

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

    /**
     * @param args 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

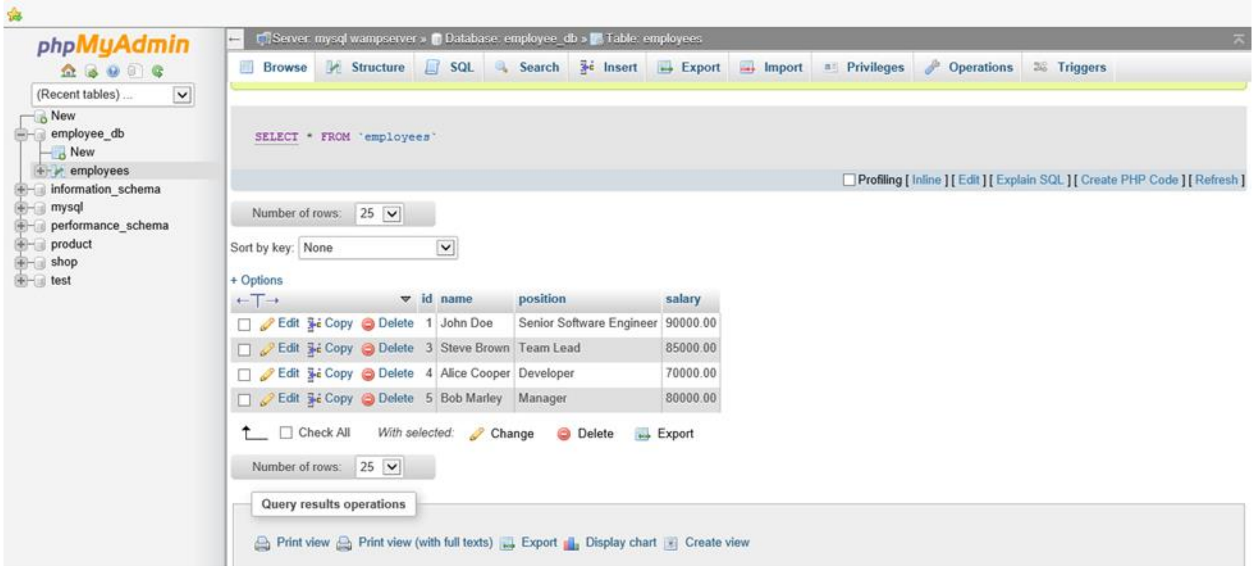


```
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args)  {
      EmployeeDAO.addEmployee("Alice Cooper", "Developer", 70000);
EmployeeDAO.addEmployee("Bob Marley", "Manager", 80000);
// Update employee
EmployeeDAO.updateEmployee(1, "John Doe", "Senior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

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

Output - JDBCExample (run)

```
Employee{id=1, name='John Doe', position='Senior Software Engineer', salary =90000.0}
Employee{id=3, name='Steve Brown', position='Team Lead', salary =85000.0}
Employee{id=4, name='Alice Cooper', position='Developer', salary =70000.0}
Employee{id=5, name='Bob Marley', position='Manager', salary =80000.0}
Employee{id=6, name='Alice Cooper', position='Developer', salary =70000.0}
Employee{id=7, name='Bob Marley', position='Manager', salary =80000.0}
Employee{id=8, name='Alice Cooper', position='Developer', salary =70000.0}
Employee{id=9, name='Bob Marley', position='Manager', salary =80000.0}
Employee{id=10, name='Alice Cooper', position='Developer', salary =70000.0}
Employee{id=11, name='Bob Marley', position='Manager', salary =80000.0}
Employee{id=12, name='Alice Cooper', position='Developer', salary =70000.0}
Employee{id=13, name='Bob Marley', position='Manager', salary =80000.0}
Employee{id=14, name='Alice Cooper', position='Developer', salary =70000.0}
Employee{id=15, name='Bob Marley', position='Manager', salary =80000.0}
Employee deleted successfully. Rows affected:0
BUILD SUCCESSFUL (total time: 0 seconds)
```

DATABASE UPDATE



```
SELECT * FROM `employees`
```

Number of rows: 25

Sort by key: None

	id	name	position	salary
□ Edit Copy Delete	1	John Doe	Senior Software Engineer	90000.00
□ Edit Copy Delete	3	Steve Brown	Team Lead	85000.00
□ Edit Copy Delete	4	Alice Cooper	Developer	70000.00
□ Edit Copy Delete	5	Bob Marley	Manager	80000.00

Number of rows: 25

XML

Lab sheet 03 (xml)

Create an XML Document:

```
<?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>
<!--
```

Create a Java Class for XML Parsing:

```
package javaproject1;
import org.w3c.dom.*;
import javax.xml.parsers.*;
/**
 *
 * @author student
 */
public class xmlpharser {
   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\\Downloads\\Amandi\\javaproject1\\src\\javaproject1\\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("-----------");
 }
```

}

} catch (Exception e) {

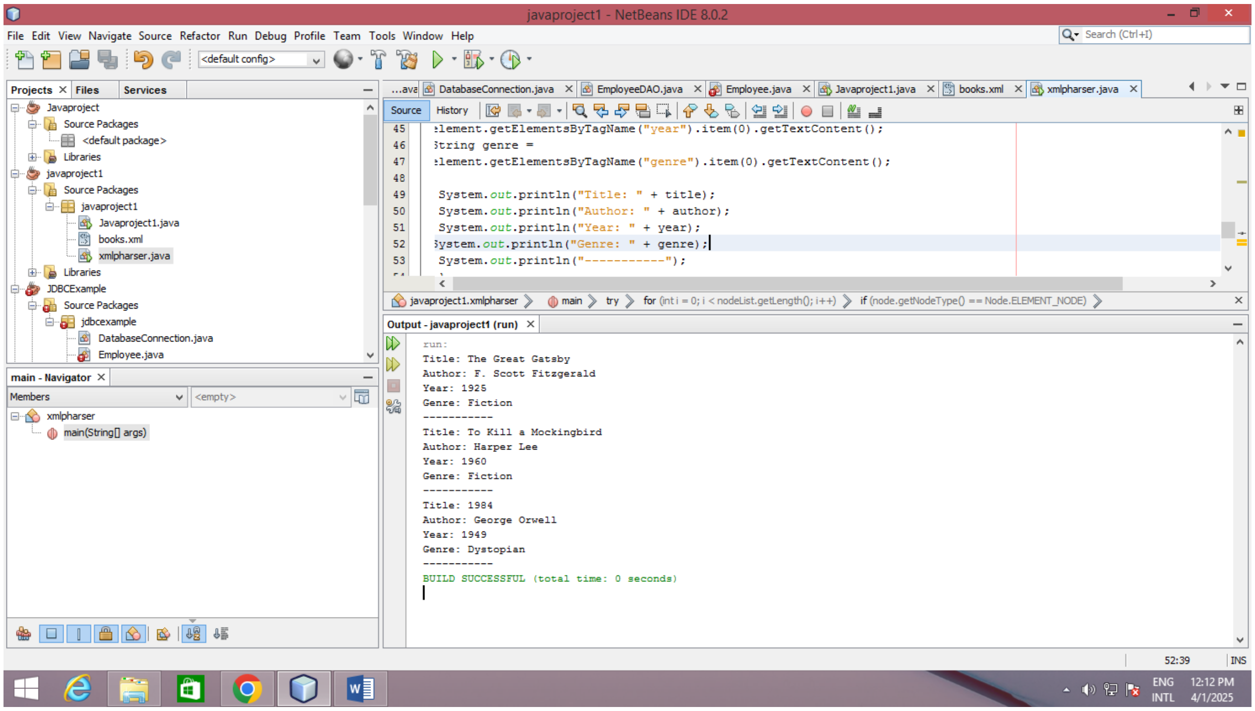
e.printStackTrace();

}

}

}

Output



```
ɛlement.getElementsByTagName("year").item(0).getTextContent();
String genre =
ɛlement.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("-----------");
```

```
run:
Title: The Great Gatsby
Author: F. Scott Fitzgerald
Year: 1925
Genre: Fiction
-----------
Title: To Kill a Mockingbird
Author: Harper Lee
Year: 1960
Genre: Fiction
-----------
Title: 1984
Author: George Orwell
Year: 1949
Genre: Dystopian
-----------
BUILD SUCCESSFUL (total time: 0 seconds)
```

Modifying XML Data

package javaproject1;

import java.io.File;

import org.w3c.dom.*;

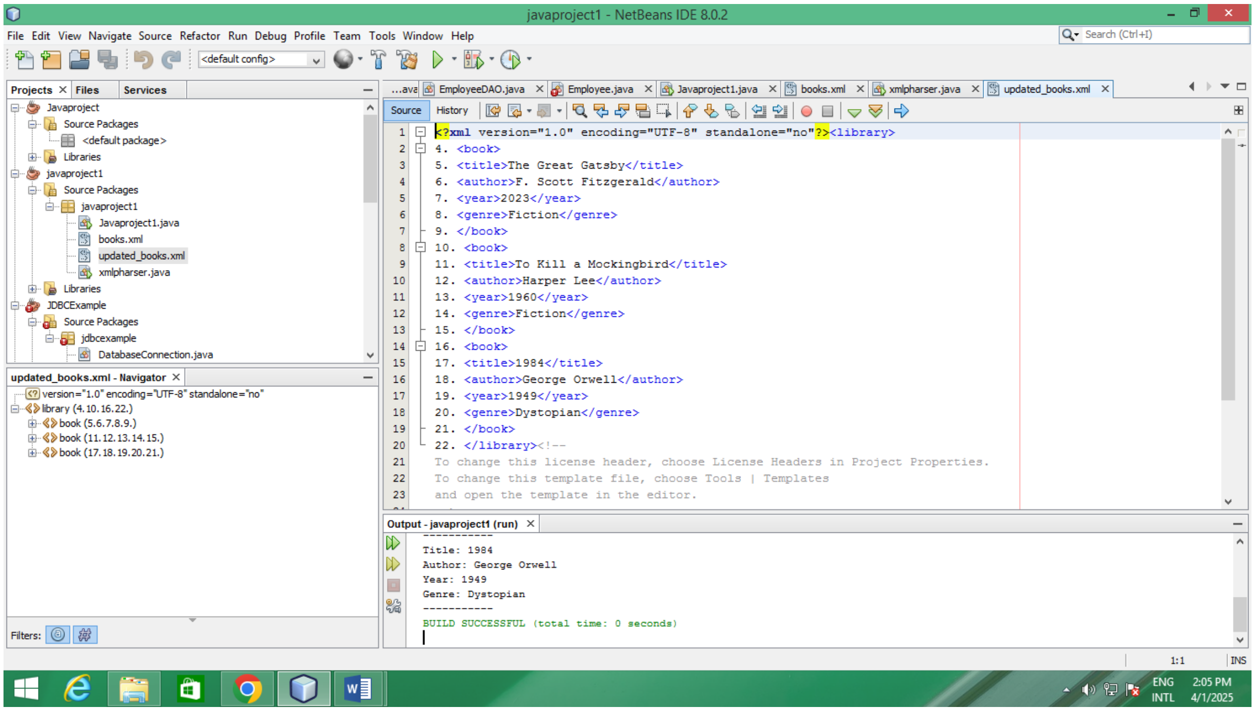
import javax.xml.parsers.*;

```
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 xmlpharser {

    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\\Downloads\\Amandi\\javaproject1\\src\\javaproject1\\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\\Downloads\\Amandi\\javaproject1\\src\\javaproject1\\updated_books.xml"))
;

 transformer.transform(source, result);


} catch (Exception e) {

 e.printStackTrace();

 }

    }

}
```

output



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><library>
4. <book>
5. <title>The Great Gatsby</title>
6. <author>F. Scott Fitzgerald</author>
7. <year>2023</year>
8. <genre>Fiction</genre>
9. </book>
10. <book>
11. <title>To Kill a Mockingbird</title>
12. <author>Harper Lee</author>
13. <year>1960</year>
14. <genre>Fiction</genre>
15. </book>
16. <book>
17. <title>1984</title>
18. <author>George Orwell</author>
19. <year>1949</year>
20. <genre>Dystopian</genre>
21. </book>
22. </library><!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
```

```
-----------
Title: 1984
Author: George Orwell
Year: 1949
Genre: Dystopian
-----------
BUILD SUCCESSFUL (total time: 0 seconds)
```

SERVLET

Java Servlet Practical Lab Sheet

implement basic and advanced concepts of Java Servlets.

Steps:

1. Create a Java Servlet (DisplayMessageServlet) that outputs a static message.

2. Configure the servlet using the @WebServlet annotation or the web.xml deployment descriptor.

Code(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>
        <div>TODO write content</div>

            <form action="getUserInput" method="POST">
Name: <input type="text" name="username" required><br>
<input type="submit" value="Submit">
</form>
 </body>
</html>
```

(DisplayMessageServlet.html)

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
 *
 * @author student
 */
@WebServlet(urlPatterns = {"/DisplayMessageServlet"})
public class DisplayMessageServlet 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)
            throws ServletException, IOException {
        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 DisplayMessageServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>welcome to java lab !"+ 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)
            throws ServletException, 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)
            throws ServletException, 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>

}

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

/**
 *
 * @author student
 */
```

```
@WebServlet(urlPatterns = {"/getUserInput"})
public class GetUserInputServlet 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)
            throws ServletException, IOException {
        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 GetUserInputServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet GetUserInputServlet 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)
            throws ServletException, 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)
            throws ServletException, IOException {

String username = request.getParameter("username");
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html><body>");
out.println("<h1>Hello, " + username + "!</h1>");
out.println("</body></html>");
        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>

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

/**
 *
 * @author student
 */
@WebServlet(urlPatterns = {"/getUserInput"})
```

```
public class GetUserInputServlet 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)
            throws ServletException, IOException {
        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 GetUserInputServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet GetUserInputServlet 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)
            throws ServletException, 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)
            throws ServletException, IOException {
            out.println("<h1>Servlet GetUserInputServlet 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)
            throws ServletException, 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)
            throws ServletException, IOException {
```

```
    /**
     * 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)
            throws ServletException, IOException {
        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 GetUserInputServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet GetUserInputServlet 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)
            throws ServletException, 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)
            throws ServletException, IOException {

String username = request.getParameter("username");
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html><body>");
out.println("<h1>Hello, " + username + "!</h1>");
out.println("</body></html>");
```

```
        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>

}
```

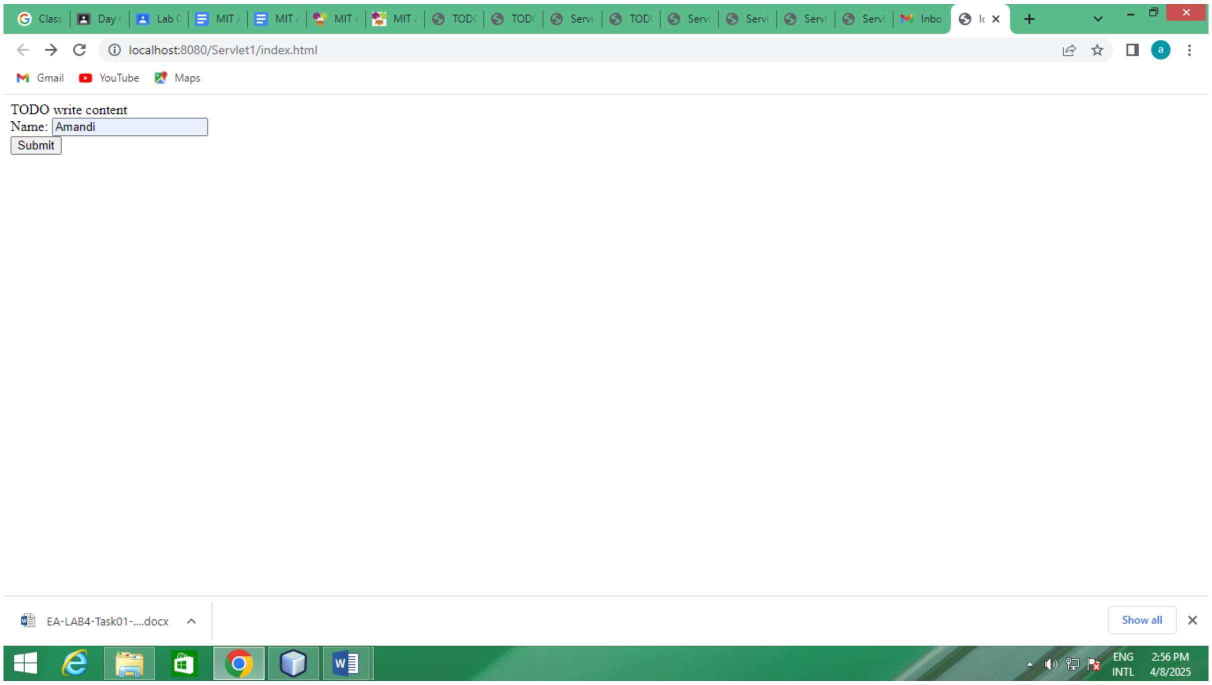
(GetUserInput.Java)

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

/**
 *
 * @author student
 */
@WebServlet(urlPatterns = {"/getUserInput"})
```

```
public class GetUserInputServlet 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)
            throws ServletException, IOException {
        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 GetUserInputServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet GetUserInputServlet 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.">
    /**
```

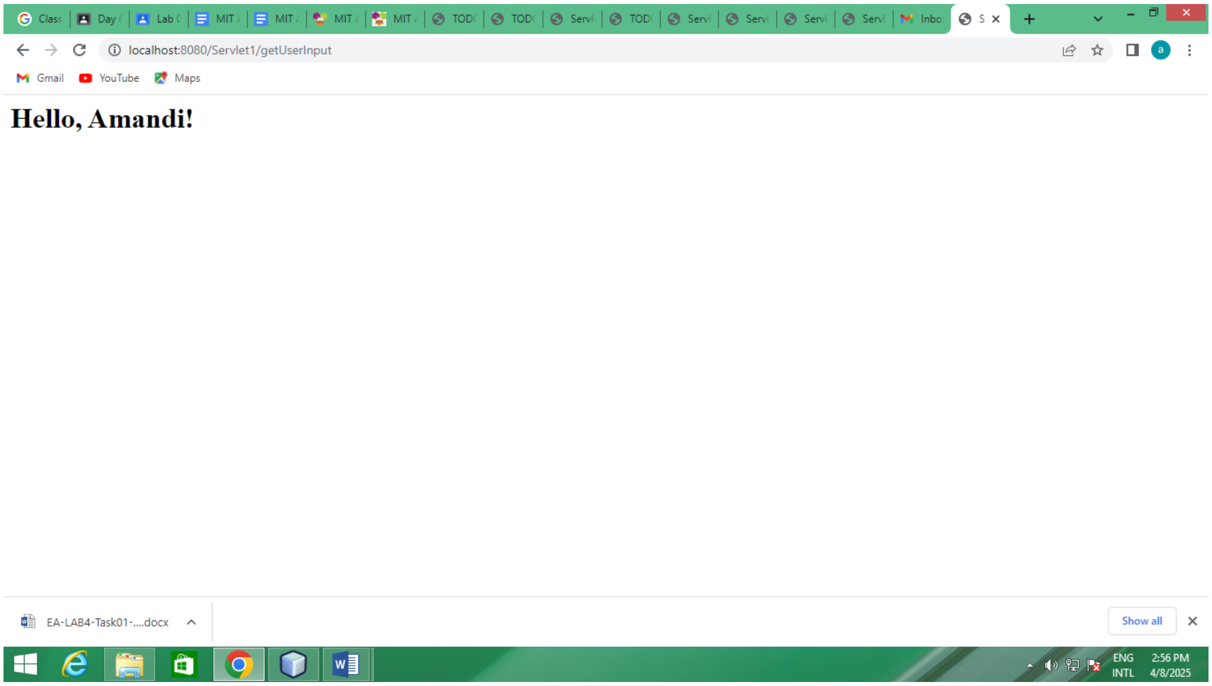
```
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest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

String username = request.getParameter("username");
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html><body>");
out.println("<h1>Hello, " + username + "!</h1>");
out.println("</body></html>");
        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>

}
```

OUTPUT



TODO write content

Name: Amandi



```
Hello, Amandi!
```

Lab Task 5: Display Data from Database on Another Web Page

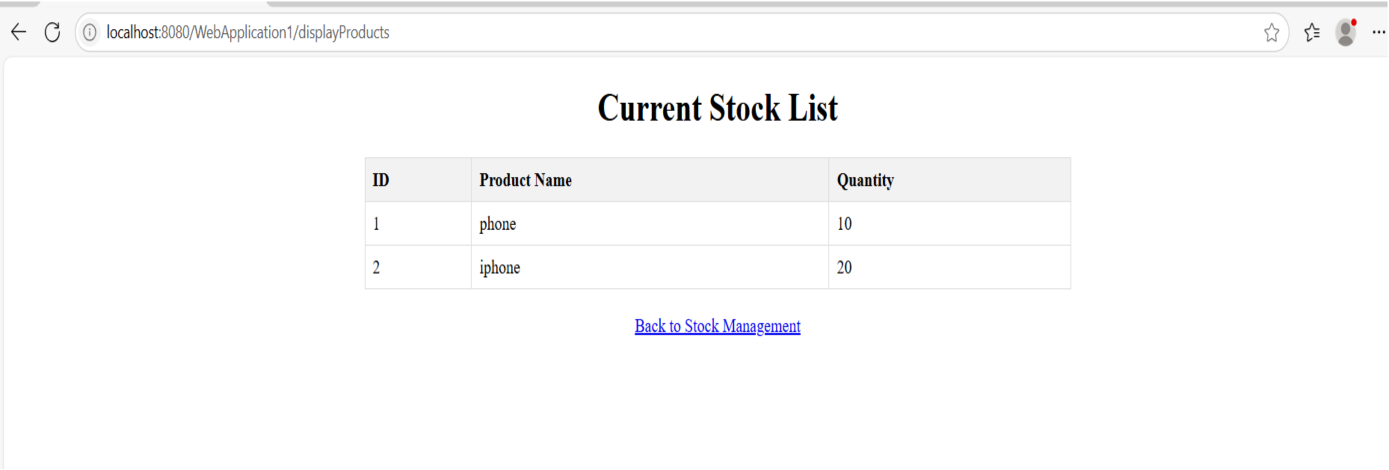
Servlet Code (DisplayProductsServlet.java):

```
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

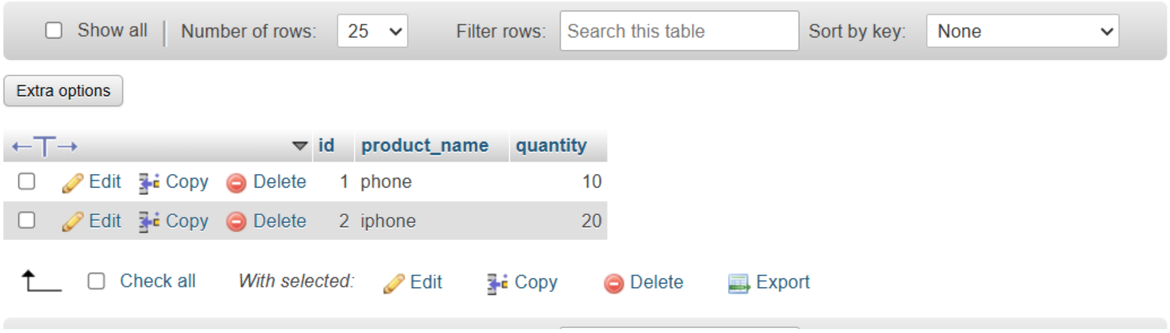


Current Stock List

ID	Product Name	Quantity
1	phone	10
2	iphone	20

Back to Stock Management

Database



Number of rows: 25

Filter rows: Search this table

Sort by key: None

	id	product_name	quantity
□ Edit Copy Delete	1	phone	10
□ Edit Copy Delete	2	iphone	20