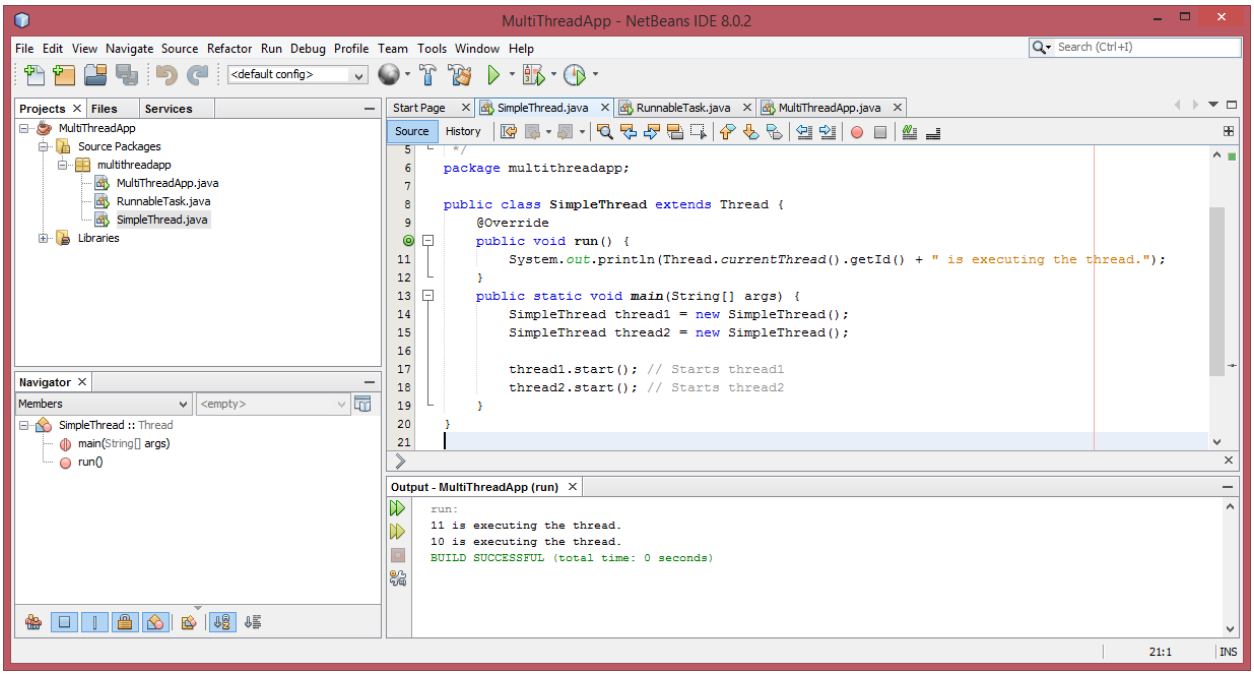
GAM/IT/2022/F/0037 - EA

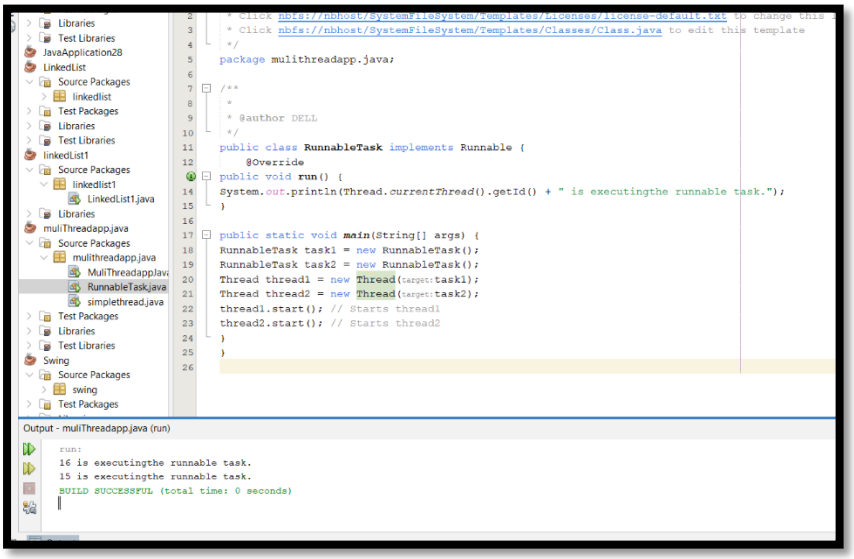
01.Create a Simple Thread Class

```
package multithreadapp;
public class SimpleThread extends Thread {
    @Override
    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
    }
}
```


MultiThreadApp - NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
Files
Services
MultiThreadApp
Source Packages
multithreadapp
MultiThreadApp.java
RunnableTask.java
SimpleThread.java
Libraries
Navigator
Members
<empty>
SimpleThread :: Thread
main(String[] args)
run()
Start Page
SimpleThread.java
RunnableTask.java
MultiThreadApp.java
Source
History
*/
package multithreadapp;
public class SimpleThread extends Thread {
@Override
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 - MultiThreadApp (run)
run:
11 is executing the thread.
10 is executing the thread.
BUILD SUCCESSFUL (total time: 0 seconds)
21:1
INS

1. Create a Runnable Class

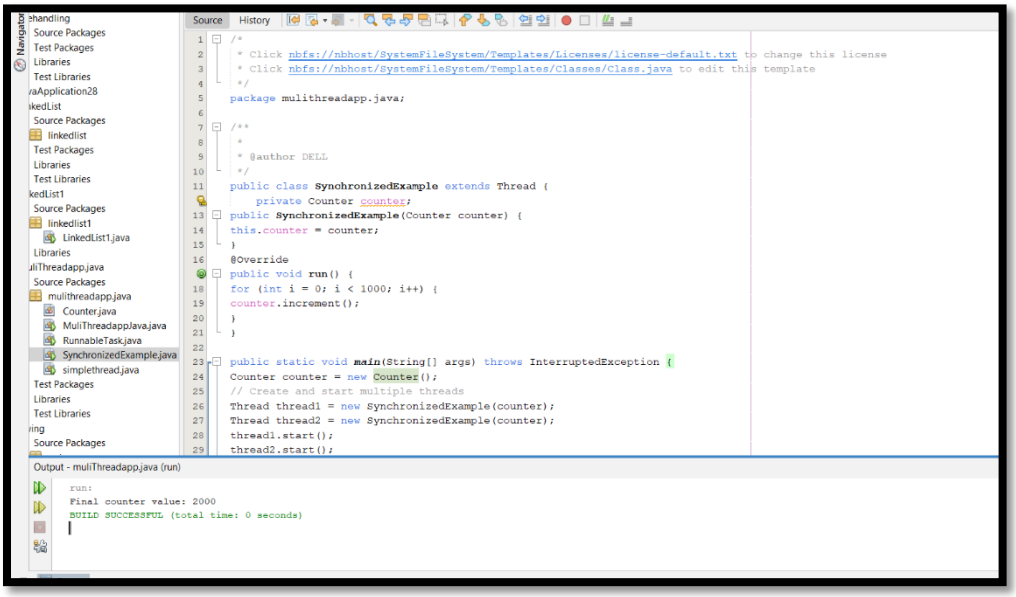
```
public class RunnableTask implements Runnable {
 @Override
public void run() {
System.out.println(Thread.currentThread().getId() + " is executingthe runnable task.");}
public static void main(String[] args) {
RunnableTask task1 = new RunnableTask();
RunnableTask task2 = new RunnableTask();
Thread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



3. Synchronizing Shared Resources

```
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);
```

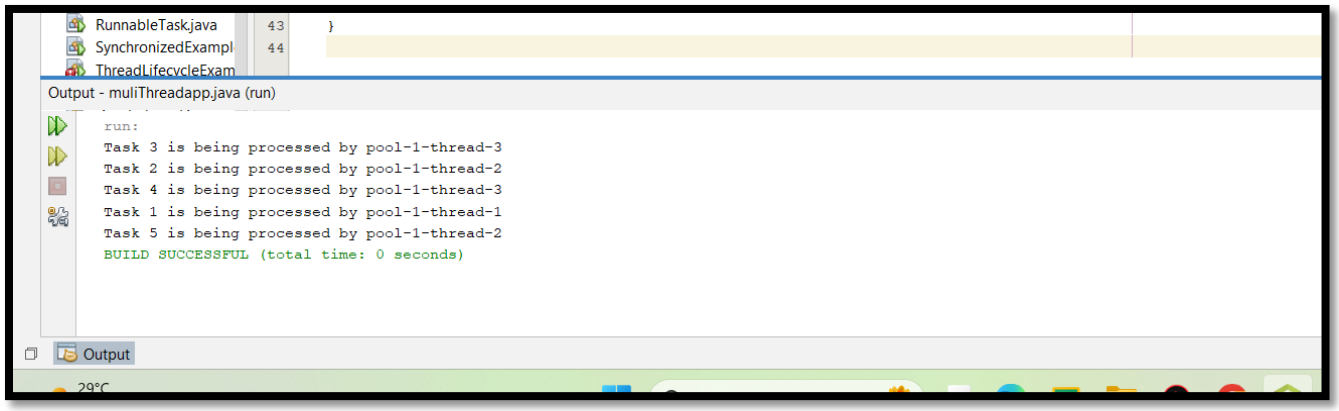
```
thread1.start();
thread2.start()
// Wait for threads to finish
thread1.join();
thread2.join();
System.out.println("Final counter value: " + counter.getCount());
}}
```



Part 4: Thread Pooling

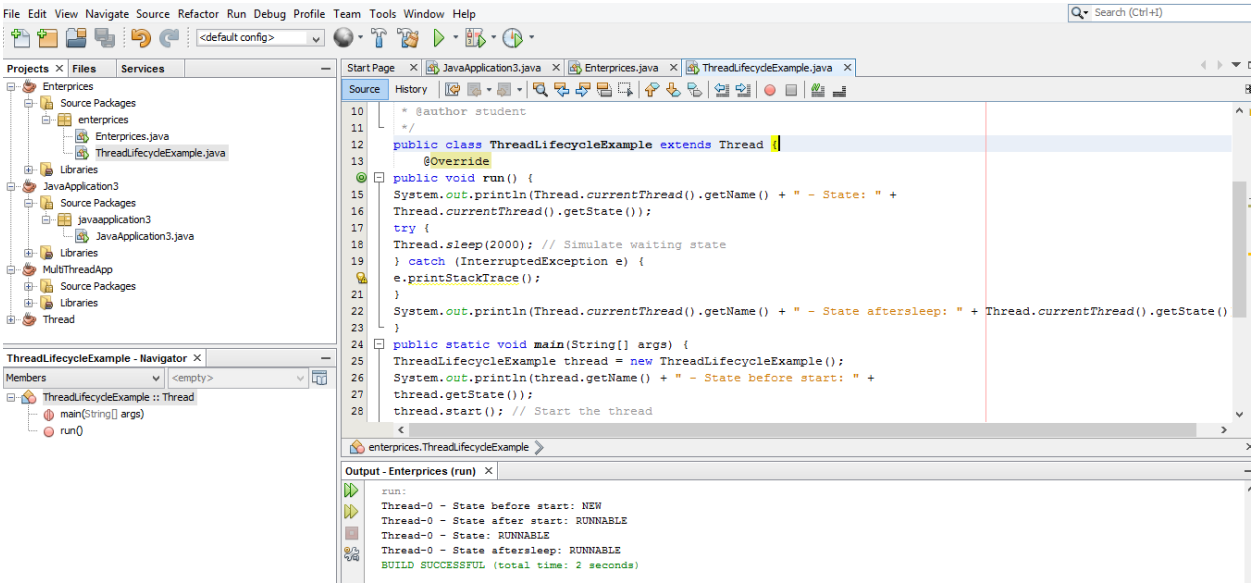
```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class ThreadPoolExample {

    // Task class implements Runnable
    static 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());
        }
    }

    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();
    }
}
```



Part 5: Thread Lifecycle and States

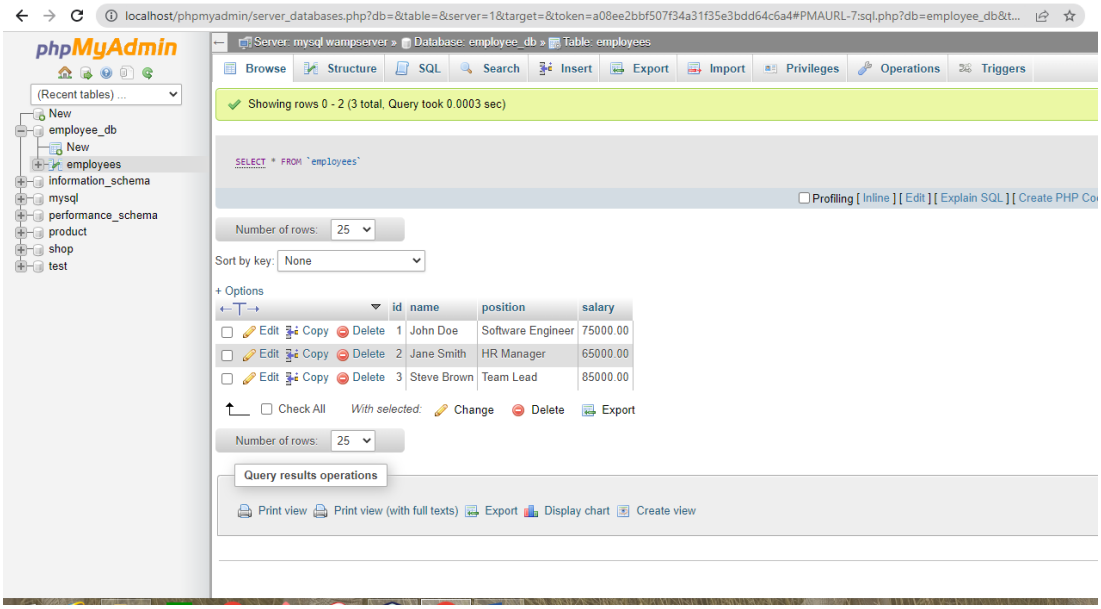
```
public class ThreadLifecycleExample extends Thread {
@Override
public void run() {
System.out.println(Thread.currentThread().getName() + " - State: " +
Thread.currentThread().getState());
try {
Thread.sleep(2000); // Simulate waiting state
} catch (InterruptedException e) {
e.printStackTrace();
}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


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
Files
Services
Enterprices
Source Packages
enterprices
Enterprices.java
ThreadLifecycleExample.java
Libraries
JavaApplication3
Source Packages
javaapplication3
JavaApplication3.java
Libraries
MultiThreadApp
Source Packages
Libraries
Thread
ThreadLifecycleExample - Navigator
Members
<empty>
ThreadLifecycleExample :: Thread
main(String[] args)
run()
Start Page
JavaApplication3.java
Enterprices.java
ThreadLifecycleExample.java
Source
History
* @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 aftersleep: " + 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
enterprices.ThreadLifecycleExample
Output - Enterprices (run)
run:
Thread-0 - State before start: NEW
Thread-0 - State after start: RUNNABLE
Thread-0 - State: RUNNABLE
Thread-0 - State aftersleep: RUNNABLE
BUILD SUCCESSFUL (total time: 2 seconds)

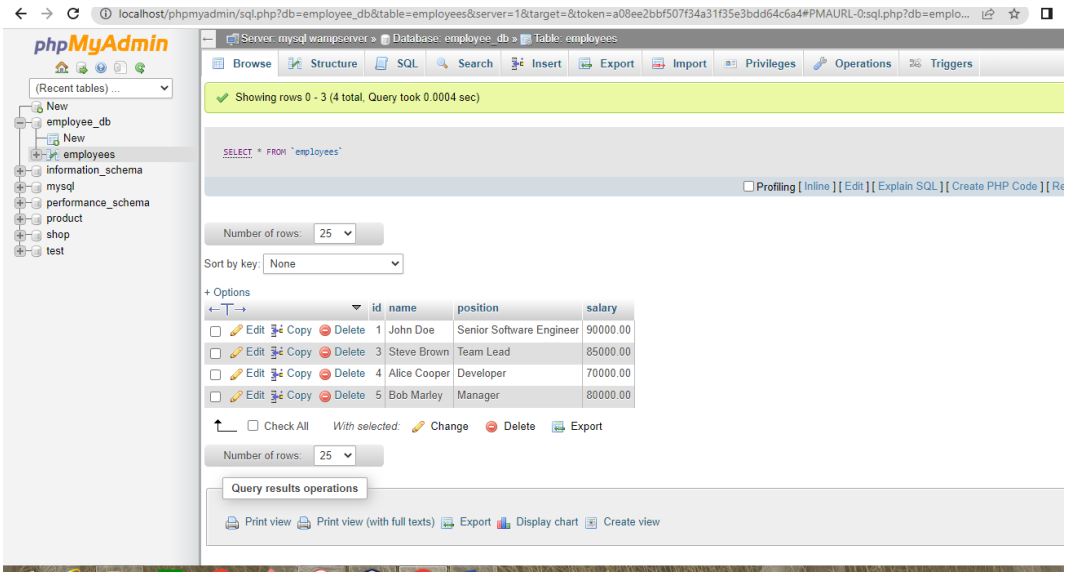
Lab sheet 03

```
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 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);
```

Before the execution



After the execution



DatabaseConnection.java

```
package jdbcexample;
```

```
import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

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 {

   public static Connection getConnection() throws SQLException {

      try {

         // Load the JDBC driver

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

         // Return the database connection

         return DriverManager.getConnection(URL, USER, PASSWORD);

      } catch (ClassNotFoundException | SQLException e) {

         System.out.println("Connection failed: " + e.getMessage());

         throw new SQLException("Failed to establish connection.");

      }
```

Employee.java

```
package jdbcexample;

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 + '}';
 }
}
```

EmployeeDAO.java

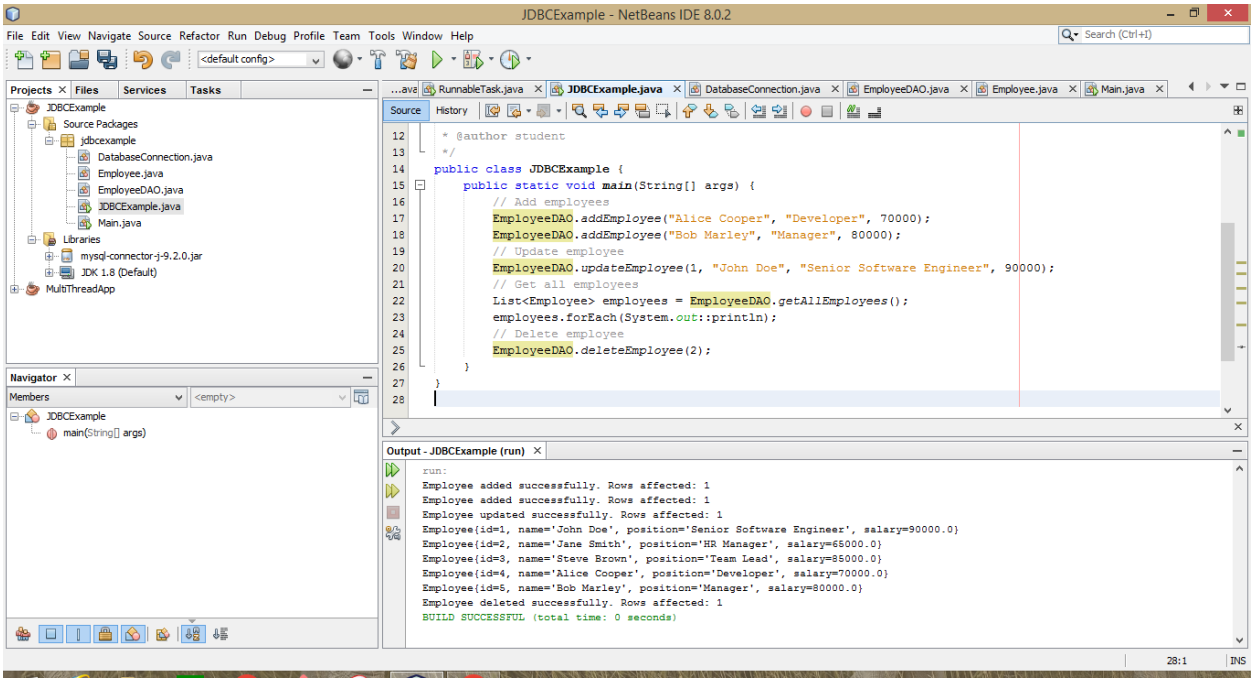
```
package jdbcexample;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
public class EmployeeDAO {
   // Create an employee
 public static void addEmployee(String name, String position, double salary) {
   String sql = "INSERT INTO employees (name, position, salary) VALUES (?, ?, ?)";
   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's information
 public static void updateEmployee(int id, String name, String position, double
salary) {
    String sql = "UPDATE employees SET name = ?, position = ?, salary = ?
WHERE id = ?";
    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();
    }
 }
}
```

Main.java

```
package jdbcexample;

import java.util.List;

public class Main {
    public static void main(String[] args) {
        // Add employees
        EmployeeDAO.addEmployee("Alice Cooper", "Developer", 70000);
        EmployeeDAO.addEmployee("Bob Marley", "Manager", 80000);
        // Update employee
        EmployeeDAO.updateEmployee(1, "John Doe", "Senior Software Engineer",
90000);
        // Get all employees
        List<Employee> employees = EmployeeDAO.getAllEmployees();
        employees.forEach(System.out::println);
        // Delete employee
        EmployeeDAO.deleteEmployee(2);
    }


}
```


JDBCExample - NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects Files Services Tasks
JDBCExample
Source Packages
jdbcexample
DatabaseConnection.java
Employee.java
EmployeeDAO.java
JDBCExample.java
Main.java
Libraries
mysql-connector-j-9.2.0.jar
JDK 1.8 (Default)
MultiThreadApp
Navigator
Members
JDBCExample
main(String[] args)
RunnableTask.java
JDBCExample.java
DatabaseConnection.java
EmployeeDAO.java
Employee.java
Main.java
Source History
* @author student
*/
public class JDBCExample {
public static void main(String[] args) {
// Add employees
EmployeeDAO.addEmployee("Alice Cooper", "Developer", 70000);
EmployeeDAO.addEmployee("Bob Marley", "Manager", 80000);
// Update employee
EmployeeDAO.updateEmployee(1, "John Doe", "Senior Software Engineer", 90000);
// Get all employees
List<Employee> employees = EmployeeDAO.getAllEmployees();
employees.forEach(System.out::println);
// Delete employee
EmployeeDAO.deleteEmployee(2);
}
}
Output - JDBCExample (run)
run:
Employee added successfully. Rows affected: 1
Employee added successfully. Rows affected: 1
Employee updated successfully. Rows affected: 1
Employee{id=1, name='John Doe', position='Senior Software Engineer', salary=90000.0}
Employee{id=2, name='Jane Smith', position='HR Manager', salary=65000.0}
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 deleted successfully. Rows affected: 1
BUILD SUCCESSFUL (total time: 0 seconds)
28:1
INS

Part 2: Creating Your First XML Document

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>
```

This XML file does not appear to have any style information associated with it. The document tree is shown below.

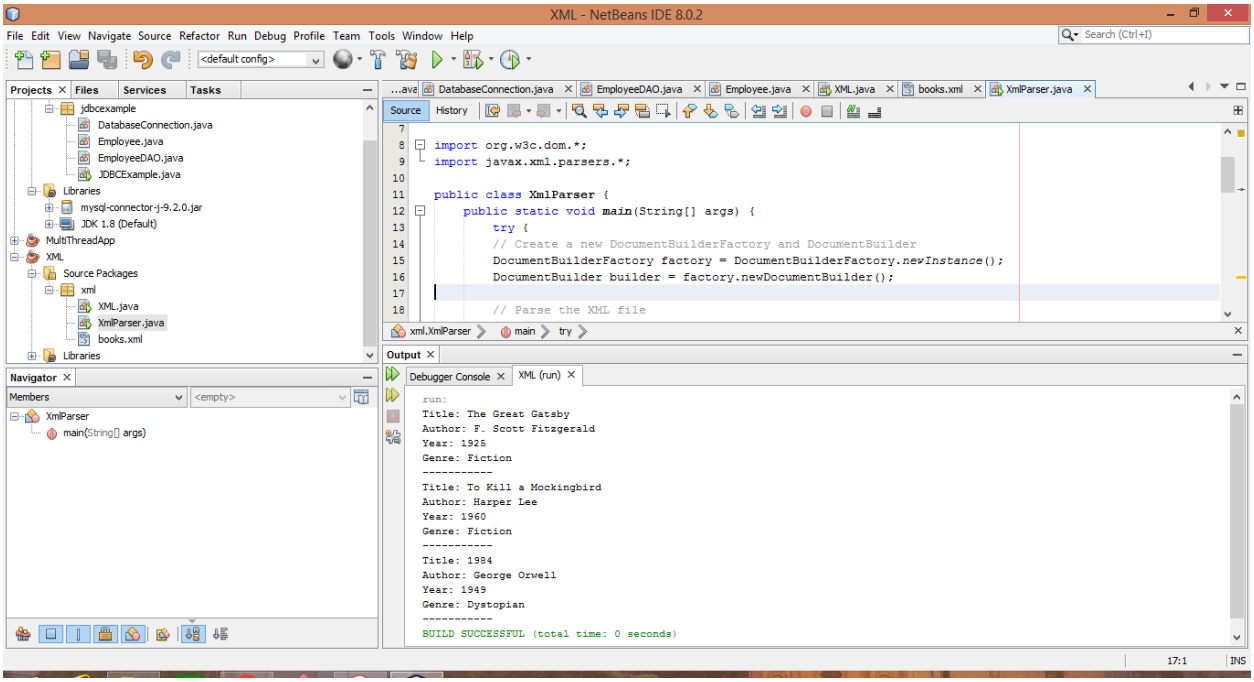
```
<library>
  <book>
    <title>The Great Gatsby</title>
    <author>F. Scott Fitzgerald</author>
    <year>1925</year>
    <genre>Fiction</genre>
  </book>
  <book>
    <title>To Kill a Mockingbird</title>
    <author>Harper Lee</author>
    <year>1960</year>
    <genre>Fiction</genre>
  </book>
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

Part 3: Parsing XML in Java

XmlParser.java

```
package xml;
import org.w3c.dom.*;
import javax.xml.parsers.*;
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\\Desktop\\EA\\XML\\src\\xml\\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();
    }
  }
}
```


XML - NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects Files Services Tasks
jdbcexample
DatabaseConnection.java
Employee.java
EmployeeDAO.java
JDBCExample.java
Libraries
mysql-connector-j-9.2.0.jar
JDK 1.8 (Default)
MultiThreadApp
XML
Source Packages
xml
XML.java
XmlParser.java
books.xml
Libraries
Navigator
Members
<empty>
XmlParser
main(String[] args)
DatabaseConnection.java
EmployeeDAO.java
Employee.java
XML.java
books.xml
XmlParser.java
Source History
7 import org.w3c.dom.*;
8 import javax.xml.parsers.*;
9
10 public class XmlParser {
11 public static void main(String[] args) {
12 try {
13 // Create a new DocumentBuilderFactory and DocumentBuilder
14 DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
15 DocumentBuilder builder = factory.newDocumentBuilder();
16
17 // Parse the XML file
18
xml.XmlParser main try
Output
Debugger Console XML (run)
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)
17:1
INS

Part 4: Modifying XML Data

XmlParser.java

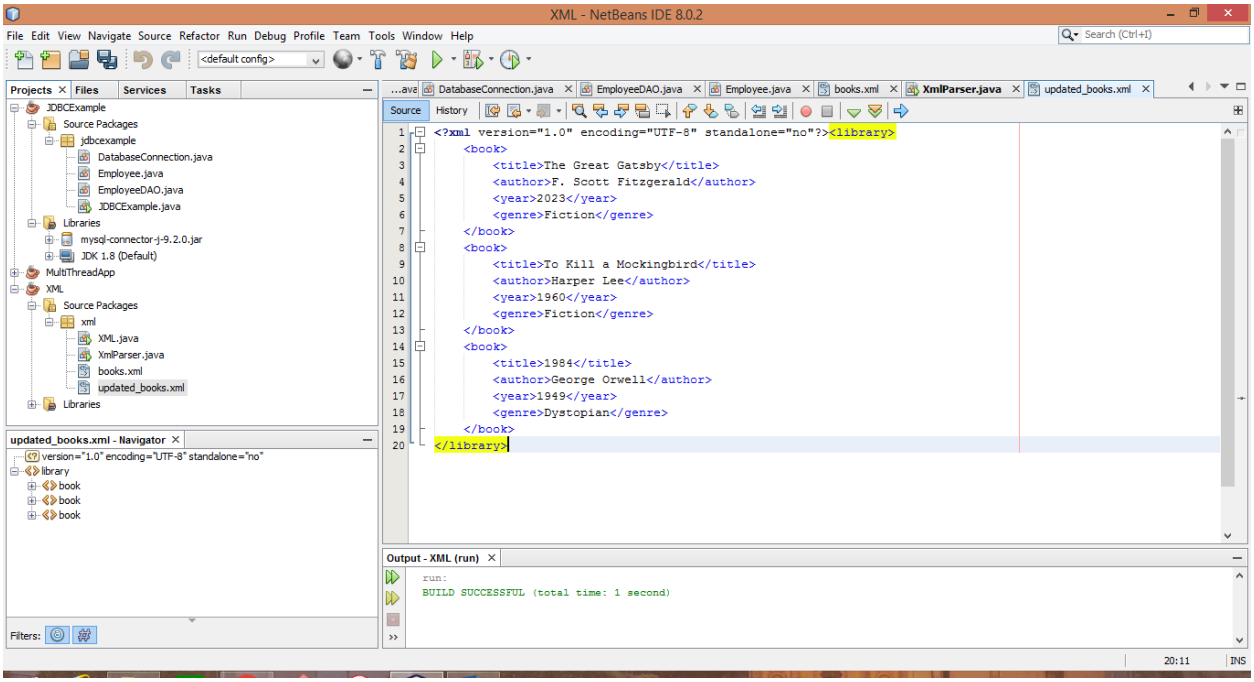
```
package xml;
import java.io.File;
import org.w3c.dom.*;
import javax.xml.parsers.*;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
public class XmlParser {
    public 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\\Desktop\\EA\\XML\\src\\xml\\books.xml");
        // Normalize the document
        document.getDocumentElement().normalize();
        // Get the root element (library)
        NodeList nodeList = document.getElementsByTagName("book");
        // 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\\Desktop\\EA\\XML\\src\\xml\\updated_books.xml"));
        transformer.transform(source, result);
        } catch (Exception e) {
                e.printStackTrace();
        }
    }
```

updated_books.xml

```
<?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>
```

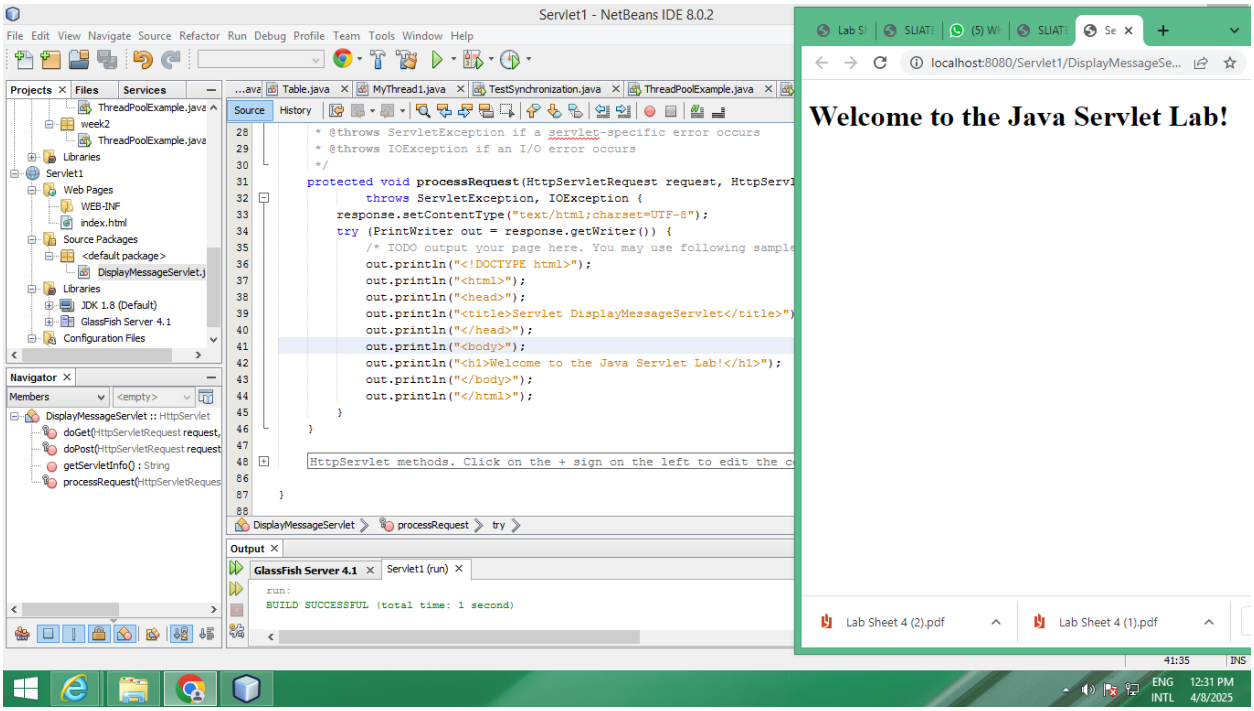
```
        <title>1984</title>
        <author>George Orwell</author>
        <year>1949</year>
        <genre>Dystopian</genre>
    </book>
</library>
```


XML - NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects Files Services Tasks
JDBCExample
Source Packages
jdbcexample
DatabaseConnection.java
Employee.java
EmployeeDAO.java
JDBCExample.java
Libraries
mysql-connector-j-9.2.0.jar
JDK 1.8 (Default)
MultiThreadApp
XML
Source Packages
xml
XML.java
XmlParser.java
books.xml
updated_books.xml
Libraries
updated_books.xml - Navigator
version="1.0" encoding="UTF-8" standalone="no"
library
book
book
book
Filters:
DatabaseConnection.java
EmployeeDAO.java
Employee.java
books.xml
XmlParser.java
updated_books.xml
Source
History
<?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 - XML (run)
run:
BUILD SUCCESSFUL (total time: 1 second)
20:11
INS

Lab Task 1: Simple Servlet - Display Static Message

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

@WebServlet(urlPatterns = {"/DisplayMessageServlet"})
public class DisplayMessageServlet extends HttpServlet {

    protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            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 the Java Servlet Lab!</h1>");
```

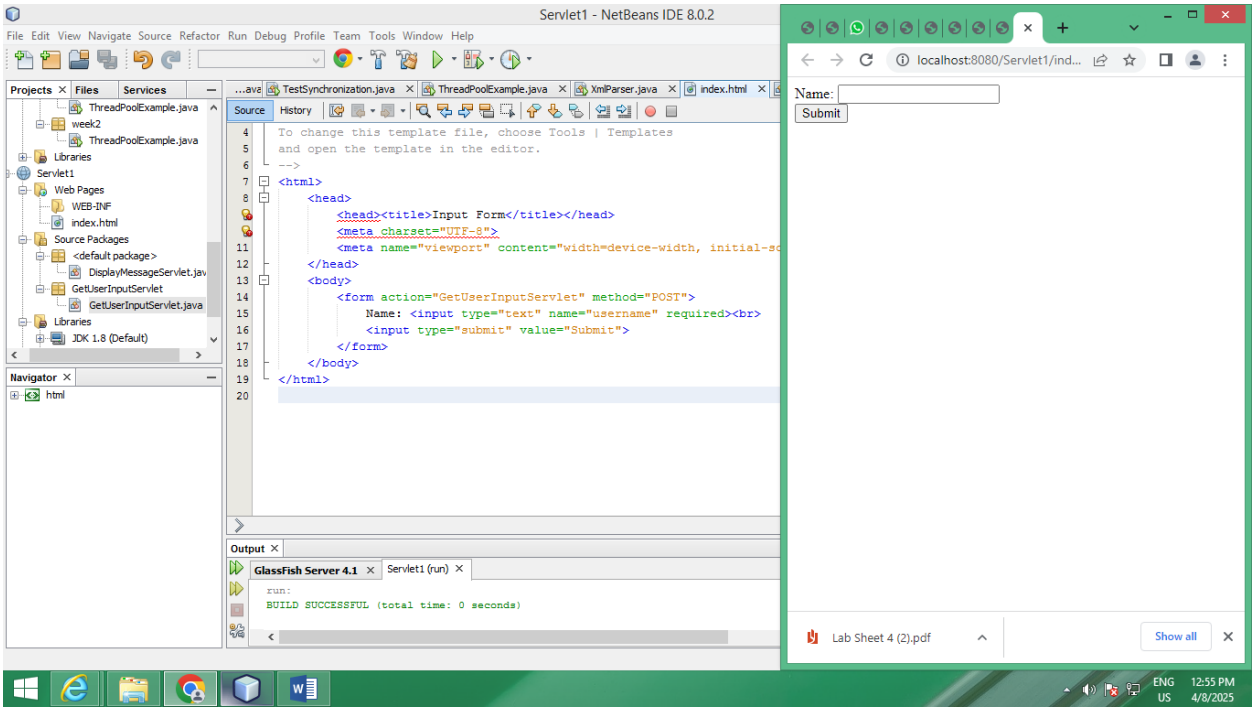
```
            out.println("</body>");
            out.println("</html>");
        }
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    @Override
    public String getServletInfo() {
        return "Short description";
    }

}
```


Servlet1 - NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Files Services
Servlet1
Web Pages
WEB-INF
index.html
Source Packages
<default package>
DisplayMessageServlet.j
Libraries
JDK 1.8 (Default)
GlassFish Server 4.1
Configuration Files
Navigator
DisplayMessageServlet :: HttpServlet
doGet(HttpServletRequest request,
doPost(HttpServletRequest request
getServletInfo() : String
processRequest(HttpServletReques
Table.java
MyThread1.java
TestSynchronization.java
ThreadPoolExample.java
Source History
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
protected void processRequest(HttpServletRequest request, HttpServl
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
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 the Java Servlet Lab!</h1>");
out.println("</body>");
out.println("</html>");
}
}
HttpServlet methods. Click on the + sign on the left to edit the c
}
DisplayMessageServlet processRequest try
Output
GlassFish Server 4.1
Servlet1 (run)
run:
BUILD SUCCESSFUL (total time: 1 second)
41:35
INS
localhost:8080/Servlet1/DisplayMessageSe...
Welcome to the Java Servlet Lab!
Lab Sheet 4 (2).pdf
Lab Sheet 4 (1).pdf
ENG
INTL
12:31 PM
4/8/2025

Lab Task 2: Get User Input from Form and Display

- Index.html

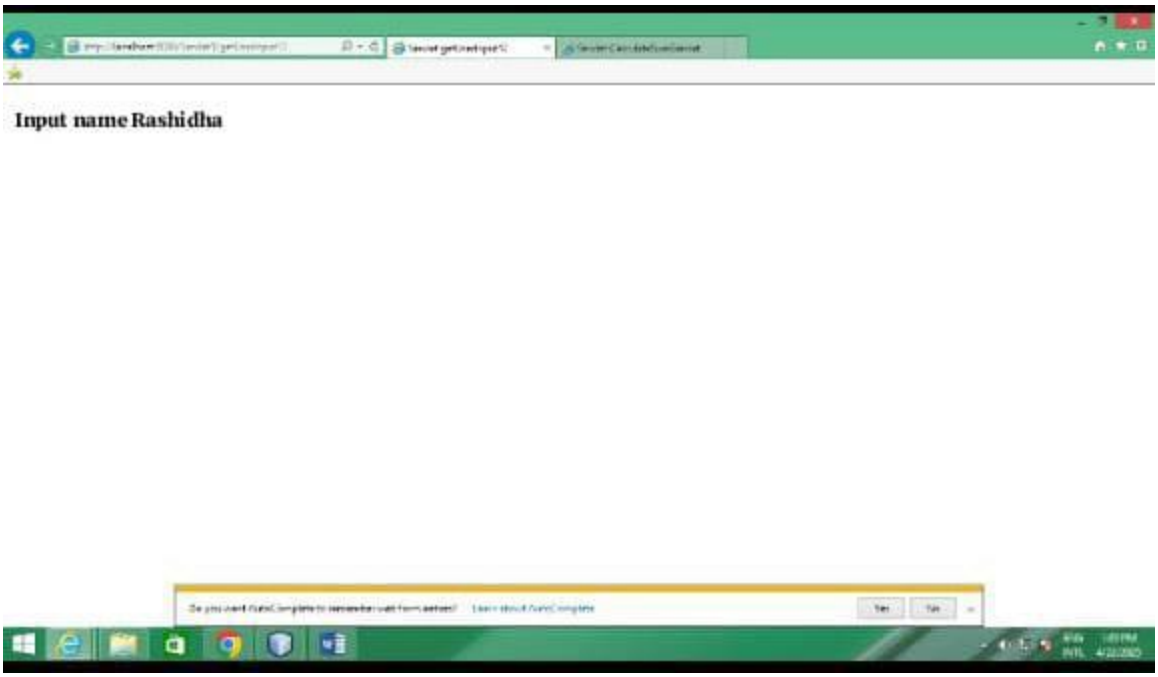
```
<html>
    <head>
        <head><title>Input Form</title></head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    </head>
    <body>
        <form action="GetUserInputServlet" method="POST">
            Name: <input type="text" name="username" required><br>
            <input type="submit" value="Submit">
        </form>
    </body>
</html>
```



- **Servlet Code (GetUserInputServlet.java)**

```
package GetUserInputServlet;

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


@WebServlet(name = "GetUserInputServlet", urlPatterns =
{"/GetUserInputServlet"})
public class GetUserInputServlet extends HttpServlet {


   protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
          throws ServletException, IOException {
       response.setContentType("text/html;charset=UTF-8");
       String username = request.getParameter("username");
       response.setContentType("text/html");

       try (PrintWriter out = response.getWriter()) {

          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>Hello, " + username + "!</h1>");
            out.println("</body>");
            out.println("</html>");
        }
    }


    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

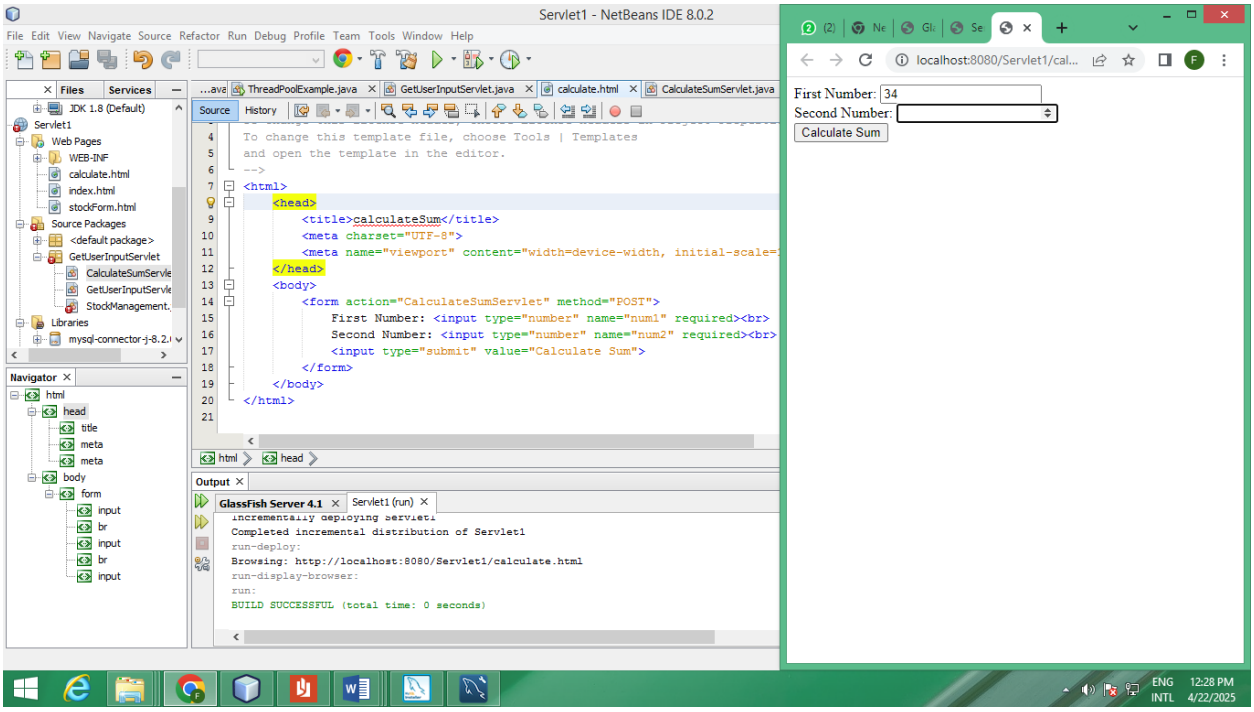


Lab Task 3: Get Multiple Inputs, Perform Calculation, and Display Result

HTML Form (calculate.html)

```
<html>
    <head>
        <title>calculateSum</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <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>
</body>
</html>
```



Servlet Code (CalculateSumServlet.java)

```
package GetUserInputServlet;

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



@WebServlet(name = "CalculateSumServlet", urlPatterns =
{"/CalculateSumServlet"})
public class CalculateSumServlet extends HttpServlet {



   protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
           throws ServletException, IOException {
       int num1 = Integer.parseInt(request.getParameter("num1"));
       int num2 = Integer.parseInt(request.getParameter("num2"));
       int sum = num1 + num2;


       response.setContentType("text/html");
```

```
        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>The sum of " + num1 + " and " + num2 + " is: " + sum + "</h1>");
            out.println("</body>");
            out.println("</html>");
        }
    }
```