



**MALAD KANDIVALI EDUCATION SOCIETY'S
NAGINDAS KHANDWALA COLLEGE OF COMMERCE,
ARTS & MANAGEMENT STUDIES & SHANTABEN NAGINDAS
KHANDWALA COLLEGE OF SCIENCE
MALAD [W], MUMBAI – 64
(AUTONOMOUS)**

**(Reaccredited 'A' Grade by NAAC)
(AFFILIATED TO UNIVERSITY OF MUMBAI)
(ISO 9001:2015)**

CERTIFICATE

Name: Mr./Ms. Shibu Mohapatra

Roll No: 2 Programme: MSc CS (Artificial Intelligence) Semester: II

This is certified to be a bonafide record of practical works done by the above student in the college laboratory for the course **ADVANCED JAVA PROGRAMMING** (Course Code: **2127PSAIPR**) for the partial fulfillment of Second Semester of MSc AI during the academic year 2021-2022.

The journal work is the original study work that has been duly approved in the year 2021-2022 by the undersigned.

External Examiner

**(Subject-In-Charge)
Dr. Pragati Hiwarkar**

Date of Examination:

(College Stamp)

Subject: Advanced Java Programming

Branch: MSc CS (Artificial Intelligence)

List of Experiments

Sr. No.	Date	Aim	Sign
SERVLET			
1.	05/02/22	Develop dynamic web application to display current system date and time using servlets.	
2	08/02/22	Develop dynamic web application to display login page with proper HTML UI elements using servlets. OR Implement a JAVA Servlet Program to implement a dynamic HTML using Servlet (user name and password should be accepted using HTML and displayed using a Servlet).	
3	08/02/22	Implement a servlet to authenticate login details, which is created previously (user name and password should be accepted using HTML and displayed using a Servlet)	
4	11/02/22	Develop dynamic web application to manage product (prodId, name, category, price) details using servlets. This app must have following pages <ol style="list-style-type: none">Home pageProduct adding pageProduct editing pageProduct displaying page	
5	15/02/22	Design a JAVA Servlet Program to Download a file and display it on the screen (A link has to be provided in HTML, when the link is clicked corresponding file has to be displayed on Screen).	
6	18/02/22	a) Design a JAVA Servlet Program to implement Request Dispatcher object using include() and forward() methods. b) Implement a JAVA Servlet Program to implement sessions using HTTP Session Interface.	
JSP			
1	21/02/22	Develop dynamic web application to manage user (userId, name, dob, address) details using JSP. This app must have following pages <ol style="list-style-type: none">Home pageUser adding pageUser editing pageUser displaying page	

2	22/02/22	Write JSP program to implement custom tag with name <product>, which display product (prodId, name, category, price) details	
3	25/02/22	Design a JAVA JSP Program to implement verification of a particular user login and display a welcome page	
4	28/02/22	Write a program to design a simple calculator using JSP	

JSF

1	03/03/22	Write a JSF application to print Hello World	
2	05/03/22	Write a JSF application for Celsius to Fahrenheit convertor	
3	11/03/22	Design following JSF application to accept username in textbox. if user textbox is empty show validation message	
4	15/03/22	A web application that takes a name as input and on submit it shows a hello <name> page where name is taken from the request. It shows the start time at the right top corner of the page and provides a logout button. On clicking this button, it should show a logout page with Thank You <name > message with the duration of usage (hint: Use session to store name and time).	

HIBERNATE

1	17/03/22	Creating hibernate example to print “Hello Word”	
2	28/03/22	A) Create Maven Project for hibernate to add dependencies B) Configure hibernate program	
3	29/03/22	Write hibernate program to save student data using annotation	
4	29/03/22	Write hibernate program to save image in database	

SPRING

1	08/04/22	Create new Maven Project to add Spring Dependencies using setter injection	
2	09/04/22	Create new Maven Project for property injection using p Schema and using value as attribute	
3	11/04/22	Create project to implement injecting collection types: List, Set Map and Properties	

4	12/04/22	Create project to implement constructor injection	
5	13/04/22	Write a Spring program for implementing Bean lifecycle method using XML	
6	18/04/22	Create Application for Spring JDBC concept (insert)	
7	19/04/22	Create Application for Spring MVC	

SERVLET

1. Write a servlet application to print the current date and time.

Code:**DateSrv.java**

```
package com.srccodes.example;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.*;
import java.io.*;
import java.util.*;

/**
 * Servlet implementation class DateSrv
 */
@SuppressWarnings("serial")
@WebServlet("/DateSrv")
public class DateSrv extends GenericServlet
{
    //implement service()
    public void service(ServletRequest req, ServletResponse res) throws IOException,
    ServletException
    {
        //set response content type
        res.setContentType("text/html");

        //get stream obj
        PrintWriter pw = res.getWriter();

        //write req processing logic
        java.util.Date date = new java.util.Date();
        pw.println("<h2> Shibu Mohapatra <h2>");
        pw.println("<h2>"+"Display current Date & Time: <h2>\n" + date.toString());

        pw.close();
    }
}
```

Name: **Shibu Mohapatra**

Branch: **MSC AI**

Roll no: **02**

```
}
```

Output:



Shibu Mohapatra

Display current Date & Time:

Tue Apr 26 12:47:24 IST 2022



2. Develop dynamic web application to display login page with proper HTML UI elements using servlets.

OR

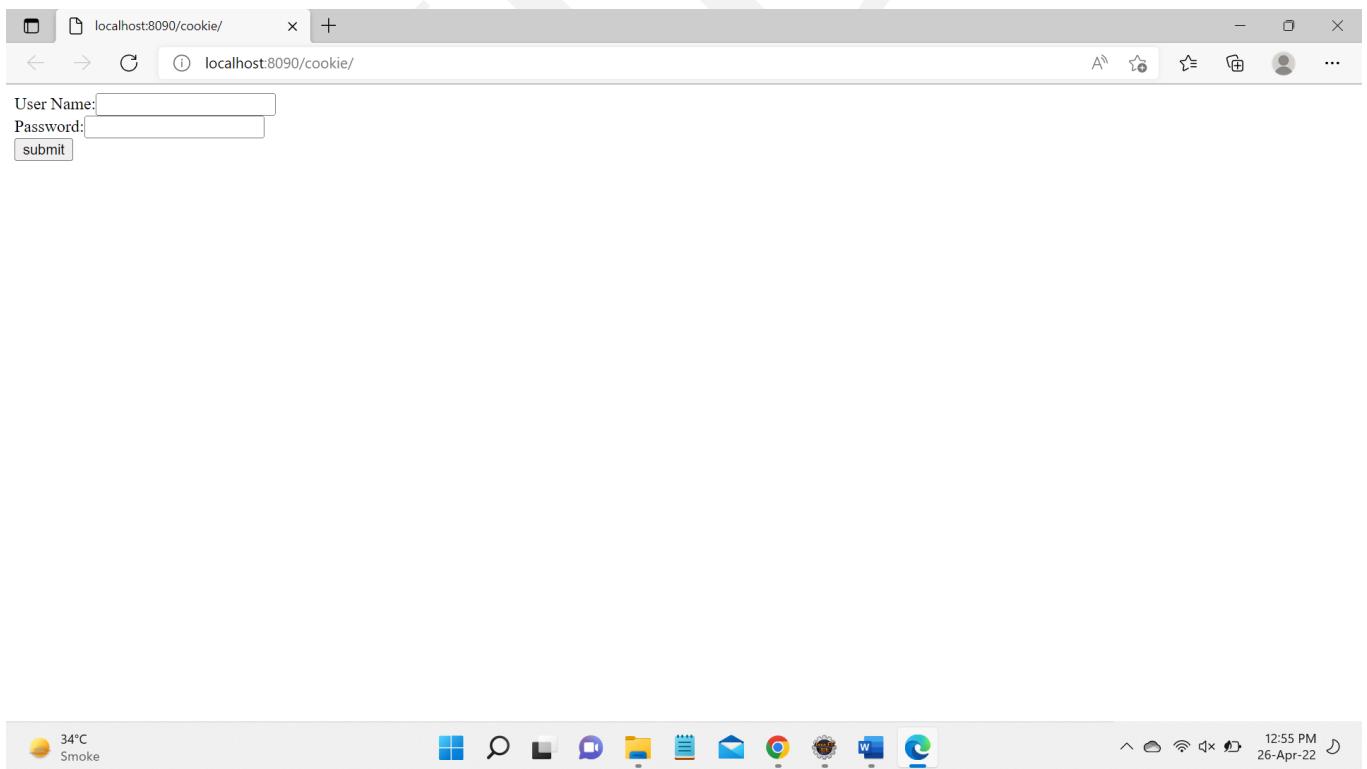
Implement a JAVA Servlet Program to implement a dynamic HTML using Servlet (user name and password should be accepted using HTML and displayed using a Servlet).

Code:

Index.html –

```
<form action="login">  
User Name:<input type="text" name="userName"/><br/>  
Password:<input type="password" name="userPassword"/><br/>  
<input type="submit" value="submit"/>  
</form>
```

Output:



Code:**MyServlet1.java –**

```
import java.io.PrintWriter;

import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@SuppressWarnings("serial")
public class MyServlet1 extends HttpServlet
{
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response) {
        try{
            response.setContentType("text/html");
            PrintWriter pwriter = response.getWriter();

            String name = request.getParameter("userName");
            String password = request.getParameter("userPassword");
            pwriter.print("Hello "+name);
            pwriter.print(" Your Password is: "+password);

            //Creating two cookies
            Cookie c0=new Cookie("userName",name);
            Cookie c1=new Cookie("userPassword",password);

            //Adding the cookies to response header
            response.addCookie(c0);
            response.addCookie(c1);
            pwriter.print("<br><a href='welcome'>View Details</a>");
            pwriter.close();
        }catch(Exception exp){
            System.out.println(exp);
        }
    }
}
```

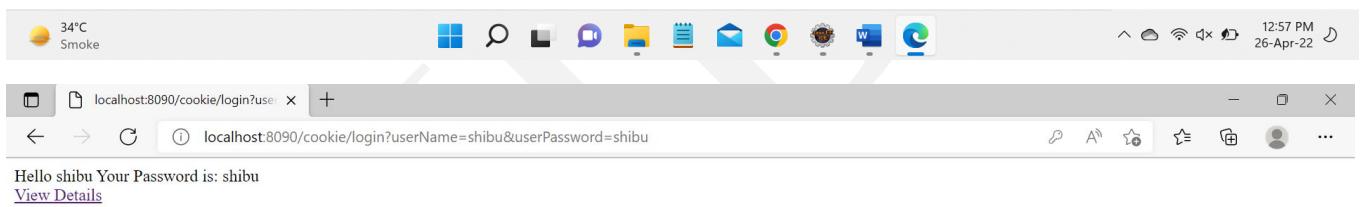
Name: **Shibu Mohapatra**

Branch: **MSC AI**

Roll no: **02**

Output:

A screenshot of a Microsoft Edge browser window. The address bar shows "localhost:8090/cookie/". The page contains a form with fields for "User Name" (shibu) and "Password" (****), and a "submit" button.



- 3. Implement a servlet to authenticate login details, which is created previously (user name and password should be accepted using HTML and displayed using a Servlet)**

Code:

MyServlet2.java –

```
import java.io.PrintWriter;

import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@SuppressWarnings("serial")
public class MyServlet2 extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response){
        try{
            response.setContentType("text/html");
            PrintWriter pwriter = response.getWriter();

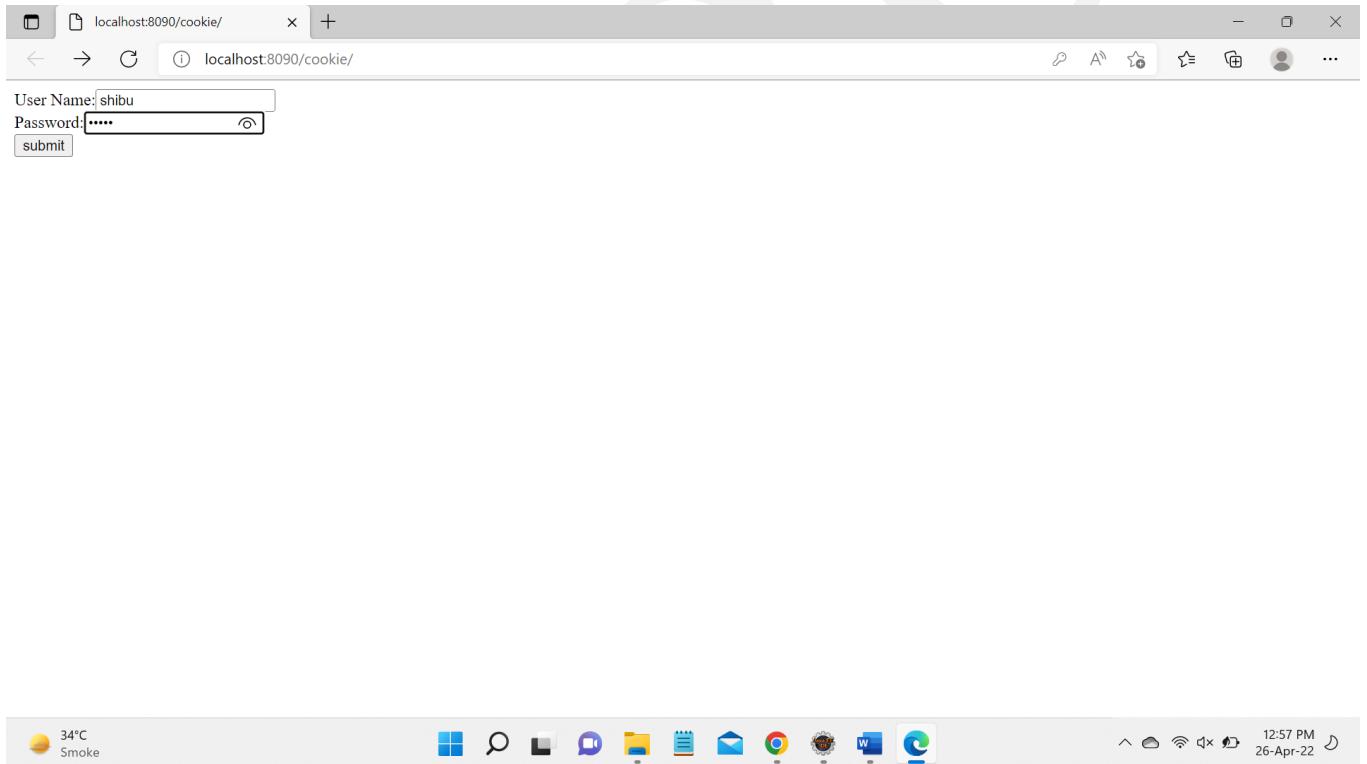
            //Reading cookies
            Cookie c[] = request.getCookies();
            //Displaying User name value from cookie
            pwriter.print("Name: " + c[0].getValue());
            //Displaying user password value from cookie
            pwriter.print("Password: " + c[1].getValue());

            pwriter.close();
        }catch(Exception exp){
            System.out.println(exp);
        }
    }
}
```

Web.xml

```
<web-app>
    <display-name>cookie</display-name>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>
    <servlet>
        <servlet-name>Servlet1</servlet-name>
        <servlet-class>MyServlet1</servlet-class>
    </servlet>
```

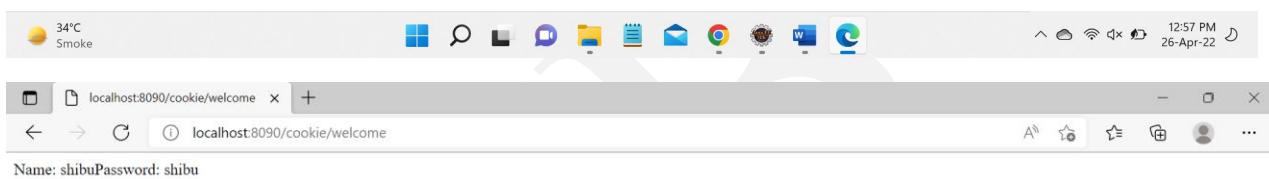
```
<servlet-mapping>
    <servlet-name>Servlet1</servlet-name>
    <url-pattern>/login</url-pattern>
</servlet-mapping>
<servlet>
    <servlet-name>Servlet2</servlet-name>
    <servlet-class>MyServlet2</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>Servlet2</servlet-name>
    <url-pattern>/welcome</url-pattern>
</servlet-mapping>
</web-app>
```

Output:

Name: **Shibu Mohapatra**

Branch: **MSC AI**

Roll no: **02**



4. Develop dynamic web application to manage product (prodId, name, category, price) details using servlets. This app must have following pages

- a) Home page
- b) Product adding page
- c) Product editing page
- d) Product displaying page

Code:

Servlet:

ProductController.java:

```
package com.gl.demo.controller;

import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.gl.demo.dto.ProductDTO;
import com.gl.demo.service.ProductService;
import com.gl.demo.service.ProductServiceImpl;

/**
 * Servlet implementation class ProductController
 */
@WebServlet("/ProductController")
public class ProductController extends HttpServlet {
    private static final long serialVersionUID = 1L;

    private ProductService productService;

    public void init() {
        String jdbcURL = getServletContext().getInitParameter("dbURL");
        String jdbcUsername = getServletContext().getInitParameter("dbUsername");
        String jdbcPassword = getServletContext().getInitParameter("dbPassword");

        this.productService = new ProductServiceImpl(jdbcURL, jdbcUsername,
                jdbcPassword);
    }
}
```

```
}

/**
 * @see HttpServlet#HttpServlet()
 */
public ProductController() {
}

/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

    String action = request.getParameter("action") != null ?
request.getParameter("action") : "none";
    try {
        switch (action) {
            case "new":
                RequestDispatcher dispatcher =
request.getRequestDispatcher("newProductForm.jsp");
                dispatcher.forward(request, response);
                break;
            case "insert":
                String name = request.getParameter("name");
                String des = request.getParameter("description");
                float price = Float.parseFloat(request.getParameter("price"));

                ProductDTO newProduct = new ProductDTO(name, des, price);
                this.productService.addNewProduct(newProduct);
                response.sendRedirect("product");
                break;
            case "delete":
                break;
            case "edit":
                this.showEditForm(request, response);
                break;
            case "update":
                this.updateProduct(request, response);
                break;
            default:
                this.getListProduct(request, response);
                break;
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

```
        }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    doGet(request, response);
}

private void getListProduct(HttpServletRequest request, HttpServletResponse response)
throws SQLException, ServletException, IOException{
    List<ProductDTO> listBook = this.productService.getAllProducts();
    request.setAttribute("listProduct", listBook);
    RequestDispatcher dispatcher = request.getRequestDispatcher("productList.jsp");
    dispatcher.forward(request, response);
}

private void showEditForm(HttpServletRequest request, HttpServletResponse response)
throws SQLException, ServletException, IOException {
    int id = Integer.parseInt(request.getParameter("id"));
    ProductDTO existingBook = this.productService.getProductById(id);
    RequestDispatcher dispatcher =
request.getRequestDispatcher("newProductForm.jsp");
    request.setAttribute("product", existingBook);
    dispatcher.forward(request, response);

}

private void updateProduct(HttpServletRequest request, HttpServletResponse response)
throws SQLException, IOException {
    int id = Integer.parseInt(request.getParameter("id"));
    String name = request.getParameter("name");
    String des = request.getParameter("description");
    float price = Float.parseFloat(request.getParameter("price"));

    ProductDTO newProduct = new ProductDTO(id, name, des, price);
    this.productService.updateProduct(newProduct);
    response.sendRedirect("product");
}
}
```

Data Access Object (DAO):**ProductDAO.java:**

```
package com.gl.demo.dao;

import java.sql.SQLException;
import java.util.List;

import com.gl.demo.dto.ProductDTO;

/**
 * Interface for Product DAO.
 *
 */

public interface ProductDAO {
    List<ProductDTO> getAllProducts() throws SQLException;
    boolean addNewProduct(ProductDTO newProduct) throws SQLException;
    boolean updateProduct(ProductDTO product) throws SQLException;
    boolean deleteProduct(ProductDTO product) throws SQLException;
    ProductDTO getProductById(int id) throws SQLException;
}
```

ProductDAOImpl.java:

```
package com.gl.demo.dao;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

import com.gl.demo.dto.ProductDTO;

/**
 * Product DAO Implement.
 *
 */

public class ProductDAOImpl implements ProductDAO{

    private String jdbcURL;
```

```
private String jdbcUsername;
private String jdbcPassword;
private Connection jdbcConnection;

public ProductDAOImpl(String jdbcURL, String jdbcUsername, String jdbcPassword) {
    super();
    this.jdbcURL = jdbcURL;
    this.jdbcUsername = jdbcUsername;
    this.jdbcPassword = jdbcPassword;
}

protected void connect() throws SQLException {
    if (jdbcConnection == null || jdbcConnection.isClosed()) {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            throw new SQLException(e);
        }
        jdbcConnection = DriverManager.getConnection(
                jdbcURL,
                jdbcUsername, jdbcPassword);
    }
}

protected void disconnect() throws SQLException {
    if (jdbcConnection != null && !jdbcConnection.isClosed()) {
        jdbcConnection.close();
    }
}

@Override
public List<ProductDTO> getAllProducts() throws SQLException {
    List<ProductDTO> listProd = new ArrayList<ProductDTO>();

    String sql = "SELECT * FROM product";
    connect();

    Statement statement = jdbcConnection.createStatement();
    ResultSet resultSet = statement.executeQuery(sql);

    while (resultSet.next()) {
        int id = resultSet.getInt("product_id");
        String name = resultSet.getString("name");
        String des = resultSet.getString("description");
        float price = resultSet.getFloat("price");
    }
}
```

```
        ProductDTO product = new ProductDTO(id, name, des, price);
        listProd.add(product);
    }
    resultSet.close();
    statement.close();

    disconnect();

    return listProd;
}

@Override
public boolean addNewProduct(ProductDTO newProduct) throws SQLException {
    String sqlInsert = "INSERT INTO product (name, description, price) VALUES
    (?, ?, ?)";
    connect();
    PreparedStatement statement = jdbcConnection.prepareStatement(sqlInsert);
    statement.setString(1, newProduct.getName());
    statement.setString(2, newProduct.getDescription());
    statement.setFloat(3, newProduct.getPrice());

    boolean rowInserted = statement.executeUpdate() > 0;
    statement.close();
    disconnect();
    return rowInserted;
}

@Override
public boolean updateProduct(ProductDTO product) throws SQLException {
    String sql = "UPDATE product SET name = ?, description = ?, price = ?
    WHERE product_id = ?";
    connect();

    PreparedStatement statement = jdbcConnection.prepareStatement(sql);
    statement.setString(1, product.getName());
    statement.setString(2, product.getDescription());
    statement.setFloat(3, product.getPrice());
    statement.setInt(4, product.getId());

    boolean rowUpdated = statement.executeUpdate() > 0;
    statement.close();
    disconnect();
    return rowUpdated;
}

@Override
```

```
public boolean deleteProduct(ProductDTO product) throws SQLException {
    String sql = "DELETE FROM book where product_id = ?";
    connect();

    PreparedStatement statement = jdbcConnection.prepareStatement(sql);
    statement.setInt(1, product.getId());

    boolean rowDeleted = statement.executeUpdate() > 0;
    statement.close();
    disconnect();
    return rowDeleted;
}

@Override
public ProductDTO getProductById(int id) throws SQLException {
    ProductDTO prod = null;
    String sql = "SELECT * FROM product WHERE product_id = ?";

    connect();

    PreparedStatement statement = jdbcConnection.prepareStatement(sql);
    statement.setInt(1, id);

    ResultSet resultSet = statement.executeQuery();

    if (resultSet.next()) {
        String title = resultSet.getString("name");
        String author = resultSet.getString("description");
        float price = resultSet.getFloat("price");

        prod = new ProductDTO(id, title, author, price);
    }

    resultSet.close();
    statement.close();

    return prod;
}
```

Data Transfer Object:**ProductDTO.java:**

```
package com.gl.demo.dto;

public class ProductDTO {
    private int id;
    private String name;
    private String description;
    private float price;

    public int getId() {
        return id;
    }

    public ProductDTO(int id, String name, String description, float price) {
        super();
        this.id = id;
        this.name = name;
        this.description = description;
        this.price = price;
    }

    public ProductDTO(String name, String description, float price) {
        super();
        this.name = name;
        this.description = description;
        this.price = price;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public float getPrice() {
        return price;
    }
}
```

```
public void setPrice(float price) {  
    this.price = price;  
}  
}
```

ProductService.java:

```
package com.gl.demo.service;  
  
import java.sql.SQLException;  
import java.util.List;  
  
import com.gl.demo.dto.ProductDTO;  
  
/**  
 * Interface for Product Service.  
 *  
 */  
public interface ProductService {  
    List<ProductDTO> getAllProducts() throws SQLException;  
    boolean addNewProduct(ProductDTO newProduct) throws SQLException;  
    boolean updateProduct(ProductDTO product) throws SQLException;  
    boolean deleteProduct(ProductDTO product) throws SQLException;  
    ProductDTO getProductById(int id) throws SQLException;  
}
```

ProductServiceImpl.java:

```
package com.gl.demo.service;  
  
import java.sql.SQLException;  
import java.util.List;  
  
import com.gl.demo.dao.ProductDAO;  
import com.gl.demo.dao.ProductDAOImpl;  
import com.gl.demo.dto.ProductDTO;  
  
/**  
 * Product Service Implement.  
 *  
 */  
public class ProductServiceImpl implements ProductService{  
    private ProductDAO productDAO;
```

```
public ProductServiceImpl(String jdbcURL, String jdbcUsername, String jdbcPassword)
{
    this.productDAO = new ProductDAOImpl(jdbcURL, jdbcUsername,
jdbcPassword);
}

@Override
public List<ProductDTO> getAllProducts() throws SQLException {
    return this.productDAO.getAllProducts();
}

@Override
public boolean addNewProduct(ProductDTO newProduct) throws SQLException {
    return this.productDAO.addNewProduct(newProduct);
}

@Override
public boolean updateProduct(ProductDTO product) throws SQLException {
    return this.productDAO.updateProduct(product);
}

@Override
public boolean deleteProduct(ProductDTO product) throws SQLException {
    return this.productDAO.deleteProduct(product);
}

@Override
public ProductDTO getProductById(int id) throws SQLException {
    return this.productDAO.getProductById(id);
}
```

JSP:**productList.jsp:**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<link rel="stylesheet" href="${pageContext.request.contextPath}/css/productList.css" />
<title>List of Products</title>
</head>
<body>
    <div class="survey-page">
        <h1 id="title">Home Page Product</h1>
        <div id="form-container">
            <h2>
                <a href="product?action=new">Add New Product</a>
            </h2>
            <table cellpadding="5">
                <tr>
                    <th>ID</th>
                    <th>Name</th>
                    <th>Description</th>
                    <th>Price</th>
                    <th>Actions</th>
                </tr>
                <c:forEach var="product" items="${listProduct}">
                    <tr>
                        <td><c:out value="${product.id}" /></td>
                        <td><c:out value="${product.name}" /></td>
                        <td><c:out value="${product.description}" /></td>
                        <td><c:out value="${product.price}" /></td>
                        <td>
                            <a href="product?action=edit&id=<c:out value='${product.id}' />">Edit</a>
                            &nbsp;&nbsp;&nbsp;
                            <a href="product?action=delete&id=<c:out value='${product.id}' />">Delete</a>
                        </td>
                    </tr>
                </c:forEach>
            </table>
        </div>
```

```
</div>
</body>
</html>
```

newProductForm.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<link rel="stylesheet" href="${pageContext.request.contextPath}/css/newProduct.css" />
<title>Add new Product</title>
</head>
<body>
    <div class="survey-page">
        <h1 id="title">Product Editing/Adding Page</h1>
        <div id="form-container">
            <p id="description">
                Input product information to add or edit a product
            </p>
            <h2>
                <a href="product">List All Products</a> &ampnbsp&ampnbsp&ampnbsp
                <a href="product?action=new">Add New Product</a>
            </h2>
            <c:if test="${product != null}">
                <form action="product?action=update" method="get">
                    <c:if>
                        <c:if test="${product == null}">
                            <form action="product?action=insert" method="get">
                                <c:if>
                                    <c:if test="${product != null}">
                                        <input type="hidden" name="id" class="input-field" value="'/>
                                    </c:if>
                                    <div class="formRow">
                                        <label id="name-label" class="label-cls" for="name">* Name: </label>
                                        <div class="input-col">
                                            <input autofocus type="text" name="name" id="name" class="input-field"
value="'>
                                            <placeholder>Enter product name</placeholder>
                                            <required></required>
                                        </div>
                                    </div>
                                </c:if>
                            </form>
                        </c:if>
                    </c:if>
                </form>
            </c:if>
        </div>
    </div>
</body>
```

```

</div>
<div class="formRow">
    <label id="email-label" class="label-cls" for="email">* Description: </label>
    <div class="input-col">
        <input type="text" name="description" id="email" class="input-field"
value=<c:out value='${product.description}'/>">
            required placeholder="Enter description" >
        </div>
    </div>
    <div class="formRow">
        <label id="number-label" class="label-cls" for="age">* Price: </label>
        <div class="input-col">
            <input type="number" name="price" id="number" class="input-field"
value=<c:out value='${product.price}'/>">
                placeholder="Enter product price" >
            </div>
        </div>
    </div>
    <button id="submit" type="submit">Submit</button>
</form>
</div>
</div>
</div>

</body>
</html>

```

Web.xml:

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://JAVA.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
app_3_0.xsd" id="WebApp_ID" version="3.0">
    <display-name>ProductManagement</display-name>
    <welcome-file-list>
        <welcome-file>ProductController</welcome-file>
    </welcome-file-list>
    <servlet>
        <servlet-name>ProductController</servlet-name>
        <servlet-class>com.gl.demo.controller.ProductController</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ProductController</servlet-name>
        <url-pattern>/product</url-pattern>
    </servlet-mapping>
</web-app>

```

```
</servlet-mapping>

<context-param>
    <param-name>dbURL</param-name>
    <param-value>jdbc:mysql://localhost:3306/ProductManagement</param-value>
</context-param>

<context-param>
    <param-name>dbUsername</param-name>
    <param-value>root</param-value>
</context-param>

<context-param>
    <param-name>dbPassword</param-name>
    <param-value>shibu@ninja</param-value>
</context-param>

</web-app>
```

Pom.xml:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>ProjectManagement</groupId>
    <artifactId>ProjectManagement</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>

    <dependencies>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>3.1.0</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>javax.servlet.jsp</groupId>
            <artifactId>javax.servlet.jsp-api</artifactId>
            <version>2.3.3</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
```

```
<groupId>javax.servlet</groupId>
<artifactId>jstl</artifactId>
<version>1.2</version>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.20</version>
</dependency>

</dependencies>

<build>
    <plugins>
        <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.1</version>
            <configuration>
                <release>17</release>
            </configuration>
        </plugin>
        <plugin>
            <artifactId>maven-war-plugin</artifactId>
            <version>3.2.3</version>
        </plugin>
    </plugins>
</build>
</project>
```

MySQL:

- CREATE DATABASE ProductManagement;
- USE ProductManagement;
- CREATE TABLE product (product_id int(11) NOT NULL AUTO_INCREMENT, name varchar(128) NOT NULL, description varchar(255) NOT NULL, price float NOT NULL, PRIMARY KEY (product_id), UNIQUE KEY product_id_UNIQUE (product_id), UNIQUE KEY name_UNIQUE (name));

```
MySQL 8.0 Command Line Client
mysql> show databases;
+-----+
| Database |
+-----+
| employee |
| information_schema |
| myhiber |
| myhiber01 |
| mysql |
| performance_schema |
| spring |
| springjdbc |
| sys |
+-----+
9 rows in set (0.00 sec)

mysql> CREATE DATABASE ProductManagement;
Query OK, 1 row affected (0.10 sec)

mysql> USE ProductManagement;
Database changed

mysql> CREATE TABLE product (product_id int(11) NOT NULL AUTO_INCREMENT, name varchar(128) NOT NULL, description varchar(255) NOT NULL, price float NOT NULL, PRIMARY KEY (product_id), UNIQUE KEY product_id_UNIQUE (product_id), UNIQUE KEY name_UNIQUE (name));
Query OK, 0 rows affected, 1 warning (1.44 sec)

mysql> desc product;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| product_id | int | NO | PRI | NULL | auto_increment |
| name | varchar(128) | NO | UNI | NULL | |
| description | varchar(255) | NO | | NULL | |
| price | float | NO | | NULL | |
+-----+-----+-----+-----+-----+
4 rows in set (0.09 sec)
```

```
MySQL 8.0 Command Line Client
mysql> select * from product;
+-----+-----+-----+-----+
| product_id | name | description | price |
+-----+-----+-----+-----+
| 1 | samosa | snack | 15 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select * from product;
+-----+-----+-----+-----+
| product_id | name | description | price |
+-----+-----+-----+-----+
| 1 | samosa | snack | 10 |
| 2 | bisleri | water | 20 |
| 3 | mango | fruit | 120 |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Workbench:

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Includes icons for New Connection, Open Connection, Save, Undo, Redo, Copy, Paste, Find, Replace, Refresh, and Help.
- Navigator:** Shows the database structure. Under the 'productmanagement' schema, there is a 'Tables' section containing 'product'. Other schemas listed include 'employee', 'myhiber', 'myhiber01', 'spring', 'springjdbc', and 'sys'.
- SQL Editor:** A tab labeled 'SQL File 1' with the query: `1 • SELECT * FROM productmanagement.product;`
- Result Grid:** Displays the data from the 'product' table:

product_id	name	description	price
1	samosa	snack	15
2	bisleri	water	20
3	mango	fruit	120
*	NULL	NULL	NULL
- Output Panel:** Shows the execution log:

```
Action Output
# Time Action
1 21:23:48 SELECT * FROM productmanagement.product LIMIT 0, 1000
Message 0 row(s) returned
Duration / Fetch 0.015 sec / 0.000 sec
```
- Help Panel:** A note stating: "Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or to toggle automatic help."

Name: **Shibu Mohapatra**

Branch: **MSC AI**

Roll no: **02**

Output:

Home Page:



Home Page Product

Add New Product

ID	Name	Description	Price	Actions
1	samosa	snack	15.0	Edit Delete
2	bisleri	water	20.0	Edit Delete



Product adding page:



Product Editing/Adding Page

Input product information to add or edit a product

List All Products Add New Product

* Name:

* Description:

* Price:



Name: **Shibu Mohapatra**

Branch: **MSC AI**

Roll no: **02**

Product editing page:

Product Editing/Adding Page

Input product information to add or edit a product

[List All Products](#) [Add New Product](#)

* Name:

* Description:

* Price:

Product displaying page

List of Products

localhost:8081/ProductManagement/product?name=mango&description=fruit&price=120

Home Page Product

[Add New Product](#)

ID	Name	Description	Price	Actions
1	samosa	snack	10.0	Edit Delete
2	bisleri	water	20.0	Edit Delete
3	mango	fruit	120.0	Edit Delete

- 5. Design a JAVA Servlet Program to Download a file and display it on the screen (A link has to be provided in HTML, when the link is clicked corresponding file has to be displayed on Screen).**

Code:

FileDownloadServlet.java:

```
package com.jcg.servlet;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.OutputStream;

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(description = "Download File From The Server", urlPatterns = {
    "/downloadServlet" })
public class FileDownloadServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    public static int BUFFER_SIZE = 1024 * 100;
    public static final String UPLOAD_DIR = "uploadedFiles";

    /**
     * This Method Is Called By The Servlet Container To Process A 'GET' Request
     */
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        handleRequest(request, response);
    }

    public void handleRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

        /**
         * Get The Absolute Path Of The File To Be Downloaded ****/
        String fileName = request.getParameter("fileName"),
               applicationPath = getServletContext().getRealPath("") ,
               downloadPath = applicationPath + File.separator +
        UPLOAD_DIR,
               filePath = downloadPath + File.separator + fileName;
```

```
File file = new File(filePath);
OutputStream outStream = null;
FileInputStream inputStream = null;

if (file.exists()) {

    /*** Setting The Content Attributes For The Response Object ***/
    String mimeType = "application/octet-stream";
    response.setContentType(mimeType);

    /*** Setting The Headers For The Response Object ***/
    String headerKey = "Content-Disposition";
    String headerValue = String.format("attachment; filename=\"%s\"", file.getName());
    response.setHeader(headerKey, headerValue);

    try {

        /*** Get The Output Stream Of The Response ***/
        outStream = response.getOutputStream();
        inputStream = new FileInputStream(file);
        byte[] buffer = new byte[BUFFER_SIZE];
        int bytesRead = -1;

        /*** Write Each Byte Of Data Read From The Input Stream
        Write Each Byte Of Data Read From The Input Stream Into The Output Stream ***/
        while ((bytesRead = inputStream.read(buffer)) != -1) {
            outStream.write(buffer, 0, bytesRead);
        }
    } catch(IOException ioExObj) {
        System.out.println("Exception While Performing The I/O
Operation?=" + ioExObj.getMessage());
    } finally {
        if (inputStream != null) {
            inputStream.close();
        }

        outStream.flush();
        if (outStream != null) {
            outStream.close();
        }
    }
} else {

    /*** Set Response Content Type ***/
}
```

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

        /***** Print The Response *****/
        response.getWriter().println("<h3>File " + fileName + " Is Not Present
.....!</h3>");
    }
}
}
```

FileUploadServlet.java:

```
package com.jcg.servlet;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.Dispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.MultipartConfig;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.Part;

@WebServlet(description = "Upload File To The Server", urlPatterns = { "/fileUploadServlet" })
@MultipartConfig(fileSizeThreshold = 1024 * 1024 * 10, maxFileSize = 1024 * 1024 * 30,
maxRequestSize = 1024 * 1024 * 50)
public class FileUploadServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;
    public static final String UPLOAD_DIR = "uploadedFiles";

    /**** This Method Is Called By The Servlet Container To Process A 'POST' Request
****/
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        handleRequest(request, response);
    }

    public void handleRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
}
```

```
***** Get The Absolute Path Of The Web Application *****
String applicationPath = getServletContext().getRealPath("");
    uploadPath = applicationPath + File.separator + UPLOAD_DIR;

File fileUploadDirectory = new File(uploadPath);
if (!fileUploadDirectory.exists()) {
    fileUploadDirectory.mkdirs();
}

String fileName = "";
UploadDetail details = null;
List<UploadDetail> fileList = new ArrayList<UploadDetail>();

for (Part part : request.getParts()) {
    fileName = extractFileName(part);
    details = new UploadDetail();
    details.setFileName(fileName);
    details.setFileSize(part.getSize() / 1024);
    try {
        part.write(uploadPath + File.separator + fileName);
        details.setUploadStatus("Success");
    } catch (IOException ioObj) {
        details.setUploadStatus("Failure : " + ioObj.getMessage());
    }
    fileList.add(details);
}

request.setAttribute("uploadedFiles", fileList);
RequestDispatcher dispatcher =
request.getRequestDispatcher("/fileuploadResponse.jsp");
dispatcher.forward(request, response);
}

***** Helper Method #1 - This Method Is Used To Read The File Names *****
private String extractFileName(Part part) {
    String fileName = "",
        contentDisposition = part.getHeader("content-disposition");
    String[] items = contentDisposition.split(",");
    for (String item : items) {
        if (item.trim().startsWith("filename")) {
            fileName = item.substring(item.indexOf("=") + 2, item.length() -
1);
        }
    }
    return fileName;
}
```

}

UploadDetail.java:

```
package com.jcg.servlet;

import java.io.Serializable;

public class UploadDetail implements Serializable {

    private long fileSize;
    private String fileName, uploadStatus;

    private static final long serialVersionUID = 1L;

    public long getFileSize() {
        return fileSize;
    }

    public void setFileSize(long fileSize) {
        this.fileSize = fileSize;
    }

    public String getFileName() {
        return fileName;
    }

    public void setFileName(String fileName) {
        this.fileName = fileName;
    }

    public String getUploadStatus() {
        return uploadStatus;
    }

    public void setUploadStatus(String uploadStatus) {
        this.uploadStatus = uploadStatus;
    }
}
```

UploadFilesServlet.java:

```
package com.jcg.servlet;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.Dispatcher;
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(description = "List The Already Uploaded Files", urlPatterns = {
    "/uploadedFilesServlet" })
public class UploadedFilesServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;
    public static final String UPLOAD_DIR = "uploadedFiles";

    /**
     * This Method Is Called By The Servlet Container To Process A 'GET' Request
     */
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        handleRequest(request, response);
    }

    public void handleRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

        /**
         * Get The Absolute Path Of The Web Application ****/
        String applicationPath = getServletContext().getRealPath("");
        uploadPath = applicationPath + File.separator + UPLOAD_DIR;

        File fileUploadDirectory = new File(uploadPath);
        if (!fileUploadDirectory.exists()) {
            fileUploadDirectory.mkdirs();
        }

        UploadDetail details = null;
        File[] allFiles = fileUploadDirectory.listFiles();
        List<UploadDetail> fileList = new ArrayList<UploadDetail>();

        for (File file : allFiles) {
```

```

        details = new UploadDetail();
        details.setFileName(file.getName());
        details.setFileSize(file.length() / 1024);
        fileList.add(details);
    }

    request.setAttribute("uploadedFiles", fileList);
    RequestDispatcher dispatcher = request.getRequestDispatcher("/allfiles.jsp");
    dispatcher.forward(request, response);
}
}

```

Fileupload.js:

```

***** jQuery To Prevent The Form Submission In Case No File Is Selected For Upload *****
$(document).ready(function() {
    $("#fileUploadErr").hide();

    ***** Hide The Error Message When The Attachment Btn Is Clicked. *****
    $("#fileAttachment").click(function(eObj) {
        $("#fileUploadErr").hide();
    });

    ***** Validating Whether The Attachment Is Uploaded Or Not. *****
    $("#uploadBtn").click(function(eObj) {
        var file = $("#fileAttachment").map(function() {
            return $(this).val().trim() ? true : false;
        }).get();
        if (file.includes(true)) {
            ***** Do Nothing...! *****
        } else {
            $("#fileUploadErr").css({'color':'red', 'font-weight': 'bold'}).show();
            eObj.preventDefault();
        }
    });
});

```

Allfiles.jsp:

```

<% @page import="java.util.List"%>
<% @page import="com.jcg.servlet.UploadDetail"%>
<% @page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>

```

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Servlet File Upload/Download</title>

</head>
<body>
<div class="panel">
    <h1>Uploaded Files</h1>
    <table class="bordered_table">
        <thead>
            <tr align="center"><th>File Name</th><th>File Size</th><th>Action</th></tr>
        </thead>
        <tbody>
            <% List<UploadDetail> uploadDetails =
(List<UploadDetail>)request.getAttribute("uploadedFiles");
if(uploadDetails != null && uploadDetails.size() > 0) {
    for(int i=0; i<uploadDetails.size(); i++) {
        %>
        <tr>
            <td align="center"><span id="fileName"><%=uploadDetails.get(i).getFileName()%></span></td>
            <td align="center"><span id="fileSize"><%=uploadDetails.get(i).getFileSize()%>
KB</span></td>
            <td align="center"><span id="fileDownload"><a id="downloadLink"
class="hyperLink"
href="<%=request.getContextPath()%>/downloadServlet?fileName=<%=uploadDetails.get(i).get
FileName()%>">Download</a></span></td>
        </tr>
        <% } %>
    } else { %>
        <tr>
            <td colspan="3" align="center"><span id="noFiles">No Files
Uploaded.....!</span></td>
        </tr>
        <% } %>
    </tbody>
</table>
<div class="margin_top_15px">
    <a id="fileUpload" class="hyperLink"
href="<%=request.getContextPath()%>/fileupload.jsp">Back</a>
    </div>
</div>
</body>
</html>
```

Fileupload.jsp:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Servlet File Upload/Download</title>

        <script type="text/javascript" src="resource/js/jquery-3.2.1.min.js"></script>
        <script type="text/javascript" src="resource/js/fileupload.js"></script>
    </head>
    <body>
        <div class="panel">
            <h1>File Upload and Download</h1>

            <form id="fileUploadForm" method="post" action="fileUploadServlet" enctype="multipart/form-data">
                <div class="form_group">
                    <label>Upload File</label><span id="colon">:</span><input id="fileAttachment" type="file" name="fileUpload" multiple="multiple" />
                    <span id="fileUploadErr">Please Upload A File!</span>
                </div>
                <br>
                <button id="uploadBtn" type="submit" class="btn btn_primary">Upload</button>
            </form>
        </div>

        <!-- List All Uploaded Files -->
        <br>
        <div class="panel">
            <a id="allFiles" class="hyperLink" href="<% request.getContextPath() %>/uploadedFilesServlet">List all uploaded files</a>
        </div>
    </body>
</html>
```

fileuploadResponse.jsp:

```
<%@page import="java.util.List"%>
<%@page import="com.jcg.servlet.UploadDetail"%>
<%@page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Servlet File Upload/Download</title>
        <!-- <link rel="stylesheet" href="resource/css/main.css" /> -->
    </head>
    <body>
        <div class="panel">
            <h1>File Upload Status</h1>
            <table class="bordered_table">
                <thead>
                    <tr align="center"><th>File Name</th><th>File Size</th><th>Upload Status</th><th>Action</th></tr>
                </thead>
                <tbody>
                    <% List<UploadDetail> uploadDetails = (List<UploadDetail>)request.getAttribute("uploadedFiles");
                    for(int i=0; i<uploadDetails.size(); i++) { %>
                        <tr>
                            <td align="center"><span id="fileName"><%=uploadDetails.get(i).getFileName()%></span></td>
                            <td align="center"><span id="fileSize"><%=uploadDetails.get(i).getFileSize()%> KB</span></td>
                            <td align="center"><span id="fileuploadStatus"><%=uploadDetails.get(i).getUploadStatus()%></span></td>
                            <td align="center"><span id="fileDownload" class="hyperLink" href="<%=request.getContextPath()%>/downloadServlet?fileName=<%=uploadDetails.get(i).getFileName()%>">Download</a></span></td>
                        </tr>
                    <% } %>
                </tbody>
            </table>
            <div class="margin_top_15px">
                <a id="fileUpload" class="hyperLink" href="<%=request.getContextPath()%>/fileupload.jsp">Back</a>
            </div>
        </div>
```

```
</body>
</html>
```

Web.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" version="3.0">
  <display-name>Servlet File Upload/Download</display-name>
  <welcome-file-list>
    <welcome-file>fileupload.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

Pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-
v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>ServletFileUploadDownload</groupId>
  <artifactId>ServletFileUploadDownload</artifactId>
  <packaging>war</packaging>
  <version>0.0.1-SNAPSHOT</version>
  <name>ServletFileUploadDownload Maven Webapp</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <!-- Servlet API Dependency -->
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>3.1.0</version>
    </dependency>
    <dependency>
      <groupId>javax.servlet.jsp</groupId>
      <artifactId>jsp-api</artifactId>
      <version>2.1</version>
    </dependency>
  </dependencies>
  <build>
    <plugins>
```

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-war-plugin</artifactId>
    <version>3.3.1</version>
</plugin>
</plugins>
<finalName>${project.artifactId}</finalName>
</build>
</project>
```

Output:



File Upload and Download

Upload File: No file chosen

[List all uploaded files](#)



Name: **Shibu Mohapatra**

Branch: **MSC AI**

Roll no: **02**



File Upload and Download

Upload File: Receipt Format.pdf

[List all uploaded files](#)



File Upload Status

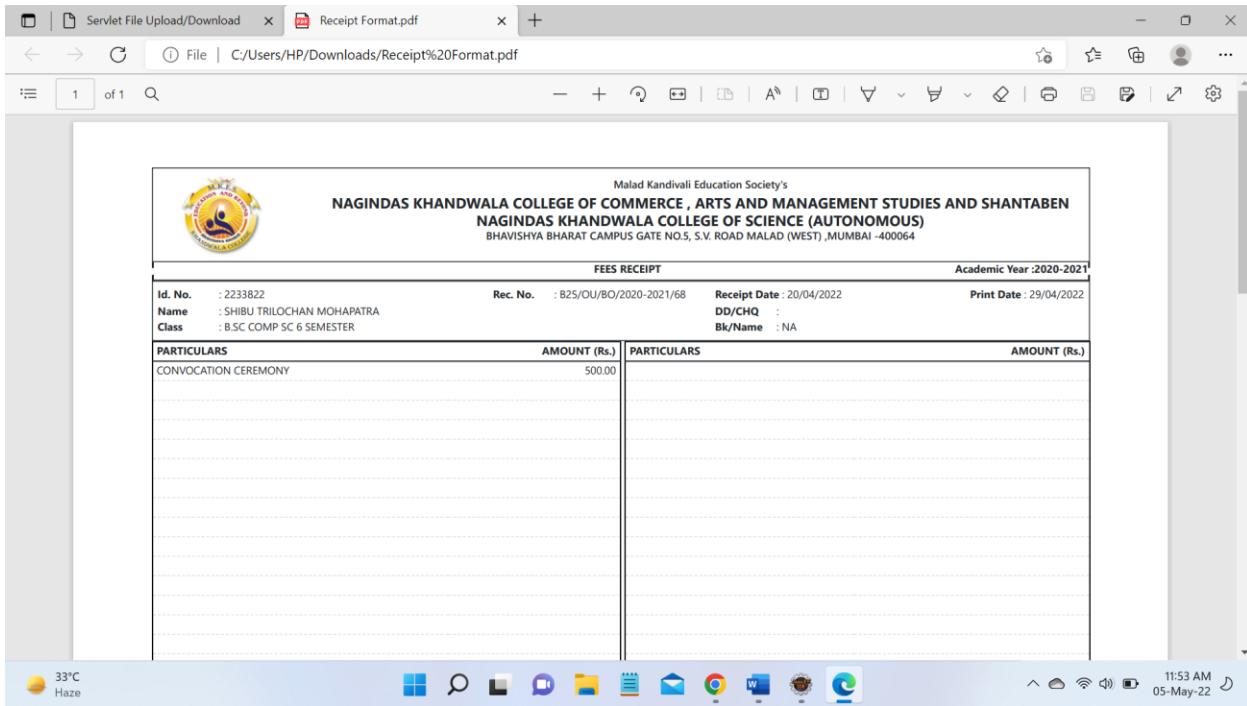
File Name	File Size	Upload Status	Action
Receipt Format.pdf	102 KB	Success	Download



Name: **Shibu Mohapatra**

Branch: **MSC AI**

Roll no: **02**



6.

- a) **Design a JAVA Servlet Program to implement Request Dispatcher object using include() and forward() methods.**

Code:

Login.java:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
```

```
public class Login extends HttpServlet {

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

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

        String n=request.getParameter("userName");
        String p=request.getParameter("userPass");

        if(p.equals("servlet")){
            RequestDispatcher rd=request.getRequestDispatcher("servlet2");
            rd.forward(request, response);
        }
        else{
            out.print("Sorry UserName or Password Error!");
            RequestDispatcher rd=request.getRequestDispatcher("/index.html");
            rd.include(request, response);

        }
    }
}
```

WelcomeServlet.java:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class WelcomeServlet extends HttpServlet {

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

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

        String n=request.getParameter("userName");
        out.print("Welcome "+n);
    }

}
```

Index.html:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Login</title>
</head>
<body>
<form action="servlet1" method="post">
Name:<input type="text" name="userName"/><br/>
Password:<input type="password" name="userPass"/><br/>
<input type="submit" value="login"/>

</form>
</body>
</html>
```

Web.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-
  app_4_0.xsd" id="WebApp_ID" version="4.0">
  <display-name>RequestDispatcher</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.jsp</welcome-file>
    <welcome-file>default.htm</welcome-file>
  </welcome-file-list>

  <servlet>
    <servlet-name>Login</servlet-name>
    <servlet-class>Login</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>WelcomeServlet</servlet-name>
    <servlet-class>WelcomeServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>Login</servlet-name>
    <url-pattern>/servlet1</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>WelcomeServlet</servlet-name>
    <url-pattern>/servlet2</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
</web-app>
```

Name: **Shibu Mohapatra**

Branch: **MSC AI**

Roll no: **02**

Output:

A screenshot of a web browser window. The address bar shows the URL `localhost:8090/RequestDispatcher/`. Below the address bar is a login form. It has two text input fields: the first is labeled "Name:" and contains the text "Shibu"; the second is labeled "Password:" and contains six black dots representing masked text. Below these fields is a blue "login" button.

With correct login details –

A screenshot of a web browser window. The address bar shows the URL `localhost:8090/RequestDispatcher/servlet1`. The main content area displays the text "Welcome Shibu".

With incorrect login details –

A screenshot of a web browser window. The address bar shows the URL `localhost:8090/RequestDispatcher/servlet1`. The main content area displays the text "Sorry UserName or Password Error!". Below this text is a login form with two text input fields: the first is labeled "Name:" and the second is labeled "Password:", both currently empty. Below these fields is a blue "login" button.

b) Implement a JAVA Servlet Program to implement sessions using HTTP Session Interface.**Code:****FirstServlet.java:**

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class FirstServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response) {
        try {
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            String n = request.getParameter("userName");
            out.print("Welcome " + n);
            HttpSession session = request.getSession();
            session.setAttribute("uname", n);
            out.print("<a href='servlet2'>visit</a>");
            out.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

SecondServlet.java:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SecondServlet extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response){
        try{

            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            HttpSession session=request.getSession(false);
```

```
String n=(String)session.getAttribute("uname");
out.print("Hello "+n);
out.close();
}catch(Exception e){System.out.println(e);}
}
}
```

Index.html:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>login</title>
</head>
<body>
<form action="servlet1">
Name:<input type="text" name="userName"/><br/>
<input type="submit" value="go"/>
</form>
</body>
</html>
```

Web.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-
app_4_0.xsd" id="WebApp_ID" version="4.0">
<display-name>HttpSession</display-name>
<welcome-file-list>
<welcome-file>index.html</welcome-file>
<welcome-file>index.jsp</welcome-file>
<welcome-file>index.htm</welcome-file>
<welcome-file>default.html</welcome-file>
<welcome-file>default.jsp</welcome-file>
<welcome-file>default.htm</welcome-file>
</welcome-file-list>
<servlet>
<servlet-name>s1</servlet-name>
<servlet-class>FirstServlet</servlet-class>
</servlet>
```

```
<servlet-mapping>
<servlet-name>s1</servlet-name>
<url-pattern>/servlet1</url-pattern>
</servlet-mapping>

<servlet>
<servlet-name>s2</servlet-name>
<servlet-class>SecondServlet</servlet-class>
</servlet>

<servlet-mapping>
<servlet-name>s2</servlet-name>
<url-pattern>/servlet2</url-pattern>
</servlet-mapping>

</web-app>
```

Output:

A screenshot of a web browser window. The address bar shows "localhost:8090/HttpSession/". Below the address bar is a form with a text input field containing "ShibuMoha" and a button labeled "go".

Servlet1 –

A screenshot of a web browser window. The address bar shows "localhost:8090/HttpSession/servlet1?userName= ShibuMoha". The page content displays the text "Welcome ShibuMoha [visit](#)".

Servlet2 –

A screenshot of a web browser window. The address bar shows "localhost:8090/HttpSession/servlet2". The page content displays the text "Hello ShibuMoha".

JSP

- 1. Develop dynamic web application to manage user (userId, name, dob, address) details using JSP. This app must have following pages**
 - a. Home page**
 - b. User adding page**
 - c. User editing page**
 - d. User displaying page**

Code:**User.java:**

```
package com.javatpoint.bean;

import java.sql.Date;

public class User {
    private int id;
    private String name,password,email,sex,address;
    private Date dob;
    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 getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public String getSex() {
```

```
        return sex;
    }
public void setSex(String sex) {
    this.sex = sex;
}
public Date getDob() {
    return dob;
}
public void setDob(Date dob) {
    this.dob = dob;
}
public String getAddress() {
    return address;
}
public void setAddress(String address) {
    this.address = address;
}
```

UserDao:

```
package com.javatpoint.dao;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

import com.javatpoint.bean.User;
public class UserDao {
    public static Connection getConnection(){
        Connection con=null;
        try{
            Class.forName("com.mysql.jdbc.Driver");

            con=DriverManager.getConnection("jdbc:mysql://localhost:3306/test","root","shibu@nin
ja");
        }catch(Exception e){System.out.println(e);}
        return con;
    }
    public static int save(User u){
        int status=0;
        try{
            Connection con=getConnection();
            PreparedStatement ps=con.prepareStatement("insert into
register(name,password,email,sex,dob,address) values(?,?,?,?,?,?)");
            ps.setString(1,u.getName());
            ps.setString(2,u.getPassword());
```

```
        ps.setString(3,u.getEmail());
        ps.setString(4,u.getSex());
        ps.setDate(5,u.getDob());
        ps.setString(6,u.getAddress());
        status=ps.executeUpdate();
    }catch(Exception e){System.out.println(e);}
    return status;
}
public static int update(User u){
    int status=0;
    try{
        Connection con=getConnection();
        PreparedStatement ps=con.prepareStatement("update register set
name=?,password=?,email=?,sex=?,dob=?,address=? where id=?");
        ps.setString(1,u.getName());
        ps.setString(2,u.getPassword());
        ps.setString(3,u.getEmail());
        ps.setString(4,u.getSex());
        ps.setDate(5,u.getDob());
        ps.setString(6,u.getAddress());
        ps.setInt(7,u.getId());
        status=ps.executeUpdate();
    }catch(Exception e){System.out.println(e);}
    return status;
}
public static int delete(User u){
    int status=0;
    try{
        Connection con=getConnection();
        PreparedStatement ps=con.prepareStatement("delete from register where id=?");
        ps.setInt(1,u.getId());
        status=ps.executeUpdate();
    }catch(Exception e){System.out.println(e);}

    return status;
}
public static List<User> getAllRecords(){
    List<User> list=new ArrayList<User>();

    try{
        Connection con=getConnection();
        PreparedStatement ps=con.prepareStatement("select * from register");
        ResultSet rs=ps.executeQuery();
        while(rs.next()){
            User u=new User();
            u.setId(rs.getInt("id"));
            u.setName(rs.getString("name"));
            u.setPassword(rs.getString("password"));
            u.setEmail(rs.getString("email"));
            u.setSex(rs.getString("sex"));
            u.setDob(rs.getDate("dob"));
            u.setAddress(rs.getString("address"));
            list.add(u);
        }
    }catch(Exception e){System.out.println(e);}
    return list;
}
```

```

        u.setName(rs.getString("name"));
        u.setPassword(rs.getString("password"));
        u.setEmail(rs.getString("email"));
        u.setSex(rs.getString("sex"));
        u.setDob(rs.getDate("dob"));
        u.setAddress(rs.getString("address"));
        list.add(u);
    }
}catch(Exception e){System.out.println(e);}
return list;
}
public static User getRecordById(int id){
    User u=null;
    try{
        Connection con=getConnection();
        PreparedStatement ps=con.prepareStatement("select * from register where id=?");
        ps.setInt(1,id);
        ResultSet rs=ps.executeQuery();
        while(rs.next()){
            u=new User();
            u.setId(rs.getInt("id"));
            u.setName(rs.getString("name"));
            u.setPassword(rs.getString("password"));
            u.setEmail(rs.getString("email"));
            u.setSex(rs.getString("sex"));
            u.setDob(rs.getDate("dob"));
            u.setAddress(rs.getString("address"));
        }
    }catch(Exception e){System.out.println(e);}
    return u;
}
}

```

Adduser.jsp:

```

<% @page import="com.javatpoint.dao.UserDao"%>
<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<% @page import="com.javatpoint.bean.User"%>
<jsp:useBean id="u" class="com.javatpoint.bean.User"></jsp:useBean>
<jsp:setProperty property="*" name="u"/>

<%
int i=UserDao.save(u);
if(i>0){
response.sendRedirect("adduser-success.jsp");
}else{

```

```
response.sendRedirect("adduser-error.jsp");
}
%>
```

Adduser-error.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Add User Success</title>
</head>
<body>

<p>Sorry, an error occured!</p>
<jsp:include page="userform.html"></jsp:include>

</body>
</html>
```

Adduserform.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Adding User Page</title>
</head>
<body>

<jsp:include page="userform.html"></jsp:include>

</body>
</html>
```

Adduser-success.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Add User Success</title>
```

```
</head>
<body>

<p>Record successfully saved!</p>
<jsp:include page="userform.html"></jsp:include>

</body>
</html>
```

Deleteuser.jsp:

```
<% @page import="com.javatpoint.dao.UserDao"%>
<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<% @page import="com.javatpoint.bean.User"%>
<jsp:useBean id="u" class="com.javatpoint.bean.User"></jsp:useBean>
<jsp:setProperty property="*" name="u"/>

<%
UserDao.delete(u);
response.sendRedirect("viewusers.jsp");
%>
```

Editform.jsp:

```
<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Edit Form</title>
</head>
<body>
<% @page import="com.javatpoint.dao.UserDao,com.javatpoint.bean.User"%>

<%
String id=request.getParameter("id");
User u=UserDao.getRecordById(Integer.parseInt(id));
%>

<h1>User Editing Page</h1>
<form action="edituser.jsp" method="post">
<input type="hidden" name="id" value="<% =u.getId() %>">
<table>
<tr><td>Name:</td><td><input type="text" name="name" value="<% =
u.getName()%>"></td></tr>
```

```

<tr><td>Password:</td><td><input type="password" name="password" value="<%=
u.getPassword()%>" /></td></tr>
<tr><td>Email:</td><td><input type="email" name="email" value="<%=
u.getEmail()%>" /></td></tr>
<tr><td>Sex:</td><td><input type="radio" name="sex" value="male"/>Male <input
type="radio" name="sex" value="female"/>Female </td></tr>
<tr><td>Dob:</td><td><input type="date">
<tr><td>Address:</td><td>
<select name="address">
<option>India</option>
<option>UK</option>
<option>USA</option>
<option>Canada</option>
<option>Other</option>
</select>
</td></tr>
<tr><td colspan="2"><input type="submit" value="Edit User"/></td></tr>
</table>
</form>

</body>
</html>

```

Edituser.jsp:

```

<% @page import="com.javatpoint.dao.UserDao"%>
<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<% @page import="com.javatpoint.bean.User"%>
<jsp:useBean id="u" class="com.javatpoint.bean.User"></jsp:useBean>
<jsp:setProperty property="*" name="u"/>

<%
int i=UserDao.update(u);
response.sendRedirect("viewusers.jsp");
%>

```

Index.jsp:

```

<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>

```

```

<body>
<h2>User details Home Page</h2>
<a href="adduserform.jsp">User Adding Page</a>
<br><br>
<a href="viewusers.jsp">User Display Page</a>

</body>
</html>

```

Userform.html:

```

<!DOCTYPE html>
<html>

<a href="viewusers.jsp">View All Records</a><br/>

<h1>Adding New User</h1>
<form action="adduser.jsp" method="post">
<table>
<tr><td>Name:</td><td><input type="text" name="name"/></td></tr>
<tr><td>Password:</td><td><input type="password" name="password"/></td></tr>
<tr><td>Email:</td><td><input type="email" name="email"/></td></tr>
<tr><td>Sex:</td><td><input type="radio" name="sex" value="male"/>Male <input type="radio" name="sex" value="female"/>Female </td></tr>
<tr><td>Dob:</td><td><input type="date">
<tr><td>Address:</td><td>
<select name="address">
<option>India</option>
<option>UK</option>
<option>USA</option>
<option>Canada</option>
<option>Other</option>
</select>
</td></tr>
<tr><td colspan="2"><input type="submit" value="Add User"/></td></tr>
</table>
</form>
</html>

```

Viewusers.jsp:

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

```

```

<title>View Users</title>
</head>
<body>

<% @page import="com.javatpoint.dao.UserDao,com.javatpoint.bean.*java.util.*"%>
<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<h1>User Displaying Page</h1>

<%
List<User> list=UserDao.getAllRecords();
request.setAttribute("list",list);
%>

<table border="1" width="90%">
<tr><th>Id</th><th>Name</th><th>Password</th><th>Email</th><th>Sex</th><th>Dob</th>
<th>Address</th><th>Edit</th><th>Delete</th></tr>
<c:forEach items="${list}" var="u">
    <tr><td>${u.getId()}</td>
    <td>${u.getName()}</td>
    <td>${u.getPassword()}</td>
    <td>${u.getEmail()}</td>
    <td>${u.getSex()}</td>
    <td>${u.getDob()}</td>
    <td>${u.getAddress()}</td>
    <td><a href="editform.jsp?id=${u.getId()}">Edit</a></td>
    <td><a href="deleteuser.jsp?id=${u.getId()}">Delete</a></td></tr>
</c:forEach>
</table>
<br/><a href="adduserform.jsp">Add New User</a>

</body>
</html>

```

Pom.xml:

```

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-
4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>JSPCrud</groupId>
    <artifactId>JSPCrud</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>
    <dependencies>

```

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.1.0</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.servlet.jsp</groupId>
    <artifactId>javax.servlet.jsp-api</artifactId>
    <version>2.3.3</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.20</version>
</dependency>
</dependencies>
<build>
<plugins>
<plugin>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.8.1</version>
    <configuration>
        <release>17</release>
    </configuration>
</plugin>
<plugin>
    <artifactId>maven-war-plugin</artifactId>
    <version>3.2.3</version>
</plugin>
</plugins>
</build>
</project>
```

MySQL:

- Create database test;
- Use test;
- Create table register(id INT(10) NOT NULL AUTO_INCREMENT, name varchar(100) NOT NULL, password varchar(100) NOT NULL, email varchar(100) NOT NULL, sex varchar(100) NOT NULL, dob varchar(100), country varchar(100) NOT NULL, PRIMARY KEY(id));
- desc register;
- alter table register MODIFY column dob DATE;
- select * from register;



```
MySQL 8.0 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 99
Server version: 8.0.28 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database          |
+--------------------+
| employee          |
| information_schema |
| myhiber           |
| myhiber01         |
| mysql              |
| performance_schema |
| productmanagement |
| spring             |
| springjdbc        |
| sys                |
+--------------------+
10 rows in set (0.18 sec)

mysql> create database test;
Query OK, 1 row affected (0.11 sec)

mysql> use test;
Database changed
mysql> create table register(id INT(10) NOT NULL AUTO_INCREMENT, name varchar(100) NOT NULL, password varchar(100) NOT NULL, email varchar(100) NOT NULL, sex varchar(100) NOT NULL, dob varchar(100), country varchar(100) NOT NULL, PRIMARY KEY(id));
Query OK, 0 rows affected, 1 warning (2.70 sec)
```

```
MySQL 8.0 Command Line Client
Records: 0 Duplicates: 0 Warnings: 0

mysql> desc register;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra       |
+-----+-----+-----+-----+-----+
| id    | int    | NO   | PRI | NULL    | auto_increment |
| name  | varchar(100) | NO  |     | NULL    |               |
| password | varchar(100) | NO  |     | NULL    |               |
| email | varchar(100) | NO  |     | NULL    |               |
| sex   | varchar(100) | NO  |     | NULL    |               |
| dob   | datetime | YES  |     | NULL    |               |
| address | varchar(100) | NO  |     | NULL    |               |
+-----+-----+-----+-----+-----+
7 rows in set (0.22 sec)

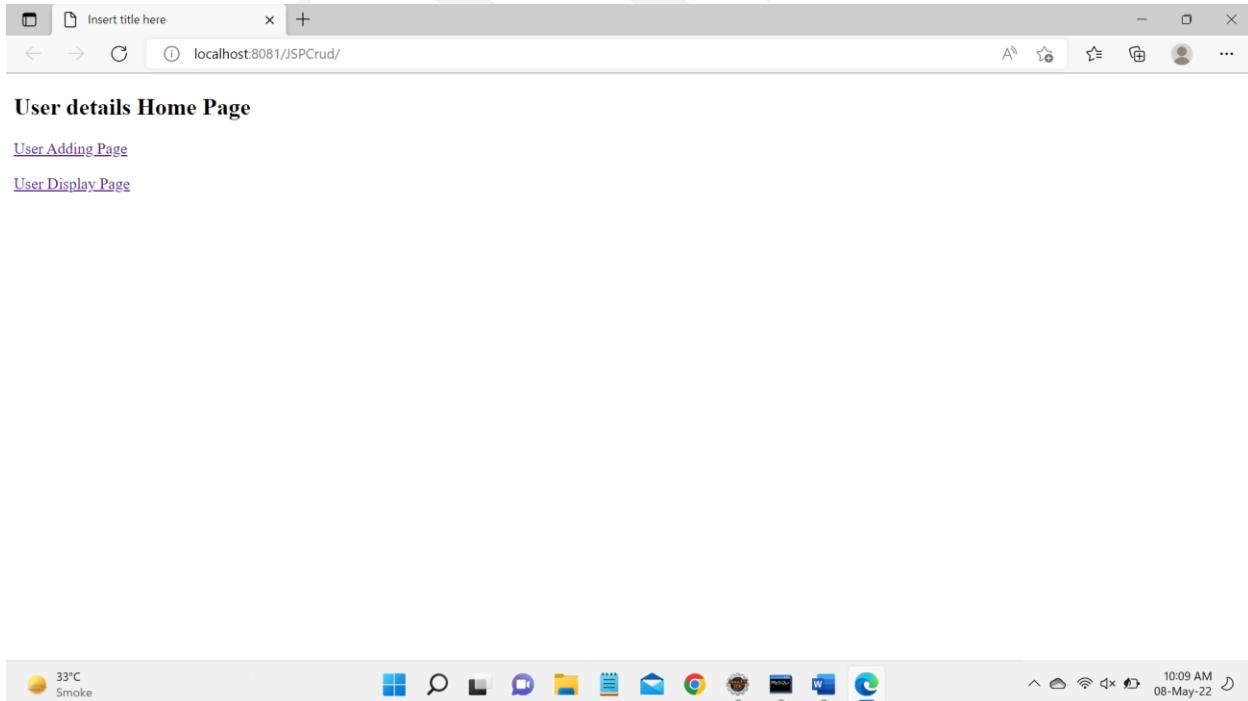
mysql> alter table register MODIFY column dob DATE;
Query OK, 2 rows affected (3.15 sec)
Records: 2 Duplicates: 0 Warnings: 0

mysql> select * from register;
+-----+-----+-----+-----+-----+
| id | name | password | email      | sex  | dob   | address |
+-----+-----+-----+-----+-----+
| 1  | test | test     | test@gmail.com | female | NULL  | India   |
| 2  | asd  | asd      | asd@gmail.com  | male  | NULL  | USA     |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> ■
```

Output:

Home page:



Name: **Shibu Mohapatra**

Branch: **MSC AI**

Roll no: **02**

User adding page:

Adding User Page

localhost:8081/JSPCrud/adduserform.jsp

[View All Records](#)

Adding New User

Name:

Password:

Email:

Sex: Male Female

Dob: dd----yyyy

Address: India



User editing page:

Edit Form

localhost:8081/JSPCrud/editform.jsp?id=2

User Editing Page

Name:

Password:

Email:

Sex: Male Female

Dob:

Address:



Name: **Shibu Mohapatra**

Branch: **MSC AI**

Roll no: **02**

User displaying page:



Id	Name	Password	Email	Sex	Dob	Address	Edit	Delete
1	test	test	test@gmail.com	female	1990-10-25	India	Edit	Delete
2	shibu	asd	abc@gmail.com	male	1999-09-29	USA	Edit	Delete

[Add New User](#)



2. Write JSP program to implement custom tag with name <product>, which display product (prodId, name, category, price) details.

Code:**ForwardTag.jsp:**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Forward Action Example</title>
</head>
<body>

<body>
<!-- > is square.jsp -->
<!-- < is OrderForm.jsp -->

<%if(Math.random()<0.5){ %>
<jsp:forward page="square.jsp"></jsp:forward>

<% } else{ %>
<jsp:forward page="OrderForm.jsp"></jsp:forward>
<% }
%>

</body>
</html>
```

Square.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>IPower of 2 for integers in the range0-10</title>
</head>
<body>

<center></center>
```

```

<table border="2" align="center">
<th>Exponent</th>
<th>2^Exponent</th>

<% for(int i=0;i<=10;i++)
{ // start the loop
%>

<tr>
<td><%=i %></td>
<td><%=Math.pow(2,i) %></td>
</tr>
<% }//end of loop %>
</table>
</center>

</body>
</html>

```

OrderForm.jsp:

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Sample Order Form</title>
</head>
<body>

<h1 align="center">A Sample Order Form</h1>

<% ! int prodid[]={1,2,3,4,5};
String name []={ "Pen", "Pencil", "Book", "Mouse", "Keyboard"} ;
String category[]={"stationary","stationary","stationary","computer","computer"};
double price[]={10.10, 5.5, 99.99, 200.00, 600.00};
int quantity[]={10,5,8,5,2};
%>

<table align="center" bgcolour="Blue" border="1" width="50%">

<tr><td>ProdId</td>
<td>Name</td>
<td>Category</td>

```

```

<td>Price</td>
<td>Quantity</td>
<td>Total Price</td>
</tr>

<%for(int i=0;i<5;i++){ %>
<tr><td><%=prodid[i] %></td>
<td><%=name[i] %></td>
<td><%=category[i] %></td>
<td><%=price[i] %></td>
<td><%=quantity[i] %></td>
<td><%=price[i] * quantity[i] %></td>

<% } //end for loop %>

</table>

</body>
</html>

```

Web.xml:

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
  app_3_0.xsd" id="WebApp_ID" version="3.0">
  <display-name>PowerOf2</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.jsp</welcome-file>
    <welcome-file>default.htm</welcome-file>
  </welcome-file-list>
</web-app>

```

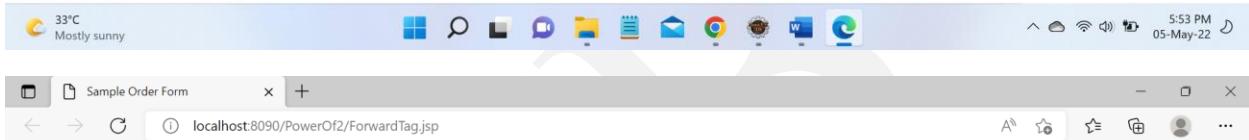
Output:

Name: **Shibu Mohapatra**

Branch: **MSC AI**

Roll no: **02**

Exponent	2^{Exponent}
0	1.0
1	2.0
2	4.0
3	8.0
4	16.0
5	32.0
6	64.0
7	128.0
8	256.0
9	512.0
10	1024.0



A Sample Order Form

ProdId	Name	Category	Price	Quantity	Total Price
1	Pen	stationary	10.1	10	101.0
2	Pencil	stationary	5.5	5	27.5
3	Book	stationary	99.99	8	799.92
4	Mouse	computer	200.0	5	1000.0
5	Keyboard	computer	600.0	2	1200.0



3. Design a JAVA JSP Program to implement verification of a particular user login and display a welcome page

Code:

Index.html:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE html>
<html>
  <head>
    <title>JSP Page</title>
  </head>
  <body>
    <form method="post" action="auth.jsp" >
      <h1>
        User name <input type="text" name="uname" />
        <br>
        Password <input type="password" name="pwd" />
        <input type="submit" value="Login" />
      </h1>
    </form>
  </body>
</html>
```

Auth.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>JSP Page</title>
  </head>
  <body>
    <%
      String u=request.getParameter("uname");
      String p=request.getParameter("pwd");
      if ((u.equals("admin"))&& (p.equals("hello")))
        out.println("Welcome "+u+" you are authenticated");
      else
        out.println("Failed Login Attempt");
    %>
```

```
</body>
</html>
```

Web.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
  app_3_0.xsd" id="WebApp_ID" version="3.0">
  <display-name>userpass</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.jsp</welcome-file>
    <welcome-file>default.htm</welcome-file>
  </welcome-file-list>
</web-app>
```

Name: **Shibu Mohapatra**

Branch: **MSC AI**

Roll no: **02**

Output:



User name
Password



Welcome admin you are authenticated



4. Write a program to design a simple calculator using JSP

Code:

Calculator.html:

```
<!DOCTYPE html>
<html>
<title>calculator</title>
<head>
<h1>
    <center>Simple Calculator</center>
</h1>
</head>
<body>
    <center>
        <form action="calculator.jsp" method="get">

            <label for="num1"><b>Number 1</b></label> <input type="text"
                name="num1"><br>
            <br> <label for="num2"><b>Number 2</b></label> <input
                type="text" name="num2"><br>
            <br> <input type="radio" name="r1" value="Add">+ <input
                type="radio" name="r1" value="Sub">-<br> <input
                type="radio" name="r1" value="mul">*<input type="radio"
                name="r1" value="div"/><br>
            <br> <input type="submit" value="submit">
        </form>
    </center>
</body>
</html>
```

Calculator.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<title>calculator</title>
<head></head>
<body>
    <%@page language="java"%>
    <%
    int num1 = Integer.parseInt(request.getParameter("num1"));
    int num2 = Integer.parseInt(request.getParameter("num2"));

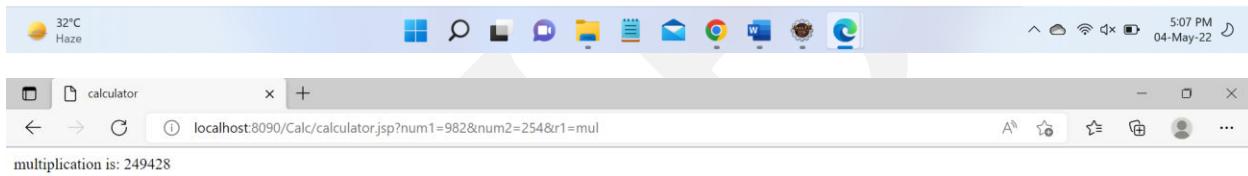
    if(r1.value == "Add")
        result = num1 + num2;
    else if(r1.value == "Sub")
        result = num1 - num2;
    else if(r1.value == "mul")
        result = num1 * num2;
    else if(r1.value == "div")
        result = num1 / num2;
    %>
</body>
</html>
```

```
String operation = request.getParameter("r1");
if(operation.equals("Add")){
int add=num1+num2;
out.println("Addition is: "+add);
}
else if(operation.equals("Sub")){
int sub=num1-num2;
out.println("Subtraction is: "+sub);
}
else if(operation.equals("mul")){
int mul=num1*num2;
out.println("multiplication is: "+mul);
}
else if(operation.equals("div")){
{
int div = num1/num2;

if(num1>=num2)
out.println("division is: "+div);
else
out.println("The division cannot be performed");
}
%>
</body>
</html>
```

Output:

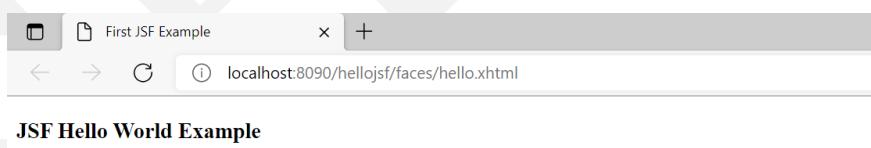
A screenshot of a Microsoft Edge browser window. The title bar shows the tab is titled "calculator". The address bar displays the URL "localhost:8090/Calc/calculator.html". The main content area contains the heading "Simple Calculator". Below it are two input fields: "Number 1" with the value "982" and "Number 2" with the value "254". Underneath these fields are four radio buttons for operators: addition (+), subtraction (-), multiplication (*), and division (/). A "submit" button is located below the operator buttons.



JSF**1. Write a JSF application to print Hello World.****Code:****Hello.xhtml:**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">

<h:head>
    <title>First JSF Example</title>
</h:head>
<h:body>
    <h3>JSF Hello World Example</h3>
</h:body>
</html>
```

Output:

2. Write a JSF application for Celsius to Fahrenheit convertor.**Code:****TemperatureConvertor.java:**

```
package shibu.jsf.jsfDemo.model;

public class TemperatureConvertor {
    private double celsius;
    private double fahrenheit;
    private boolean initial = true;

    public double getCelsius() {
        return celsius;
    }

    public void setCelsius(double celsius) {
        this.celsius = celsius;
    }

    public double getFahrenheit() {
        return fahrenheit;
    }

    public boolean getInitial() {
        return initial;
    }

    public String reset() {
        initial = true;
        fahrenheit = 0;
        celsius = 0;
        return "reset";
    }

    public String celsiusToFahrenheit() {
        initial = false;
        fahrenheit = (celsius * 9 / 5) + 32;
        return "calculated";
    }
}
```

Faces-config.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config
    xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig_2_2.xsd"
    version="2.2">
    <managed-bean>
        <managed-bean-name>temperatureConvertor</managed-bean-name>
        <managed-bean-
class>shibu.jsf.jsfDemo.model.TemperatureConvertor</managed-bean-class>
        <managed-bean-scope>session</managed-bean-scope>
    </managed-bean>

</faces-config>
```

Web.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
app_3_0.xsd" id="WebApp_ID" version="3.0">
    <display-name>jsfDemo</display-name>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.jsp</welcome-file>
        <welcome-file>index.htm</welcome-file>
        <welcome-file>default.html</welcome-file>
        <welcome-file>default.jsp</welcome-file>
        <welcome-file>default.htm</welcome-file>
    </welcome-file-list>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>/faces/*</url-pattern>
    </servlet-mapping>
    <context-param>
        <description>State saving method: 'client' or 'server' (=default). See JSF Specification
2.5.2</description>
```

```
<param-name>javax.faces.STATE_SAVING_METHOD</param-name>
<param-value>client</param-value>
</context-param>
<context-param>
<param-name>javax.servlet.jsp.jstl.fmt.localizationContext</param-name>
<param-value>resources.application</param-value>
</context-param>
<listener>
<listener-class>com.sun.faces.config.ConfigureListener</listener-class>
</listener>
</web-app>
```

Convertor.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="f" uri="http://java.sun.com/jsf/core"%>
<%@ taglib prefix="h" uri="http://java.sun.com/jsf/html"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Celsius to Fahrenheit Convertor</title>
</head>
<body>
<f:view>
<h:form>
<h:panelGrid columns="2">
<h:outputLabel value="Celsius"></h:outputLabel>
<h:inputText value="#{temperatureConvertor.celsius}"></h:inputText>
</h:panelGrid>
<h:commandButton action="#{temperatureConvertor.celsiusToFahrenheit}"
value="Calculate"></h:commandButton>
<h:commandButton action="#{temperatureConvertor.reset}"
value="Reset"></h:commandButton>
<h:messages layout="table"></h:messages>
</h:form>

<h:panelGroup rendered="#{temperatureConvertor.initial!=true}">
<h3> Result </h3>
<h:outputLabel value="Fahrenheit "></h:outputLabel>
<h:outputLabel value="#{temperatureConvertor.fahrenheit}"></h:outputLabel>
</h:panelGroup>
</f:view>
```

Name: **Shibu Mohapatra**

Branch: **MSC AI**

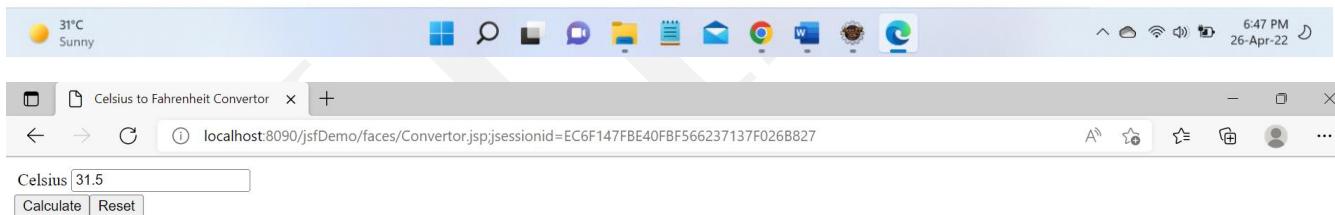
Roll no: **02**

</body>
</html>

Output:



Celsius to Fahrenheit Convertor x +
localhost:8090/jsfDemo/faces/Convertor.jsp
Celsius 0.0
Calculate Reset



31°C Sunny 6:47 PM 26-Apr-22
Celsius to Fahrenheit Convertor x +
localhost:8090/jsfDemo/faces/Convertor.jsp?jsessionid=EC6F147FBE40FBF566237137F026B827
Celsius 31.5
Calculate Reset

Result

Fahrenheit 88.7



- 3. Design following JSF application to accept username in textbox. if user textbox is empty show validation message.**

Code:

HelloBean.java;

```
package com.shibu.hellojsf;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import java.io.Serializable;

@ManagedBean
@SessionScoped
public class HelloBean implements Serializable {

    private static final long serialVersionUID = 1L;

    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

Hello.xhtml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">

<h:head>
    <title>Userform</title>
</h:head>
```

```
<h:body>
    <h3>JSF Hello World Example</h3>
    <h:form>
        What's your name?
        <h:inputText value="#{helloBean.name}"></h:inputText>
            <h:commandButton value="Submit" action="welcome"></h:commandButton>
        </h:form>
</h:body>
</html>
```

Welcome.xhtml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">

<h:head>
    <title>Welcome</title>
</h:head>
<h:body bgcolor="white">
    <h3>Everything went right!</h3>
    <h4>Welcome #{helloBean.name}</h4>
</h:body>
</html>
```

Faces-config.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config
    xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig_2_2.xsd"
    version="2.2">

</faces-config>
```

Web.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://JAVA.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
    id="WebApp_ID" version="3.0">
    <display-name>HelloJSF</display-name>
    <welcome-file-list>
        <welcome-file>faces/hello.xhtml</welcome-file>
    </welcome-file-list>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>/faces/*</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.jsf</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.faces</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.xhtml</url-pattern>
    </servlet-mapping>
    <context-param>
        <param-name>javax.faces.PROJECT_STAGE</param-name>
        <param-value>Development</param-value>
    </context-param>
    <listener>
        <listener-class>com.sun.faces.config.ConfigureListener
        </listener-class>
    </listener>
</web-app>
```

Name: **Shibu Mohapatra**

Branch: **MSC AI**

Roll no: **02**

Output:

A screenshot of a Microsoft Edge browser window. The title bar says "Userform". The address bar shows "localhost:8090/namejsf/hello.jsf". The page content is a simple form with a text input field containing "What's your name?" and a submit button.

A screenshot of a Microsoft Edge browser window, identical to the one above but with the text input field now containing "shibu". The rest of the interface is the same, showing the browser toolbar and taskbar at the bottom.



Name: **Shibu Mohapatra**

Branch: **MSC AI**

Roll no: **02**



Everything went right!

Welcome shibu



- 4. A web application that takes a name as input and on submit it shows a hello <name> page where name is taken from the request. It shows the start time at the right top corner of the page and provides a logout button. On clicking this button, it should show a logout page with Thank You <name > message with the duration of usage
(hint: Use session to store name and time).**

Code:

Session.xhtml:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head> <title> SESSION LOGIN </title> </head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
<body>
<center>
<form action="Session1.jsp" method="get">
Enter Name: <input type="text" name="uname" />
<input type="submit" value="LOGIN" name="register" />
</form>
</center>
</body>
</html>
```

Session1.jsp:

```
<%@page language="java" import="java.util.*" errorPage=""%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="f" uri="http://java.sun.com/jsf/core"%>
<%@ taglib prefix="h" uri="http://java.sun.com/jsf/html"%>
<form method="get" action="Session2.jsp">
<%
Date d=new Date();
%>
<p align="right"> Time;<%=d.getTime()%></p>
<%
String un=request.getParameter("uname");
session.setAttribute("user",un);
session.setAttribute("time",d.getTime());
%>
Hello <%=un%>
<br><br>
```

```
<input type="submit" value="logout">
</form>
```

Session2.jsp:

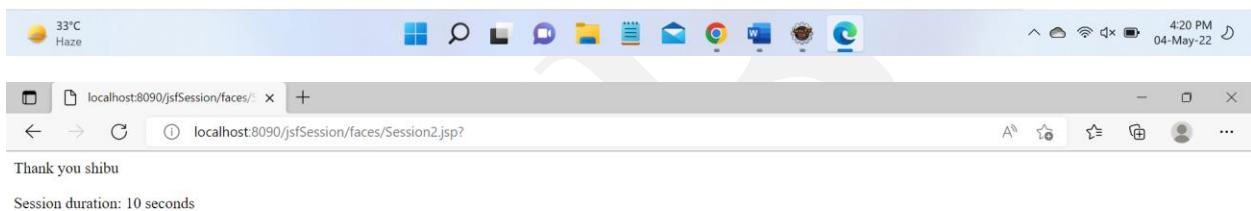
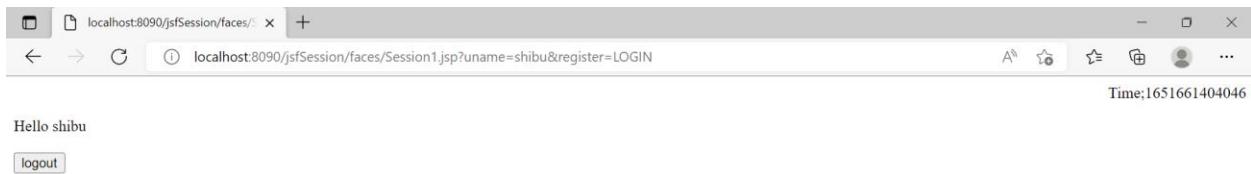
```
<%@page language="java" import="java.util.*" errorPage=""%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="f" uri="http://java.sun.com/jsf/core"%>
<%@ taglib prefix="h" uri="http://java.sun.com/jsf/html"%>
<%
Date d2=new Date();
String un=(String)session.getAttribute("user");
Long t1=(Long)session.getAttribute("time");
Long t2=d2.getTime();
%>
Thank you <%=un%>
<br><br>
Session duration: <%=(t2-t1)/(60*60)%> seconds
<% session.invalidate();%>
```

Output:

Name: **Shibu Mohapatra**

Branch: **MSC AI**

Roll no: **02**



HIBERNATE**1. Creating hibernate example to print “Hello Word”.****Code:****hello.java:**

```
package com.hello;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Hello World!" );
    }
}
```

Pom.xml:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

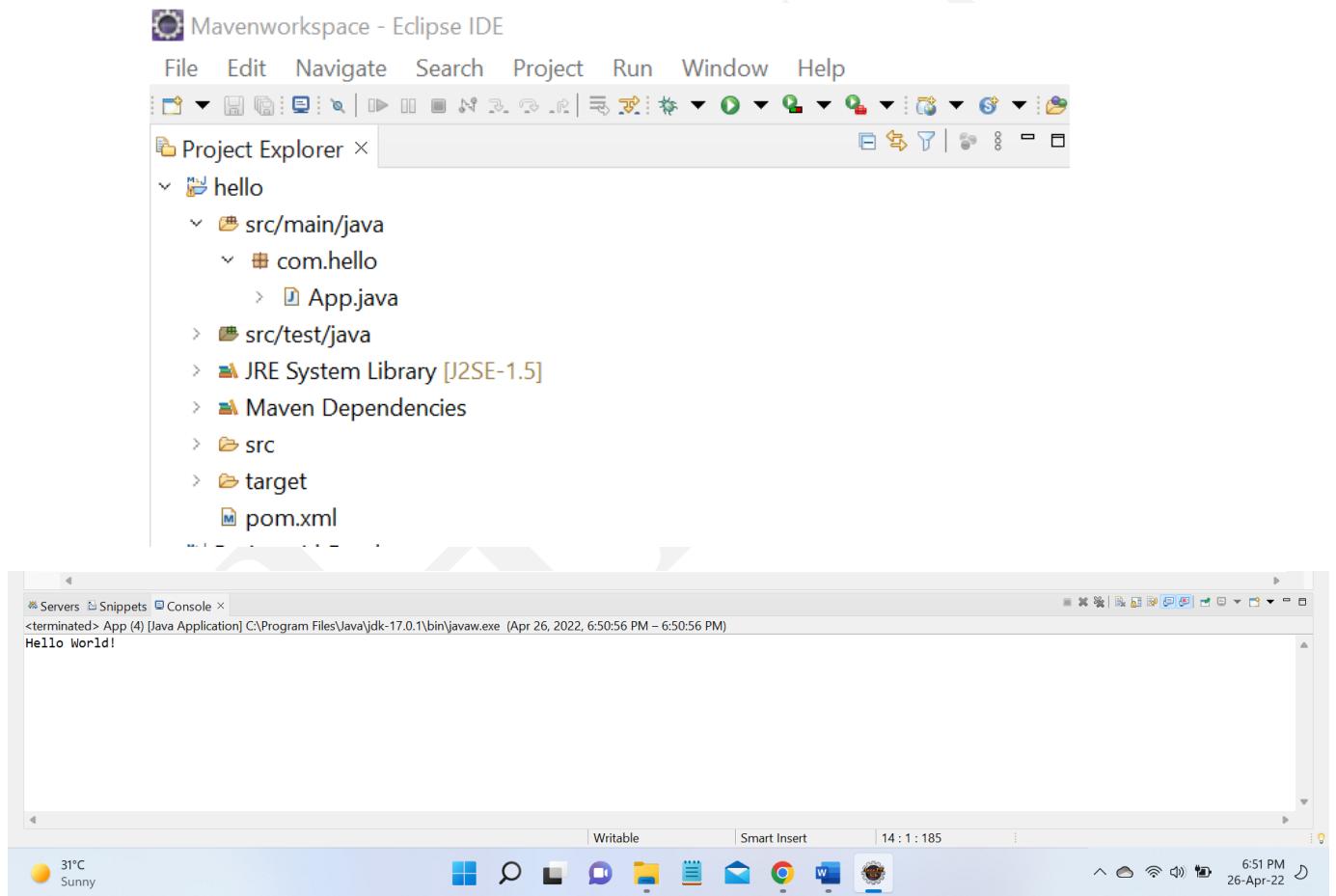
  <groupId>com.spring.mvc</groupId>
  <artifactId>hello</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>hello</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
```

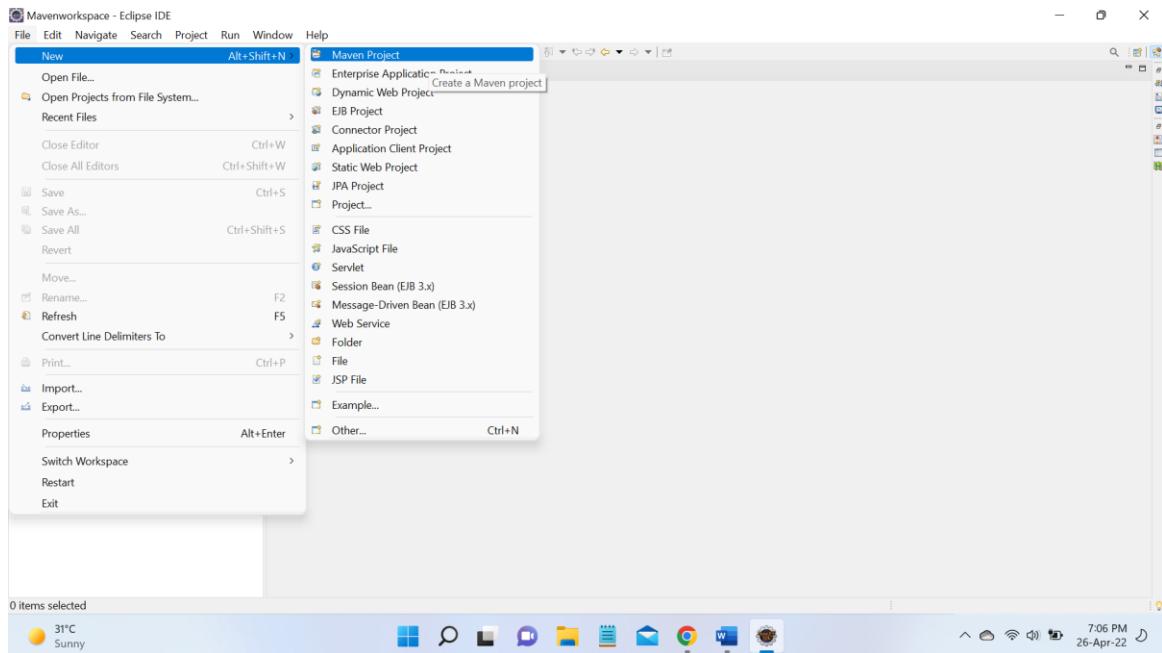
```
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>3.8.1</version>
<scope>test</scope>
</dependency>
</dependencies>
</project>
```

Output:

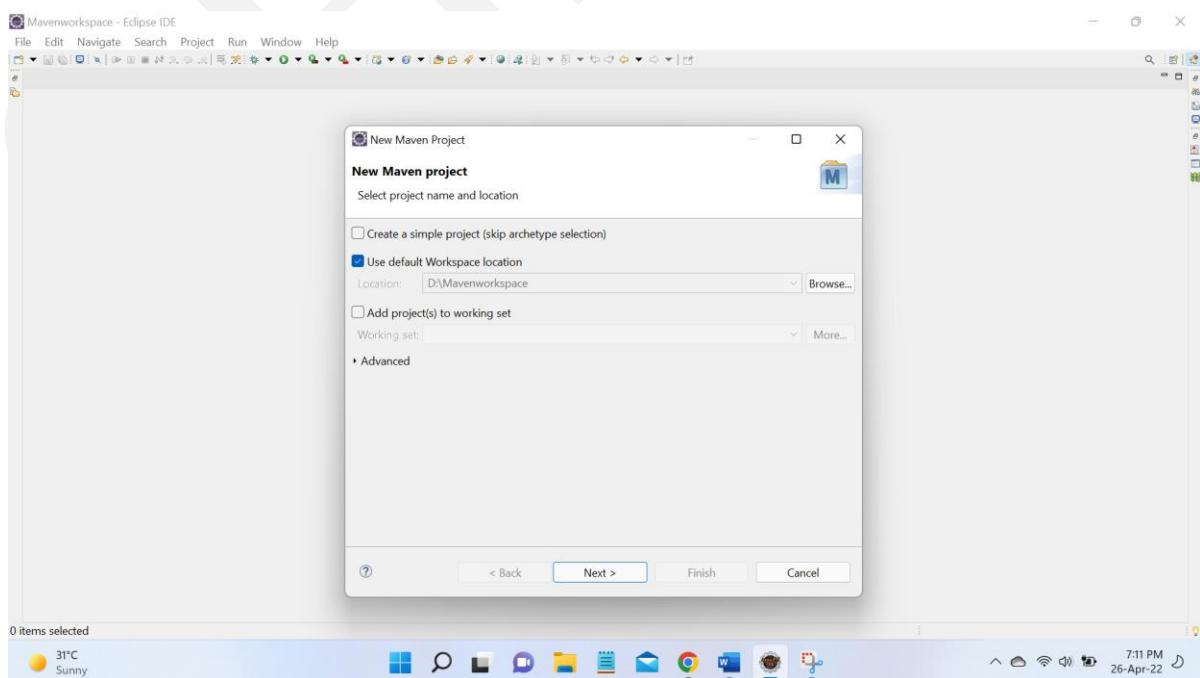
**2. A) Create Maven Project for hibernate to add dependencies
B) Configure hibernate program**

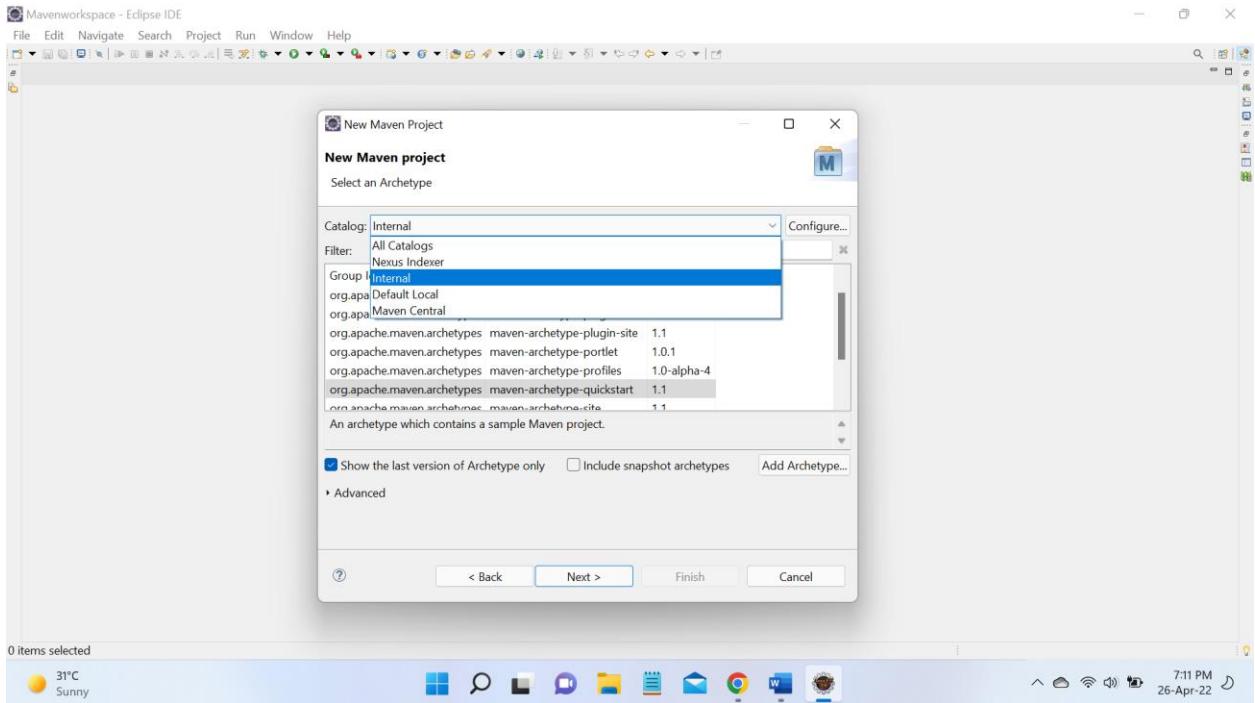
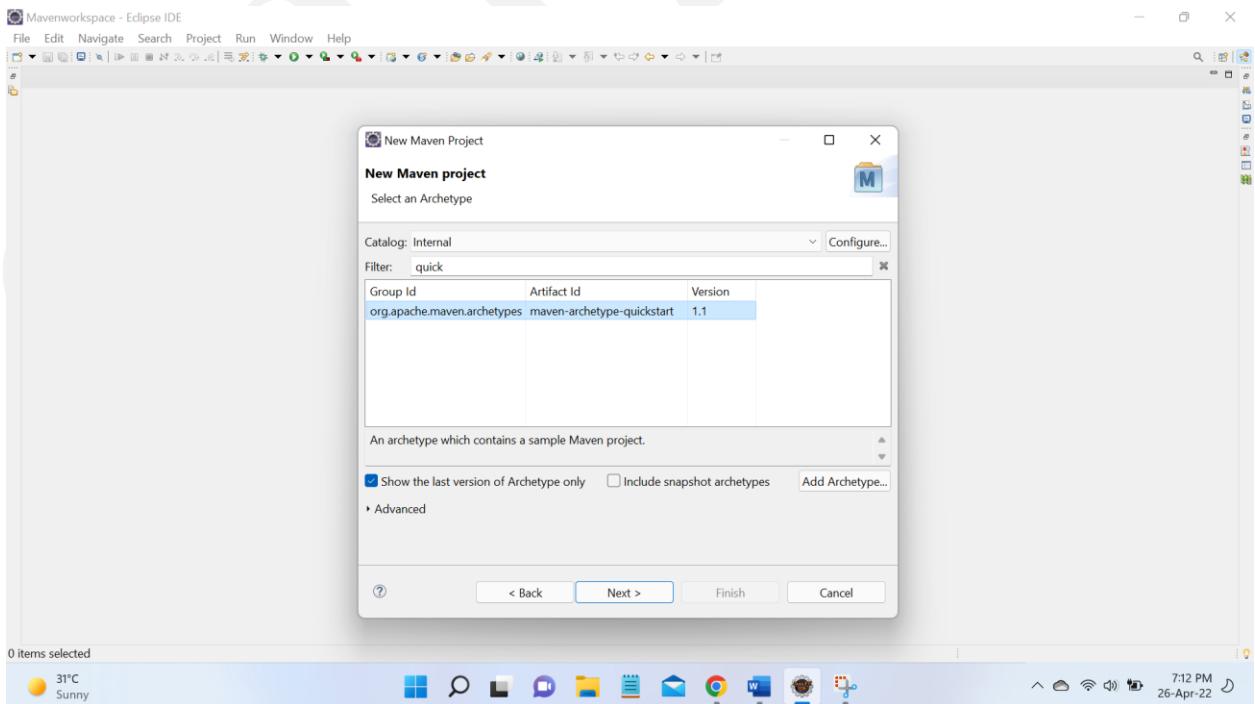
A. Create Maven Project for hibernate to add dependencies

- i. New => maven project

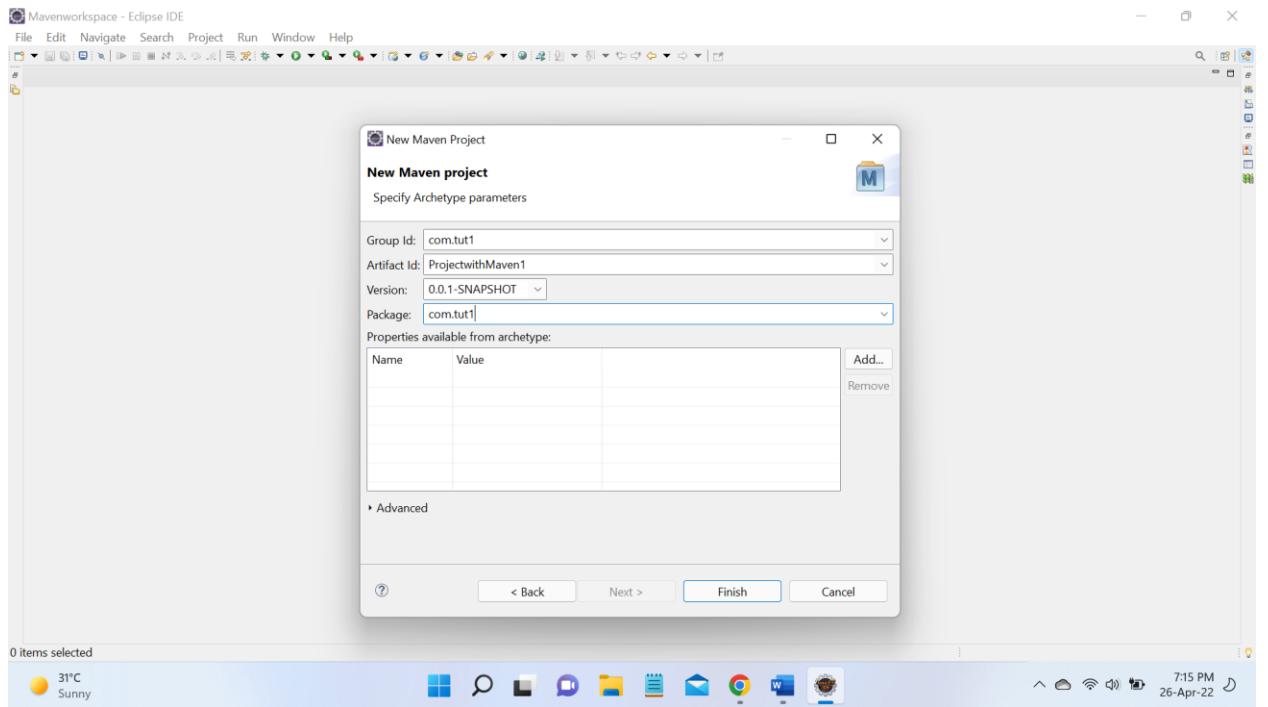


- ii. switch directory (if necessary) or just click next

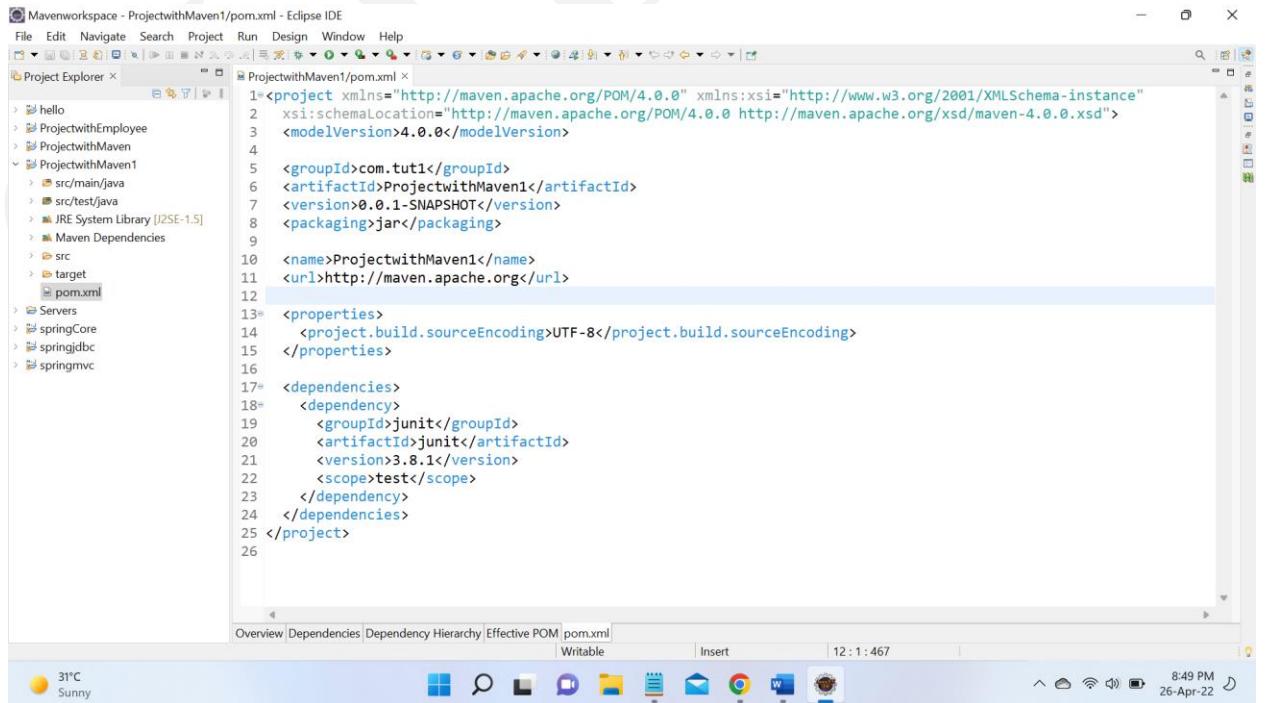


iii. catalog (select internal)**iv. filter (type quick) => select quickstart and click on Next**

- v. Write group id (com.tut) => artifact Id (ProjectwithMaven) => Package (com.tut) =>
Click on finish.



- vi. Explore project, POM file is important



- vii. Copy the maven code in the website & paste it in pom.xml under <dependencies> tag from below website

<https://mvnrepository.com/artifact/org.hibernate/hibernate-core/5.6.2.Final>

The screenshot shows the Maven Repository page for the hibernate-core artifact. The page includes a graph titled 'Indexed Artifacts (27.1M)' showing the number of projects indexed over time from 2009 to 2018. A sidebar lists 'Popular Categories' such as Aspect Oriented, Actor Frameworks, Application Metrics, Build Tools, Bytecode Libraries, Command Line Parsers, Cache Implementations, Cloud Computing, Code Analyzers, Collections, Configuration Libraries, Core Utilities, Date and Time Utilities, Dependency Injection, Embedded SQL Databases, and HTML Parsers. The main content area displays the 'Hibernate Core Relocation > 5.6.2.Final' page, which includes the following details:

- License:** LGPL 2.1
- Categories:** Object/Relational Mapping
- Organization:** Hibernate.org
- HomePage:** <https://hibernate.org/orm>
- Date:** (Dec 08, 2021)
- Files:** pom (5 KB) | jar (7.1 MB) | View All
- Repositories:** Central
- Used By:** 3,676 artifacts

A note at the bottom states: "Note: There is a new version for this artifact" followed by "New Version 6.0.0.Final". Below this, a code block shows the Maven dependency XML:

```
<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.6.2.Final</version>
</dependency>
```

```
<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.6.2.Final</version>
</dependency>
```

Mavenworkspace - ProjectwithMaven1/pom.xml - Eclipse IDE

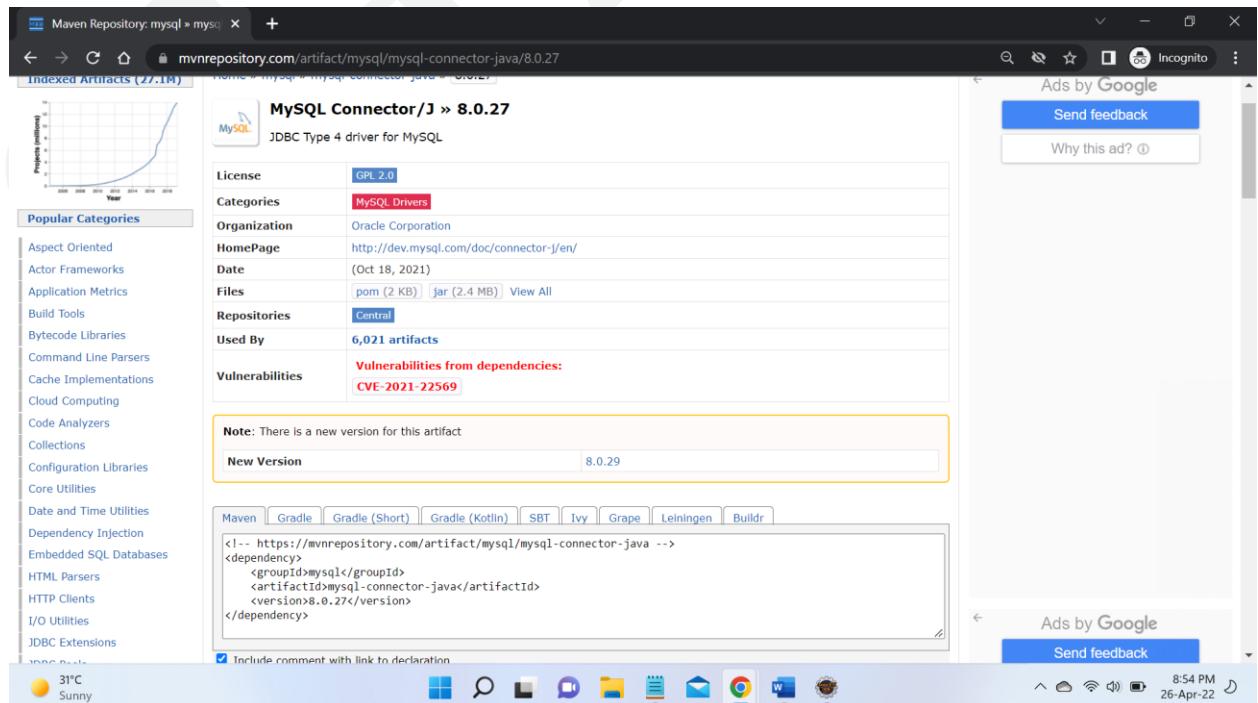
File Edit Navigate Search Project Run Design Window Help

ProjectwithMaven1/pom.xml

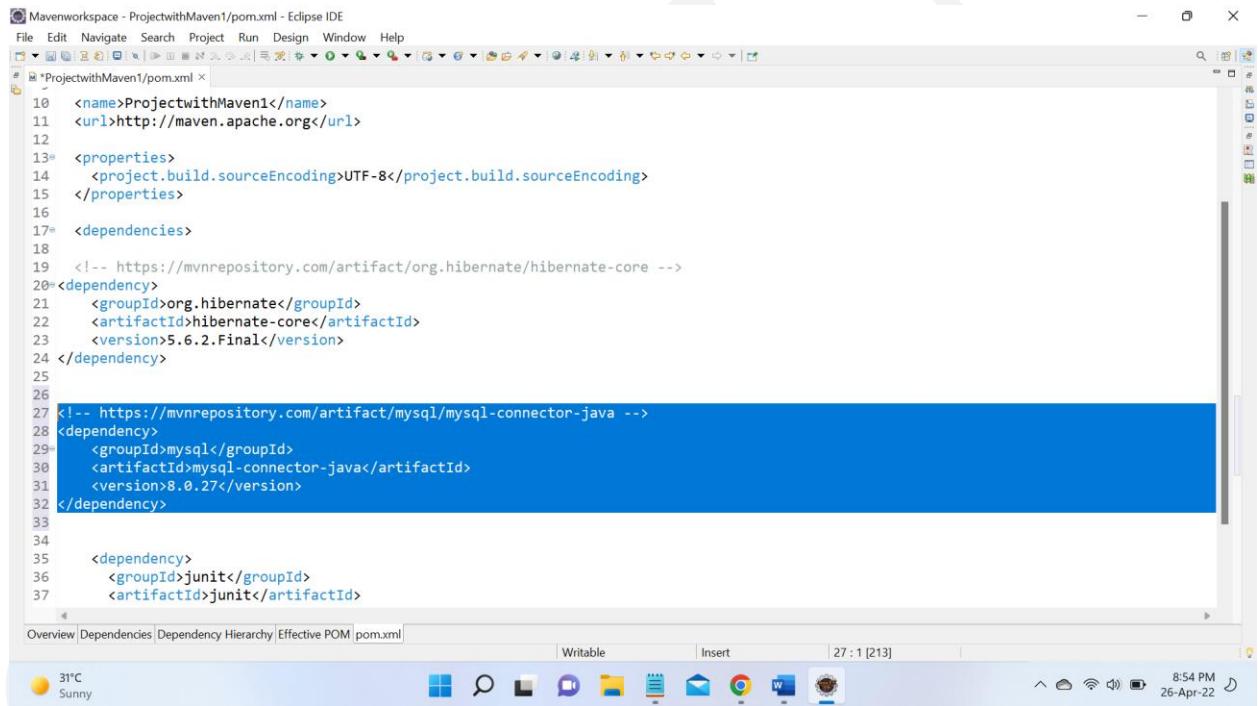
```
1<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
3  <modelVersion>4.0.0</modelVersion>
4
5  <groupId>com.tut1</groupId>
6  <artifactId>ProjectwithMaven1</artifactId>
7  <version>0.0.1-SNAPSHOT</version>
8  <packaging>jar</packaging>
9
10 <name>ProjectwithMaven1</name>
11 <url>http://maven.apache.org</url>
12
13<properties>
14   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15 </properties>
16
17<dependencies>
18
19  <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
20<dependency>
21   <groupId>org.hibernate</groupId>
22   <artifactId>hibernate-core</artifactId>
23   <version>5.6.2.Final</version>
24 </dependency>
25
26<
27 <dependency>
28   <groupId>junit</groupId>
29   <artifactId>junit</artifactId>
30 </dependency>
```

- viii. Again do same for mysql maven and paste it on pom.xml under <dependencies> tag from below website

<https://mvnrepository.com/artifact/mysql/mysql-connector-java/8.0.27>



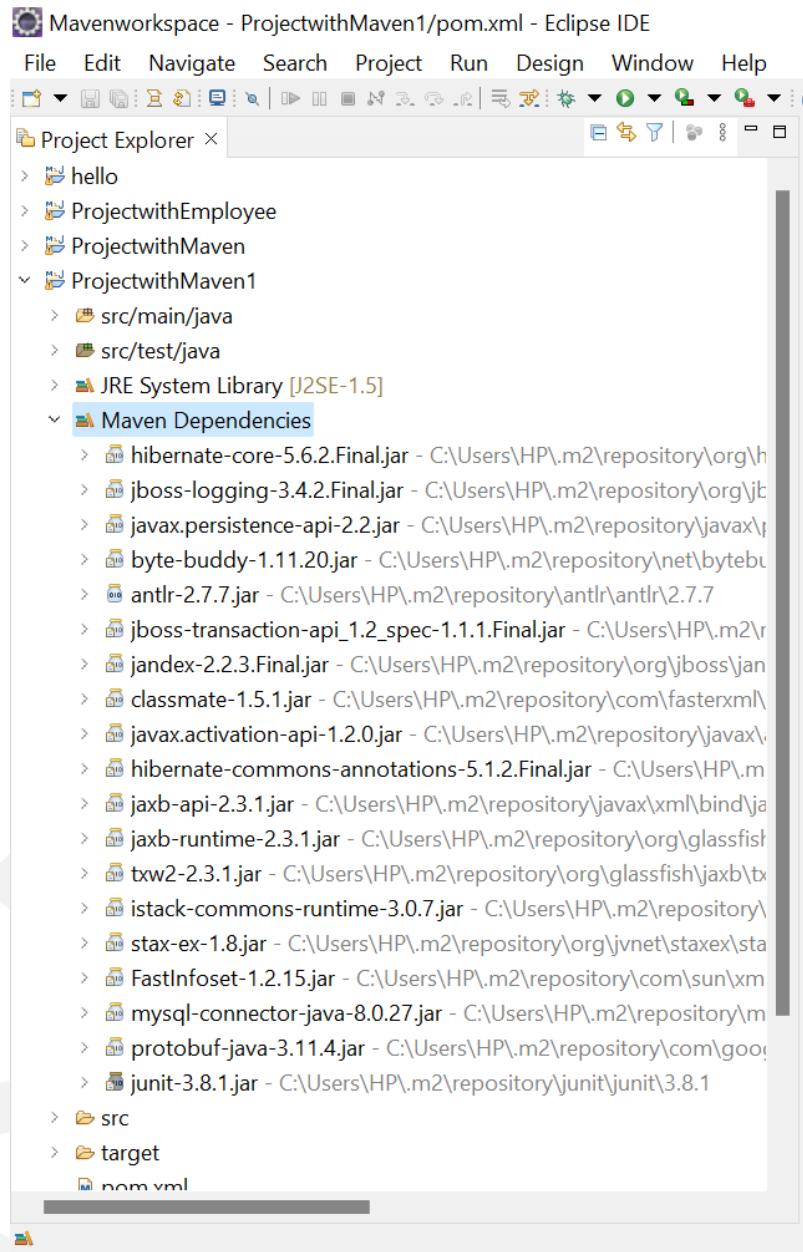
```
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>8.0.27</version>
</dependency>
```



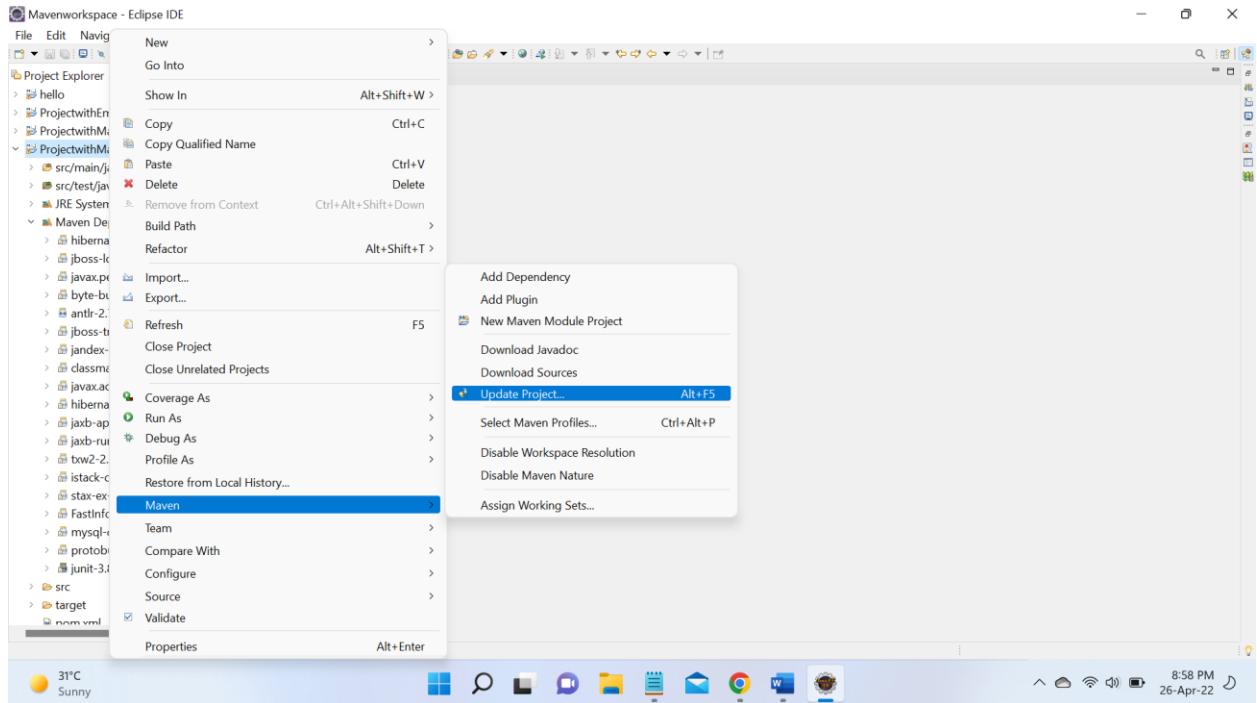
The screenshot shows the Eclipse IDE interface with the 'pom.xml' file open in the central editor window. The dependency code for MySQL connector Java is highlighted with a blue rectangular selection. The code itself is as follows:

```
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>8.0.27</version>
</dependency>
```

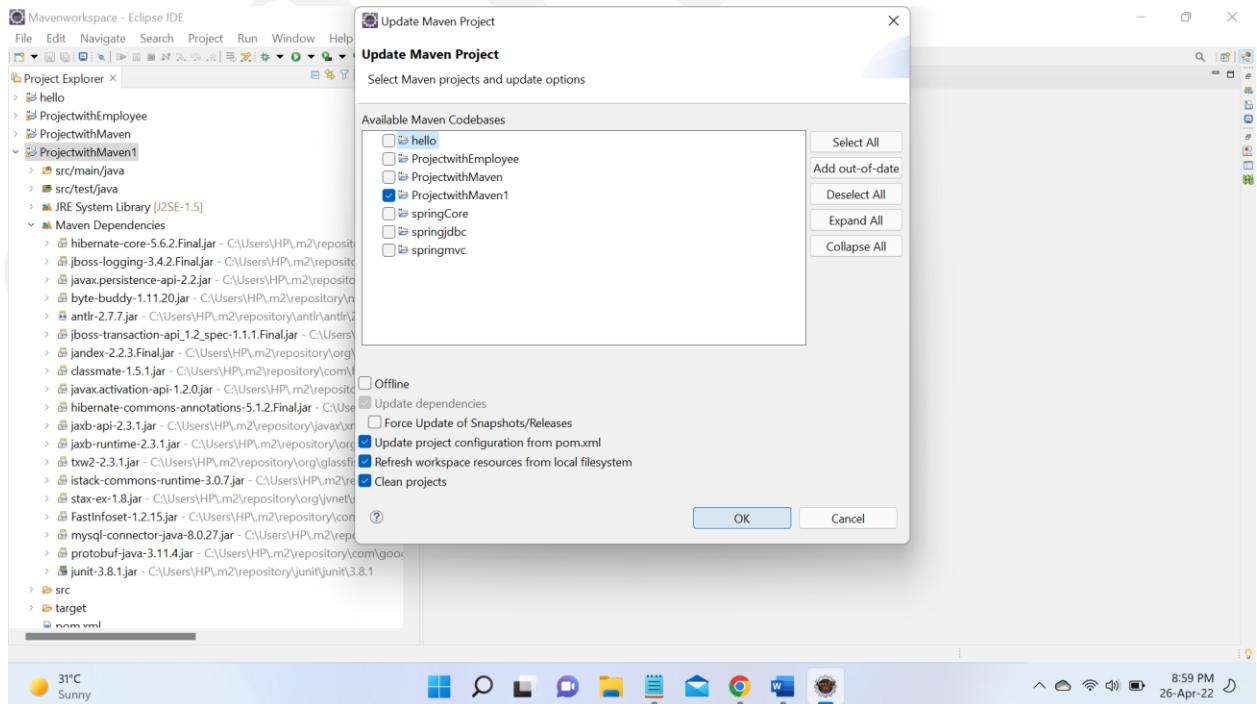
ix. Explore project and all the dependencies will be auto downloaded



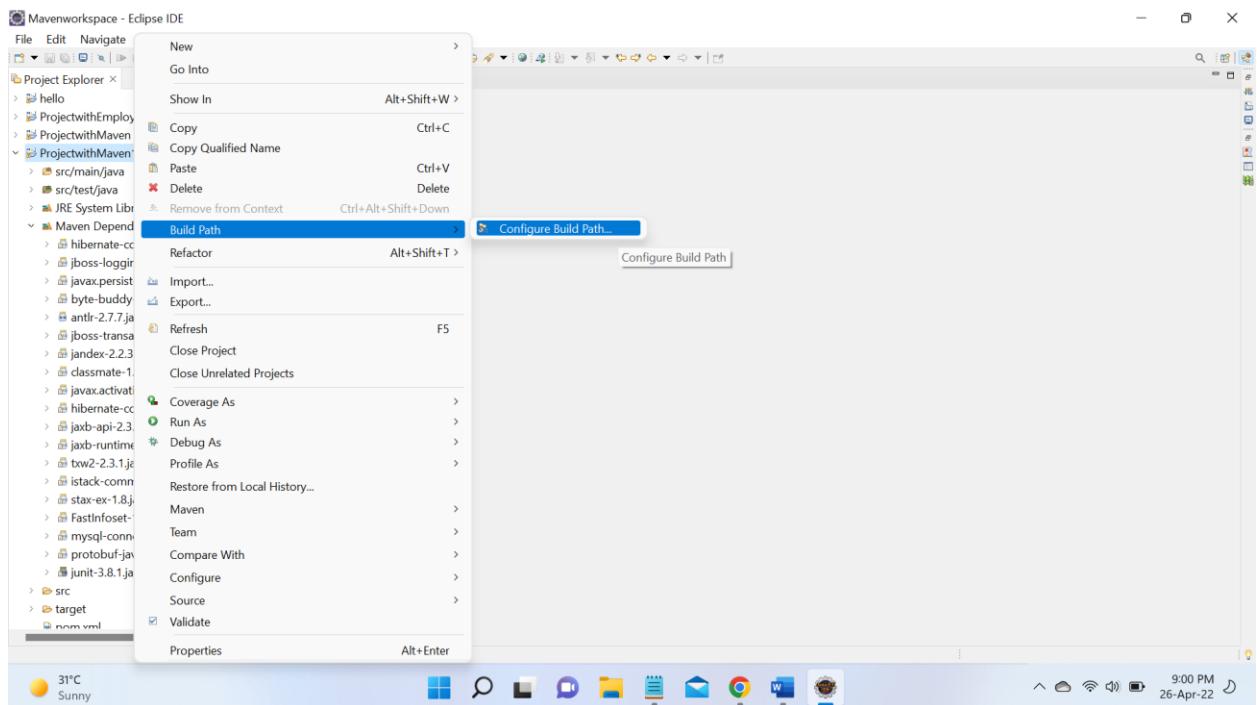
- x. Now right click on project => click on maven => update project



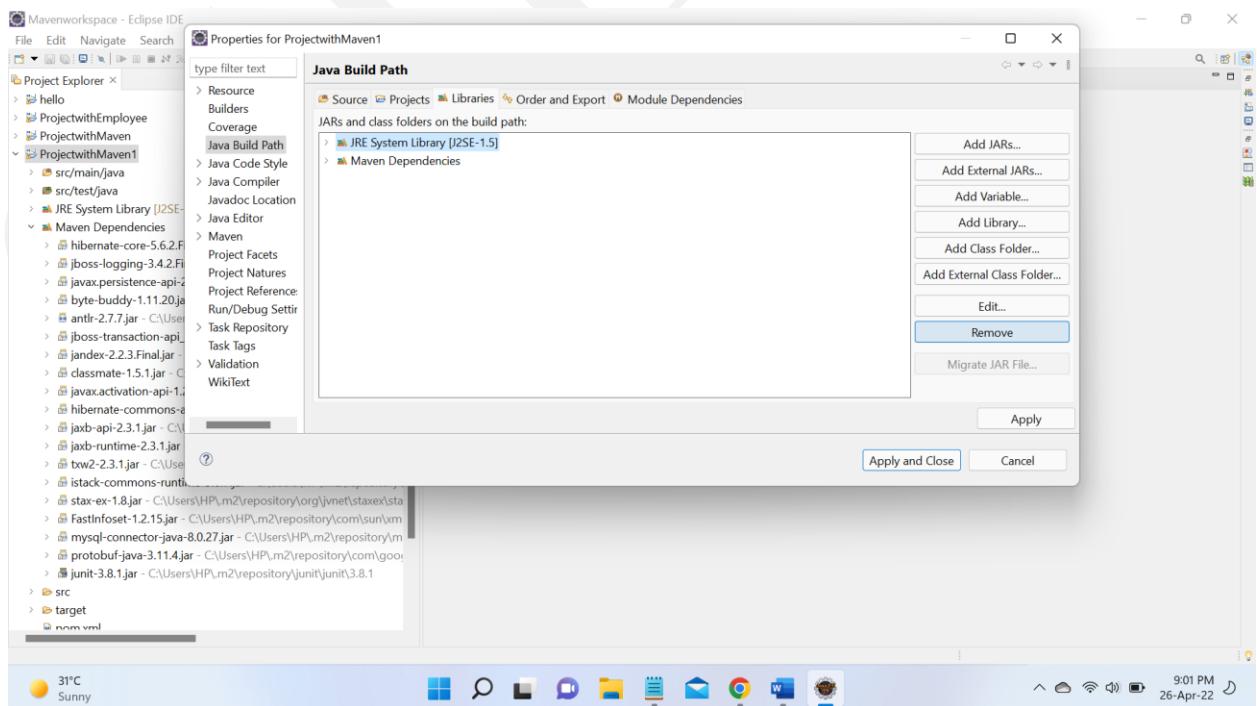
- xi. Click ok



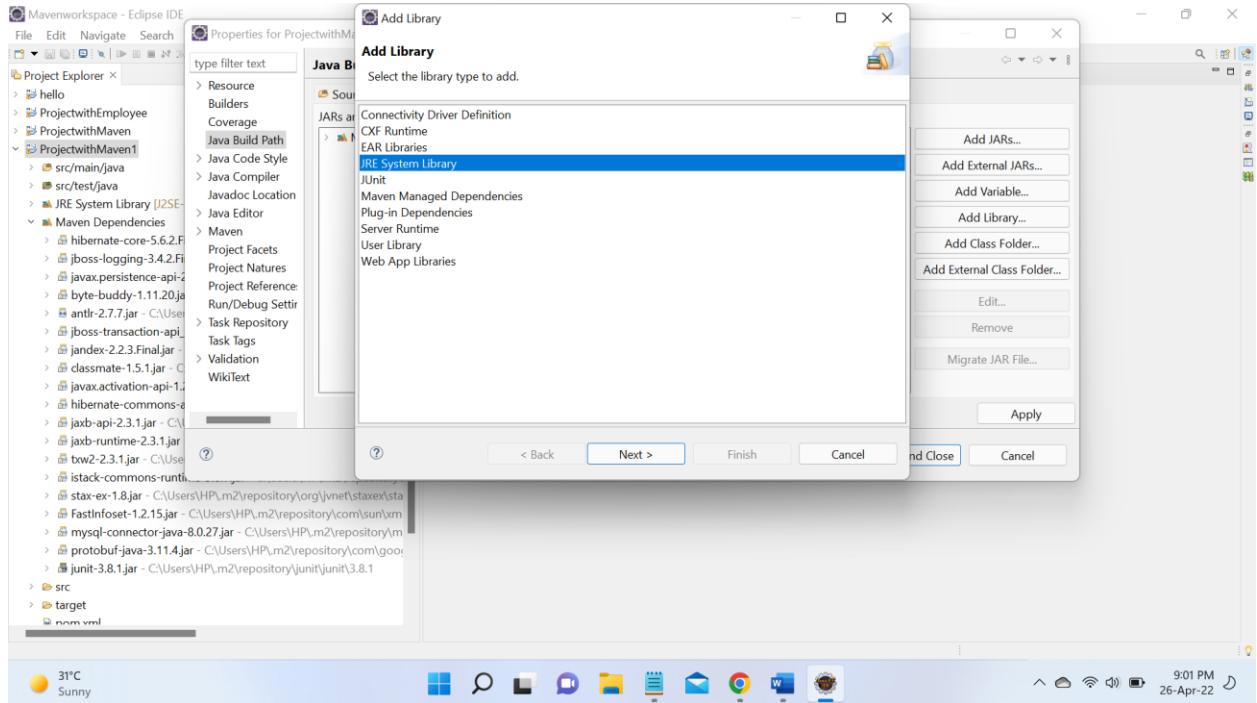
- xii. To change JRE environment of your own => right click JRE system => build path => configure build path



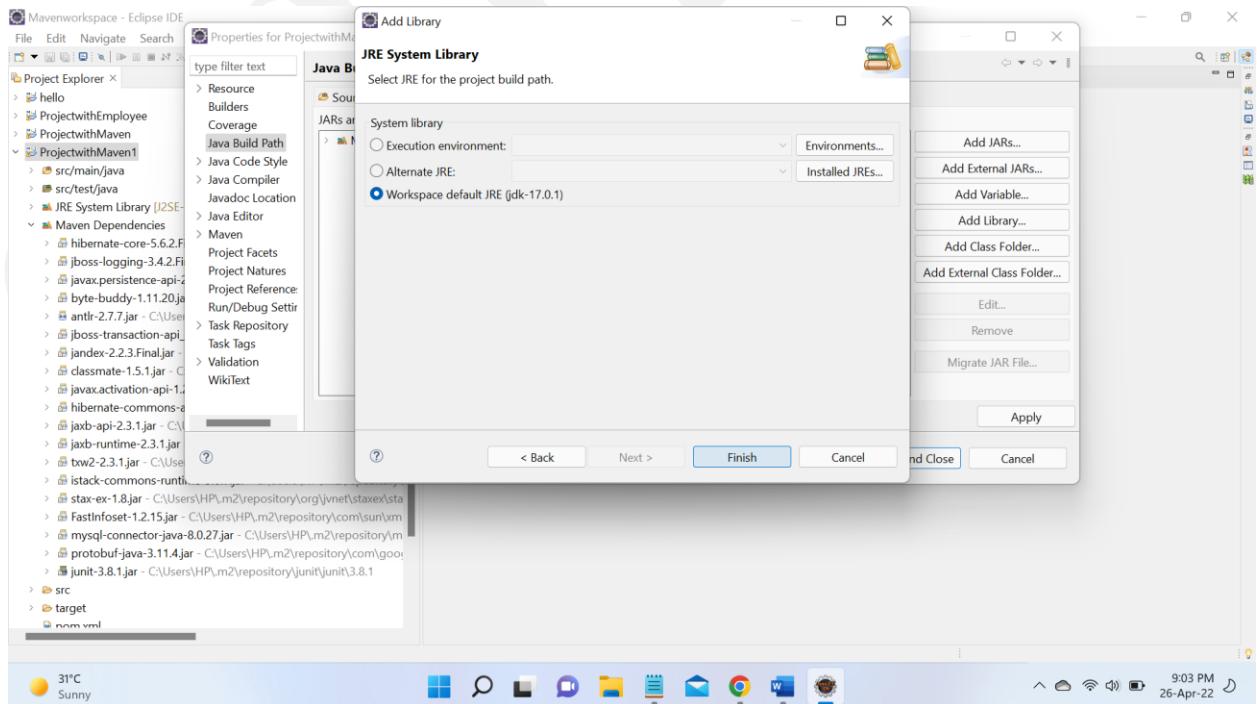
- xiii. Select JRE system & click on remove



xiv. Now click add library => click JRE system library => click Next



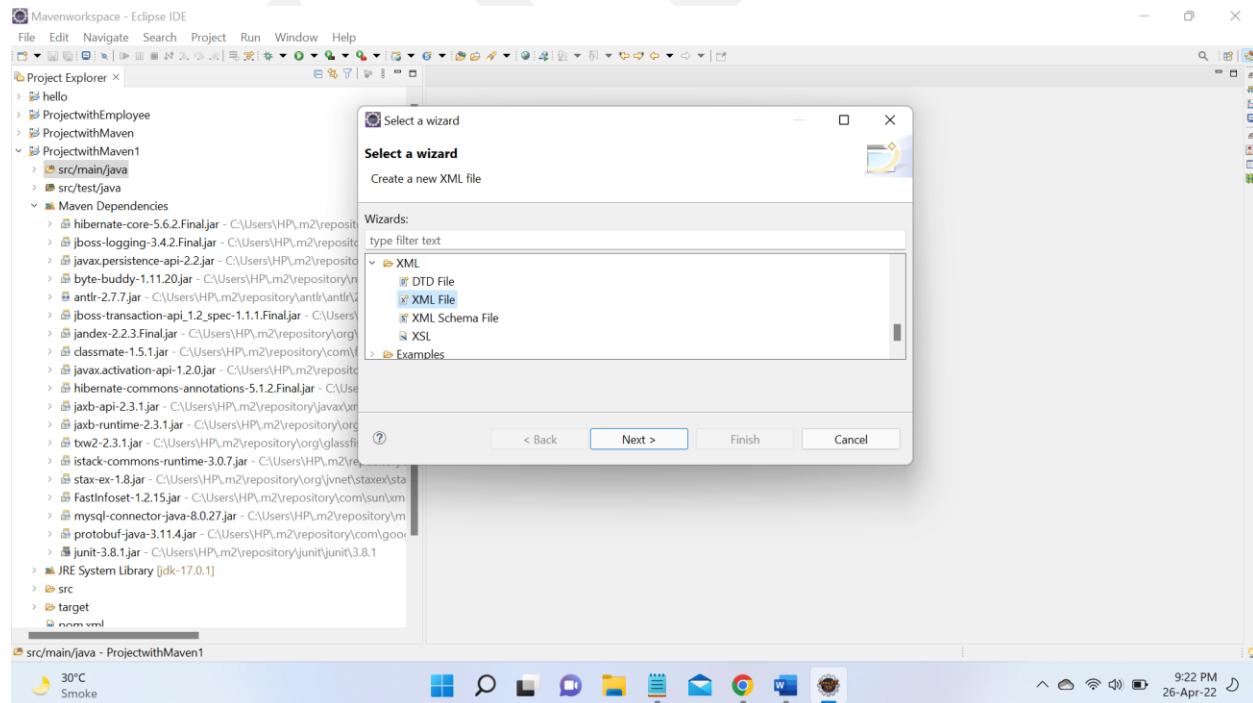
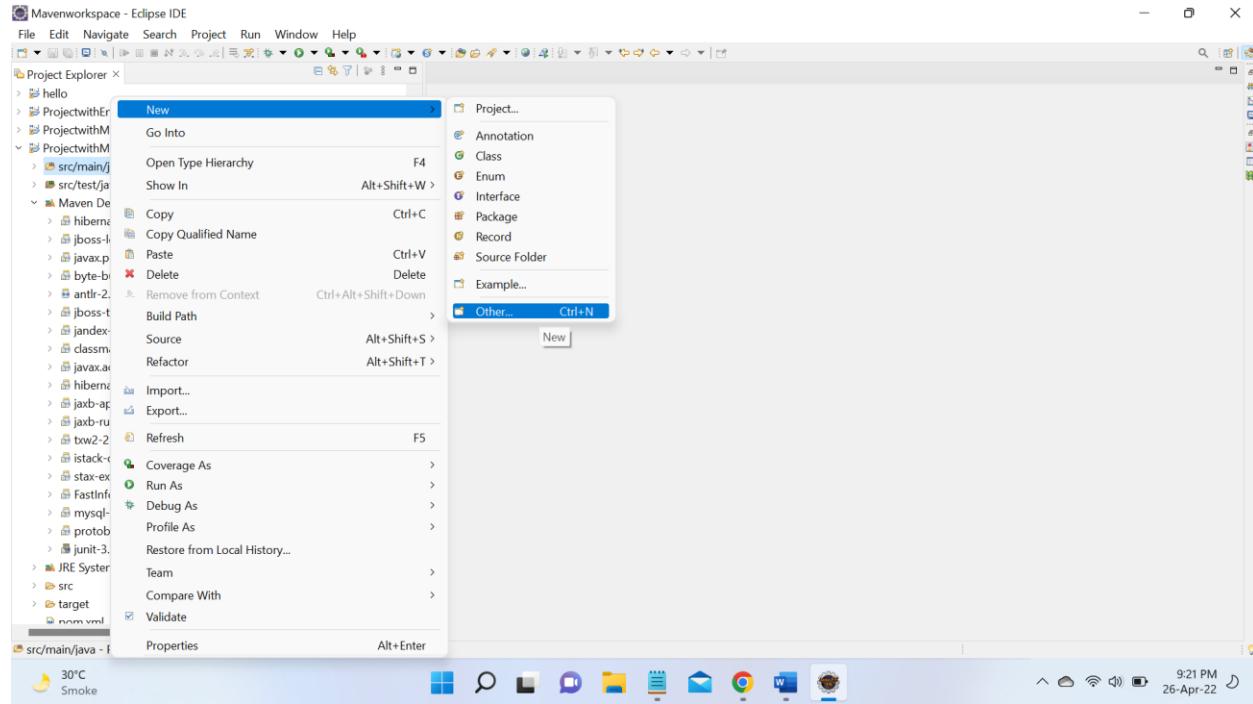
xv. click workspace default library (jdk 17 or whatever user has) => click finish



xvi. Click on apply => apply and close

B. Configure hibernate program

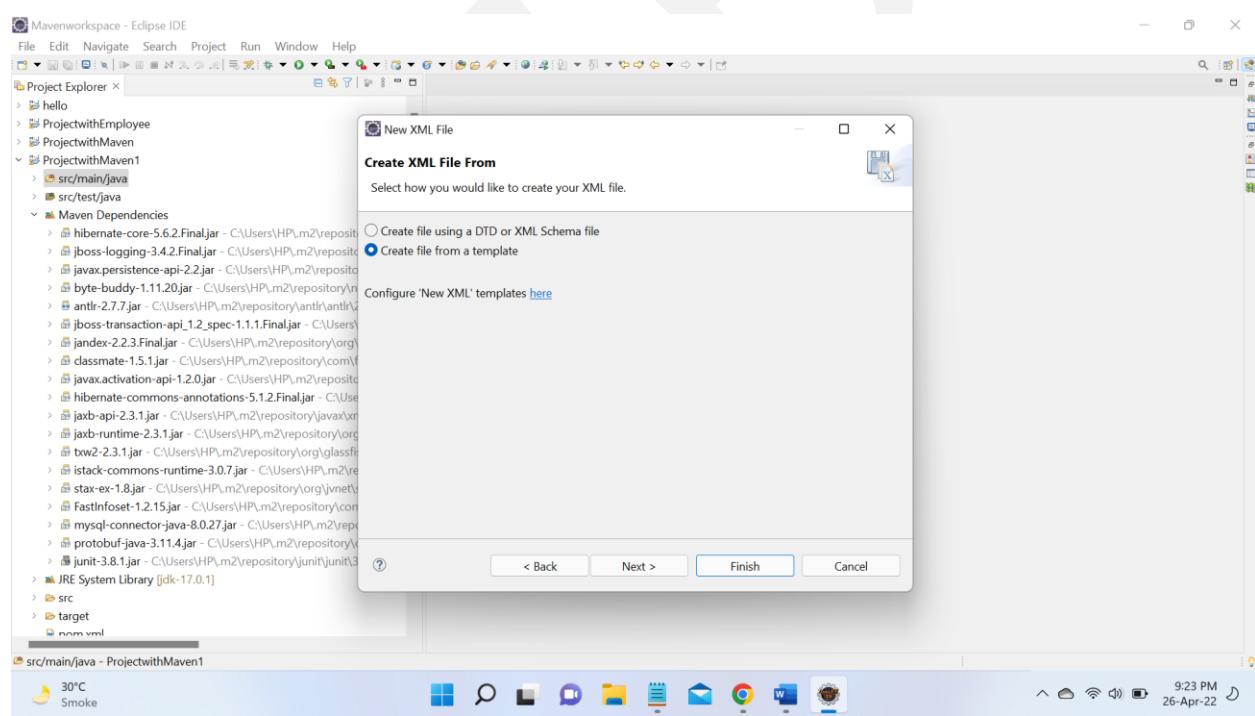
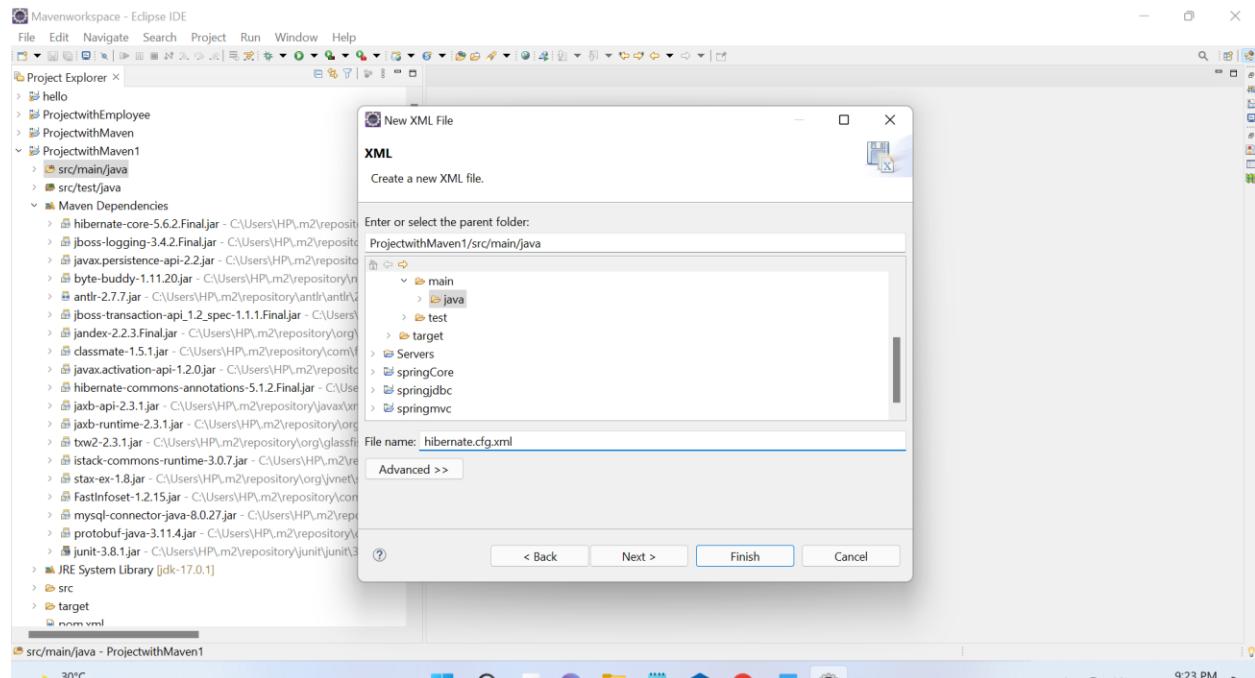
- Right click on src/java/main
- New => XML file => name it hibernate.cfg.xml => Next => click on Finish



Name: Shibu Mohapatra

Branch: MSC AI

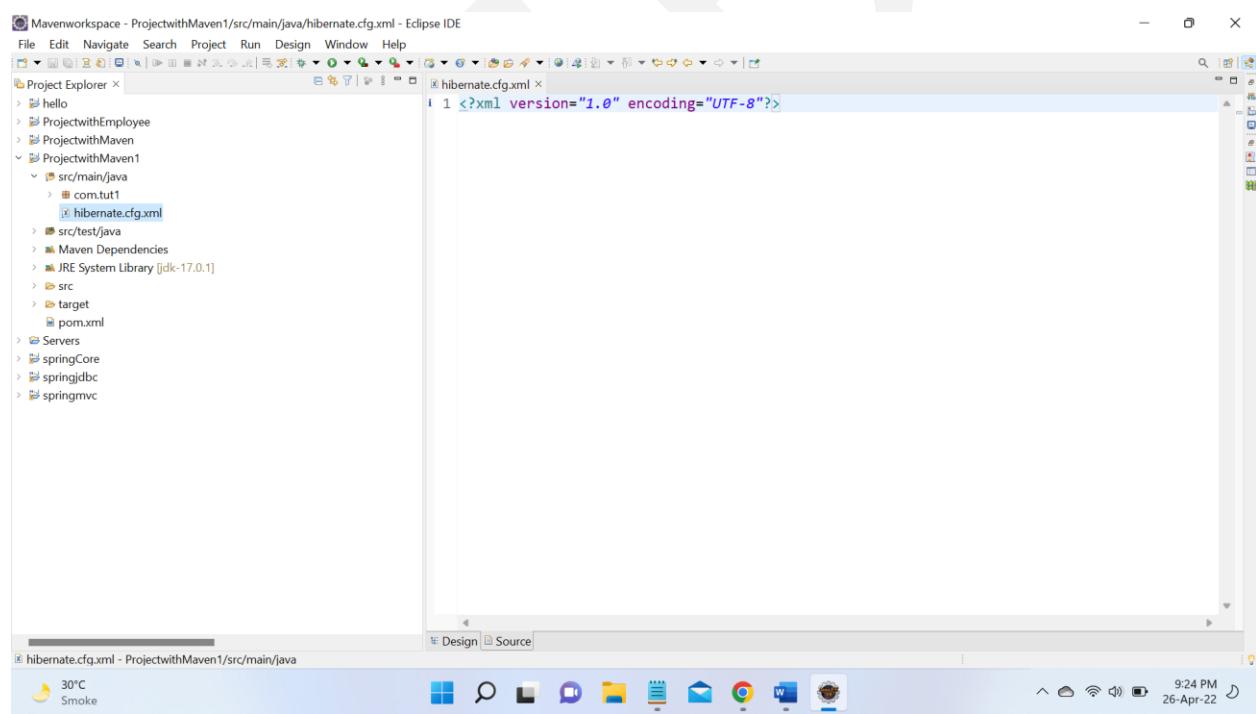
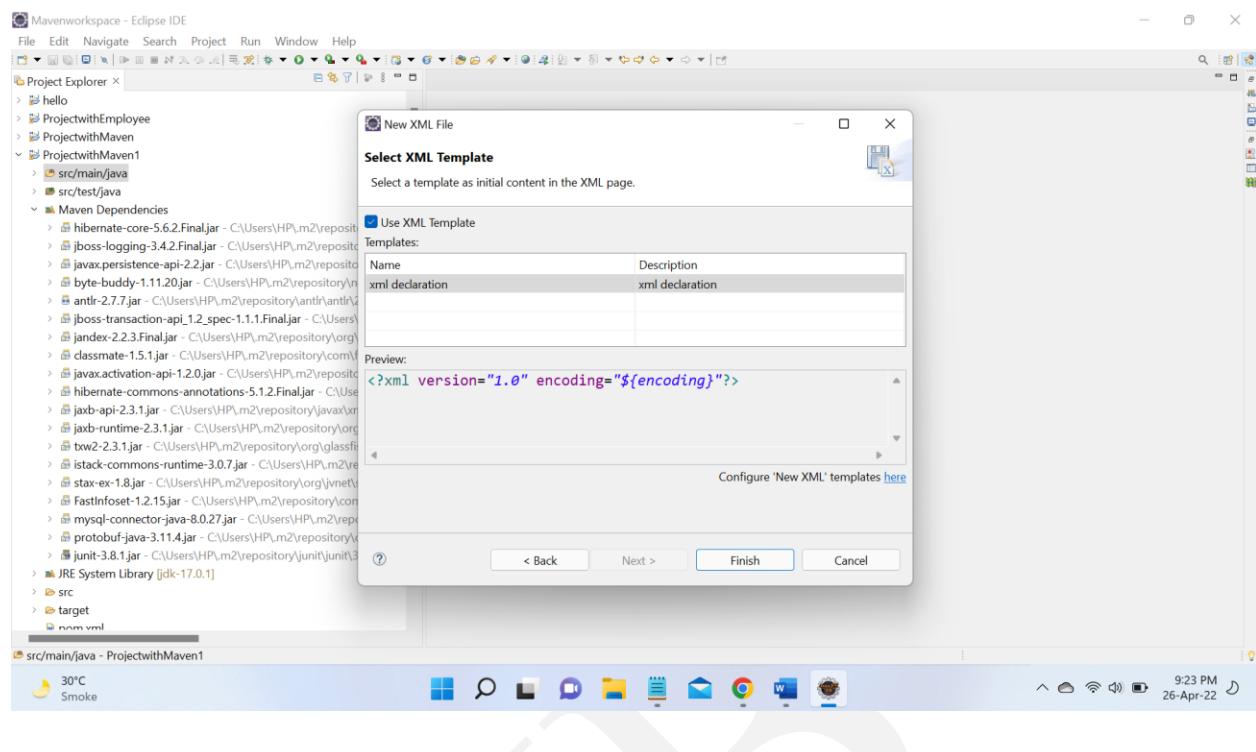
Roll no: 02



Name: Shibu Mohapatra

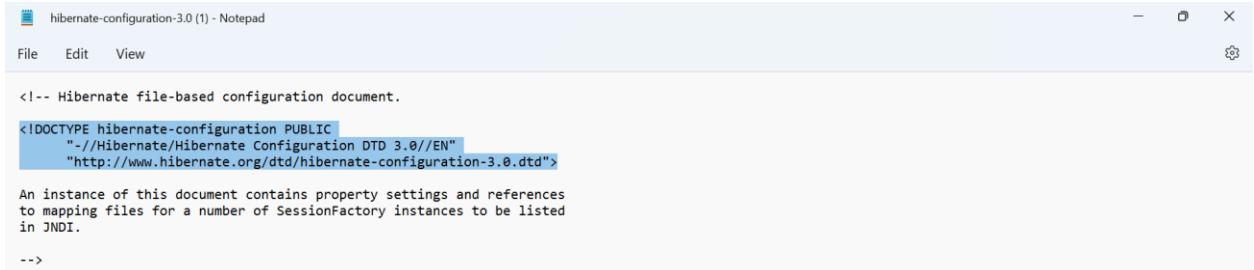
Branch: MSC AI

Roll no: 02



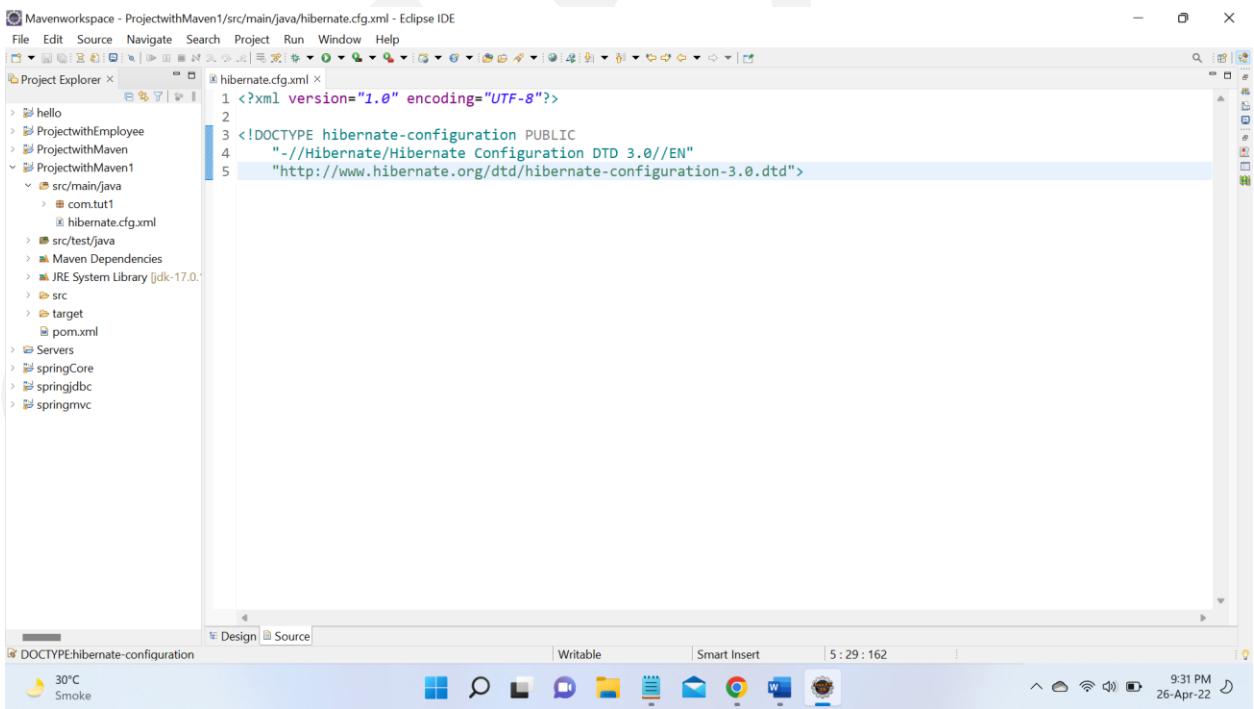
iii. Now browse to the website - <https://hibernate.org/dtd/> and select [hibernate-configuration-3.0.dtd](#), after clicking it will download automatically

iv. From downloaded DTD file, copy these lines & paste it on hibernate.cfg.xml file:



```
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
```

```
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
```



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
```

v. Now write further code for configuration in the file –

Like for Database, Image displaying and etc

3. Write hibernate program to save student data using annotation**Code:****App.java:**

```
package com.tut;

import java.io.IOException;
import java.util.Date;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

/**
 * Hello world!
 *
 */
public class App {

    public static void main(String[] args) throws IOException {
        System.out.println("Project started..");
        Configuration cfg = new Configuration();
        cfg.configure("hibernate.cfg.xml");
        SessionFactory factory = cfg.buildSessionFactory();

        System.out.println(factory);

        // creating student
        Student st = new Student();
        st.setId(101);
        st.setName("Shibu");
        st.setCity("Mumbai");
        System.out.println(st);

        // create object of address class
        Address ad = new Address();
        ad.setStreet("strrt1");
        ad.setCity("Mumbai");
        ad.setOpen(true);
        ad.setAddedDate(new Date());
        ad.setX(1234);
```

```
Session session = factory.openSession();
Transaction tx = session.beginTransaction();
session.save(st);

session.save(ad);
tx.commit();
session.close();

System.out.println("Done..");
}

}
```

Student.java:

```
package com.tut;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Student {

    @Id
    private int id;
    private String name;
    private String city;

    public Student() {
        super();
        // TODO Auto-generated constructor stub
    }

    public Student(int id, String name, String city) {
        super();
        this.id = id;
        this.name = name;
        this.city = city;
    }

    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 getCity() {
    return city;
}

public void setCity(String city) {
    this.city = city;
}

@Override
public String toString() {
    // TODO Auto-generated method stub
    return this.id + " :" + this.name + " :" + this.city;
}

}
```

Address.java:

```
package com.tut;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Lob;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.persistence.Transient;
```

```
@Entity
@Table(name = "Student_Address")
public class Address {

    @Column(name = "addressId")
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private int addressId;

    @Column(length = 50, name = "STREET")
    private String street;

    @Column(length = 100, name = "CITY")
    private String city;

    @Column(name = "is_open")
    private boolean isOpen;

    @Transient
    private double x;

    @Column(name = "added_time")
    @Temporal(TemporalType.DATE)
    private Date addedDate;

    @Lob
    private byte[] image;

    public Address() {
        super();
        // TODO Auto-generated constructor stub
    }

    public Address(int addressId, String street, String city, boolean isOpen, double x, Date
addedDate, byte[] image) {
        super();
        this.addressId = addressId;
        this.street = street;
        this.city = city;
        this.isOpen = isOpen;
        this.x = x;
        this.addedDate = addedDate;
        this.image = image;
    }
}
```

```
public int getAddressId() {
    return addressId;
}

public void setAddressId(int addressId) {
    this.addressId = addressId;
}

public String getStreet() {
    return street;
}

public void setStreet(String street) {
    this.street = street;
}

public String getCity() {
    return city;
}

public void setCity(String city) {
    this.city = city;
}

public boolean isOpen() {
    return isOpen;
}

public void setOpen(boolean isOpen) {
    this.isOpen = isOpen;
}

public double getX() {
    return x;
}

public void setX(double x) {
    this.x = x;
}

public Date getAddedDate() {
    return addedDate;
}

public void setAddedDate(Date addedDate) {
    this.addedDate = addedDate;
}
```

```
}

public byte[] getImage() {
    return image;
}

public void setImage(byte[] image) {
    this.image = image;
}

}
```

Hibernate.cfg.xml (CONFIGURATION FILE):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
    <session-factory>
        <property name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property
name="connection.url">jdbc:mysql://localhost:3306/myhiber01</property>
        <property name="connection.username">root</property>
        <property name="connection.password">shibu@ninja</property>
        <property name="dialect">org.hibernate.dialect.MySQL5Dialect</property>
        <property name="hbm2ddl.auto">create</property>
        <property name="show_sql">true</property>

        <mapping class="com.tut.Student" />
        <mapping class="com.tut.Address" />

    </session-factory>
</hibernate-configuration>
```

Pom.xml:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.tut</groupId>
  <artifactId>ProjectwithMaven</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>ProjectwithMaven</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>

    <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
    <dependency>
      <groupId>org.hibernate</groupId>
      <artifactId>hibernate-core</artifactId>
      <version>5.6.2.Final</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>8.0.27</version>
    </dependency>

    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

MySQL Database:**Step 1: Create database**

```
MySQL 8.0 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 8.0.28 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database      |
+-----+
| employee      |
| information_schema |
| myhiber       |
| mysql          |
| performance_schema |
| spring         |
| springjdbc    |
| sys            |
+-----+
8 rows in set (0.03 sec)

mysql> create database myhiber01;
Query OK, 1 row affected (0.18 sec)
```

33°C
Smoke

**Step 2: Create table student**

```
MySQL 8.0 Command Line Client

mysql> create table student(id int primary key,city varchar(100), name varchar(100) not null);
ERROR 1046 (3D000): No database selected
mysql> use myhiber01;
Database changed
mysql> create table student(id int primary key,city varchar(100), name varchar(100) not null);
Query OK, 0 rows affected (1.24 sec)

mysql> show tables;
+-----+
| Tables_in_myhiber01 |
+-----+
| student           |
+-----+
1 row in set (0.02 sec)

mysql>
```

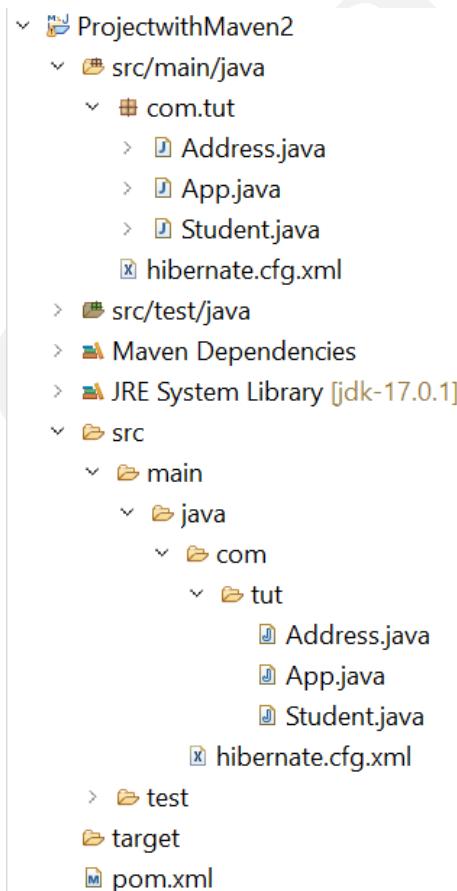
Step 3: Create table student_address;

```
mysql> create table student_address(addressId int primary key, added_time date, CITY varchar(100), image longblob, is_open bit(1), STREET varchar(50));
Query OK, 0 rows affected (0.93 sec)

mysql> desc student_address;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key  | Default | Extra |
+-----+-----+-----+-----+-----+
| addressId | int    | NO   | PRI   | NULL    |       |
| added_time | date   | YES  |       | NULL    |       |
| CITY        | varchar(100) | YES  |       | NULL    |       |
| image       | longblob | YES  |       | NULL    |       |
| is_open     | bit(1)  | YES  |       | NULL    |       |
| STREET      | varchar(50) | YES  |       | NULL    |       |
+-----+-----+-----+-----+-----+
6 rows in set (0.29 sec)

mysql>
```

Now the database and the tables have been created, switch back to Eclipse and run App.java (run java application) to display the output.

Eclipse Output:

Mavenworkspace - ProjectwithMaven2/src/main/java/com/tut/Address.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

App.java Student.java hibernate.cfg.xml ProjectwithMaven2/pom.xml Address.java

```
1 package com.tut;
```

Servers Snippets Console ×

<terminated> App (5) [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (Apr 28, 2022, 9:02:45 AM – 9:02:53 AM)

Project started..

Apr 28, 2022 9:02:46 AM org.hibernate.Version logVersion

```
Hibernate: drop table if exists Student
Apr 28, 2022 9:02:49 AM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionPr
Hibernate: drop table if exists Student_Address
Hibernate: create table Student (id integer not null, city varchar(255), name varchar(255), primary key (id)) engine=MyISAM
Apr 28, 2022 9:02:52 AM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionPr
Hibernate: create table Student_Address (addressId integer not null auto_increment, added_time date, CITY varchar(100), image longblob, is_ope
Apr 28, 2022 9:02:52 AM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
org.hibernate.internal.SessionFactoryImpl@1e1b061
101 :Shibu : Mumbai
Hibernate: insert into Student (city, name, id) values (?, ?, ?)
Hibernate: insert into Student_Address (added_time, CITY, image, is_open, STREET) values (?, ?, ?, ?, ?)
Done..|
```

33°C Smoke 9:05 AM 28-Apr-22

MySQL Output:

MySQL 8.0 Command Line Client

```
mysql> select * from student;
+----+-----+-----+
| id | city | name |
+----+-----+-----+
| 101 | Mumbai | Shibu |
+----+-----+-----+
1 row in set (0.00 sec)

mysql> select * from student_address;
+-----+-----+-----+-----+-----+
| addressId | added_time | CITY | image | is_open | STREET |
+-----+-----+-----+-----+-----+
| 1 | 2022-04-28 | Mumbai | NULL | 0x01 | strrt1 |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

33°C Smoke 9:14 AM 28-Apr-22

4. Write hibernate program to save image in database**Code:****App.java:**

```
package com.tut;

import java.io.FileInputStream;
import java.io.IOException;
import java.util.Date;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

/**
 * Hello world!
 *
 */
public class App {

    public static void main(String[] args) throws IOException {
        System.out.println("Project started..");
        Configuration cfg = new Configuration();
        cfg.configure("hibernate.cfg.xml");
        SessionFactory factory = cfg.buildSessionFactory();

        System.out.println(factory);

        // creating student
        Student st = new Student();
        st.setId(101);
        st.setName("Shibu");
        st.setCity("Mumbai");
        System.out.println(st);

        // create object of address class
        Address ad = new Address();
        ad.setStreet("strrt1");
        ad.setCity("Mumbai");
        ad.setOpen(true);
        ad.setAddedDate(new Date());
        ad.setX(1234);
```

```
// reading an image
FileInputStream fis = new FileInputStream("src/main/java/pic.jpeg");
byte[] data = new byte[fis.available()];
fis.read(data);
ad.setImage(data);

Session session = factory.openSession();
Transaction tx = session.beginTransaction();
session.save(st);

session.save(ad);
tx.commit();
session.close();

System.out.println("Done..");
}

}
```

Student.java:

```
package com.tut;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Student {

    @Id
    private int id;
    private String name;
    private String city;

    public Student() {
        super();
        // TODO Auto-generated constructor stub
    }

    public Student(int id, String name, String city) {
        super();
        this.id = id;
        this.name = name;
        this.city = city;
    }
}
```

```
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 getCity() {
    return city;
}

public void setCity(String city) {
    this.city = city;
}

@Override
public String toString() {
    // TODO Auto-generated method stub
    return this.id + ":" + this.name + ":" + this.city;
}
}
```

Address.java:

```
package com.tut;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
```

```
import javax.persistence.Lob;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.persistence.Transient;

@Entity
@Table(name = "Student_Address")
public class Address {

    @Column(name = "addressId")
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private int addressId;

    @Column(length = 50, name = "STREET")
    private String street;

    @Column(length = 100, name = "CITY")
    private String city;

    @Column(name = "is_open")
    private boolean isOpen;

    @Transient
    private double x;

    @Column(name = "added_time")
    @Temporal(TemporalType.DATE)
    private Date addedDate;

    @Lob
    private byte[] image;

    public Address() {
        super();
        // TODO Auto-generated constructor stub
    }

    public Address(int addressId, String street, String city, boolean isOpen, double x, Date
addedDate, byte[] image) {
        super();
        this.addressId = addressId;
        this.street = street;
        this.city = city;
    }
}
```

```
this.isOpen = isOpen;
this.x = x;
this.addedDate = addedDate;
this.image = image;
}

public int getAddressId() {
    return addressId;
}

public void setAddressId(int addressId) {
    this.addressId = addressId;
}

public String getStreet() {
    return street;
}

public void setStreet(String street) {
    this.street = street;
}

public String getCity() {
    return city;
}

public void setCity(String city) {
    this.city = city;
}

public boolean isOpen() {
    return isOpen;
}

public void setOpen(boolean isOpen) {
    this.isOpen = isOpen;
}

public double getX() {
    return x;
}

public void setX(double x) {
    this.x = x;
}
```

```
public Date getAddedDate() {
    return addedDate;
}

public void setAddedDate(Date addedDate) {
    this.addedDate = addedDate;
}

public byte[] getImage() {
    return image;
}

public void setImage(byte[] image) {
    this.image = image;
}

}
```

Hibernate.cfg.xml (CONFIGURATION FILE):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
    <session-factory>
        <property name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property
name="connection.url">jdbc:mysql://localhost:3306/myhiber</property>
        <property name="connection.username">root</property>
        <property name="connection.password">shibu@ninja</property>
        <property name="dialect">org.hibernate.dialect.MySQL5Dialect</property>
        <property name="hbm2ddl.auto">create</property>
        <property name="show_sql">true</property>

        <mapping class="com.tut.Student" />
        <mapping class="com.tut.Address" />

    </session-factory>
</hibernate-configuration>
```

Pom.xml:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
  4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.tut</groupId>
  <artifactId>ProjectwithMaven</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>ProjectwithMaven</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>

    <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
    <dependency>
      <groupId>org.hibernate</groupId>
      <artifactId>hibernate-core</artifactId>
      <version>5.6.2.Final</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>8.0.27</version>
    </dependency>

    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Eclipse Output:

Eclipse IDE Screenshot:

The screenshot shows the Eclipse IDE interface. The Project Explorer on the left displays the project structure:

```

ProjectwithMaven
  - src/main/java
    - com.tut
      - Address.java
      - App.java
      - Student.java
      - hibernate.cfg.xml
      - pic.jpeg
      - shibu.jpg
    - src/test/java
    - Maven Dependencies
    - JRE System Library [jdk-17.0.1]
  - src
    - main
      - com
        - tut
          - Address.java
          - App.java
          - Student.java
          - hibernate.cfg.xml
          - pic.jpeg
          - shibu.jpg
      - test
  - target
  - pom.xml

```

The Java Editor (bottom) shows the `App.java` file:

```

package com.tut;

public class App {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}

```

The Console tab shows the output of the application run:

```

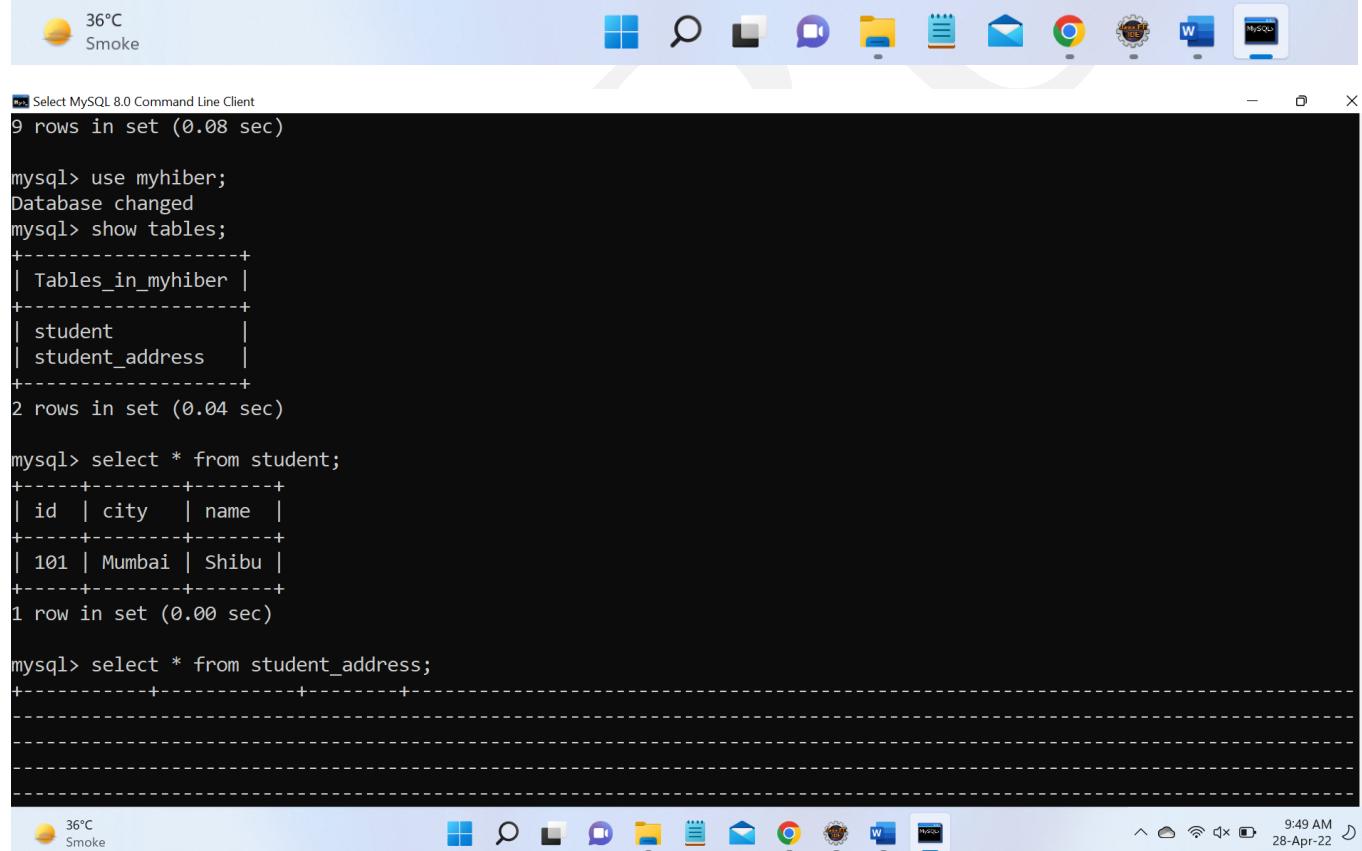
Project started..
Apr 28, 2022 9:46:45 AM org.hibernate.Version logVersion
Hibernate: drop table if exists Student
Apr 28, 2022 9:46:45 AM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JDBCConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionPr
Hibernate: drop table if exists Student_Address
Hibernate: create table Student (id integer not null, city varchar(255), name varchar(255), primary key (id)) engine=MyISAM
Apr 28, 2022 9:46:45 AM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JDBCConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionPr
Hibernate: create table Student_Address (addressId integer not null auto_increment, added_time date, CITY varchar(100), image longblob, is_o
Apr 28, 2022 9:46:45 AM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
org.hibernate.internal.SessionFactoryImpl@1e1b061
101 :Shibu : Mumbai
Hibernate: insert into Student (city, name, id) values (?, ?, ?)
Hibernate: insert into Student_Address (added_time, CITY, image, is_open, STREET) values (?, ?, ?, ?, ?)
Done..

```

The system tray at the bottom indicates it's 9:47 AM on April 28, 2022, with a temperature of 33°C and a smoke alert.

MySQL Database Output:

```
mysql> show databases;
+-----+
| Database |
+-----+
| employee |
| information_schema |
| myhiber |
| myhiber01 |
| mysql |
| performance_schema |
| spring |
| springjdbc |
| sys |
+-----+
```



```
Select MySQL 8.0 Command Line Client
9 rows in set (0.08 sec)

mysql> use myhiber;
Database changed
mysql> show tables;
+-----+
| Tables_in_myhiber |
+-----+
| student |
| student_address |
+-----+
2 rows in set (0.04 sec)

mysql> select * from student;
+---+---+---+
| id | city | name |
+---+---+---+
| 101 | Mumbai | Shibu |
+---+---+---+
1 row in set (0.00 sec)

mysql> select * from student_address;
```

36°C Smoke 9:49 AM 28-Apr-22

Name: **Shibu Mohapatra**

Branch: **MSC AI**

Roll no: **02**

SPRING**1. Create new Maven Project to add Spring Dependencies using setter injection.****Code:****Config.xml (CONFIGURATION FILE):**

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

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:p="http://www.springframework.org/schema/p"

       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- this is for bean -->

    <bean class="com.springCore.Student" name="student1">

        <!-- first way by using element tags -->

        <property name="studentId">
            <value>101</value>
        </property>

        <property name="studentName">
            <value>Shibu</value>
        </property>

        <property name="studentAddress">
            <value>California</value>
        </property>

    </bean>
</beans>
```

App.java:

```
package com.springCore;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Hello World!" );

        ApplicationContext context = new ClassPathXmlApplicationContext("config.xml");
        Student student1 = (Student)context.getBean("student1");
        System.out.println(student1);
    }
}
```

Student.java:

```
package com.springCore;

public class Student {

    private int studentId;
    private String studentName;
    private String studentAddress;

    public int getStudentId() {
        return studentId;
    }

    public void setStudentId(int studentId) {
        System.out.println("Setter ID");
        this.studentId = studentId;
    }
}
```

```
public String getStudentName() {
    return studentName;
}
public void setStudentName(String studentName) {
    System.out.println("Setter Name");
    this.studentName = studentName;
}
public String getStudentAddress() {
    return studentAddress;
}
public void setStudentAddress(String studentAddress) {
    System.out.println("Setter Address");
    this.studentAddress = studentAddress;
}

public Student(int studentId, String studentName, String studentAddress) {
    super();
    this.studentId = studentId;
    this.studentName = studentName;
    this.studentAddress = studentAddress;
}
public Student() {
    super();
    // TODO Auto-generated constructor stub
}

@Override
public String toString() {
    return "Student [studentId=" + studentId + ", studentName=" + studentName + ",
studentAddress=" + studentAddress
               + "]";
}
}
```

Pom.xml:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>

<groupId>com.springcore</groupId>
<artifactId>springCore</artifactId>
```

```
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>

<name>springCore</name>
<url>http://maven.apache.org</url>

<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>

<dependencies>

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>5.2.3.RELEASE</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.2.3.RELEASE</version>
    </dependency>

    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>3.8.1</version>
        <scope>test</scope>
    </dependency>
</dependencies>
</project>
```

Output:

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer View:** Displays the project structure for "springCore".
 - src/main/java/com.springCore:
 - App.java
 - Student.java
 - src/test/java
 - Maven Dependencies
 - JRE System Library [jdk-17.0.1]
 - src
 - target
- POM View:** Shows the pom.xml file.
- Run View:** Shows the output of the application.

```
<terminated> App (6) [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (Apr 29, 2022, 12:01:52 PM – 12:01:53 PM)
Hello World!
Setter ID
Setter Name
Setter Address
Student [studentId=101, studentName=Shibu, studentAddress=California]
```
- System Tray:** Shows weather (35°C Haze), taskbar icons (File Explorer, Search, Chat, Task View, File, Mail, Google Chrome, Microsoft Edge), and system status (Battery, Network, Volume, Date/Time).

2. Create new Maven Project for property injection using p Schema and using value as attribute**P Schema****Code:****Config.xml (CONFIGURATION FILE):**

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

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:p="http://www.springframework.org/schema/p"

       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- this is for bean -->
    <!-- using P schema -->

    <bean class="com.springCore.Student" name="student1"
          p:studentId="1001" p:studentName="Mohapatra" p:studentAddress="Mumbai">
        </bean>
</beans>
```

App.java:

```
package com.springCore;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

/**
 * Hello world!
 *
```

```
/*
public class App
{
    public static void main( String[] args )
    {

        ApplicationContext context = new ClassPathXmlApplicationContext("config.xml");
        Student student1 = (Student)context.getBean("student1");
        System.out.println(student1);
    }
}
```

Student.java:

```
package com.springCore;

public class Student {

    private int studentId;
    private String studentName;
    private String studentAddress;

    public int getStudentId() {
        return studentId;
    }

    public void setStudentId(int studentId) {
        System.out.println("Setter ID");
        this.studentId = studentId;
    }

    public String getStudentName() {
        return studentName;
    }

    public void setStudentName(String studentName) {
        System.out.println("Setter Name");
        this.studentName = studentName;
    }

    public String getStudentAddress() {
        return studentAddress;
    }

    public void setStudentAddress(String studentAddress) {
        System.out.println("Setter Address");
    }
}
```

```
        this.studentAddress = studentAddress;
    }

public Student(int studentId, String studentName, String studentAddress) {
    super();
    this.studentId = studentId;
    this.studentName = studentName;
    this.studentAddress = studentAddress;
}
public Student() {
    super();
    // TODO Auto-generated constructor stub
}

@Override
public String toString() {
    return "Student [studentId=" + studentId + ", studentName=" + studentName +",
studentAddress=" + studentAddress
            + "]";
}
}
```

Pom.xml:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.springcore</groupId>
    <artifactId>springCore</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>springCore</name>
    <url>http://maven.apache.org</url>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>

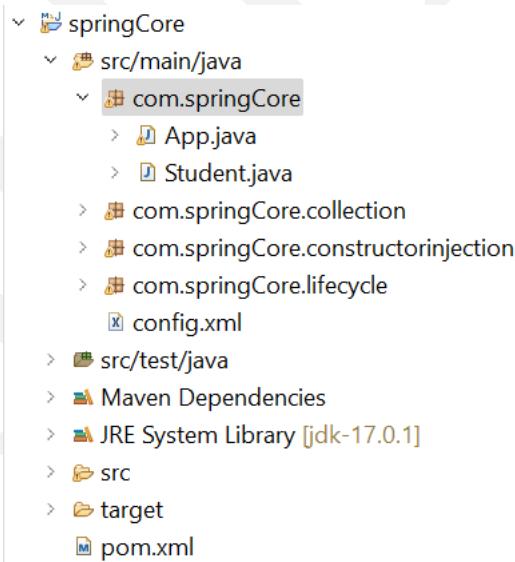
        <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
        <dependency>
```

```
<groupId>org.springframework</groupId>
<artifactId>spring-core</artifactId>
<version>5.2.3.RELEASE</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.2.3.RELEASE</version>
</dependency>

<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
</dependency>
</dependencies>
</project>
```

Output:



Name: **Shibu Mohapatra**

Branch: **MSC AI**

Roll no: **02**



Mavenworkspace - springCore/src/main/java/com/springCore/App.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Student.java config.xml App.java springCore/pom.xml

```
1 package com.springCore;
```

Servers Snippets Console

<terminated> App (6) [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (Apr 29, 2022, 12:11:56 PM – 12:11:57 PM)

Setter Address
Setter ID
Setter Name
Student [studentId=1001, studentName=Mohapatra, studentAddress=Mumbai]

35°C Haze 12:12 PM 29-Apr-22

The screenshot shows an Eclipse IDE interface with a Java project named "springCore". The code editor displays the "App.java" file with the following content:

```
1 package com.springCore;
```

The Java code defines a class "Student" with three fields: "studentId", "studentName", and "studentAddress", each with a corresponding setter method. The "studentId" field is annotated with "@GeneratedValue(strategy = GenerationType.IDENTITY)". The "studentName" and "studentAddress" fields are annotated with "@Column(name = "student_name")" and "@Column(name = "student_address")" respectively.

The "Console" tab shows the output of a Java application run. It displays the output of the setters being called for a new "Student" object with id 1001, name "Mohapatra", and address "Mumbai".

The system tray at the bottom shows the weather as 35°C Haze, the date and time as 12:12 PM on 29-Apr-22, and various system icons.

Value as Attribute**CODE:****Config.xml (CONFIGURATION FILE):**

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

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:p="http://www.springframework.org/schema/p"

       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd
                           http://www.springframework.org/schema/context
                           http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- this is for bean -->
    <!-- Using attribute value -->

    <bean class="com.springCore.Student" name="student1">
        <property name="studentId" value="202" />
        <property name="studentName" value="Shivaay" />
        <property name="studentAddress" value="London" />
    </bean>

</beans>
```

App.java:

```
package com.springCore;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

/**
 * Hello world!
 *
 */
```

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

        ApplicationContext context = new ClassPathXmlApplicationContext("config.xml");
        Student student1 = (Student)context.getBean("student1");
        System.out.println(student1);
    }
}
```

Student.java:

```
package com.springCore;

public class Student {

    private int studentId;
    private String studentName;
    private String studentAddress;

    public int getStudentId() {
        return studentId;
    }

    public void setStudentId(int studentId) {
        System.out.println("Setter ID");
        this.studentId = studentId;
    }

    public String getStudentName() {
        return studentName;
    }

    public void setStudentName(String studentName) {
        System.out.println("Setter Name");
        this.studentName = studentName;
    }

    public String getStudentAddress() {
        return studentAddress;
    }

    public void setStudentAddress(String studentAddress) {
        System.out.println("Setter Address");
        this.studentAddress = studentAddress;
    }
}
```

```

}

public Student(int studentId, String studentName, String studentAddress) {
    super();
    this.studentId = studentId;
    this.studentName = studentName;
    this.studentAddress = studentAddress;
}
public Student() {
    super();
    // TODO Auto-generated constructor stub
}

@Override
public String toString() {
    return "Student [studentId=" + studentId + ", studentName=" + studentName + ",
studentAddress=" + studentAddress
                + "]";
}
}

```

Pom.xml:

```

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.springcore</groupId>
    <artifactId>springCore</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>springCore</name>
    <url>http://maven.apache.org</url>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>

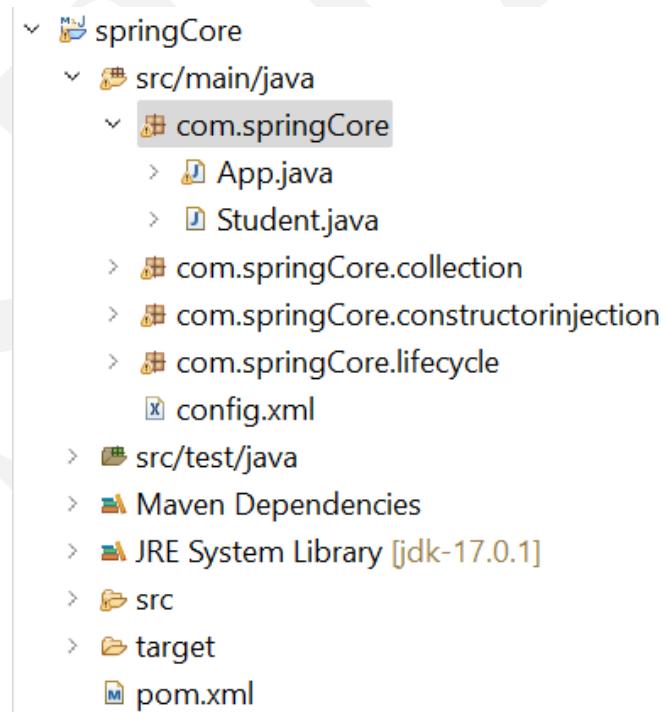
        <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>

```

```
<version>5.2.3.RELEASE</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.2.3.RELEASE</version>
</dependency>

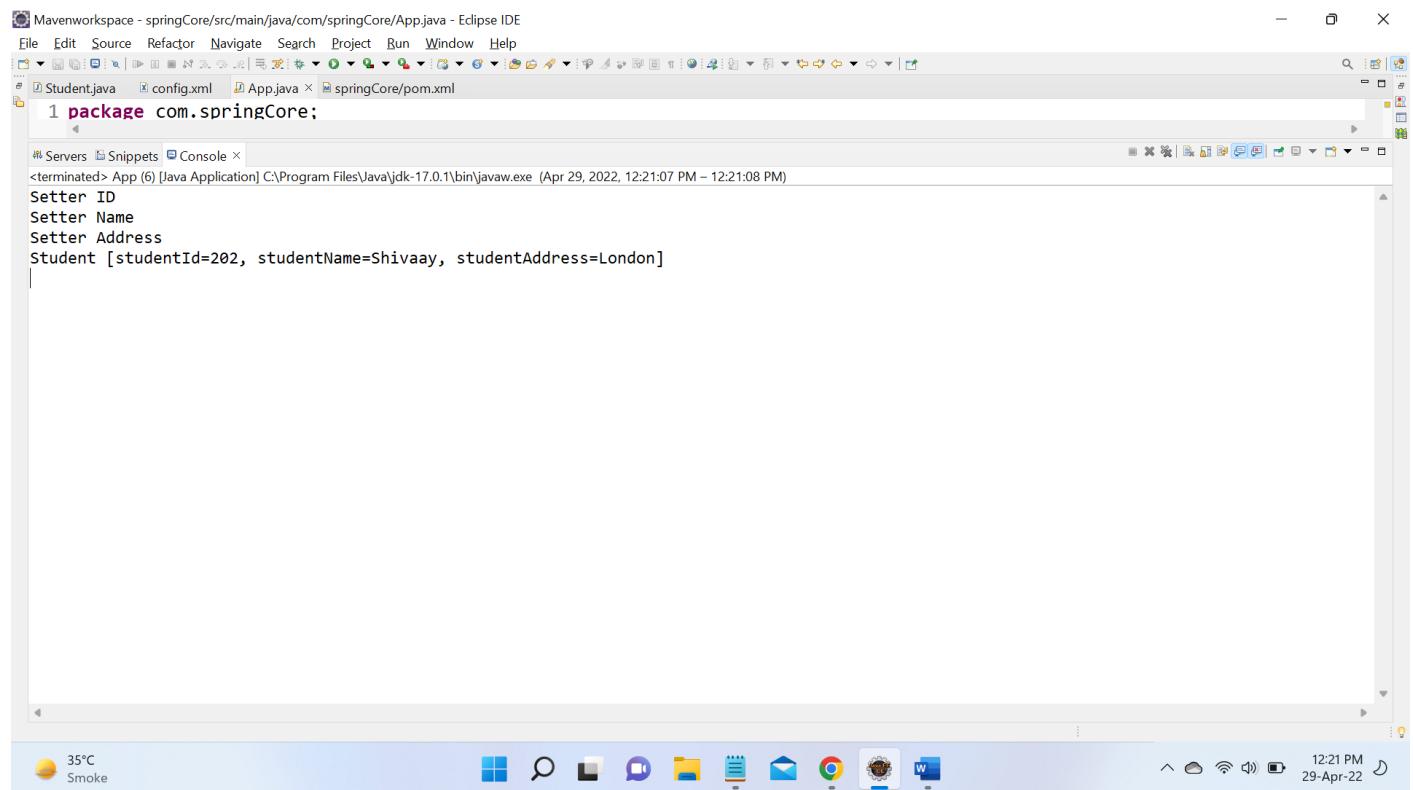
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
</dependency>
</dependencies>
</project>
```

Output:

Name: **Shibu Mohapatra**

Branch: **MSC AI**

Roll no: **02**



```
1 package com.springCore;
2
3 public class Student {
4     private int studentId;
5     private String studentName;
6     private String studentAddress;
7
8     public void setId(int id) {
9         studentId = id;
10    }
11
12    public void setName(String name) {
13        studentName = name;
14    }
15
16    public void setAddress(String address) {
17        studentAddress = address;
18    }
19
20    public int getId() {
21        return studentId;
22    }
23
24    public String getName() {
25        return studentName;
26    }
27
28    public String getAddress() {
29        return studentAddress;
30    }
31 }
```

Output in Console:

```
Setter ID
Setter Name
Setter Address
Student [studentId=202, studentName=Shivaay, studentAddress=London]
```

3. Create project to implement injecting collection types: List, Set Map and Properties.**Code:****Collectionconfig.xml (CONFIGURATION FILE):**

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

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:p="http://www.springframework.org/schema/p"

       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd
                           http://www.springframework.org/schema/context
                           http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- this is for bean -->

    <bean name="emp1" class="com.springCore.collection.Emp">
        <property name="name" value="Shibu" />

        <!-- LIST -->
        <property name="phones">
            <list>
                <value>1234567890</value>
                <value>8208213211</value>
                <value>8271639576</value>
            </list>
        </property>

        <!-- SET -->
        <property name="address">
            <set>
                <value>Mumbai</value>
                <value>New York</value>
                <value>London</value>
            </set>
        </property>

        <!-- MAP -->
        <property name="courses">
            <map>
                <entry key="Java" value="2months" />
            </map>
        </property>
    </bean>
</beans>
```

```
<entry key="Python" value="2months" />
<entry key="NLP" value="3months" />
</map>
</property>

<!-- Properties -->
<property name="info">
<props>
<prop key="name">Shibu Mohapatra</prop>
<prop key="subject">DIP</prop>
</props>
</property>

</bean>
</beans>
```

Test.java:

```
package com.springCore.collection;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Test {

    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext(
                "/com/springCore/collection/collectionconfig.xml");

        Emp emp1 = (Emp) context.getBean("emp1");
        System.out.println(emp1.getName());
        System.out.println(emp1.getPhones());
        System.out.println(emp1.getAddress());
        System.out.println(emp1.getCourses());

        System.out.println(emp1.getInfo());
    }
}
```

Emp.java:

```
//how to inject collection type List, Set, Map and properties in spring

package com.springCore.collection;

import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.Properties;

public class Emp {
    private String name;
    private List<String> phones;
    private Set<String> address;
    private Map<String, String> courses;
    private Properties info;

    public Emp() {
        super();
        // TODO Auto-generated constructor stub
    }
    public Emp(String name, List<String> phones, Set<String> address, Map<String,
String> courses, Properties info) {
        super();
        this.name = name;
        this.phones = phones;
        this.address = address;
        this.courses = courses;
        this.info = info;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public List<String> getPhones() {
        return phones;
    }
    public void setPhones(List<String> phones) {
        this.phones = phones;
    }
    public Set<String> getAddress() {
        return address;
    }
```

```
    }
    public void setAddress(Set<String> address) {
        this.address = address;
    }
    public Map<String, String> getCourses() {
        return courses;
    }
    public void setCourses(Map<String, String> courses) {
        this.courses = courses;
    }
    public Properties getInfo() {
        return info;
    }
    public void setInfo(Properties info) {
        this.info = info;
    }
}
```

Pom.xml:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.springcore</groupId>
    <artifactId>springCore</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>springCore</name>
    <url>http://maven.apache.org</url>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

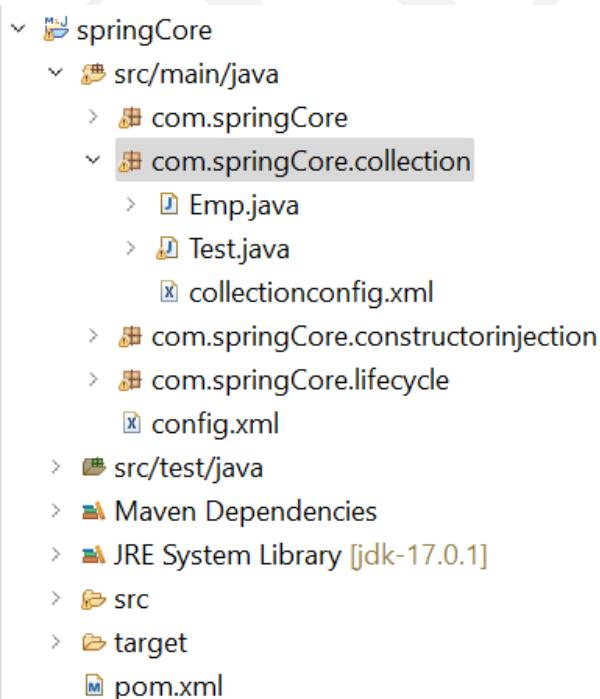
    <dependencies>

        <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
```

```
<version>5.2.3.RELEASE</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.2.3.RELEASE</version>
</dependency>

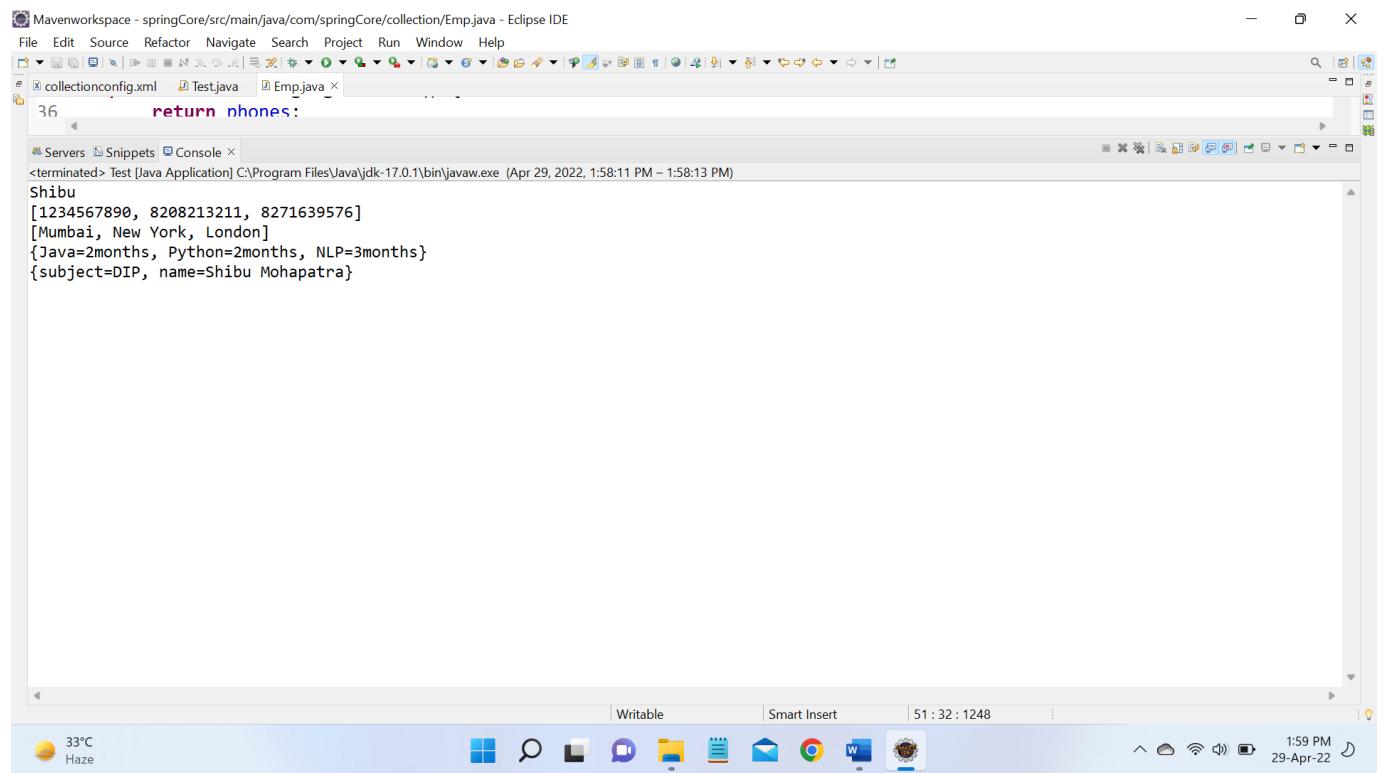
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
</dependency>
</dependencies>
</project>
```

Output:

Name: **Shibu Mohapatra**

Branch: **MSC AI**

Roll no: **02**



Mavenworkspace - springCore/src/main/java/com/springCore/collection/Emp.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

collectionconfig.xml Test.java Emp.java

```
36     return phones;
```

Servers Snippets Console <terminated> Test [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (Apr 29, 2022, 1:58:11 PM – 1:58:13 PM)

```
Shibu
[1234567890, 8208213211, 8271639576]
[Mumbai, New York, London]
{Java=2months, Python=2months, NLP=3months}
{subject=DIP, name=Shibu Mohapatra}
```

Writable Smart Insert 51 : 32 : 1248

33°C Haze 1:59 PM 29-Apr-22

This screenshot shows the Eclipse IDE interface with a Java project named 'springCore'. The 'Emp.java' file is open, containing a single line of code: 'return phones;'. Below the code editor is the 'Console' tab, which displays the output of a Java application run. The output shows the object 'Shibu' with three phone numbers and three locations: Mumbai, New York, and London. It also shows Java, Python, and NLP skill levels. At the bottom of the screen, the Windows taskbar is visible, showing the date and time (29-Apr-22, 1:59 PM), system icons, and a weather widget indicating 33°C and Haze.

4. Create project to implement constructor injection**Code:****ciconfig.xml (CONFIGURATION FILE):**

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

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:p="http://www.springframework.org/schema/p"

       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- using constructor injection -->
    <bean class="com.springCore.constructorinjection.Person"
          name="person">

        <!-- using constructor as element -->
        <constructor-arg>
            <value>Shibu</value>
        </constructor-arg>

        <!-- using constructor as attribute -->
        <!-- <constructor-arg value = "Shiv" /> -->

        <!-- <constructor-arg> <value>10</value> </constructor-arg> -->
        <constructor-arg value="10" type="int" />
    </bean>

</beans>
```

Test.java:

```
package com.springCore.constructorinjection;
```

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

```
import org.springframework.context.support.ClassPathXmlApplicationContext;
```

```
public class Test {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
        ApplicationContext context = new  
        ClassPathXmlApplicationContext("com/springCore/constructorinjection/ciconfig.xml");  
        Person p = (Person)context.getBean("person");  
  
        System.out.println(p);  
  
    }  
  
}
```

Person.java:

```
package com.springCore.constructorinjection;  
  
public class Person {  
    private String name;  
    private int personId;  
  
    public Person(String name, int personId) {  
        this.name = name;  
        this.personId = personId;  
    }  
  
    @Override  
    public String toString() {  
        return this.name + " : " + this.personId;  
    }  
}
```

Pom.xml:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
  4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.springcore</groupId>
  <artifactId>springCore</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>springCore</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>5.2.3.RELEASE</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.2.3.RELEASE</version>
    </dependency>

    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Output:

springCore

- src/main/java
 - com.springCore
 - com.springCore.collection
 - com.springCore.constructorinjection
 - Person.java
 - Test.java
 - ciconfig.xml
 - config.xml
 - src/test/java
 - Maven Dependencies
 - JRE System Library [jdk-17.0.1]
 - src
 - target
- pom.xml

Mavenworkspace - springCore/src/main/java/com/springCore/constructorinjection/Test.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Person.java Test.java ciconfig.xml

```
1 package com.springCore.constructorinjection;
```

Server Snippets Console

<terminated> Test (1) [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (May 5, 2022, 4:04:55 PM – 4:04:56 PM)

Shibu : 10

Writable Smart Insert 13 : 9 : 461

33°C Haze 4:12 PM 05-May-22

5. Write a Spring program for implementing Bean lifecycle method using XML**Code:****Samosaconfig.xml (CONFIGURATION FILE):**

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

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:p="http://www.springframework.org/schema/p"

       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">

<!-- init is only signature not passing argument so no () this bracket -->

<bean class="com.springCore.lifecycle.Samosa" name="s1" init-method="init" destroy-
method="destroy">

<property name="price" value="10" />
</bean>

</beans>
```

Test.java:

```
// program for bean object life cycle.
```

```
// implementing Spring Bean object Life cycle methods using XML
```

```
package com.springCore.lifecycle;
```

```
//import org.springframework.context.ApplicationContext;
import org.springframework.context.support.AbstractApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
```

```
public class Test {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
        // for application context  
        //ApplicationContext context = new  
        ClassPathXmlApplicationContext("com/springCore/lifecycle/samosaconfig.xml");  
  
        // for interface  
        AbstractApplicationContext context = new  
        ClassPathXmlApplicationContext("com/springCore/lifecycle/samosaconfig.xml");  
  
        Samosa s1=(Samosa)context.getBean("s1");  
        System.out.println(s1);  
  
        // for shutdown  
        // ioc method call destroy method from config.xml and shutdown  
        // registering shutdown hook enable  
  
        context.registerShutdownHook();  
  
    }  
}
```

Samosa.java:

```
// program for bean object life cycle.

package com.springCore.lifecycle;

public class Samosa {
    private double price;

    public Samosa() {
        super();
        // TODO Auto-generated constructor stub
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        System.out.println("setting price");
        this.price = price;
    }

    @Override
    public String toString() {
        return "Samosa [price=" + price + "]";
    }

    // object created
    public void init() {
        System.out.println("Inside init method, Hello I am created... ");
    }

    public void destroy() {
        System.out.println("Inside destroy method, bye I am going to die... ");
    }
}
```

Pom.xml:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.springcore</groupId>
  <artifactId>springCore</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>springCore</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>5.2.3.RELEASE</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.2.3.RELEASE</version>
    </dependency>

    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Output:

springCore

- src/main/java
 - com.springCore
 - com.springCore.collection
 - com.springCore.constructorinjection
 - com.springCore.lifecycle
 - Samosa.java
 - Test.java
 - samosaconfig.xml
 - config.xml
- src/test/java
- Maven Dependencies
- JRE System Library [jdk-17.0.1]
- src
- target
- pom.xml

Mavenworkspace - springCore/src/main/java/com/springCore/lifecycle/Test.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

samosaconfig.xml Test.java Samosa.java

```
1*// program for bean object life cycle.
```

Servers Snippets Console

<terminated> Test (2) [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (May 5, 2022, 4:22:25 PM – 4:22:25 PM)

setting price
Inside init method, Hello I am created...
Samosa [price=10.0]
Inside destroy method, bye I am going to die...

33°C Haze

6. Create Application for Spring JDBC concept (insert)**Code:****App.java:**

```
package com.spring.jdbc;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.jdbc.core.JdbcTemplate;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "My program stareded..." );

        // spring jdbc -> jdbcTemplate

        ApplicationContext context = new
ClassPathXmlApplicationContext("com/spring/jdbc/config.xml");

        JdbcTemplate template = context.getBean("jdbcTemplate", JdbcTemplate.class);

        // insert query
        String query="insert into student(id,name,city)values(?, ?, ?)";

    }
}
```

```
//fire query  
int result = template.update(query,880,"Shiv","Jaipur");  
  
//int result = template.update(query,778,"Ninja","Goa");  
  
System.out.println("number of record inserted..."+result);  
}  
}
```

StudentDao.java:

```
package com.spring.jdbc.dao;  
  
public interface StudentDao {  
    public int insert();  
}
```

StudentDaoImpl:

```
package com.spring.jdbc.dao;  
  
import org.springframework.jdbc.core.JdbcTemplate;  
  
import com.spring.jdbc.entities.Student;  
  
public class StudentDaoImpl implements StudentDao{  
  
    private JdbcTemplate jdbcTemplate;  
  
    public int insert(Student student) {  
  
        //insert query  
        String query="insert into student(id,name,city)values(?, ?, ?)";  
        int  
r=this.jdbcTemplate.update(query,student.getId(),student.getName(),student.getCity());  
  
        return r;  
    }  
}
```

```
public JdbcTemplate getJdbcTemplate() {
    return jdbcTemplate;
}

public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
    this.jdbcTemplate = jdbcTemplate;
}

public int insert() {
    // TODO Auto-generated method stub
    return 0;
}
}
```

Student.java:

```
package com.spring.jdbc.entities;

public class Student {

    private int id;
    private String name;
    private String city;

    public Student() {
    }

    public Student(int id, String name, String city) {
        super();
        this.id = id;
        this.name = name;
        this.city = city;
    }

    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 getCity() {
        return city;
    }
    public void setCity(String city) {
        this.city = city;
    }

    @Override
    public String toString() {
        return "Student [id=" + id + ", name=" + name + ", city=" + city + "]";
    }
}

```

Config.xml (CONFIGURATION FILE):

```

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

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:p="http://www.springframework.org/schema/p"

       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">

    <bean class="org.springframework.jdbc.datasource.DriverManagerDataSource" name="ds">
        <property name="driverClassName" value="com.mysql.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/springjdbc?useSSL=false" />
        <property name="username" value="root" />
        <property name="password" value="shibu@ninja"/>
    </bean>

    <!-- useSSL=false to not to display SSL exception -->

    <!-- <bean class="org.springframework.jdbc.core.JdbcTemplate" name="jdbcTemplate">
        <property name="datasource">
            <ref bean="ds"/>
        </property>
    </bean> -->

    <bean class="org.springframework.jdbc.core.JdbcTemplate" name="jdbcTemplate"

```

p:dataSource-ref="ds"/>

</beans>

Pom.xml:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.spring.jdbc</groupId>
  <artifactId>springjdbc</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>springjdbc</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>5.2.3.RELEASE</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.2.3.RELEASE</version>
    </dependency>

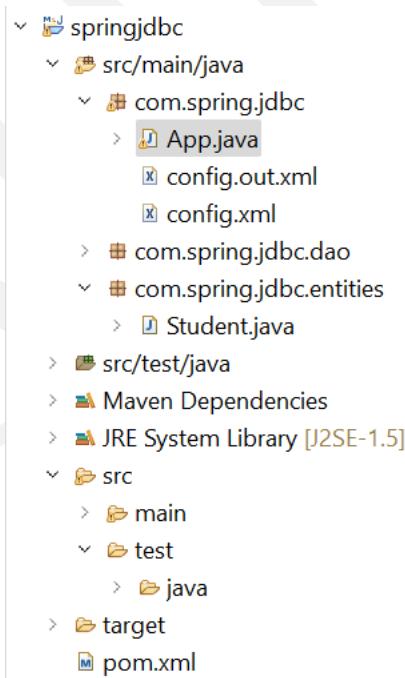
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
    <dependency>
      <groupId>org.springframework</groupId>
```

```
<artifactId>spring-jdbc</artifactId>
<version>5.2.3.RELEASE</version>
</dependency>

<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.47</version>
</dependency>

<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
</dependency>

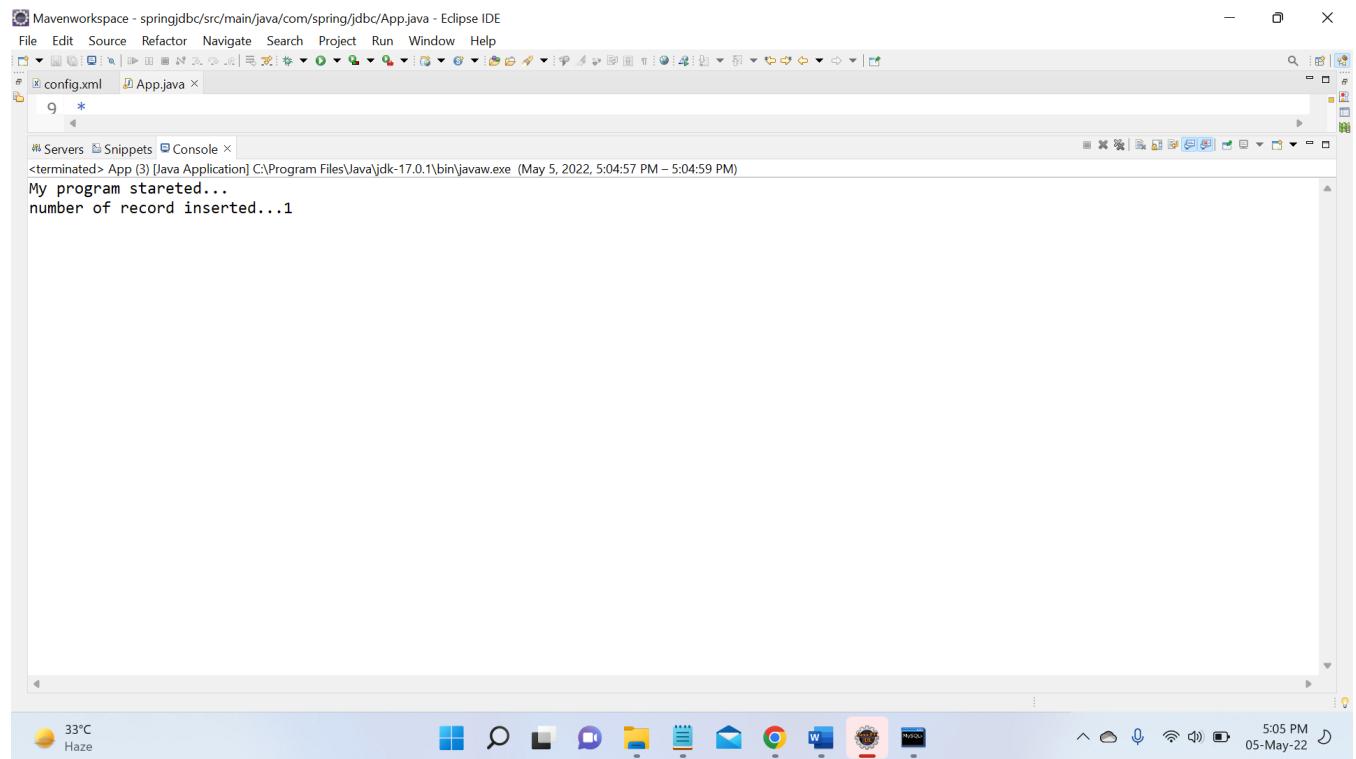
</dependencies>
</project>
```

Output:

Name: **Shibu Mohapatra**

Branch: **MSC AI**

Roll no: **02**



A screenshot of the Eclipse IDE interface. The title bar reads "Mavenworkspace - springjdbc/src/main/java/com/spring/jdbc/App.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The left sidebar shows "Servers", "Snippets", and "Console". The main editor area shows "config.xml" and "App.java". The "App.java" tab is active, displaying the following code:

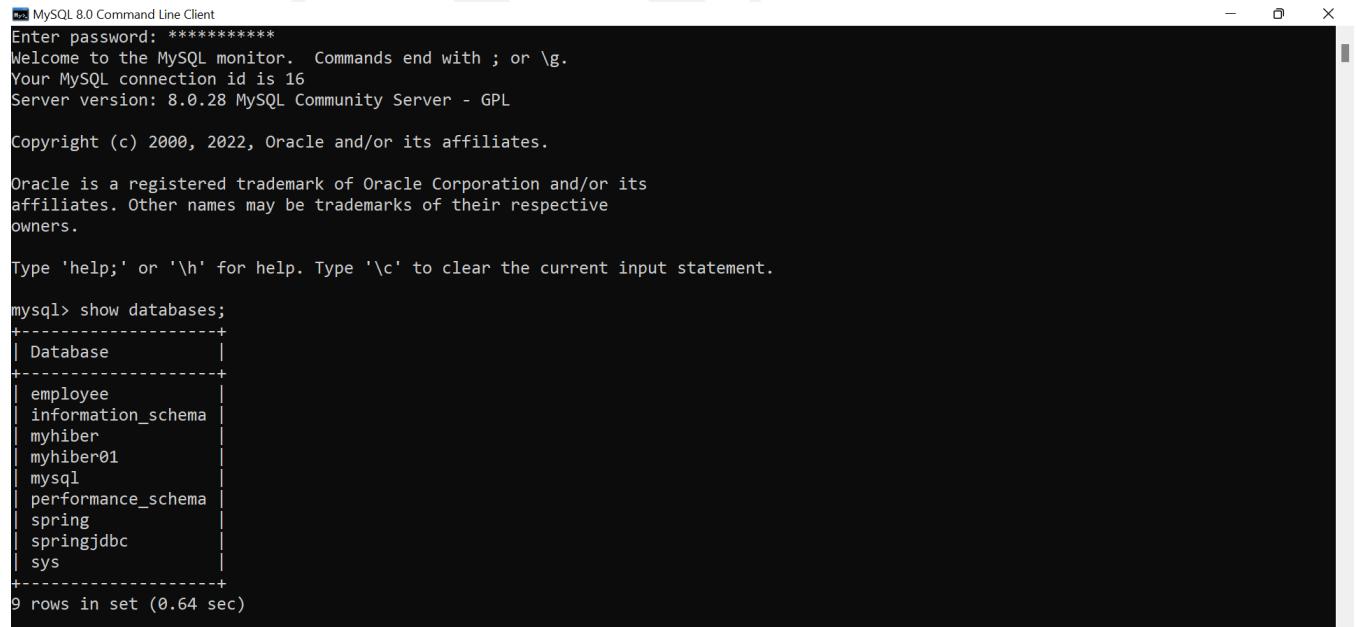
```
public class App {
    public static void main(String[] args) {
        SpringApplication.run(App.class, args);
        System.out.println("My program started...");
        System.out.println("number of record inserted...1");
    }
}
```

The "Console" tab shows the output of the application's run:

```
<terminated> App [3] [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (May 5, 2022, 5:04:57 PM – 5:04:59 PM)
My program started...
number of record inserted...1
```

The system tray at the bottom shows weather (33°C Haze), date (05-May-22), and time (5:05 PM).

MySQL Output:



A screenshot of the MySQL Command Line Client window. The title bar reads "MySQL 8.0 Command Line Client". The password prompt "Enter password: *****" is shown. The client welcome message and version information are displayed:

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 16
Server version: 8.0.28 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

The command "mysql> show databases;" is run, and the output shows the following databases:

Database
employee
information_schema
myhiber
myhiber01
mysql
performance_schema
spring
springjdbc
sys

9 rows in set (0.64 sec)

```
mysql> use springjdbc;
Database changed
mysql> show tables;
+-----+
| Tables_in_springjdbc |
+-----+
| student               |
+-----+
1 row in set (0.19 sec)

mysql> select * from student;
+----+-----+-----+
| id | name   | city   |
+----+-----+-----+
| 111 | shibu  | mumbai |
| 777 | Mohapatra | Delhi  |
| 778 | Ninja   | Goa    |
| 779 | Mohapatra | Delhi  |
+----+-----+-----+
4 rows in set (0.00 sec)
```

```
mysql> select * from student;
+----+-----+-----+
| id | name   | city   |
+----+-----+-----+
| 111 | shibu  | mumbai |
| 777 | Mohapatra | Delhi  |
| 778 | Ninja   | Goa    |
| 779 | Mohapatra | Delhi  |
| 880 | Shiv    | Jaipur |
+----+-----+-----+
5 rows in set (0.00 sec)
```

```
mysql> █
```



7. Create Application for Spring MVC.

Code:

HomeController.java:

```
package springmvc.controller;

import org.springframework.web.bind.annotation.RequestMapping;

public class HomeController {

    @RequestMapping("/home")
    public String home() {
        System.out.println("This is Home URL..");
        return "index";
    }

}
```

Index.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Home Page</title>
</head>
<body>
    <h1>This is home page</h1>
    <h1>called by home controller. URL/Home</h1>
</body>
</html>
```

Web.xml:

```
<!DOCTYPE web-app PUBLIC
    "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
    <display-name>Archetype Created Web Application</display-name>
```

```
<!-- configuration dispatcher servlet -->
<servlet>
<servlet-name>Spring</servlet-name>

<!-- got from DispatcherServlet.class or
make a demo test.java class file and press ctrl + shift + T and select DispatcherServlet class
name from the opened file
-->
<servlet-class>package org.springframework.web.servlet.DispatcherServlet</servlet-class>
</servlet>

<!-- mapping -->
<servlet-mapping>
<servlet-name>spring</servlet-name>
<url-pattern>/</url-pattern>
</servlet-mapping>

</web-app>
```

Spring-servlet.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:p="http://www.springframework.org/schema/p"

       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd
                           http://www.springframework.org/schema/context
                           http://www.springframework.org/schema/context/spring-context.xsd">

    <context:component-scan base-package="springmvc.controller"></context:component-scan>

    <bean
        class="org.springframework.web.servlet.view.InternalResourceViewResolver"
        name="viewResolver">

        <property name="prefix" value="/WEB-INF/views/" />
        <property name="suffix" value=".jsp" />

    </bean>
</beans>
```

Pom.xml:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-
v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.spring.mvc</groupId>
  <artifactId>springmvc1</artifactId>
  <packaging>war</packaging>
  <version>0.0.1-SNAPSHOT</version>
  <name>springmvc1 Maven Webapp</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>

    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>3.0.1</version>
      <scope>provided</scope>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>5.2.4.RELEASE</version>
    </dependency>

  </dependencies>

  <build>
    <finalName>springmvc</finalName>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
```

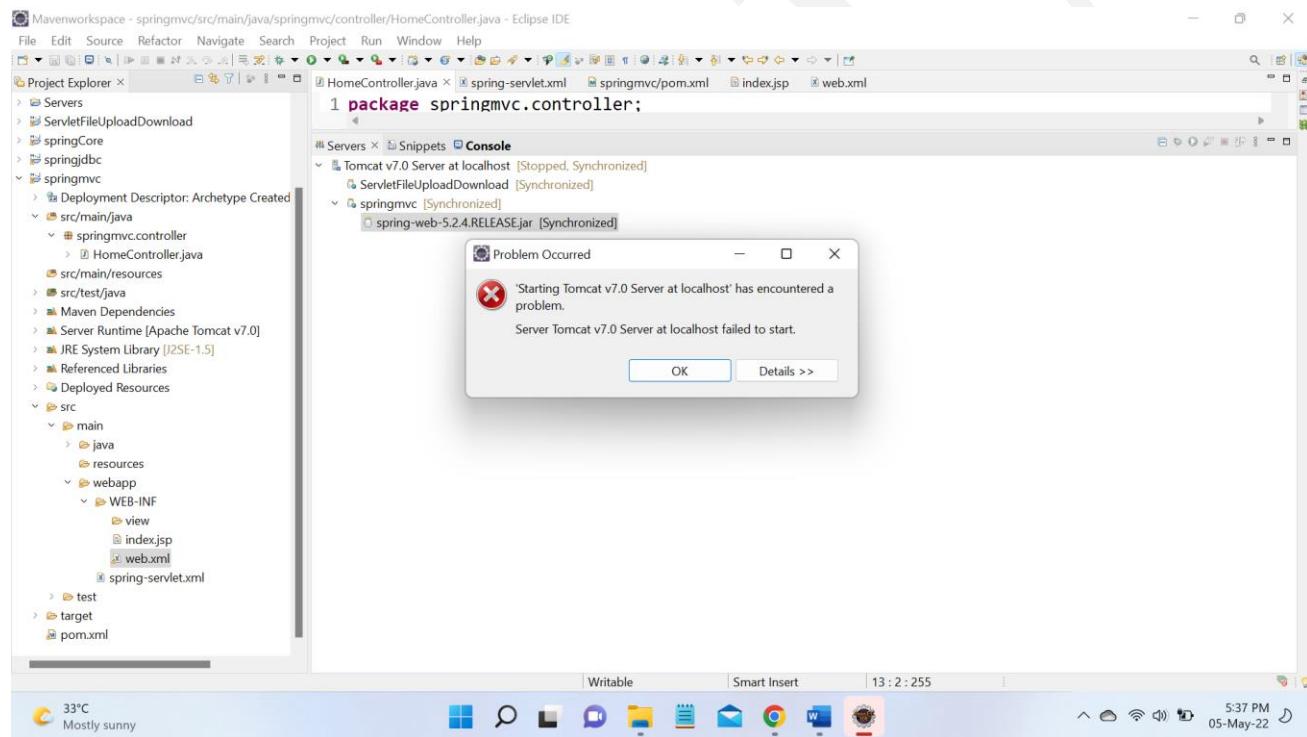
```

<artifactId>maven-war-plugin</artifactId>
<version>3.3.1</version>
</plugin>
</plugins>
</build>
</project>

```

Output:**Note:**

Due to the Apache Tomcat not starting the output for the following is displayed as below in the eclipse, with also changing of port number the output of the program remains the same.



“

Once the program is successfully executed the output in the browser will be displayed as:

This is home page

called by home controller.

”