Experiment 8

Student Name: Harshad Fozdar **UID: 22BCS10263**
Branch: CSE **Section/Group: 901'A**
Semester: 6 **Date of Performance: 02/04/25**
Subject Name: Advanced Programming-II **Subject Code: 22CSP-35**

3.1.1 Easy Level:
Write a servlet to accept user credentials through an HTML form and display a personalized welcome message if the login is successful.

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Login</title>
</head>
<body>
        <h2>Login Form</h2>
        <form action="LoginServlet" method="post">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username" required><br><br>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password" required><br><br>
        <button type="submit">Login</button>
        </form>
</body>
</html>
```

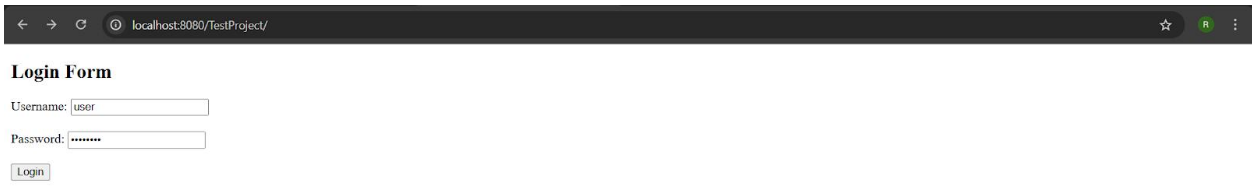
LoginServlet.java

```
import java.io.*;


import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

public class LoginServlet extends HttpServlet {
	// Dummy user credentials for demonstration (in a real-world app, you'd query a database)
	private static final String VALID_USERNAME = "user";
	private static final String VALID_PASSWORD = "password";

	@Override
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
	// Get user input from the form
	String username = request.getParameter("username");
```

```
            String password = request.getParameter("password");

            // Set content type for the response
            response.setContentType("text/html");

            // Get the PrintWriter to write the response
            PrintWriter out = response.getWriter();

            // Check if the credentials are correct
            if (VALID_USERNAME.equals(username) && VALID_PASSWORD.equals(password)) {
            // If login is successful, display a personalized welcome message
            out.println("<html><body>");
            out.println("<h2>Welcome, " + username + "!</h2>");
            out.println("<p>You have logged in successfully.</p>");
            out.println("</body></html>");
            } else {
            // If login fails, display an error message
            out.println("<html><body>");
            out.println("<h2>Invalid credentials. Please try again.</h2>");
            out.println("<a href='login.html'>Go back to login</a>");
            out.println("</body></html>");
            }
            }
}
```

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="https://jakarta.ee/xml/ns/jakartaee" xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd" id="WebApp_ID" version="6.0">

  <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>LoginServlet</servlet-name>
            <servlet-class>LoginServlet</servlet-class>
            </servlet>

            <servlet-mapping>
            <servlet-name>LoginServlet</servlet-name>
            <url-pattern>/LoginServlet</url-pattern>
            </servlet-mapping>

</web-app>
```

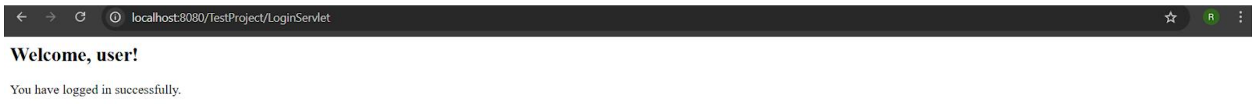

localhost:8080/TestProject/

Login Form

Username: user

Password: ••••••••

Login


localhost:8080/TestProject/LoginServlet

Welcome, user!

You have logged in successfully.

3.1.2

Create a servlet integrated with JDBC to display a list of employees from a database. Include a search form to fetch employee details by ID.

MySQL

CREATE DATABASE employeeDB;

```
USE employeeDB;

CREATE   TABLE   employees
        (   id   INT   PRIMARY
        KEY,                name
        VARCHAR(50),
        department VARCHAR(50),
        salary DECIMAL(10, 2)
);

INSERT INTO employees (id, name, department, salary) VALUES
(1, 'Kapil', 'TT', 5000),
(2, 'Devendra', 'TT', 6000),
(3, 'Tusar', 'TT', 7000);
```

Create the Employee Servlet (`EmployeeServlet.java`):

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

public class EmployeeServlet extends HttpServlet {

        private static final String JDBC_URL = "jdbc:mysql://localhost:3306/employeeDB";
        private static final String JDBC_USER = "Your-username";
        private static final String JDBC_PASSWORD = "Your_Pasword";
```

```
            protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

          response.setContentType("text/html");



            PrintWriter out = response.getWriter();



            String action = request.getParameter("action");



            if (action == null || action.equals("list"))

            { List<Employee> employeeList = getAllEmployees();

            out.println("<h1>Employee List</h1>");

            out.println("<table
border='1'><tr><th>ID</th><th>Name</th><th>Department</th><th>Salary</th></tr>");

            for (Employee emp : employeeList) {

                out.println("<tr><td>" + emp.getId() + "</td><td>" + emp.getName() + "</td><td>" +
emp.getDepartment() + "</td><td>" + emp.getSalary() + "</td></tr>");

            }

            out.println("</table>");

                 } else if (action.equals("search")) {

            String empId = request.getParameter("empId");

            Employee emp = getEmployeeById(Integer.parseInt(empId));

            if (emp != null) {

                out.println("<h1>Employee Details</h1>");

                out.println("<p>ID: " + emp.getId() + "</p>");

                out.println("<p>Name: " + emp.getName() + "</p>");

                out.println("<p>Department: " + emp.getDepartment() + "</p>");

                out.println("<p>Salary: " + emp.getSalary() + "</p>");
```

```
        } else {
            out.println("<h1>No employee found with ID " + empId + "</h1>");
        }
        }

    out.println("<br/><a href='EmployeeServlet?action=list'>Back to Employee List</a>");
        out.close();
        }

        private List<Employee> getAllEmployees()
    { List<Employee> employees = new
    ArrayList<>();
        try (Connection conn = DriverManager.getConnection(JDBC_URL, JDBC_USER,
JDBC_PASSWORD);
        Statement stmt = conn.createStatement())
        { String query = "SELECT * FROM employees";
        ResultSet rs = stmt.executeQuery(query);
        while (rs.next()) {
            Employee emp = new Employee(rs.getInt("id"), rs.getString("name"),
rs.getString("department"), rs.getBigDecimal("salary"));
            employees.add(emp);
        }
        } catch (SQLException e)
        { e.printStackTrace();
        }
        return employees;
        }
        private Employee getEmployeeById(int id)

      { Employee emp = null;

        try (Connection conn = DriverManager.getConnection(JDBC_URL, JDBC_USER,
JDBC_PASSWORD);

         PreparedStatement stmt = conn.prepareStatement("SELECT * FROM employees
WHERE id = ?")) {

        stmt.setInt(1, id);

        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {

                emp = new Employee(rs.getInt("id"), rs.getString("name"),
rs.getString("department"), rs.getBigDecimal("salary"));

        }

        } catch (SQLException e)

        { e.printStackTrace();

        }

        return emp;

        }


        // Employee class to hold employee data

        static class Employee {

        private int id;

        private String name;

        private String department;

        private BigDecimal salary;


        public Employee(int id, String name, String department, BigDecimal salary)

        { this.id = id;

        this.name = name;
```

```
            this.department = department;

            this.salary = salary;

            }


            public int getId()

            { return id;

            }


            public String getName()

            { return name;

            }


            public String getDepartment()

            { return department;

            }


            public BigDecimal getSalary()

              { return salary;

            }

            }

    }
```

Create the HTML Form for Search

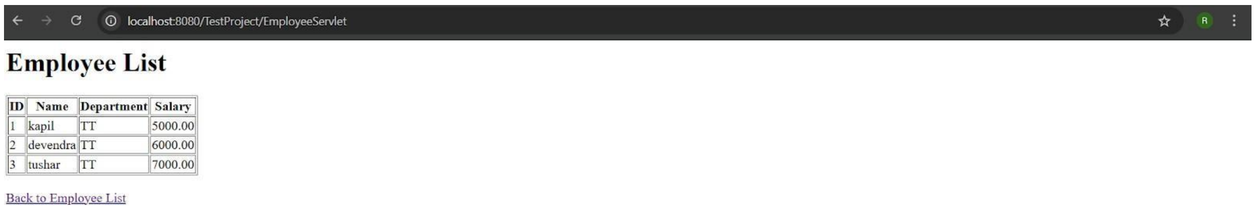
```
<!DOCTYPE html>
```

```html
<html lang="en">
<head>
            <meta charset="UTF-8">
            <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Employee Search</title>
</head>
<body>
            <h1>Search Employee</h1>
            <form action="EmployeeServlet" method="get">
            <label for="empId">Employee ID:</label>
            <input type="text" name="empId" id="empId" required>
            <input type="hidden" name="action" value="search">
        <input type="submit" value="Search">
            </form>
            <br/>
            <a href="EmployeeServlet?action=list">all Employee List</a>
</body>
</html>
```

Web.xml:

```xml
<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_1.xsd"
         version="3.1">
            <servlet>
```

```
        <servlet-name>EmployeeServlet</servlet-name>

        <servlet-class>EmployeeServlet</servlet-class>

            </servlet>

    <servlet-mapping>

        <servlet-name>EmployeeServlet</servlet-name>

        <url-pattern>/EmployeeServlet</url-pattern>

    </servlet-mapping>

</web-app>
```

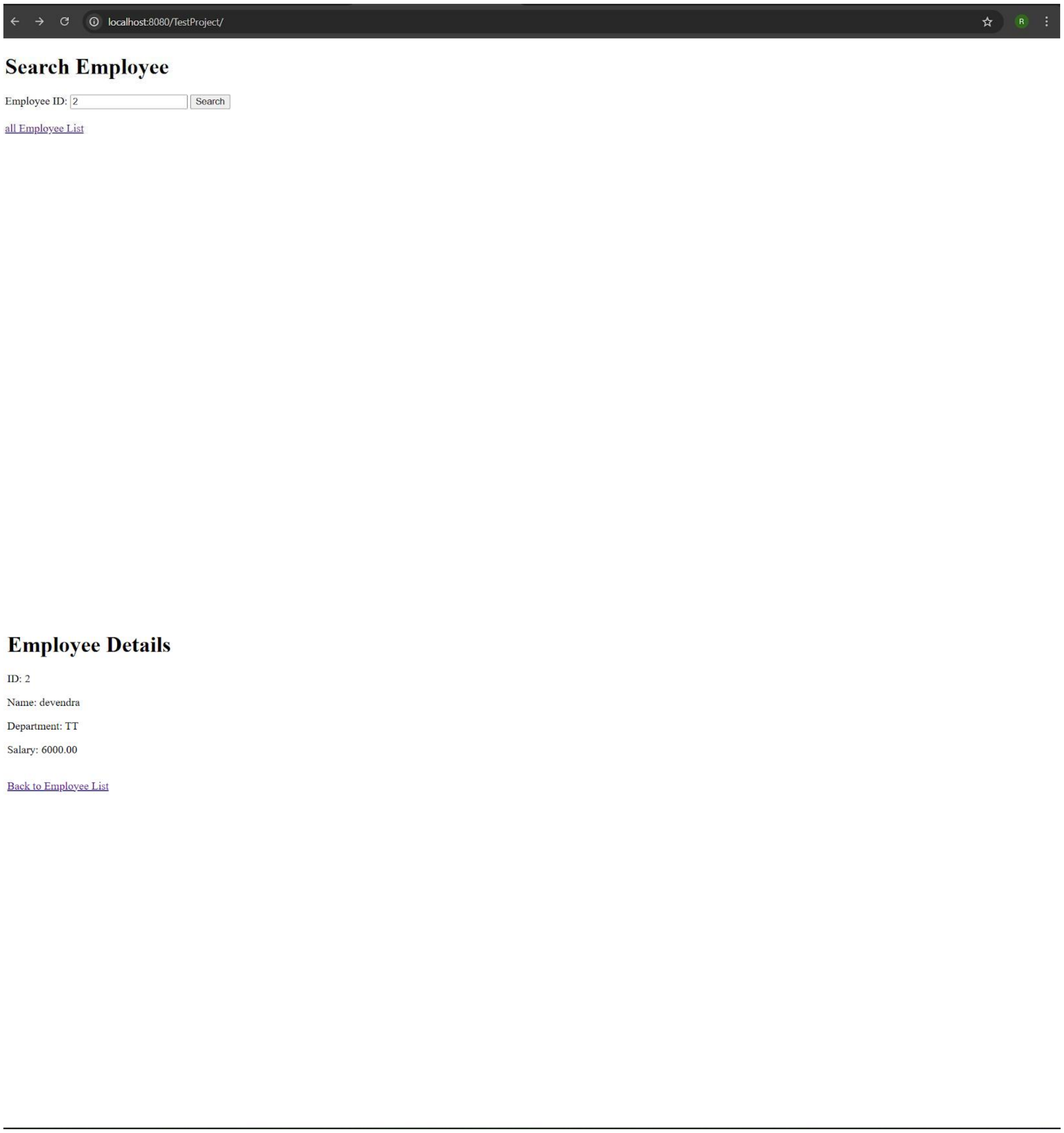


localhost:8080/TestProject/EmployeeServlet

Employee List

ID	Name	Department	Salary
1	kapil	TT	5000.00
2	devendra	TT	6000.00
3	tushar	TT	7000.00

Back to Employee List


localhost:8080/TestProject/

Search Employee

Employee ID: 2 Search

all Employee List

Employee Details

ID: 2

Name: devendra

Department: TT

Salary: 6000.00

Back to Employee List

3.2.1

Create a simple Spring application that demonstrates Dependency Injection (DI) using Java-based configuration instead of XML. Define a Student class that depends on a Course class. Use Spring's @Configuration and @Bean annotations to inject dependencies.

Requirements:

1. **Define a Course class with attributes courseName and duration.**
2. **Define a Student class with attributes name and a reference to Course.**
3. **Use Java-based configuration (@Configuration and @Bean) to configure the beans.**
4. **Load the Spring context in the main method and print student details.**

Create java class in under springdi package:-

```
package com.example.springdi;

public class Course {
        private String courseName;
        private int duration;  // Duration in hours

        public Course(String courseName, int duration) {
        this.courseName = courseName;
        this.duration = duration;
        }

        public String getCourseName() {
        return courseName;
        }

        public void setCourseName(String courseName) {
        this.courseName = courseName;
        }

        public int getDuration() {
    return duration;
        }

        public void setDuration(int duration) {
        this.duration = duration;
        }

        @Override
        public String toString() {
        return "Course [courseName=" + courseName + ", duration=" + duration + " hours]";
        }
}
```

Create the Student class under springdi package:-

```
package com.example.springdi;

public class Student {
	private String name;
	private Course course;

	public Student(String name, Course course) {
	this.name = name;
	this.course = course;
	}

	public String getName() {
	return name;
	}

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

	public Course getCourse() {
	return course;
	}

	public void setCourse(Course course) {
	this.course = course;
	}

	public void printDetails()
	{ System.out.println("Student Name: " +
	name); System.out.println("Enrolled in: " +
	course);
	}
}
```

Create AppConfig class under springdi package

```
package com.example.springdi;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class AppConfig {

	// Define a Course bean
	@Bean
	public Course course() {
	return new Course("Spring Framework", 40);  // Course with 40 hours duration
	}
```

```
        // Define a Student bean, injecting the Course bean
        @Bean
        public Student student() {
        return new Student("John Doe", course());  // Injecting the course bean into student
        }
}
```

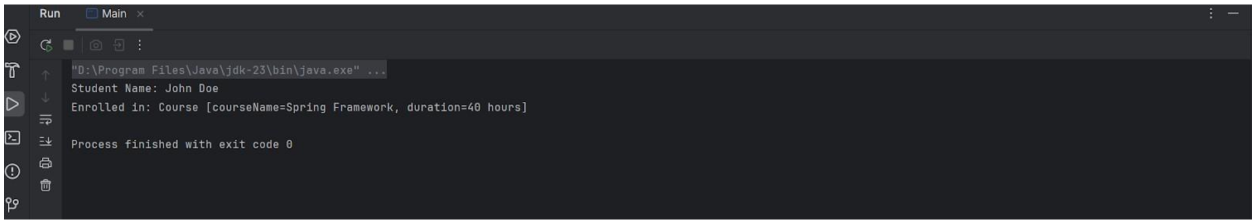
Create Main Class under sppringdi package:-

```
package com.example.springdi;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class Main {
        public static void main(String[] args) {
        // Initialize Spring context with the configuration class
     AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext(AppConfig.class);

        // Retrieve the Student bean from the context
        Student student = context.getBean(Student.class);

        // Print student details
        student.printDetails();

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

Pom.xml

Add dependency in dependencies section

```
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.23</version> <!-- Use the latest version -->
        </dependency>
```



```
"D:\Program Files\Java\jdk-23\bin\java.exe" ...
Student Name: John Doe
Enrolled in: Course [courseName=Spring Framework, duration=40 hours]

Process finished with exit code 0
```

3.2.2
Develop a Hibernate-based application to perform CRUD (Create, Read, Update, Delete) operations on a Student entity using Hibernate ORM with MySQL.

Requirements:

1. **Configure Hibernate using hibernate.cfg.xml.**
2. **Create an Entity class (Student.java) with attributes: id, name, and age.**
3. **Implement Hibernate SessionFactory to perform CRUD operations.**
4. **Test the CRUD functionality with sample data.**

```
hibernate-crud-app/
│
├── src/
│   └── main/
│       ├── java/
│       │   └── com/
│       │       └── example/
│       │           ├── model/
│       │           │   └── Student.java         # Entity Class (Student)
│       │           ├── dao/
│       │           │   └── StudentDAO.java      # DAO class for CRUD operations
│       │           ├── util/
│       │           │   └── HibernateUtil.java   # Utility class for Hibernate SessionFactory
│       │           └── Main.java                # Main class to test CRUD operations
│       ├── resources/
│       │   ├── hibernate.cfg.xml                # Hibernate configuration file
│       │   └── log4j.properties                 # Logging configuration (optional)
├── pom.xml                                      # Maven dependencies
```

```
└── target/                        # Compiled and packaged classes
```

Hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD
3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <!-- JDBC Database connection settings -->
        <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/your_database_name</property>
        <property name="hibernate.connection.username">your_username</property>
        <property name="hibernate.connection.password">your_password</property>

        <!-- JDBC connection pool settings -->
        <property name="hibernate.c3p0.min_size">5</property>
        <property name="hibernate.c3p0.max_size">20</property>
        <property name="hibernate.c3p0.timeout">300</property>
        <property name="hibernate.c3p0.max_statements">50</property>
        <property name="hibernate.c3p0.idle_test_period">3000</property>

        <!-- Specify dialect -->
        <property name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>

        <!-- Enable Hibernate's automatic session context management -->
        <property name="hibernate.current_session_context_class">thread</property>

        <!-- Echo all executed SQL to stdout -->
        <property name="hibernate.show_sql">true</property>

        <!-- Drop and re-create the database schema on startup -->
        <property name="hibernate.hbm2ddl.auto">update</property>

        <!-- Mention annotated class -->
        <mapping class="Student"/>
    </session-factory>
</hibernate-configuration>
```

Student.java

```
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "student")
public class Student {

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

     public Student() {}

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

     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 int getAge()
        { return age;
    }

    public void setAge(int age)
        { this.age = age;
    }

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

StudentDAO.java

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

import java.util.List;

public class StudentDAO {

    private static SessionFactory factory;

    // Static block to initialize Hibernate SessionFactory
    static {
        factory = new
Configuration().configure("hibernate.cfg.xml").addAnnotatedClass(Student.class).buildSessionF
actory();
    }

    // Create a new student
    public void createStudent(Student student)
        { Session session = factory.getCurrentSession();
        try {
            session.beginTransaction();
            session.save(student);
            session.getTransaction().commit();
        } finally
            { session.close();
        }
    }

    // Read a student by id
    public Student getStudent(int studentId)
        {  Session session =
        factory.getCurrentSession(); Student student =
        null;
        try {
            session.beginTransaction();
            student = session.get(Student.class, studentId);
            session.getTransaction().commit();
        } finally
            { session.close();
        }
        return student;
    }

    // Update an existing student
    public void updateStudent(Student student)
        { Session session = factory.getCurrentSession();
        try {
            session.beginTransaction();
            session.update(student);
            session.getTransaction().commit();
        } finally
            { session.close();
        }
    }

    // Delete a student by id
    public void deleteStudent(int studentId) {
        Session session = factory.getCurrentSession();
        try {
            session.beginTransaction();
            Student student = session.get(Student.class, studentId);
            if (student != null) {
                session.delete(student);
            }
            session.getTransaction().commit();
        } finally
            { session.close();
        }
    }

    // Get all students
    public List<Student> getAllStudents() {
        Session session = factory.getCurrentSession();
        List<Student> students = null;
        try {
            session.beginTransaction();
            students = session.createQuery("from Student", Student.class).getResultList();
            session.getTransaction().commit();
        } finally
            { session.close();
        }
        return students;
    }
}
```

Main.java

```
public class Main {

                public static void main(String[] args)
                    { StudentDAO studentDAO = new
                    StudentDAO();

                    // Create student objects
                    Student student1 = new Student(1, "John", 22);
                    Student student2 = new Student(2, "Emma", 20);
```

```
            // CREATE operation
            System.out.println("Creating new students...");
            studentDAO.createStudent(student1);
            studentDAO.createStudent(student2);

            // READ operation
            System.out.println("Getting student with ID 1...");
            Student fetchedStudent = studentDAO.getStudent(1);
            System.out.println(fetchedStudent);

            // UPDATE operation
            System.out.println("Updating student with ID 2...");
            student2.setName("Emily");
            studentDAO.updateStudent(student2);

            // DELETE operation
            System.out.println("Deleting student with ID 1...");
            studentDAO.deleteStudent(1);

            // Get all students
            System.out.println("All students:");
            studentDAO.getAllStudents().forEach(System.out::println);
        }
}
```

MySQL

```
CREATE DATABASE your_database_name;

USE your_database_name;

CREATE TABLE student
   ( id INT PRIMARY KEY,
   name VARCHAR(255),
   age INT
);
```

Pom.xml

```
<dependencies>
   <!-- Hibernate Dependencies -->
   <dependency>
      <groupId>org.hibernate</groupId>
      <artifactId>hibernate-core</artifactId>
      <version>5.4.30.Final</version>
   </dependency>

   <!-- MySQL Connector Dependency -->
   <dependency>
       <groupId>mysql</groupId>
       <artifactId>mysql-connector-java</artifactId>
       <version>8.0.27</version>
    </dependency>
</dependencies>
```