DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

EXPERIMENT-7

Name: Saras Sahu
Branch: BE-CSE
Semester: 6th
Subject Name: Java Lab

UID: 22BCS10125
Section/Group: NTPP-DL-903(A)
Date of Performance:
Subject Code: 22CSH-359

AIM: Create Java applications with JDBC for database connectivity, CRUD operations, and MVC architecture.

Objective: The objective of creating Java applications with JDBC for database connectivity, CRUD operations, and MVC architecture is to design and implement a robust, scalable, and maintainable application that follows the Model-View-Controller (MVC) design pattern. This approach will allow for separation of concerns, easier management of data, and efficient interaction with relational databases.

Problem 1: Create a Java program to connect to a MySQL database and fetch data from a single table. The program should: Use DriverManager and Connection objects. Retrieve and display all records from a table named Employee with columns EmpID, Name, and Salary.

Code:

```
CREATE DATABASE company;
USE company;
CREATE TABLE Employee (
    EmpID INT PRIMARY KEY,
    Name VARCHAR(100),
    Salary DECIMAL(10, 2)
);
INSERT INTO Employee (EmpID, Name, Salary) VALUES
(1, 'John Doe', 50000.00),
(2, 'Jane Smith', 55000.00),
(3, 'Sam Brown', 45000.00);
import java.sql.*;
public class MySQLJDBCExample {
    // Database URL, username, and password
    static final String DB_URL = "jdbc:mysql://localhost:3306/company";  // Update the URL based on
your setup
    static final String USER = "root";  // MySQL username
    static final String PASS = "password";  // MySQL password (replace with your actual password)
    public static void main(String[] args) {
        // Step 1: Establish a connection to the database
        try (Connection connection = DriverManager.getConnection(DB_URL, USER, PASS)) {
            // Step 2: Create a statement object to execute SQL queries
            String sql = "SELECT EmpID, Name, Salary FROM Employee";
            try (Statement stmt = connection.createStatement()) {
                // Step 3: Execute the query and obtain the result set
```

```
            ResultSet rs = stmt.executeQuery(sql);
            // Step 4: Process the result set and display the records
            System.out.println("EmpID | Name          | Salary");
            System.out.println("---------------------------------");
            while (rs.next()) {
                int empID = rs.getInt("EmpID");
                String name = rs.getString("Name");
                double salary = rs.getDouble("Salary");
                System.out.printf("%-6d | %-15s | %.2f%n", empID, name, salary);
            }
        }
    } catch (SQLException e) {
        // Handle SQL exceptions
        e.printStackTrace();
    }
  }
}
```

Output:

```
EmpID | Name          | Salary
-----------------------------------
1     | John Doe      | 50000.00
2     | Jane Smith    | 55000.00
3     | Sam Brown     | 45000.00
```

Problem 2: Build a program to perform CRUD operations (Create, Read, Update, Delete) on a database table Product with columns: ProductID, ProductName, Price, and Quantity.
The program should include: Menu-driven options for each operation. Transaction handling to ensure data integrity.

Code:

```
CREATE DATABASE store;
USE store;
CREATE TABLE Product (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(100),
    Price DECIMAL(10, 2),
    Quantity INT
);
-- Example insertion
INSERT INTO Product (ProductID, ProductName, Price, Quantity) VALUES
(1, 'Laptop', 800.00, 10),
(2, 'Smartphone', 500.00, 15);
import java.sql.*;
import java.util.Scanner;
public class ProductCRUDApp {
    // Database URL, username, and password
    static final String DB_URL = "jdbc:mysql://localhost:3306/store";  // Update based on your setup
    static final String USER = "root";  // MySQL username
    static final String PASS = "password";  // MySQL password (replace with actual)
    public static void main(String[] args) {
```

```
    Scanner scanner = new Scanner(System.in);
    Connection connection = null;
    try {
        // Step 1: Establish connection to the database
        connection = DriverManager.getConnection(DB_URL, USER, PASS);
        connection.setAutoCommit(false);  // Disable auto-commit for transaction handling
        while (true) {
            // Display menu
            System.out.println("\nMenu:");
            System.out.println("1. Create Product");
            System.out.println("2. Read Product");
            System.out.println("3. Update Product");
            System.out.println("4. Delete Product");
            System.out.println("5. Exit");
            System.out.print("Enter choice: ");
            int choice = scanner.nextInt();
            switch (choice) {
                case 1:
                    createProduct(connection, scanner);
                    break;
                case 2:
                    readProduct(connection, scanner);
                    break;
                case 3:
                    updateProduct(connection, scanner);
                    break;
                case 4:
                    deleteProduct(connection, scanner);
                    break;
                case 5:
                    System.out.println("Exiting...");
                    return;
                default:
                    System.out.println("Invalid choice! Please try again.");
            }
        }
    } catch (SQLException e) {
        System.out.println("Error: " + e.getMessage());
        try {
            if (connection != null) {
                connection.rollback();  // Rollback transaction in case of error
            }
        } catch (SQLException ex) {
            System.out.println("Rollback failed: " + ex.getMessage());
        }
    } finally {
        try {
            if (connection != null) {
                connection.close();  // Close connection
            }
        } catch (SQLException e) {
            System.out.println("Error closing connection: " + e.getMessage());
        }
    }
}
```

```
    // Create Product
    private static void createProduct(Connection connection, Scanner scanner) throws SQLException {
        System.out.println("\nEnter Product Details:");
        System.out.print("ProductID: ");
        int productID = scanner.nextInt();
        scanner.nextLine();  // Consume newline
        System.out.print("ProductName: ");
        String productName = scanner.nextLine();
        System.out.print("Price: ");
        double price = scanner.nextDouble();
        System.out.print("Quantity: ");
        int quantity = scanner.nextInt();

        String sql = "INSERT INTO Product (ProductID, ProductName, Price, Quantity) VALUES
(?, ?, ?, ?)";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setInt(1, productID);
            stmt.setString(2, productName);
            stmt.setDouble(3, price);
            stmt.setInt(4, quantity);
            int rowsAffected = stmt.executeUpdate();
            if (rowsAffected > 0) {
                System.out.println("Product added successfully.");
                connection.commit();  // Commit transaction
            } else {
                System.out.println("Error: Product not added.");
                connection.rollback();  // Rollback transaction
            }
        }
    }

    // Read Product
    private static void readProduct(Connection connection, Scanner scanner) throws SQLException {
        System.out.print("\nEnter ProductID to view details: ");
        int productID = scanner.nextInt();

        String sql = "SELECT * FROM Product WHERE ProductID = ?";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setInt(1, productID);
            ResultSet rs = stmt.executeQuery();

            if (rs.next()) {
                System.out.println("\nProduct Details:");
                System.out.println("ProductID: " + rs.getInt("ProductID"));
                System.out.println("ProductName: " + rs.getString("ProductName"));
                System.out.println("Price: " + rs.getDouble("Price"));
                System.out.println("Quantity: " + rs.getInt("Quantity"));
            } else {
                System.out.println("Product not found.");
            }
        }
    }
    // Update Product
    private static void updateProduct(Connection connection, Scanner scanner) throws SQLException {
        System.out.print("\nEnter ProductID to update: ");
        int productID = scanner.nextInt();
```

```
        String sql = "SELECT * FROM Product WHERE ProductID = ?";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setInt(1, productID);
            ResultSet rs = stmt.executeQuery();

            if (rs.next()) {
                // Product exists, now update it
                System.out.print("Enter new ProductName: ");
                scanner.nextLine();  // Consume newline
                String productName = scanner.nextLine();
                System.out.print("Enter new Price: ");
                double price = scanner.nextDouble();
                System.out.print("Enter new Quantity: ");
                int quantity = scanner.nextInt();

                String updateSql = "UPDATE Product SET ProductName = ?, Price = ?, Quantity = ? WHERE
ProductID = ?";
                try (PreparedStatement updateStmt = connection.prepareStatement(updateSql)) {
                    updateStmt.setString(1, productName);
                    updateStmt.setDouble(2, price);
                    updateStmt.setInt(3, quantity);
                    updateStmt.setInt(4, productID);

                    int rowsAffected = updateStmt.executeUpdate();
                    if (rowsAffected > 0) {
                        System.out.println("Product updated successfully.");
                        connection.commit();  // Commit transaction
                    } else {
                        System.out.println("Error: Product not updated.");
                        connection.rollback();  // Rollback transaction
                    }
                }
            } else {
                System.out.println("Product not found.");
            }
        }
    }
    // Delete Product
    private static void deleteProduct(Connection connection, Scanner scanner) throws SQLException {
        System.out.print("\nEnter ProductID to delete: ");
        int productID = scanner.nextInt();

        String sql = "DELETE FROM Product WHERE ProductID = ?";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setInt(1, productID);
            int rowsAffected = stmt.executeUpdate();

            if (rowsAffected > 0) {
                System.out.println("Product deleted successfully.");
                connection.commit();  // Commit transaction
            } else {
                System.out.println("Product not found.");
                connection.rollback();  // Rollback transaction
            }
        }
```

```
    }
}
```

Output:

```
Menu:
1. Create Product
2. Read Product
3. Update Product
4. Delete Product
5. Exit
Enter choice: 1

Enter Product Details:
ProductID: 3
ProductName: Tablet
Price: 300.00
Quantity: 20
Product added successfully.
```

Problem 3: Develop a Java application using JDBC and MVC architecture to manage student data. The application should: Use a Student class as the model with fields like StudentID, Name, Department, and Marks. Include a database table to store student data. Allow the user to perform CRUD operations through a simple menu-driven view. Implement database operations in a separate controller class.

Code:

```
CREATE DATABASE school;

USE school;

CREATE TABLE Student (

    StudentID INT PRIMARY KEY,

    Name VARCHAR(100),

    Department VARCHAR(100),

    Marks DECIMAL(5, 2)

);

-- Example insertion

INSERT INTO Student (StudentID, Name, Department, Marks) VALUES

(1, 'John Doe', 'Computer Science', 88.50),
```

```
(2, 'Jane Smith', 'Mathematics', 92.00);

import java.sql.*;
import java.util.ArrayList;
import java.util.List;


public class StudentController {
    // Database URL, username, and password
    static final String DB_URL = "jdbc:mysql://localhost:3306/school";  // Update based on your setup
    static final String USER = "root";  // MySQL username
    static final String PASS = "password";  // MySQL password (replace with actual)
    // Create a new student
    public void createStudent(Student student) throws SQLException {
        try (Connection connection = DriverManager.getConnection(DB_URL, USER, PASS)) {
            String sql = "INSERT INTO Student (StudentID, Name, Department, Marks) VALUES (?, ?, ?, ?)";
            try (PreparedStatement stmt = connection.prepareStatement(sql)) {
                stmt.setInt(1, student.getStudentID());
                stmt.setString(2, student.getName());
                stmt.setString(3, student.getDepartment());
                stmt.setDouble(4, student.getMarks());
                stmt.executeUpdate();
            }
        }
    }
    // Read all students
    public List<Student> readAllStudents() throws SQLException {
        List<Student> students = new ArrayList<>();
        try (Connection connection = DriverManager.getConnection(DB_URL, USER, PASS)) {
            String sql = "SELECT * FROM Student";
```

```
        try (Statement stmt = connection.createStatement(); ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                int studentID = rs.getInt("StudentID");
                String name = rs.getString("Name");
                String department = rs.getString("Department");
                double marks = rs.getDouble("Marks");

                students.add(new Student(studentID, name, department, marks));
            }
        }
    }
    return students;
}
// Update a student's data
public void updateStudent(Student student) throws SQLException {
    try (Connection connection = DriverManager.getConnection(DB_URL, USER, PASS)) {
        String sql = "UPDATE Student SET Name = ?, Department = ?, Marks = ? WHERE StudentID = ?";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setString(1, student.getName());
            stmt.setString(2, student.getDepartment());
            stmt.setDouble(3, student.getMarks());
            stmt.setInt(4, student.getStudentID());
            stmt.executeUpdate();
        }
    }
}
// Delete a student by ID
public void deleteStudent(int studentID) throws SQLException {
```

```
        try (Connection connection = DriverManager.getConnection(DB_URL, USER, PASS)) {
            String sql = "DELETE FROM Student WHERE StudentID = ?";
            try (PreparedStatement stmt = connection.prepareStatement(sql)) {
                stmt.setInt(1, studentID);
                stmt.executeUpdate();
            }
        }
    }
}
import java.sql.SQLException;
import java.util.List;
import java.util.Scanner;
public class StudentView {
    private StudentController controller;
    public StudentView() {
        controller = new StudentController();
    }
    // Display menu and process user input
    public void displayMenu() {
        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.println("\nMenu:");
            System.out.println("1. Add Student");
            System.out.println("2. View All Students");
            System.out.println("3. Update Student");
            System.out.println("4. Delete Student");
            System.out.println("5. Exit");
            System.out.print("Enter choice: ");
```

```
        int choice = scanner.nextInt();
        switch (choice) {
            case 1:
                addStudent(scanner);
                break;
            case 2:
                viewAllStudents();
                break;
            case 3:
                updateStudent(scanner);
                break;
            case 4:
                deleteStudent(scanner);
                break;
            case 5:
                System.out.println("Exiting...");
                return;
            default:
                System.out.println("Invalid choice. Please try again.");
        }
    }
}
// Add new student
private void addStudent(Scanner scanner) {
    System.out.print("\nEnter StudentID: ");
    int studentID = scanner.nextInt();
    scanner.nextLine();  // Consume newline
    System.out.print("Enter Name: ");
```

```
    String name = scanner.nextLine();
    System.out.print("Enter Department: ");
    String department = scanner.nextLine();
    System.out.print("Enter Marks: ");
    double marks = scanner.nextDouble();
    Student student = new Student(studentID, name, department, marks);
    try {
        controller.createStudent(student);
        System.out.println("Student added successfully.");
    } catch (SQLException e) {
        System.out.println("Error adding student: " + e.getMessage());
    }
}
// View all students
private void viewAllStudents() {
    try {
        List<Student> students = controller.readAllStudents();
        if (students.isEmpty()) {
            System.out.println("No students found.");
        } else {
            System.out.println("\nAll Students:");
            for (Student student : students) {
                System.out.println(student);
            }
        }
    } catch (SQLException e) {
        System.out.println("Error fetching students: " + e.getMessage());
    }
```

```
}
// Update student data
private void updateStudent(Scanner scanner) {
    System.out.print("\nEnter StudentID to update: ");
    int studentID = scanner.nextInt();
    scanner.nextLine();  // Consume newline
    System.out.print("Enter new Name: ");
    String name = scanner.nextLine();
    System.out.print("Enter new Department: ");
    String department = scanner.nextLine();
    System.out.print("Enter new Marks: ");
    double marks = scanner.nextDouble();
    Student student = new Student(studentID, name, department, marks);
    try {
        controller.updateStudent(student);
        System.out.println("Student updated successfully.");
    } catch (SQLException e) {
        System.out.println("Error updating student: " + e.getMessage());
    }
}
// Delete student
private void deleteStudent(Scanner scanner) {
    System.out.print("\nEnter StudentID to delete: ");
    int studentID = scanner.nextInt();
    try {
        controller.deleteStudent(studentID);
        System.out.println("Student deleted successfully.");
    } catch (SQLException e) {
```

```
            System.out.println("Error deleting student: " + e.getMessage());
        }
    }
}
```

Output:

```
Menu:
1. Add Student
2. View All Students
3. Update Student
4. Delete Student
5. Exit
Enter choice: 1

Enter StudentID: 3
Enter Name: Alice
Enter Department: Physics
Enter Marks: 89.00
Student added successfully.

Menu:
1. Add Student
2. View All Students
3. Update Student
4. Delete Student
5. Exit
Enter choice: 2

All Students:
StudentID: 1, Name: John Doe, Department: Computer Science, Marks: 88.50
StudentID: 2, Name: Jane Smith, Department: Mathematics, Marks: 92.00
StudentID: 3, Name: Alice, Department: Physic Marks: 89.00
```

Learning Outcome:

1. Understanding JDBC and Database Connectivity.
2. Proficiency in Performing CRUD Operations.
3. Applying MVC (Model-View-Controller) Architecture in Java Applications.
4. Design and Implementation of Menu-Driven Applications.