Experiment- 7

Student Name: Kamal Mehta
UID: 22BET10097
Branch: B.E - IT
Section/Group: 22BET-701/A
Semester: 6th
Date of Performance: 04-03-25
Subject Name: PBLJ Lab
Subject Code: 22ITH-359

Problem 1

1. **Aim:** To develop a Java application that connects to a MySQL database and retrieves data from a specified table using JDBC.

2. **Objective:**
   - To understand the process of establishing a database connection using JDBC.
   - To learn how to use DriverManager and Connection objects for database connectivity.
   - To retrieve and display all records from the "Employee" table with columns EmpID, Name, and Salary.

3. **Code:**

```
package student;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class product {

    private static final String DB_URL = "jdbc:mysql://localhost:3306/employee";
    private static final String USER = "kamal";
    private static final String PASS = "P@ssword97";
```

```
public static void main(String[] args) {
    Connection connection = null;
    Statement statement = null;

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

        System.out.println("Connecting to the database...");
        connection = DriverManager.getConnection(DB_URL, USER, PASS);

        System.out.println("Creating statement...");
        statement = connection.createStatement();
        String sql = "SELECT EmpID, Name, Salary FROM Employee";
        ResultSet resultSet = statement.executeQuery(sql);

        System.out.println("Employee Data:");
        while (resultSet.next()) {

            int empID = resultSet.getInt("EmpID");
            String name = resultSet.getString("Name");
            double salary = resultSet.getDouble("Salary");

            System.out.println("EmpID: " + empID + ", Name: " + name + ", Salary: " +
            salary);
        }

        resultSet.close();
        statement.close();
        connection.close();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (statement != null) statement.close();
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
        try {
            if (connection != null) connection.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
  }
}
```

4. Output:

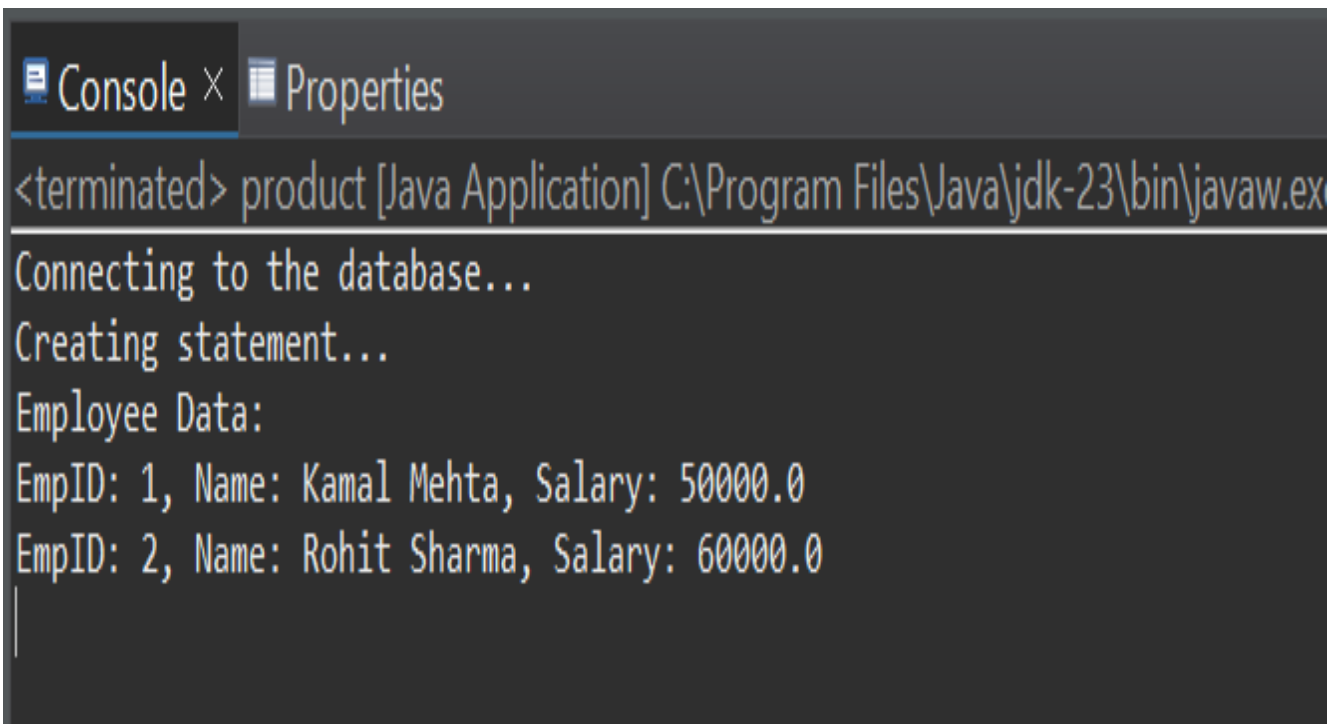

Fig 1: Output for Problem 1

Problem 2

1. **Aim:** To create a Java application that performs CRUD (Create, Read, Update, Delete) operations on a "Product" table in a database using JDBC.

2. **Objective:**
   - To design a menu-driven program that allows users to perform CRUD operations.
   - To implement transaction handling to ensure data integrity during database operations.
   - To gain experience in writing SQL queries for inserting, updating, deleting, and retrieving data.

3. **Code:**

```
package exp7b;

import java.sql.*;
import java.util.Scanner;

public class Product {
    private static final String DB_URL = "jdbc:mysql://localhost:3306/Product";
    private static final String USER = "kamal";
    private static final String PASS = "P@ssword97";

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Connection connection = null;

        try {
            connection = DriverManager.getConnection(DB_URL, USER, PASS);
            connection.setAutoCommit(false);
            while (true) {
```

```
System.out.println("\nMenu:");
System.out.println("1. Create Product");
System.out.println("2. Read Products");
System.out.println("3. Update Product");
System.out.println("4. Delete Product");
System.out.println("5. Exit");
System.out.print("Choose an option: ");
int choice = scanner.nextInt();
scanner.nextLine();

switch (choice) {
    case 1:
        System.out.print("Enter Product Name: ");
        String name = scanner.nextLine();
        System.out.print("Enter Price: ");
        double price = scanner.nextDouble();
        System.out.print("Enter Quantity: ");
        int quantity = scanner.nextInt();
        createProduct(connection, name, price, quantity);
        break;

    case 2:
        readProducts(connection);
        break;

    case 3:
        System.out.print("Enter Product ID to update: ");
        int productIdToUpdate = scanner.nextInt();
        System.out.print("Enter new Product Name: ");
        String newName = scanner.next();
        System.out.print("Enter new Price: ");
        double newPrice = scanner.nextDouble();
        System.out.print("Enter new Quantity: ");
        int newQuantity = scanner.nextInt();
        updateProduct(connection, productIdToUpdate, newName, newPrice,
        newQuantity);
```

```
                break;

            case 4:
                System.out.print("Enter Product ID to delete: ");
                int productIdToDelete = scanner.nextInt();
                deleteProduct(connection, productIdToDelete);
                break;

            case 5:
                System.out.println("Exiting...");
                connection.commit();
                connection.close();
                scanner.close();
                return;

            default:
                System.out.println("Invalid choice. Please try again.");
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
        try {
            if (connection != null) {
                connection.rollback();
            }
        } catch (SQLException rollbackEx) {
            rollbackEx.printStackTrace();
        }
    }
}

private static void deleteProduct(Connection connection, int productIdToDelete) {

    }
    private static void createProduct(Connection connection, String name, double
price, int quantity) throws SQLException {
```

```
        String sql = "INSERT INTO Product (ProductName, Price, Quantity) VALUES (?,
?, ?)";
        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
            pstmt.setString(1, name);
            pstmt.setDouble(2, price);
            pstmt.setInt(3, quantity);
            pstmt.executeUpdate();
            connection.commit();
            System.out.println("Product created successfully.");
        }
    }

    private static void readProducts(Connection connection) throws SQLException {
        String sql = "SELECT * FROM Product";
        try (Statement stmt = connection.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            System.out.println("Product List:");
            while (rs.next()) {
                int id = rs.getInt("ProductID");
                String name = rs.getString("ProductName");
                double price = rs.getDouble("Price");
                int quantity = rs.getInt("Quantity");
                System.out.printf("ID: %d, Name: %s, Price: %.2f, Quantity: %d%n", id,
                name, price, quantity);
            }
        }
    }

    private static void updateProduct(Connection connection, int productId, String name,
double price, int quantity) throws SQLException {
        String sql = "UPDATE Product SET ProductName = ?, Price = ?, Quantity = ?
WHERE ProductID = ?";
        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
            pstmt.setString(1, name);
            pstmt.setDouble(2, price);
            pstmt.setInt(3, quantity);
```

```
            pstmt.setInt(4, productId);
            int rowsAffected = pstmt.executeUpdate();
            connection.commit();
            if (rowsAffected > 0) {
                System.out.println("Product updated successfully.");
            } else {
                System.out.println("Product not found.");
            }
        }
    }
}
```

4. Output:

```
Console ×  Properties
<terminated> Product [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe

Menu:
1. Create Product
2. Read Products
3. Update Product
4. Delete Product
5. Exit
Choose an option: 1
Enter Product Name: PS5
Enter Price: 50000
Enter Quantity: 1
Product created successfully.

Menu:
1. Create Product
2. Read Products
3. Update Product
4. Delete Product
5. Exit
Choose an option: 2
Product List:
ID: 1, Name: PS5, Price: 50000.00, Quantity: 1
```

Fig 2: Output for Problem 2

Problem 3

1. **Aim:** To develop a Java application using JDBC and MVC architecture to manage student data in a database.

2. **Objective:**

- To design a student class as the model with fields like StudentID, Name, Department, and Marks.
- To create a database table to store student data and perform CRUD operations.
- To implement a simple menu-driven view for user interaction.
- To separate database operations into a controller class, adhering to the MVC architecture.

3. **Code:**

```
package Exp7c;

import java.sql.*;
import java.util.Scanner;

public class Student {

    private static final String DB_URL = "jdbc:mysql://localhost:3306/Student";
    private static final String USER = "kamal";
    private static final String PASS = "P@ssword97";

    public static void main(String[] args) {
        try (Scanner scanner = new Scanner(System.in)) {
                        Connection connection = null;

                        try {
                            connection = DriverManager.getConnection(DB_URL, USER,
```

```
PASS);
connection.setAutoCommit(false);

while (true) {
    System.out.println("\nMenu:");
    System.out.println("1. Create Student");
    System.out.println("2. Read Students");
    System.out.println("3. Update Student");
    System.out.println("4. Delete Student");
    System.out.println("5. Exit");
    System.out.print("Choose an option: ");
    int choice = scanner.nextInt();
    scanner.nextLine();

    switch (choice) {
        case 1:
            System.out.print("Enter Student Name: ");
            String name = scanner.nextLine();
            System.out.print("Enter Age: ");
            int age = scanner.nextInt();
            scanner.nextLine();
            System.out.print("Enter Major: ");
            String major = scanner.nextLine();
            createStudent(connection, name, age, major);
            break;

        case 2:
            readStudents(connection);
            break;

        case 3:
            System.out.print("Enter Student ID to update: ");
            int studentIdToUpdate = scanner.nextInt();
            scanner.nextLine();
            System.out.print("Enter new Student Name: ");
            String newName = scanner.nextLine();
```

```
                System.out.print("Enter new Age: ");
                int newAge = scanner.nextInt();
                scanner.nextLine();
                System.out.print("Enter new Major: ");
                String newMajor = scanner.nextLine();
                updateStudent(connection, studentIdToUpdate, newName,
                newAge, newMajor);
                break;

            case 4:
                System.out.print("Enter Student ID to delete: ");
                int studentIdToDelete = scanner.nextInt();
                deleteStudent(connection, studentIdToDelete);
                break;

            case 5:
                System.out.println("Exiting...");
                connection.commit();
                connection.close();
                scanner.close();
                return;

            default:
                System.out.println("Invalid choice. Please try again.");
        }
    }

} catch (SQLException e) {
    e.printStackTrace();
    try {
        if (connection != null) {
            connection.rollback();
        }
    } catch (SQLException rollbackEx) {
        rollbackEx.printStackTrace();
    }
```

```
                        }
                }
        }
    private static void createStudent(Connection connection, String name, int age, String
major) throws SQLException {
        String sql = "INSERT INTO Student (StudentName, Age, Major) VALUES (?, ?,
        ?)";
        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
            pstmt.setString(1, name);
            pstmt.setInt(2, age);
            pstmt.setString(3, major);
            pstmt.executeUpdate();
            connection.commit();
            System.out.println("Student created successfully.");
        }
    }
    private static void readStudents(Connection connection) throws SQLException {
        String sql = "SELECT * FROM Student";
        try (Statement stmt = connection.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            System.out.println("Student List:");
            while (rs.next()) {
                int id = rs.getInt("StudentID");
                String name = rs.getString("StudentName");
                int age = rs.getInt("Age");
                ResultSet major = rs;
                        String major1 = rs.getString("Major");
                        System.out.printf("ID: %d, Name: %s, Age: %d, Major: %s%n", id, name,
age, major1);
                }
            }
        }
            private static void updateStudent(Connection connection, int studentId, String
name, int age, String major) throws SQLException {
                String sql = "UPDATE Student SET StudentName = ?, Age = ?, Major = ?
WHERE StudentID = ?";
```

```
        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
            pstmt.setString(1, name);
            pstmt.setInt(2, age);
            pstmt.setString(3, major);
            pstmt.setInt(4, studentId);
            int rowsAffected = pstmt.executeUpdate();
            connection.commit();
            if (rowsAffected > 0) {
                System.out.println("Student updated successfully.");
            } else {
                System.out.println("Student not found.");
            }
        }
    }
    private static void deleteStudent(Connection connection, int studentId) throws
    SQLException {
        String sql = "DELETE FROM Student WHERE StudentID = ?";
        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
            pstmt.setInt(1, studentId);
            int rowsAffected = pstmt.executeUpdate();
            connection.commit();
            if (rowsAffected > 0) {
                System.out.println("Student deleted successfully.");
            } else {
                System.out.println("Student not found.");
            }
        }
    }
}
```

4. **Output:**

```
Console ×  Properties
Student [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe  (Ma

Menu:
1. Create Student
2. Read Students
3. Update Student
4. Delete Student
5. Exit
Choose an option: 1
Enter Student Name: Kamal Mehta
Enter Age: 20
Enter Major: IT
Student created successfully.

Menu:
1. Create Student
2. Read Students
3. Update Student
4. Delete Student
5. Exit
Choose an option: 2
Student List:
ID: 1, Name: Kamal Mehta, Age: 20, Major: IT
```

Fig 3: Output for Problem 3

5. Learning Outcome:

I. Gained a comprehensive understanding of how to establish a connection between a Java application and a MySQL database using JDBC.

II. Developed the ability to write and execute SQL queries within a Java application to perform various operations such as data retrieval, insertion, updating, and deletion.

III. Acquired practical skills in implementing CRUD operations on database tables, ensuring a solid grasp of data manipulation techniques.

IV. Understood the importance of transaction management in maintaining data integrity and consistency during database operations.

V. Developed proficiency in handling exceptions that may arise during database operations, ensuring robust and error-resistant applications.

VI. Gained insights into the Model-View-Controller (MVC) design pattern and its application in Java applications.

VII. Enhanced object-oriented programming skills by designing and implementing classes that represent real-world entities, such as students and products, in the application.