Experiment 7

Student Name: Venky.PR
UID: 22BCS15514
Branch: BE/CSE
Section/Group: 22BCS_IOT-642/B
Semester: 6th
Date of Performance: 17/03/25
Subject Name: Project Based Learning in JAVA with Lab
Subject Code: 22CSH-359

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

2. **Objective:** The objective of this practical is to implement Java programs using Create Java applications with JDBC for database connectivity, CRUD operations, and MVC architecture.

3. **Implementaion\Code:**

7.1 : 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:

```
import com.sun.source.tree.StatementTree;

import java.sql.*;

public class Main {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/java";
        String user = "root";
        String password = "Sanjay_1";

        try {

            Connection conn = DriverManager.getConnection(url, user, password);
            Statement s= conn.createStatement();
            ResultSet rs = s.executeQuery("SELECT * FROM users");
```

```
            while (rs.next()) {
                System.out.println("ID: " + rs.getInt("idUsers") + ", Name: " + rs.getString("name"));
            }
            System.out.println("Connected to the database successfully!");

            conn.close();
        } catch (SQLException e) {
            System.out.println("Connection failed!");
            e.printStackTrace();
        }
    }
}
```

Output:

```
Connected to MySQL Successfully
id 1nameAlicesalary70000depart2manager0
id 2nameBobsalary50000depart2manager1
id 3nameCharliesalary60000depart1manager0
id 4nameDavidsalary40000depart2manager2
id 5nameEvesalary55000depart3manager0
id 6nameFranksalary45000depart3manager5
```

7.2: 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:

```
import java.sql.*;

class Sql {
    private Connection c;
    private static int numberOfPeople = 1;

    Sql() throws Exception {
        c = DriverManager.getConnection("jdbc:mysql://localhost:3306/java", "root", "Raghav_1");
        System.out.println("Database connected successfully!");
    }

    public void addRecord(String name, int number) throws Exception {
        String query = "INSERT INTO java.users VALUES (" +numberOfPeople+",'"+ name + "', "
```

```
+ number + ");";
        System.out.println("Executing Query: " + query);

        try (Statement s = c.createStatement()) {
            int rowsInserted = s.executeUpdate(query);
            System.out.println("Rows affected: " + rowsInserted);

            if (rowsInserted > 0) {
                System.out.println("Record inserted successfully!");
                numberOfPeople++;
            } else {
                System.out.println("Insertion failed!");
            }       }    }
    public void update(int id, String name) throws Exception {
        String query = "UPDATE users SET name='" + name + "' WHERE id=" + id;
        System.out.println("Executing Query: " + query);

        try (Statement s = c.createStatement()) {
            int rowsUpdated = s.executeUpdate(query);
            System.out.println("Rows affected: " + rowsUpdated);

            if (rowsUpdated > 0) {
                System.out.println("Record updated successfully!");
            } else {
                System.out.println("Update failed! Record not found.");
            }       }    }
    public void readRecords() throws Exception {
        String query = "SELECT * FROM users";
        System.out.println("Executing Query: " + query);

        try (Statement s = c.createStatement(); ResultSet rs = s.executeQuery(query)) {
            System.out.println("\nUser Records:");
            boolean foundRecords = false;

            while (rs.next()) {
                foundRecords = true;
                System.out.println("ID: " + rs.getInt("id") +
                        ", Name: " + rs.getString("name") +
                        ", Number: " + rs.getInt("number"));
            }
            if (!foundRecords) {
                System.out.println("No records found in the database.");
            }       }    }

    public void closeConnection() throws Exception {
        c.close();
        System.out.println("Database connection closed.");
    }
}
```

```
public class CRUD {
    public static void main(String[] args) {
        try {
            Sql db = new Sql();

            db.addRecord("Raghav", 100);
            db.readRecords();
            db.update(1, "John Doe");
            db.readRecords(); // Read again to confirm update

            db.closeConnection();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
Connected to MySQL Successfully
Table checked/created successfully.

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

Product List:
ID: 1, Name: Shampoo, Price: 100.0, Quantity: 5
ID: 3, Name: perfume, Price: 200.0, Quantity: 15
ID: 4, Name: Soap , Price: 30.0, Quantity: 10

Menu:
1. Create Product
2. Read Products
3. Update Product
4. Delete Product
5. Exit
Enter your choice: 1
Enter Product Name: FaceWash
Enter Price: 200
Enter Quantity: 2
Product added successfully!
```

7.3: 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:

```
class Student {
    int studentID;
    String name;
    String department;
    double marks;

    Student(int studentID, String name, String department, double marks) {
        this.studentID = studentID;
        this.name = name;
        this.department = department;
        this.marks = marks;
    }
}

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

class StudentController {
    private static final String URL = "jdbc:mysql://localhost:3306/student";
    private static final String USER = "root";
    private static final String PASSWORD = "Mysql@1234";

    public void addStudent(Student student) throws SQLException {
        String query = "INSERT INTO Student (StudentID, Name, Department, Marks) VALUES
(?, ?, ?, ?)";
        try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);
             PreparedStatement ps = conn.prepareStatement(query)) {
            ps.setInt(1, student.studentID);
            ps.setString(2, student.name);
            ps.setString(3, student.department);
            ps.setDouble(4, student.marks);
            ps.executeUpdate();
            System.out.println("Student added successfully.");
        }
    }
```

```
    public void viewStudents() throws SQLException {
        String query = "SELECT * FROM Student";
        try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(query)) {
            while (rs.next()) {
                System.out.println("ID: " + rs.getInt("StudentID") + ", Name: " + rs.getString("Name")
+
                                   ", Department: " + rs.getString("Department") + ", Marks: " +
rs.getDouble("Marks"));
            }
        }
    }

    public void updateStudent(int id, double marks) throws SQLException {
        String query = "UPDATE Student SET Marks = ? WHERE StudentID = ?";
        try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);
             PreparedStatement ps = conn.prepareStatement(query)) {
            ps.setDouble(1, marks);
            ps.setInt(2, id);
            ps.executeUpdate();
            System.out.println("Student record updated successfully.");
        }
    }

    public void deleteStudent(int id) throws SQLException {
        String query = "DELETE FROM Student WHERE StudentID = ?";
        try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);
             PreparedStatement ps = conn.prepareStatement(query)) {
            ps.setInt(1, id);
            ps.executeUpdate();
            System.out.println("Student record deleted successfully.");
        }
    }
}

// View (Menu-Driven)
public class StudentApp {
    public static void main(String[] args) {
        StudentController controller = new StudentController();
        Scanner sc = new Scanner(System.in);
        boolean exit = false;

        while (!exit) {
            System.out.println("1. Add Student\n2. View Students\n3. Update Student\n4. Delete
```

```
Student\n5. Exit");
            System.out.print("Choose an option: ");
            int choice = sc.nextInt();

            try {
                switch (choice) {
                    case 1 -> {
                        System.out.print("Enter ID: ");
                        int id = sc.nextInt();
                        sc.nextLine();
                        System.out.print("Enter Name: ");
                        String name = sc.nextLine();
                        System.out.print("Enter Department: ");
                        String dept = sc.nextLine();
                        System.out.print("Enter Marks: ");
                        double marks = sc.nextDouble();
                        controller.addStudent(new Student(id, name, dept, marks));
                    }
                    case 2 -> controller.viewStudents();
                    case 3 -> {
                        System.out.print("Enter Student ID to update: ");
                        int id = sc.nextInt();
                        System.out.print("Enter New Marks: ");
                        double marks = sc.nextDouble();
                        controller.updateStudent(id, marks);
                    }
                    case 4 -> {
                        System.out.print("Enter Student ID to delete: ");
                        int id = sc.nextInt();
                        controller.deleteStudent(id);
                    }
                    case 5 -> exit = true;
                    default -> System.out.println("Invalid option. Try again.");
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Output:

```
Database checked/created successfully.
Table checked/created successfully.

Menu:
1. Add Student
2. View Students
3. Update Student
4. Delete Student
5. Exit
Enter your choice: 1
Enter Student Name: Thakur
Enter Department: CSE-IT
Enter Marks: 95
Student added successfully!

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

Student List:
ID: 1, Name: Ishika, Department: CSE, Marks: 95.0
ID: 2, Name: Thakur, Department: CSE-IT, Marks: 95.0

Menu:
1. Add Student
2. View Students
3. Update Student
4. Delete Student
5. Exit
Enter your choice: 3
Enter Student ID to update: 2
Enter new Name: Ishika Thakur
Enter new Department: CSE-IT
```

4. Learning Outcomes:

- Database Connectivity with JDBC – Understood how to connect Java programs to MySQL databases and perform CRUD operations.
- MVC Architecture in Java – Implemented the Model-View-Controller (MVC) pattern for better separation of concerns in database applications.
- Data Processing & Aggregation – Used Java streams to group, filter, and compute statistics (like average price and max values) on datasets.
- Functional Programming Concepts – Practiced method references, functional interfaces, and stream operations for cleaner and more efficient code.