Experiment 7.1

Student Name: Vivek Garg **UID: 22BCS16972**
Branch: CSE **Section: 22BCS_EPAM-801/B**
Semester: 6th **DOP:17/03/25**
Subject: PBLJ **Subject Code:22CSH-359**

1. Aim:
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.

2. Objective:
To develop a Java program that connects to a MySQL database using JDBC and retrieves all records from an **Employee** table. The program utilizes the **DriverManager** and **Connection** classes to establish the connection, and uses SQL **SELECT** statements to display employee details such as **EmpID, Name, and Salary.**

3. Algorithm:
Step 1: Import SQL packages and define database connection details.
Step 2: Load JDBC driver and establish connection using **DriverManager.**
Step 3: Create Statement and execute SQL query to fetch employee records.
Step 4: Iterate ResultSet to display **EmpID, Name, and Salary.**
Step 5: Close resources and handle exceptions using try-catch.

4. Code:
```
import java.sql.*;

public class FetchEmployeeData {
    public static void main(String[] args) {
        String jdbcURL = "jdbc:mysql://localhost:3306/testdb";
        String username = "Vivek";
        String password = "Root1234";

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

            Connection conn = DriverManager.getConnection(jdbcURL, username, password);
            Statement stmt = conn.createStatement();

            String query = "SELECT EmpID, Name, Salary FROM Employee";
            ResultSet rs = stmt.executeQuery(query);

            while (rs.next()) {
                int empId = rs.getInt("EmpID");
                String name = rs.getString("Name");
                double salary = rs.getDouble("Salary");
```

```
                System.out.printf("EmpID: %d | Name: %s | Salary: %.2f\n", empId, name, salary);
            }

            rs.close();
            stmt.close();
            conn.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

5. Output:

```
PS D:\PBLJ\Exp7> javac -cp ".;mysql-connector-j-9.2.0.jar" FetchEmployeeData.java
PS D:\PBLJ\Exp7> java -cp ".;mysql-connector-j-9.2.0.jar" FetchEmployeeData
EmpID: 1 | Name: Vivek Garg | Salary: 100000.00
EmpID: 2 | Name: Vidhi | Salary: 100000.00
EmpID: 3 | Name: Rajat | Salary: 80000.00
PS D:\PBLJ\Exp7>
```

6. Learning Outcomes:

- Learn how to establish a connection between a Java application and a MySQL database using **DriverManager**.
- Gain hands-on experience in writing and executing SQL **SELECT** statements using Java.
- Learn how to retrieve and process data from a database using the **ResultSet** object.
- Understand how to use try-catch blocks to manage SQL and connection-related exceptions gracefully.
- Practice proper closing of JDBC resources like **Connection**, **Statement**, and **ResultSet** to avoid memory leaks.

Experiment 7.2

1. **Aim:** 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.

2. **Objective:** To develop a Java program that performs CRUD operations on a **Product** database table using JDBC, incorporating a menu-driven interface and transaction handling to ensure data integrity and smooth user interaction.

3. Algorithm:

Step 1: Start the program and connect to the MySQL database.

Step 2: Display menu with options: Add, View, Update, Delete, Exit.

Step 3: Take user input to select an option.

Step 4: Perform action (CRUD) based on input using SQL queries.

Step 5: Repeat until Exit is selected, then close the database connection.

4. Code:

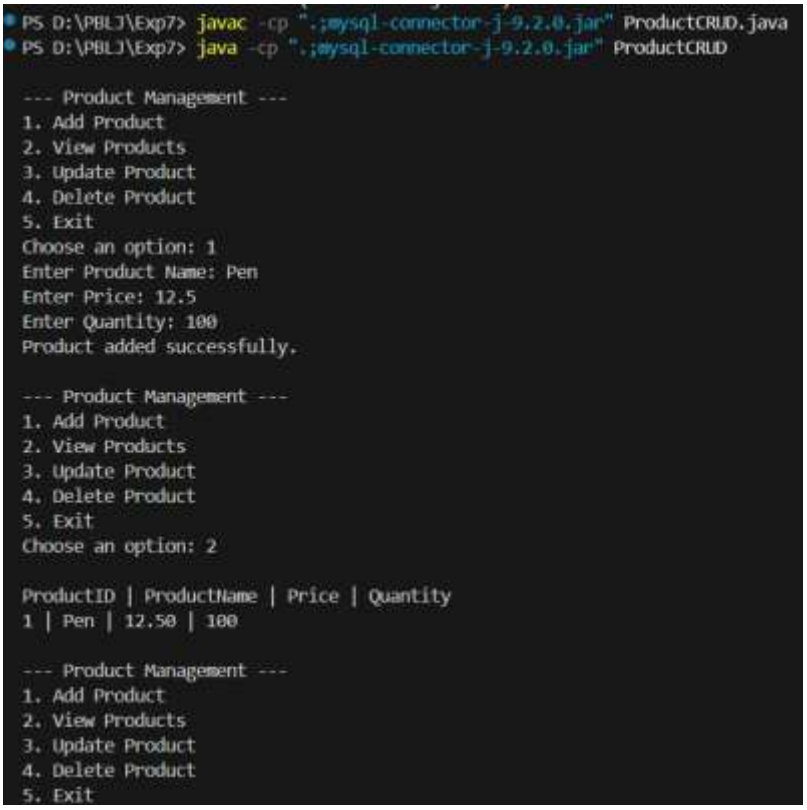
```
import java.sql.*;
import java.util.Scanner;

public class ProductCRUD {
    static final String JDBC_URL = "jdbc:mysql://localhost:3306/testdb";
    static final String USERNAME = "Vivek";
    static final String PASSWORD = "Root1234";

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

        try (Connection conn = DriverManager.getConnection(JDBC_URL, USERNAME,
PASSWORD)) {
            Class.forName("com.mysql.cj.jdbc.Driver");
            conn.setAutoCommit(false);

            while (true) {
                System.out.println("\n--- Product Management ---");
                System.out.println("1. Add Product");
                System.out.println("2. View 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();

                switch (choice) {
                    case 1:
```

```
                    addProduct(conn, scanner);
                    break;
                case 2:
                    viewProducts(conn);
                    break;
                case 3:
                    updateProduct(conn, scanner);
                    break;
                case 4:
                    deleteProduct(conn, scanner);
                    break;
                case 5:
                    conn.close();
                    System.out.println("Program terminated.");
                    return;
                default:
                    System.out.println("Invalid option.");
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

static void addProduct(Connection conn, Scanner scanner) {
    try {
        System.out.print("Enter Product Name: ");
        String name = scanner.next();
        System.out.print("Enter Price: ");
        double price = scanner.nextDouble();
        System.out.print("Enter Quantity: ");
        int quantity = scanner.nextInt();

        String sql = "INSERT INTO Product(ProductName, Price, Quantity) VALUES (?, ?, ?)";
        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, name);
            pstmt.setDouble(2, price);
            pstmt.setInt(3, quantity);
            pstmt.executeUpdate();
            conn.commit();
            System.out.println("Product added successfully.");
        }
    } catch (Exception e) {
        try {
            conn.rollback();
            System.out.println("Transaction rolled back.");
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
        e.printStackTrace();
    }
}
```

```
static void viewProducts(Connection conn) {
    try (Statement stmt = conn.createStatement()) {
        ResultSet rs = stmt.executeQuery("SELECT * FROM Product");
        System.out.println("\nProductID | ProductName | Price | Quantity");
        while (rs.next()) {
            System.out.printf("%d | %s | %.2f | %d\n",
                    rs.getInt("ProductID"),
                    rs.getString("ProductName"),
                    rs.getDouble("Price"),
                    rs.getInt("Quantity"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

static void updateProduct(Connection conn, Scanner scanner) {
    try {
        System.out.print("Enter Product ID to update: ");
        int id = scanner.nextInt();
        System.out.print("Enter new Price: ");
        double price = scanner.nextDouble();
        System.out.print("Enter new Quantity: ");
        int quantity = scanner.nextInt();

        String sql = "UPDATE Product SET Price=?, Quantity=? WHERE ProductID=?";
        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setDouble(1, price);
            pstmt.setInt(2, quantity);
            pstmt.setInt(3, id);
            int rows = pstmt.executeUpdate();
            if (rows > 0) {
                conn.commit();
                System.out.println("Product updated successfully.");
            } else {
                System.out.println("Product not found.");
            }
        }
    } catch (Exception e) {
        try {
            conn.rollback();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
        e.printStackTrace();
    }
}

static void deleteProduct(Connection conn, Scanner scanner) {
    try {
        System.out.print("Enter Product ID to delete: ");
        int id = scanner.nextInt();
```

```
            String sql = "DELETE FROM Product WHERE ProductID=?";
            try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
                pstmt.setInt(1, id);
                int rows = pstmt.executeUpdate();
                if (rows > 0) {
                    conn.commit();
                    System.out.println("Product deleted successfully.");
                } else {
                    System.out.println("Product not found.");
                }
            }
        } catch (Exception e) {
            try {
                conn.rollback();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
            e.printStackTrace();
        }
    }
}
```

5. Output:



```
Choose an option: 3
Enter Product ID to update: 1
Enter new Price: 15
Enter new Quantity: 100
Product updated successfully.

--- Product Management ---
1. Add Product
2. View Products
3. Update Product
4. Delete Product
5. Exit
Choose an option: 4
Enter Product ID to delete: 1
Product deleted successfully.

--- Product Management ---
1. Add Product
2. View Products
3. Update Product
4. Delete Product
5. Exit
Choose an option: 5
```

6. Learning Outcomes:

- Learn how to use JDBC to connect Java applications with MySQL databases.
- Gain hands-on experience with Create, Read, Update, and Delete operations in databases through Java.
- Understand how to use transactions to ensure data integrity during multiple operations.
- Learn how to design and code interactive, menu-based console applications.
- Improve skills in managing runtime errors (like **InputMismatchException**) to make programs user-friendly.

Experiment 7.3

1. **Aim:** 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.

2. **Objective:** To develop a Java application using JDBC and MVC architecture that allows users to perform CRUD operations on student data stored in a MySQL database, ensuring modularity, maintainability, and a clear separation of concerns between model, view, and controller components.

3. Algorithm:

Step 1: Start the application and display the menu with CRUD options.
Step 2: Capture user input for the selected operation (Add, View, Update, Delete).
Step 3: Perform database operations using the controller class based on input.
Step 4: Use the model (Student class) to represent and store student data.
Step 5: Repeat menu until the user selects exit.

4. Code:

- **Student.java**

```
public class Student {
    private int studentId;
    private String name;
    private String department;
    private double marks;
    public Student(int studentId, String name, String department, double marks) {
        this.studentId = studentId;
        this.name = name;
        this.department = department;
        this.marks = marks;
    }
    public int getStudentId() { return studentId; }
    public String getName() { return name; }
    public String getDepartment() { return department; }
    public double getMarks() { return marks; }

    @Override
    public String toString() {
        return String.format("ID: %d | Name: %s | Dept: %s | Marks: %.2f", studentId, name,
department, marks);
    }
}
```

- **StudentController.java**

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

public class StudentController {
    private Connection conn;
    public StudentController() {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/testdb", "Vivek",
"Root1234");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public void addStudent(Student student) {
        String sql = "INSERT INTO Student VALUES (?, ?, ?, ?)";
        try (PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setInt(1, student.getStudentId());
            stmt.setString(2, student.getName());
            stmt.setString(3, student.getDepartment());
            stmt.setDouble(4, student.getMarks());
            stmt.executeUpdate();
            System.out.println("Student added successfully.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public List<Student> getAllStudents() {
        List<Student> list = new ArrayList<>();
        String sql = "SELECT * FROM Student";
        try (Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                list.add(new Student(rs.getInt("StudentID"), rs.getString("Name"),
rs.getString("Department"), rs.getDouble("Marks")));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return list;
    }
    public void updateStudentMarks(int id, double newMarks) {
        String sql = "UPDATE Student SET Marks = ? WHERE StudentID = ?";
        try (PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setDouble(1, newMarks);
            stmt.setInt(2, id);
            int rows = stmt.executeUpdate();
            System.out.println(rows > 0 ? "Marks updated." : "Student not found.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

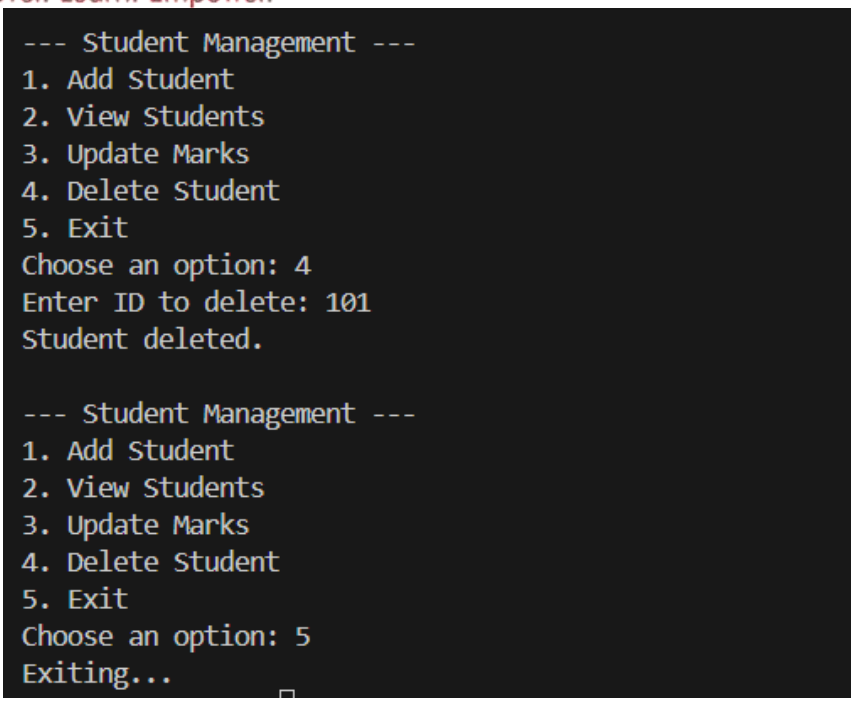
```
    public void deleteStudent(int id) {
        String sql = "DELETE FROM Student WHERE StudentID = ?";
        try (PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setInt(1, id);
            int rows = stmt.executeUpdate();
            System.out.println(rows > 0 ? "Student deleted." : "Student not found.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public void closeConnection() {
        try {
            if (conn != null) conn.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

- **StudentApp.java**

```
import java.util.List;
import java.util.Scanner;

public class StudentApp {
    public static void main(String[] args) {
        StudentController controller = new StudentController();
        Scanner sc = new Scanner(System.in);
        while (true) {
            System.out.println("\n--- Student Management ---");
            System.out.println("1. Add Student");
            System.out.println("2. View Students");
            System.out.println("3. Update Marks");
            System.out.println("4. Delete Student");
            System.out.println("5. Exit");
            System.out.print("Choose an option: ");
            int choice = sc.nextInt();
            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();
                    Student s = new Student(id, name, dept, marks);
                    controller.addStudent(s);
                    break;
                case 2:
                    List<Student> students = controller.getAllStudents();
                    for (Student stu : students) {
```

```
                    System.out.println(stu);
                }
                break;
            case 3:
                System.out.print("Enter ID to update marks: ");
                int uid = sc.nextInt();
                System.out.print("Enter new marks: ");
                double newMarks = sc.nextDouble();
                controller.updateStudentMarks(uid, newMarks);
                break;
            case 4:
                System.out.print("Enter ID to delete: ");
                int did = sc.nextInt();
                controller.deleteStudent(did);
                break;
            case 5:
                controller.closeConnection();
                System.out.println("Exiting...");
                return;

            default:
                System.out.println("Invalid option.");
        }
    }
  }
}
```

5. Output:



6. Learning Outcomes:

- Gained experience in structuring code using Model-View-Controller for better modularity and maintenance.
- Learned how to connect and interact with a MySQL database using Java JDBC.
- Successfully performed Create, Read, Update, and Delete operations on student data.
- Developed a user-friendly CLI interface for managing student records.
- Understood how to handle SQL and input-related exceptions for reliable user interaction.