

Experiment 7

Student Name: Jobanjeet Singh

UID: 22BCS15377

Branch: CSE 3rd Year

Section/Group: 640-A

Semester: 6th

Date of Performance: 17-03-25

Subject Name: Project Based Learning with JAVA

Subject Code: 22CSH-359

1. Aim:

Easy Level

Problem Statement:

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

Medium Level

Problem Statement:

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

Hard Level

Problem Statement:

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

Implementation/Code:

```
import java.sql.*;
```

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

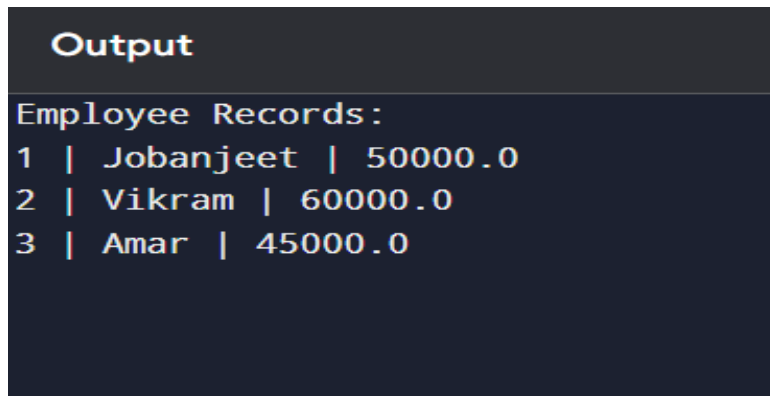
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection conn = DriverManager.getConnection(url, user, password);

            String sql = "SELECT EmpID, Name, Salary FROM Employee";
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(sql);

            System.out.println("Employee Records:");
            while (rs.next()) {
                int empId = rs.getInt("EmpID");
                String name = rs.getString("Name");
                double salary = rs.getDouble("Salary");
                System.out.println(empId + " | " + name + " | " + salary);
            }

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

OUTPUT:



```
Output
Employee Records:
1 | Jobanjeet | 50000.0
2 | Vikram | 60000.0
3 | Amar | 45000.0
```

Medium Level

Implementation/Code:


```

        System.out.println("Invalid choice!");
    }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

scanner.close();
try { conn.close(); } catch (SQLException e) {}
}

private static void createProduct(Scanner scanner) throws SQLException {
    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();

    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);
        int rows = pstmt.executeUpdate();
        System.out.println(rows + " product inserted successfully.");
    }
}

private static void readProducts() throws SQLException {
    String sql = "SELECT ProductID, ProductName, Price, Quantity FROM
Product";
    try (Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(sql)) {

        System.out.println("\nProduct 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("%d | %s | $%.2f | %d\n", id, name, price,
quantity);
        }
    }
}

private static void updateProduct(Scanner scanner) throws SQLException {

```

```
readProducts();
System.out.print("Enter ProductID to update: ");
int id = scanner.nextInt();
scanner.nextLine();

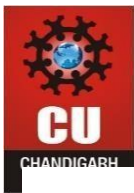
System.out.print("Enter new Product Name: ");
String name = scanner.nextLine();
System.out.print("Enter new Price: ");
double price = scanner.nextDouble();
System.out.print("Enter new Quantity: ");
int quantity = scanner.nextInt();

String sql = "UPDATE Product SET ProductName = ?, Price = ?, Quantity =
? WHERE ProductID = ?";
try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
    pstmt.setString(1, name);
    pstmt.setDouble(2, price);
    pstmt.setInt(3, quantity);
    pstmt.setInt(4, id);
    int rows = pstmt.executeUpdate();
    System.out.println(rows + " product updated successfully.");
}
}

private static void deleteProduct(Scanner scanner) throws SQLException {
    readProducts();
    System.out.print("Enter ProductID 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();
        System.out.println(rows + " product deleted successfully.");
    }
}
}
```

OUTPUT:



```
Output

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

Product List:
1 | Laptop | $999.99 | 10
2 | Phone | $599.99 | 20
3 | Shirt | $29.99 | 50|
```

Hard Level

Implementation/Code:

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

    public Student() {}

    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 void setStudentID(int studentID) { this.studentID = studentID; }

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

    public String getDepartment() { return department; }
```

```
public void setDepartment(String department) { this.department = department; }

public double getMarks() { return marks; }
public void setMarks(double marks) { this.marks = marks; }

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

Controller - StudentController.java

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

public class StudentController {
    private Connection conn;

    public StudentController() {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            conn = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/university", "root", "password");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void createStudent(Student student) {
        String sql = "INSERT INTO Student (Name, Department, Marks) VALUES (?, ?, ?)";
        try (PreparedStatement pstmt = conn.prepareStatement(sql,
            Statement.RETURN_GENERATED_KEYS)) {
            pstmt.setString(1, student.getName());
            pstmt.setString(2, student.getDepartment());
            pstmt.setDouble(3, student.getMarks());

            int affectedRows = pstmt.executeUpdate();
            if (affectedRows > 0) {
                try (ResultSet rs = pstmt.getGeneratedKeys()) {
                    if (rs.next()) {
                        student.setStudentID(rs.getInt(1));
                    }
                }
            }
        }
    }
}
```

```
} catch (SQLException e) {
    e.printStackTrace();
}

}

public List<Student> getAllStudents() {
    List<Student> students = new ArrayList<>();
    String sql = "SELECT StudentID, Name, Department, Marks FROM Student";

    try (Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(sql)) {

        while (rs.next()) {
            Student student = new Student();
            student.setStudentID(rs.getInt("StudentID"));
            student.setName(rs.getString("Name"));
            student.setDepartment(rs.getString("Department"));
            student.setMarks(rs.getDouble("Marks"));
            students.add(student);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return students;
}

public boolean updateStudent(Student student) {
    String sql = "UPDATE Student SET Name = ?, Department = ?, Marks = ?
WHERE StudentID = ?";
    try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setString(1, student.getName());
        pstmt.setString(2, student.getDepartment());
        pstmt.setDouble(3, student.getMarks());
        pstmt.setInt(4, student.getStudentID());

        return pstmt.executeUpdate() > 0;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
}

public boolean deleteStudent(int studentID) {
    String sql = "DELETE FROM Student WHERE StudentID = ?";
    try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setInt(1, studentID);

        return pstmt.executeUpdate() > 0;
    }
}
```



```
    } catch (SQLException e) {  
        e.printStackTrace();  
        return false;  
    }  
}  
  
public void closeConnection() {  
    try { if (conn != null) conn.close(); } catch (SQLException e) {}  
}  
}
```

View - StudentView.java

javaCopy

```
import java.util.List;  
import java.util.Scanner;  
  
public class StudentView {  
    private Scanner scanner = new Scanner(System.in);  
  
    public int showMainMenu() {  
        System.out.println("\nStudent Management System");  
        System.out.println("1. Add New 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 your choice: ");  
        return scanner.nextInt();  
    }  
  
    public Student getNewStudentDetails() {  
        scanner.nextLine(); // Consume newline  
        System.out.print("Enter Student Name: ");  
        String name = scanner.nextLine();  
        System.out.print("Enter Department: ");  
        String department = scanner.nextLine();  
        System.out.print("Enter Marks: ");  
        double marks = scanner.nextDouble();  
  
        return new Student(0, name, department, marks);  
    }  
  
    public void displayStudents(List<Student> students) {  
        System.out.println("\nList of Students:");  
        for (Student student : students) {  
            System.out.println(student);  
        }  
    }  
}
```

```
public int getStudentIDToUpdate() {
    System.out.print("Enter Student ID to update: ");
    return scanner.nextInt();
}

public Student getUpdatedStudentDetails(Student existingStudent) {
    scanner.nextLine(); // Consume newline
    System.out.print("Enter New Name (" + existingStudent.getName() + "):");
    String name = scanner.nextLine().trim();
    if (name.isEmpty()) name = existingStudent.getName();

    System.out.print("Enter New Department (" +
existingStudent.getDepartment() + "): ");
    String department = scanner.nextLine().trim();
    if (department.isEmpty()) department = existingStudent.getDepartment();

    System.out.print("Enter New Marks (" + existingStudent.getMarks() + "):");
    double marks = scanner.nextDouble();
    if (marks == 0) marks = existingStudent.getMarks();

    return new Student(existingStudent.getStudentID(), name, department,
marks);
}

public int getStudentIDToDelete() {
    System.out.print("Enter Student ID to delete: ");
    return scanner.nextInt();
}
}
```

Main Application - StudentManagementApp.java

```
public class StudentManagementApp {
    public static void main(String[] args) {
        StudentController controller = new StudentController();
        StudentView view = new StudentView();

        boolean running = true;
        while (running) {
            int choice = view.showMainMenu();

            switch (choice) {
                case 1:
                    Student newStudent = view.getNewStudentDetails();
                    controller.createStudent(newStudent);
                    System.out.println("Student added successfully!");
                    break;
                case 2:
```

```
List<Student> students = controller.getAllStudents();
view.displayStudents(students);
break;
case 3:
    int updateId = view.getStudentIDToUpdate();
    List<Student> currentStudents = controller.getAllStudents();
    Student existingStudent = null;
    for (Student s : currentStudents) {
        if (s.getStudentID() == updateId) {
            existingStudent = s;
            break;
        }
    }
    if (existingStudent != null) {
        Student updatedStudent =
view.getUpdatedStudentDetails(existingStudent);
        if (controller.updateStudent(updatedStudent)) {
            System.out.println("Student updated successfully!");
        } else {
            System.out.println("Failed to update student.");
        }
    } else {
        System.out.println("Student not found!");
    }
    break;
case 4:
    int deleteId = view.getStudentIDToDelete();
    if (controller.deleteStudent(deleteId)) {
        System.out.println("Student deleted successfully!");
    } else {
        System.out.println("Failed to delete student or student
not found.");
    }
    break;
case 5:
    running = false;
    break;
default:
    System.out.println("Invalid choice!");
}
}

controller.closeConnection();
System.out.println("Thank you for using Student Management System!");
}
}
```



OUTPUT:

Output

Student Management System

1. Add New Student
2. View All Students
3. Update Student
4. Delete Student
5. Exit

Enter your choice: 2

List of Students:

ID: 1 | Name: Jobanjeet | Dept: CSE | Marks: 85.50
ID: 2 | Name: Vikram | Dept: ECE | Marks: 90.75
ID: 3 | Name: Amar | Dept: ME | Marks: 88.25
ID: 4 | Name: Simran | Dept: CSE | Marks: 92.00
ID: 5 | Name: Chandan | Dept: ECE | Marks: 87.50

Learning Outcome:

- Implement MVC architecture in Java applications
- Separate concerns between model, view, and controller components
- Manage database connections efficiently in larger applications
- Create maintainable and scalable database applications
- Understand transaction management in complex operations