Assignment: Database Connectivity using JDBC

Name: Saurav Ugalkar

Roll No: 66

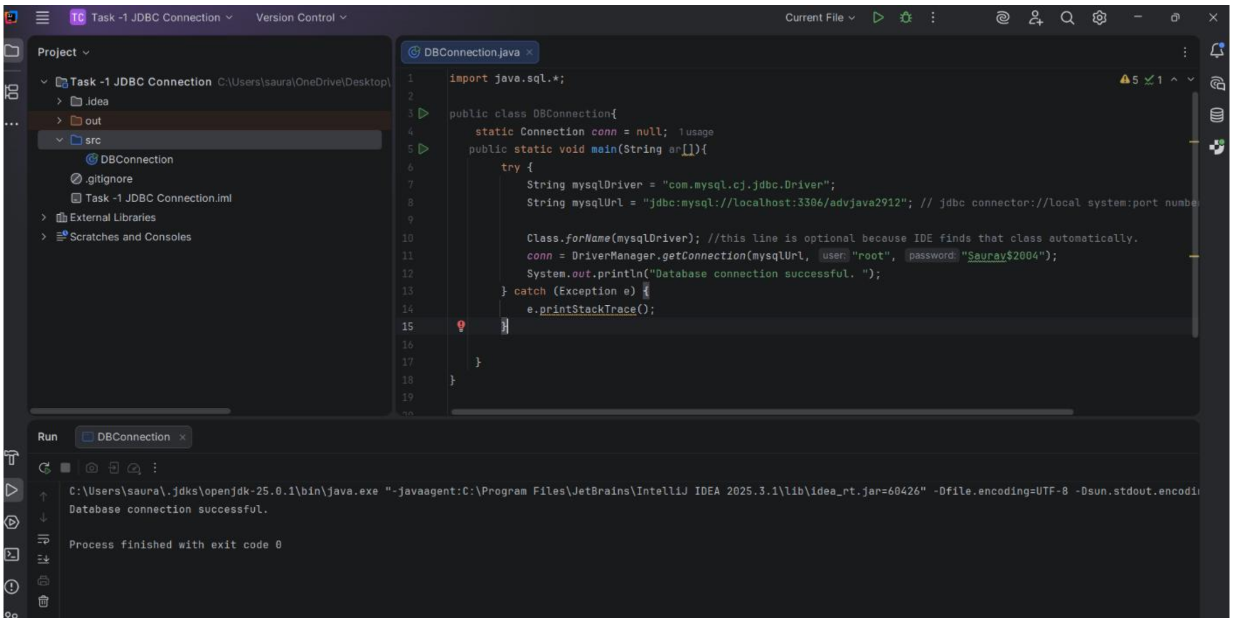
Task 1: Establishing Database Connection

File Name: DBConnection.java

1. Description

The primary objective of this task is to establish a connection between a Java application and a MySQL database named advjava. It utilizes the DriverManager class and the MySQL JDBC driver.

2. Source Code with output



```java
import java.sql.*;

public class DBConnection{
    static Connection conn = null;
   public static void main(String ar[]){
        try {
            String mysqlDriver = "com.mysql.cj.jdbc.Driver";
            String mysqlUrl = "jdbc:mysql://localhost:3306/advjava2912"; // jdbc connector://local system:port numbe

            Class.forName(mysqlDriver); //this line is optional because IDE finds that class automatically.
            conn = DriverManager.getConnection(mysqlUrl, "root", "Saurav$2004");
            System.out.println("Database connection successful. ");
        } catch (Exception e) {
            e.printStackTrace();
        }

    }
}
```

```
C:\Users\saura\.jdks\openjdk-25.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1\lib\idea_rt.jar=60426" -Dfile.encoding=UTF-8 -Dsun.stdout.encodi
Database connection successful.

Process finished with exit code 0
```

3. Execution Logic

- **Driver Loading:** Uses “com.mysql.cj.jdbc.Driver” to communicate with the MySQL server.
- **Connection String:** The URL “jdbc:mysql://localhost:3306/advjava” specifies the protocol, host, port, and database name.
- **Authentication:** Connects using the username root and the specified password.

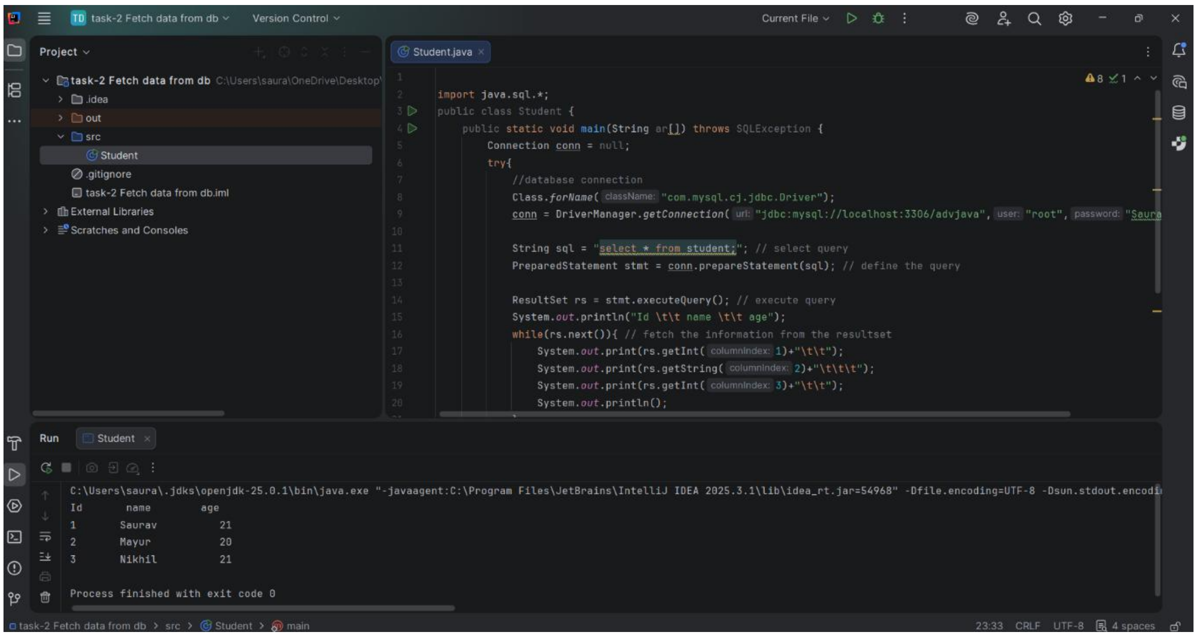
Task 2: Retrieving and Displaying Data

File Name: Student.java

1. Description

This task focuses on fetching all records from the student table and displaying them in a formatted tabular manner in the Java console.

2. Source Code with output


task-2 Fetch data from db
Version Control
Current File
Project
task-2 Fetch data from db C:\Users\saura\OneDrive\Desktop\
.idea
out
src
Student
.gitignore
task-2 Fetch data from db.iml
External Libraries
Scratches and Consoles
Student.java
Run
Student
task-2 Fetch data from db > src > Student > main
23:33 CRLF UTF-8 4 spaces

```java
import java.sql.*;
public class Student {
    public static void main(String ar[]) throws SQLException {
        Connection conn = null;
        try{
            //database connection
            Class.forName("com.mysql.cj.jdbc.Driver");
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/advjava", "root", "Saura

            String sql = "select * from student;"; // select query
            PreparedStatement stmt = conn.prepareStatement(sql); // define the query

            ResultSet rs = stmt.executeQuery(); // execute query
            System.out.println("Id \t\t name \t\t age");
            while(rs.next()){ // fetch the information from the resultset
                System.out.print(rs.getInt(1)+"\t\t");
                System.out.print(rs.getString(2)+"\t\t\t");
                System.out.print(rs.getInt(3)+"\t\t");
                System.out.println();
```

```
C:\Users\saura\.jdks\openjdk-25.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1\lib\idea_rt.jar=54968" -Dfile.encoding=UTF-8 -Dsun.stdout.encodi
Id       name        age
1       Saurav         21
2       Mayur          20
3       Nikhil         21

Process finished with exit code 0
```

3. Execution Logic

- **ResultSet:** Stores the data returned by the SELECT query.
- **While Loop:** Iterates through the rs.next() cursor to print each row's ID, Name, and Age.
- **Formatting:** Uses \t (tabs) to align columns in the output.

4. MySQL Verification

select * from student;

```
| id | name   | age |
+----+--------+------+
| 1  | Saurav |  21 |
| 2  | Mayur  |  20 |
| 3  | Nikhil  | 21 |
```

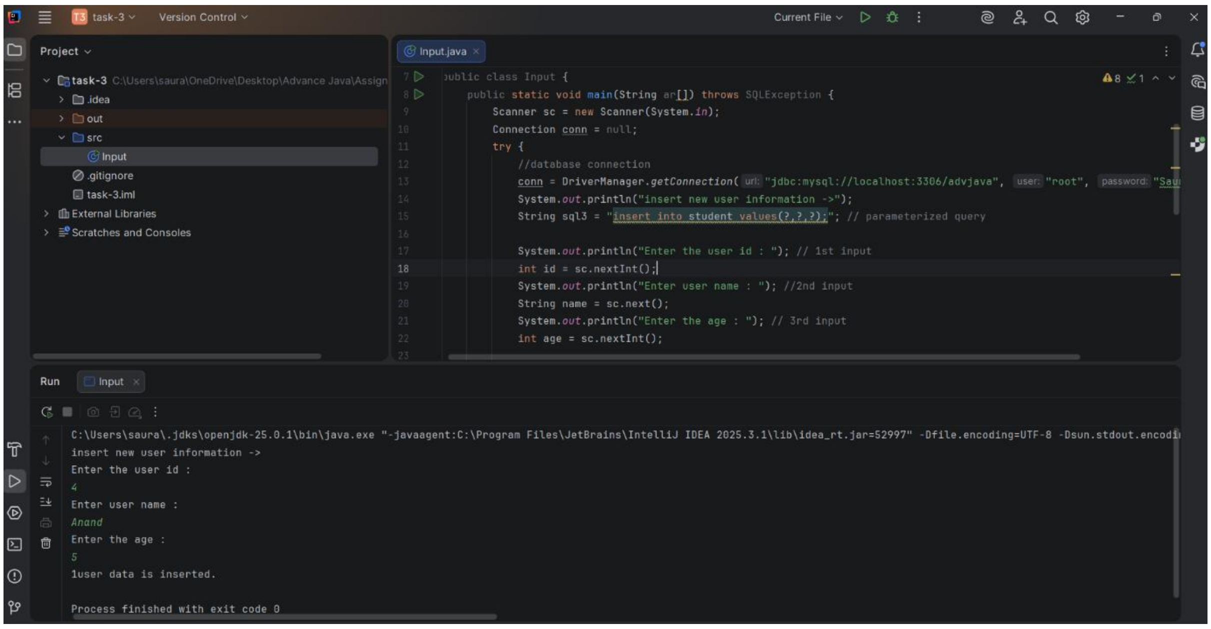
Task 3: Inserting Data into Database

File Name: Input.java

1. Description

This task demonstrates how to accept user input from the console and insert a new record into the student table using a “PreparedStatement”.

2. Source Code with output



```java
public class Input {
    public static void main(String ar[]) throws SQLException {
        Scanner sc = new Scanner(System.in);
        Connection conn = null;
        try {
            //database connection
            conn = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/advjava",  user: "root",  password: "Sau
            System.out.println("insert new user information ->");
            String sql3 = "insert into student values(?,?,?);"; // parameterized query

            System.out.println("Enter the user id : "); // 1st input
            int id = sc.nextInt();
            System.out.println("Enter user name : "); //2nd input
            String name = sc.next();
            System.out.println("Enter the age : "); // 3rd input
            int age = sc.nextInt();
```

```
C:\Users\saura\.jdks\openjdk-25.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1\lib\idea_rt.jar=52997" -Dfile.encoding=UTF-8 -Dsun.stdout.encodi
insert new user information ->
Enter the user id :
4
Enter user name :
Anand
Enter the age :
5
1user data is inserted.

Process finished with exit code 0
```

3. Execution Logic

- **Parameterized Query:** Uses ? placeholders to prevent SQL injection.
- **Scanner Class:** Captures id, name, and age from the user.
- **executeUpdate():** This method is called to perform the DML (Insert) operation.

4. MySQL Verification

select * from student;

| id | name | age |

+----+--------+------+

| 1 | Saurav | 21 |

| 2 | Mayur | 20 |

| 3 | Nikhil | 21 |

| 4 | Anand | 5 |

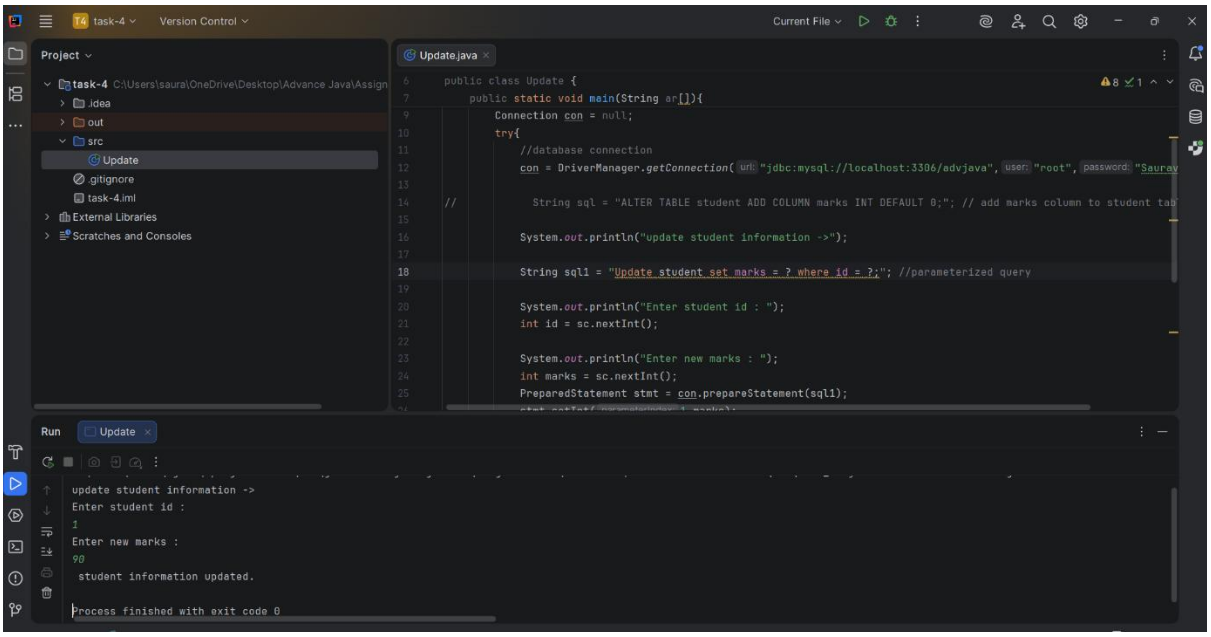
Task 4: Updating Existing Records

File Name: Update.java

1. Description

This task performs an update operation. It specifically takes a student's ID and a new value for 'marks', then updates the corresponding record in the database.

2. Source Code with output



```java
public class Update {
    public static void main(String ar[]){
        Connection con = null;
        try{
            //database connection
            con = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/advjava", user: "root", password: "Saurav
//            String sql = "ALTER TABLE student ADD COLUMN marks INT DEFAULT 0;"; // add marks column to student tab

            System.out.println("update student information ->");

            String sql1 = "Update student set marks = ? where id = ?;"; //parameterized query

            System.out.println("Enter student id : ");
            int id = sc.nextInt();

            System.out.println("Enter new marks : ");
            int marks = sc.nextInt();
            PreparedStatement stmt = con.prepareStatement(sql1);
```

```
update student information ->
Enter student id :
1
Enter new marks :
90
 student information updated.

Process finished with exit code 0
```

3. Execution Logic

- **SQL Query:** UPDATE student SET marks = ? WHERE id = ?;
- **Input Handling:** Allows dynamic updating by asking the user which ID needs a mark modification.
- **Result:** Confirms the update with a success message.

4. MySQL Verification

BEFORE UPDATION

select * from student;

```
| id | name   | age  | marks |
+----+--------+------+-------+
|  1 | Saurav |   21 |     0 |
|  2 | Mayur  |   20 |     0 |
|  3 | Nikhil |   21 |    0 |

|  4 | Anand  |    5 |    0 |
```

AFTER UPDATION

```
select * from student;

| id | name   | age  | marks |

+----+--------+------+-------+

|  1 | Saurav |   21 |   90 |

|  2 | Mayur  |   20 |    0 |

|  3 | Nikhil |   21 |    0 |

|  4 | Anand  |    5 |    0 |

|  5 | Akash  |   21 |    0 |
```

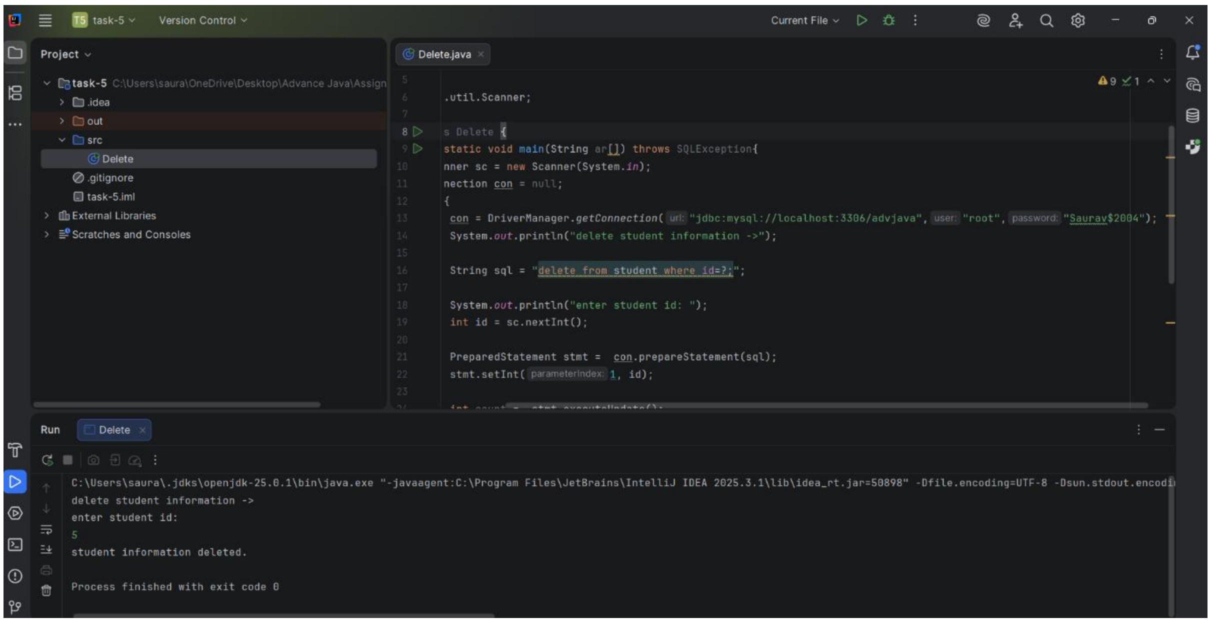
Task 5: Deleting Records from Database

File Name: Delete.java

1. Description

This task handles the removal of data. It prompts the user for a Student ID and deletes that specific record from the student table.

2. Source Code with output



```java
.util.Scanner;

s Delete {
static void main(String ar[]) throws SQLException{
nner sc = new Scanner(System.in);
nection con = null;
{
 con = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/advjava", user: "root", password: "Saurav$2004");
 System.out.println("delete student information ->");

 String sql = "delete from student where id=?;";

 System.out.println("enter student id: ");
 int id = sc.nextInt();

 PreparedStatement stmt =  con.prepareStatement(sql);
 stmt.setInt( parameterIndex: 1, id);
```

```
C:\Users\saura\.jdks\openjdk-25.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1\lib\idea_rt.jar=50898" -Dfile.encoding=UTF-8 -Dsun.stdout.encodi
delete student information ->
enter student id:
5
student information deleted.

Process finished with exit code 0
```

3. Execution Logic

- **Input:** The user provides the id of the student to be removed.
- **Statement Execution:** The PreparedStatement binds the ID and executes the delete command.
- **Cleanup:** Closes the connection after the operation.

4. MySQL Verification

```
select * from student;

| id | name   | age  | marks |

+----+--------+------+-------+

|  1 | Saurav |   21 |    90 |

|  2 | Mayur  |   20 |     0 |

|  3 | Nikhil |   21 |     0 |

|  4 | Anand  |    5 |     0 |
```