AMITY UNIVERSITY, PATNA

AMITY INSTITUTE OF INFORMATION TECHNOLOGY

Advanced Java Lab
LAB FILE

BCA

Name: Nishant Kumar
Program/Semester: BCA – 6 ‘A’
Enroll. Number: A45304821038
Submitted to : Dr. Naveen Kumar Singh

CRUD OPERATIONS

Problem description :

Develop a simple Java application that utilizes JDBC (Java Database Connectivity) to establish a connection with a relational database system and perform basic CRUD (Create, Read, Update, Delete) operations on a specified database table. The application should:

1. Provide options to perform CRUD operations including inserting new records into the database table, retrieving existing records from the table based on specified criteria, updating records in the table and deleting records from the table.

2. Implement error handling to manage connection failures and database operation exceptions gracefully.

The application should focus on simplicity and functionality, serving as a basic template for JDBC usage in CRUD operations

DESIGN

The design of the problem statement for creating a simple Java application that establishes JDBC connection and performs CRUD operations involves several key components and considerations:

1. User Interface Design :

Upon running the application, users will be presented with a menu containing 5 options, with 4 of them representing crud operations and the last option for exiting the application gracefully. Based on the user's choice, the application will invoke the appropriate method from the Student class to perform the CRUD operation.

2. Database Connection Management:

The application needs to establish a JDBC connection with the relational database system using the correct connection details.

3. Error Handling:

Error handling should be implemented to manage exceptions during database operations.

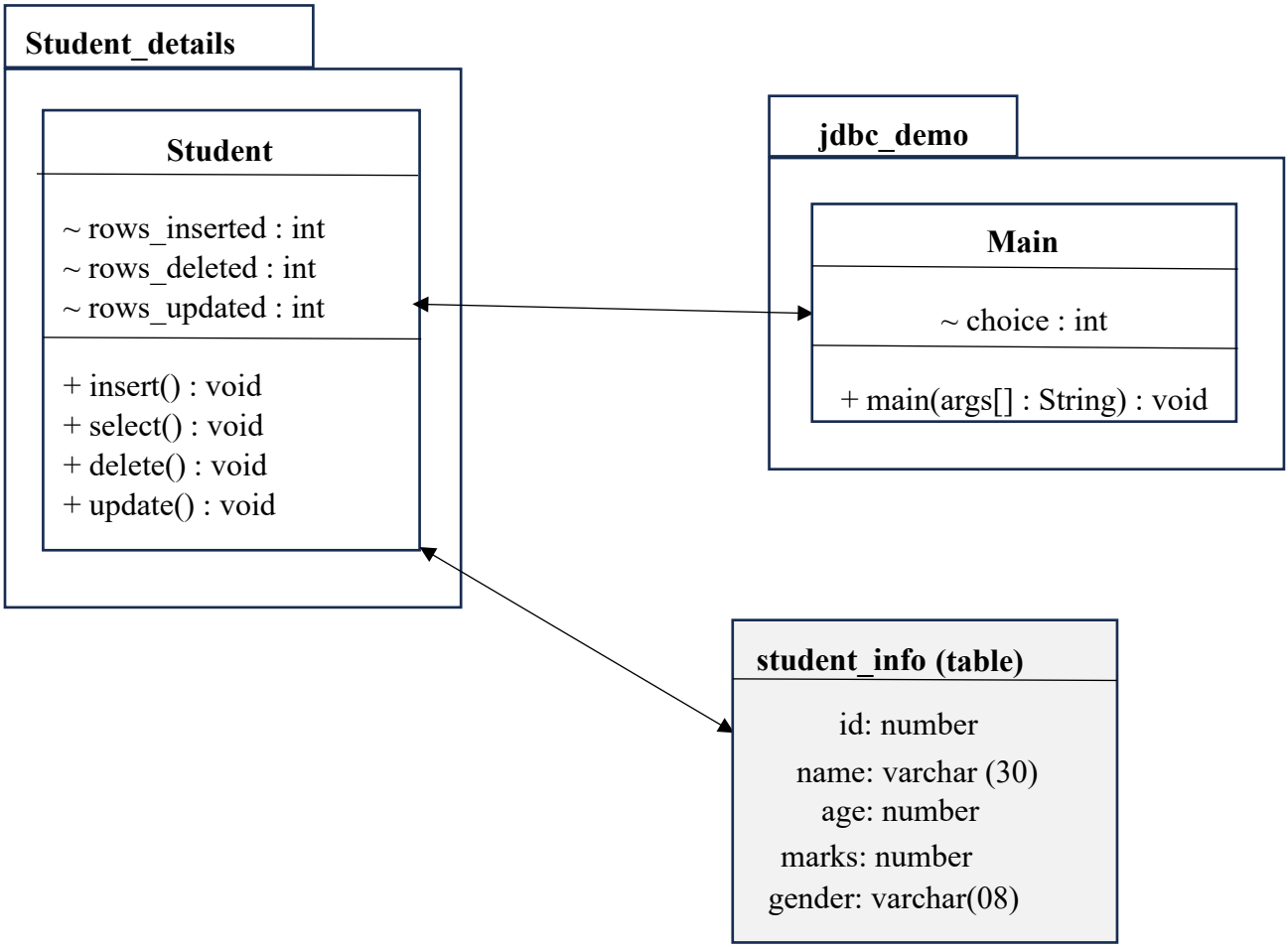
4. Code Modularity and Maintainability:

The application's code should be modular and well-organized, following best practices in software design and development. It should be easy to maintain and extend, allowing for future enhancements or modifications without significant refactoring.

5. Class Diagram:

A class diagram is crucial for design purposes as it visually illustrates the structure, relationships, and behavior of classes within a system. It aids in organizing and conceptualizing software components, facilitating communication among developers, guiding implementation, and ensuring consistency and scalability

throughout the design process. Here's a class diagram demonstrating our problem statement -



CODE

Student.java

```
 package bca.model;


import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.Scanner;


public class teacher {


public teacher() {

super();

// TODO Auto-generated constructor stub

}

public void addTeacher(Connection con, Scanner sc) throws SQLException {

//create statement

Statement st = con.createStatement();


//read teacher details

System.out.println("Enter teacher Id: ");

int id = sc.nextInt();


System.out.println("Enter teacher Name: ");

String name = sc.next();


System.out.println("Enter teacher Age: ");

int age = sc.nextInt();


System.out.println("Enter teacher Gender: ");

String gender = sc.next();
```

```
//create sql squery string

String query = String.format("Insert Into Teacher_info values(%d, '%s', %d,
'%s') ", id, name, age, gender);


//execute sql query

int rows = st.executeUpdate(query);


System.out.println(rows + " record inserted!!!");


}


public void displayTeachers(Connection con) throws SQLException {

Statement st = con.createStatement();


ResultSet rs = st.executeQuery("select * from Teacher_info");


while(rs.next()) {

      System.out.println(rs.getInt(1)+      "\t"+rs.getString(2)+      "\t"+
rs.getInt(3)+"\t"+rs.getString(4));

}

}




public void updateTeacherName(Connection con, Scanner sc) throws SQLException
{

Statement st = con.createStatement();

System.out.println("Enter teacher ID: ");

int id = sc.nextInt();

System.out.println("Enter teacher New Name: ");

String name = sc.next();
```

```
String query = String.format("update Teacher_info set name='%s' where id =
%d", name, id);

int rowsAffected = st.executeUpdate(query);

System.out.println(rowsAffected+" recored updated!!!");


}


public void deleteTeacher(Connection con, Scanner sc) throws SQLException {

Statement st = con.createStatement();

System.out.println("Enter teacher ID: ");

int id = sc.nextInt();


int rowAffected = st.executeUpdate("delete from Teacher_info where id =
"+id);

System.out.println(rowAffected + " recored deleted!!!");

}}
```

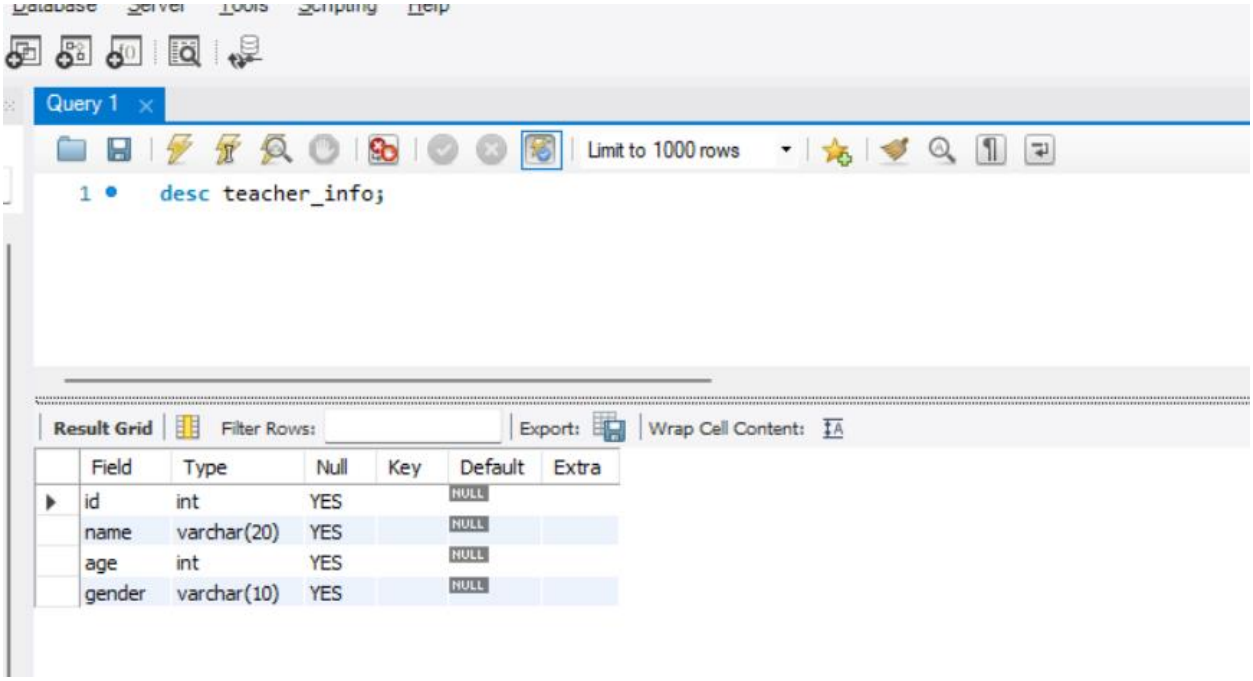
Main.java

```
package bca.drive;


import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.util.InputMismatchException;

import java.util.Scanner;


import bca.model.teacher;


public class Main {


public static void main(String[] args) throws ClassNotFoundException,
SQLException, InputMismatchException {

// TODO Auto-generated method stub

//1. load and register

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


//2

String url = "jdbc:mysql://localhost:3306/teachers";

String username = "root";

String pwd = "Mysql@2024";

Connection con = DriverManager.getConnection(url, username, pwd);

Scanner sc = new Scanner(System.in);

teacher s = new teacher();

//insert

//s.addTeacher(con, sc);

while(true) {


      menu();

      int choice = sc.nextInt();

      switch(choice) {

      case 1: s.addTeacher(con, sc);
```

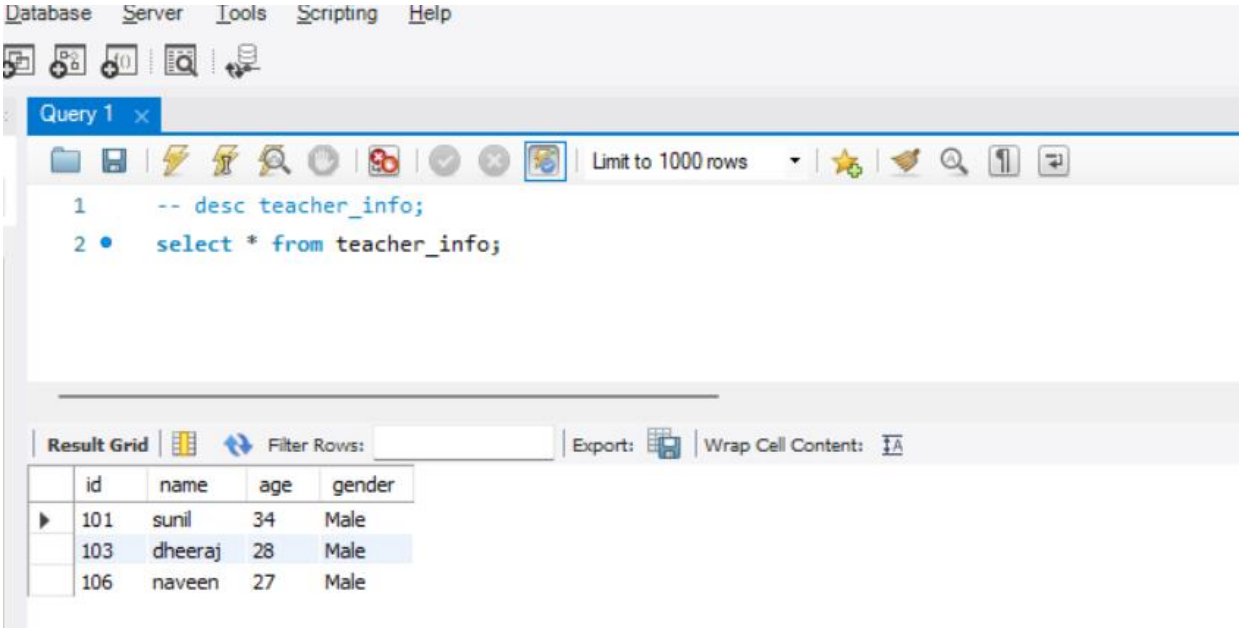
```
            break;

      case 2: s.displayTeachers(con);
            break;

      case 3: s.updateTeacherName(con, sc);
            break;

      case 4: s.deleteTeacher(con, sc);
            break;

      case 5:
            System.out.println("Bye Bye ...");
            System.exit(0);

      default:
            System.out.println("Wrong Choice...");
}}}
public static void menu() {
System.out.println("------------Menu-----------");
System.out.println("1. Add New teacher");
System.out.println("2. Display All Teachers");
System.out.println("3. Update Name of teacher");
System.out.println("4. Delete a teacher");
System.out.println("5. Exit");
System.out.println("Your Choice...");
}}
```

INPUT/OUTPUT

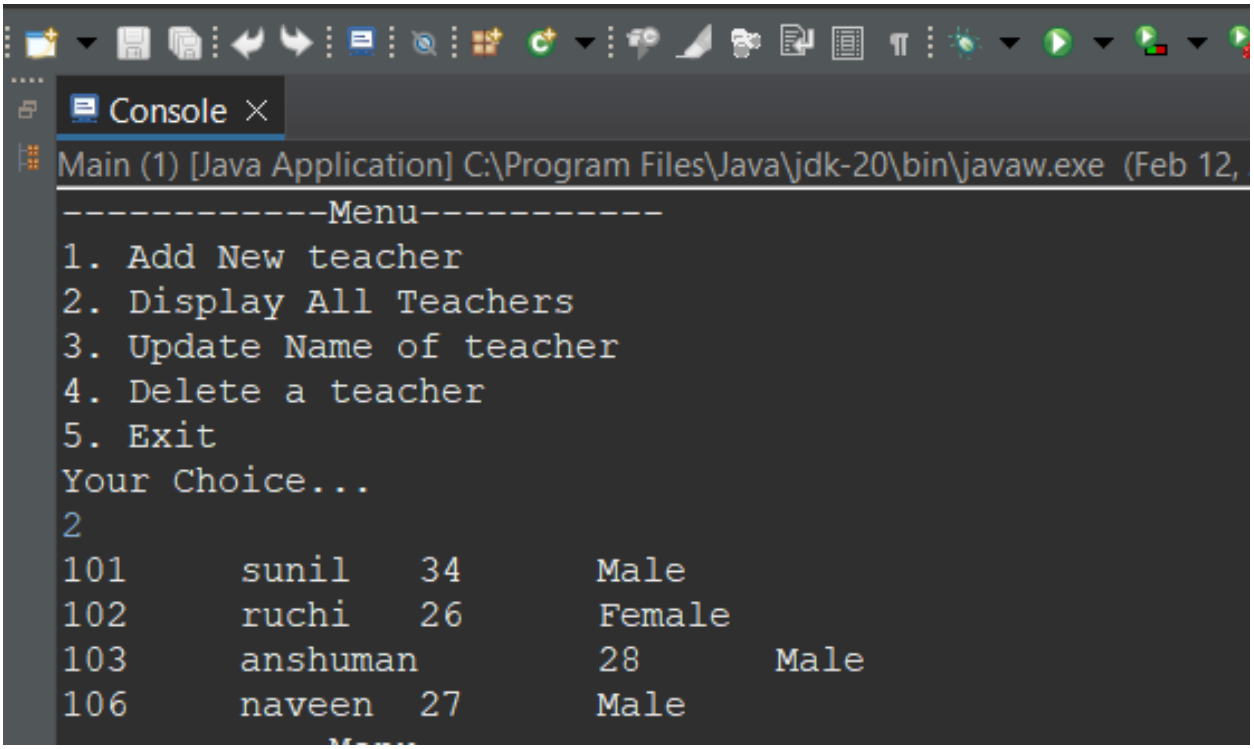
Describing the table



Selecting the table

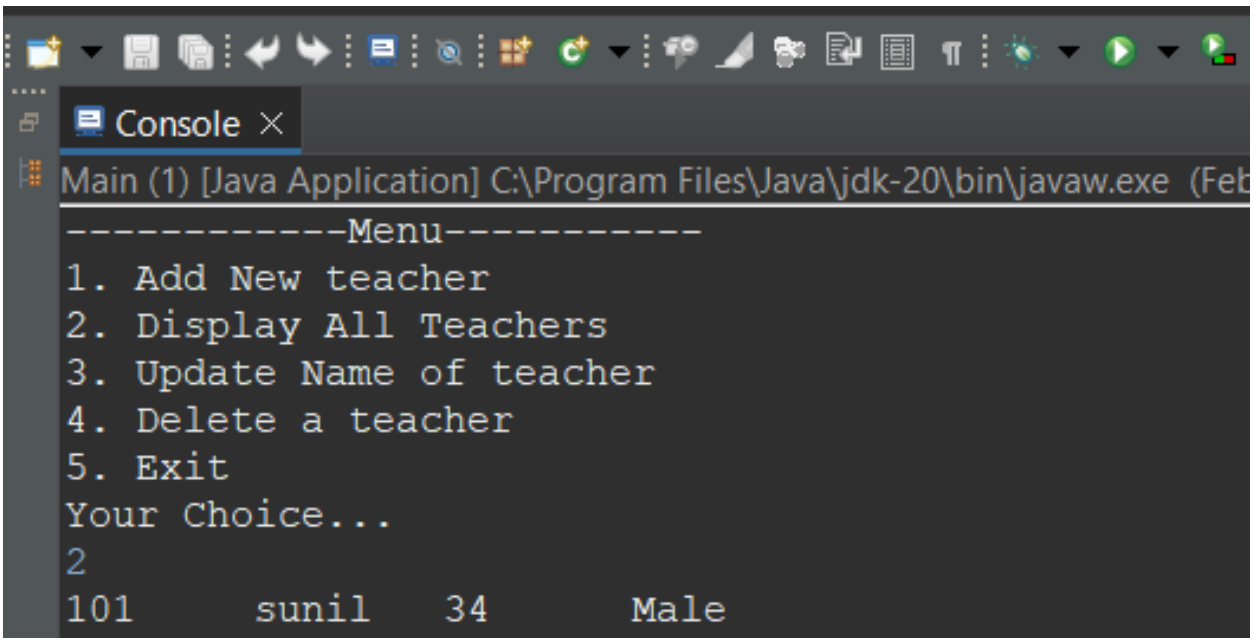


Display operation

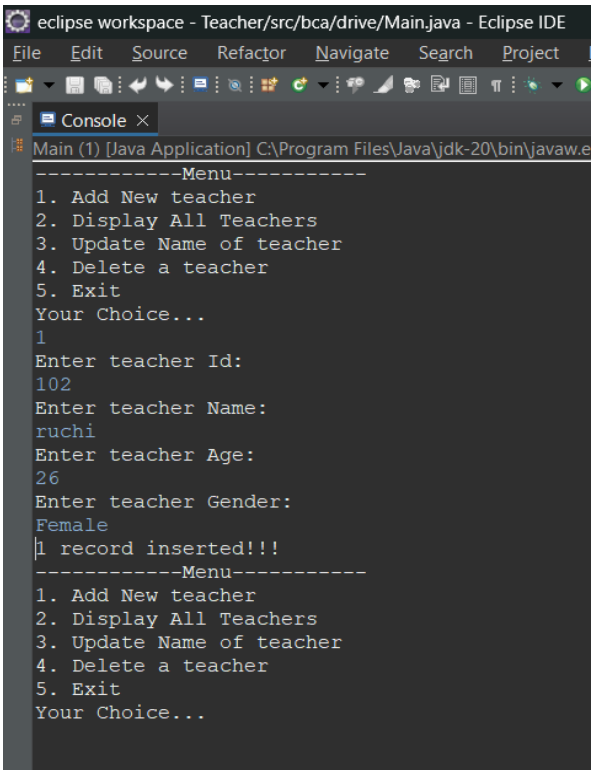


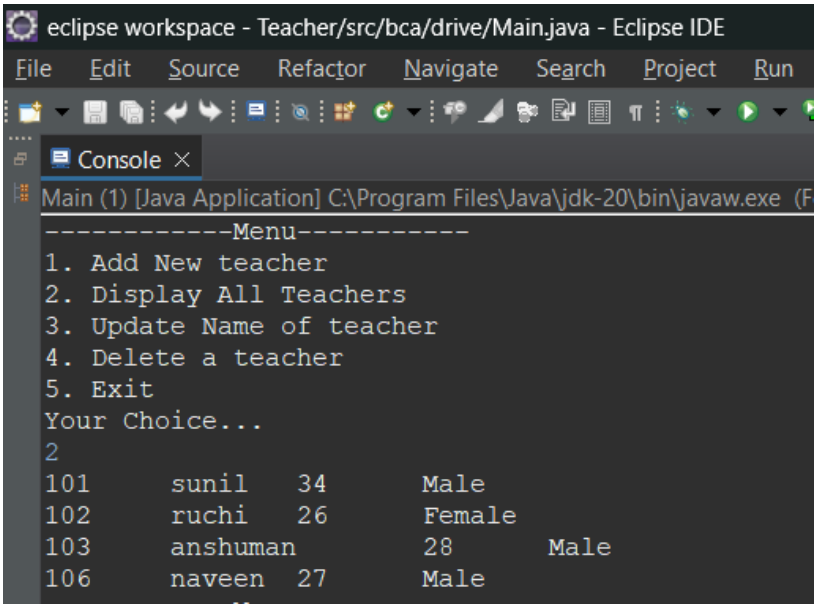
Insert operation

Before insertion:



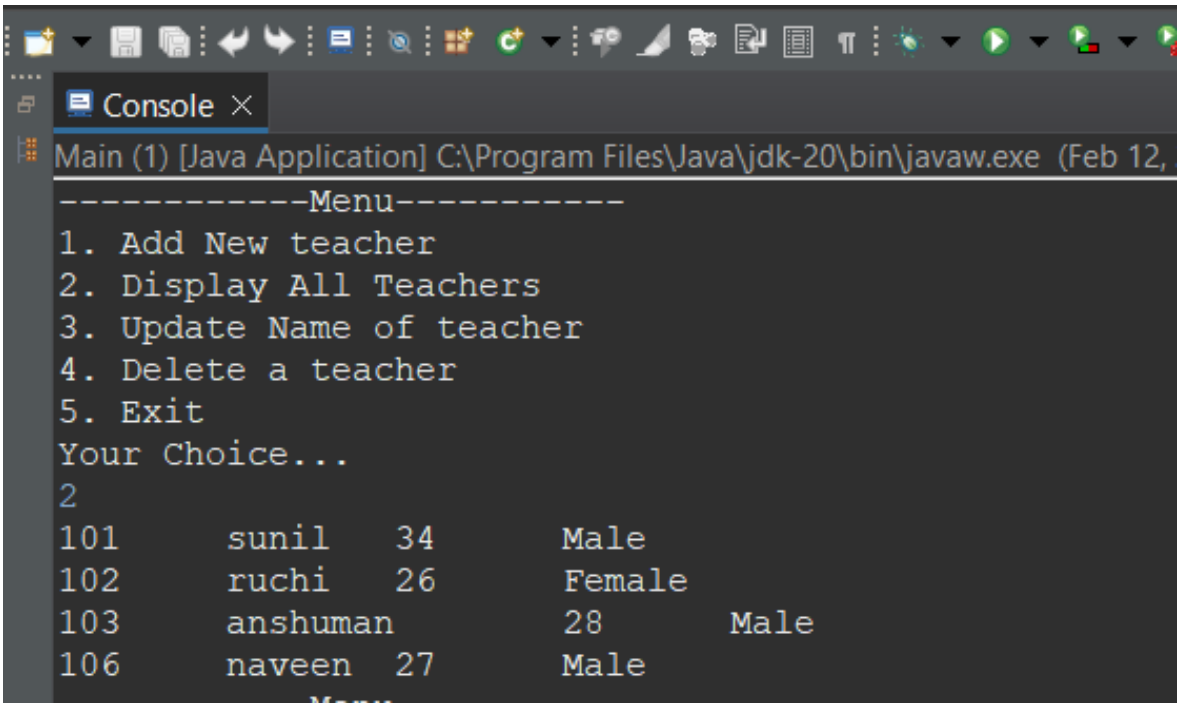
After insertion :



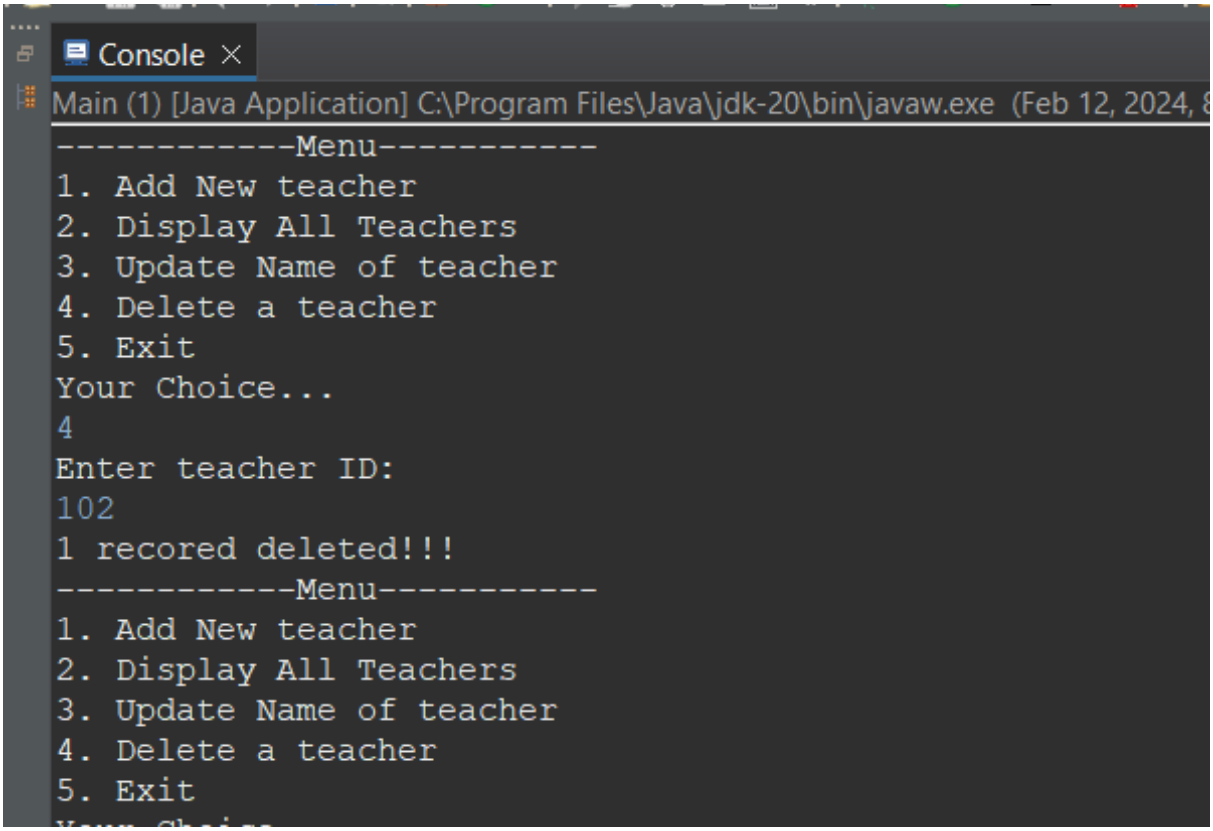


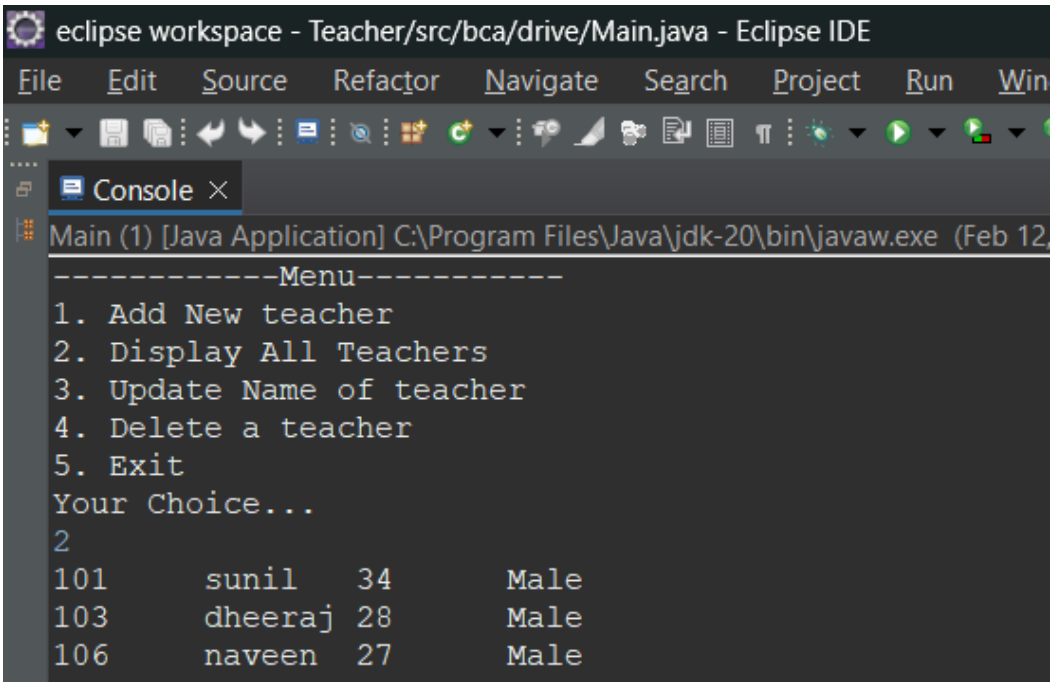
Delete operation

Before Deletion:



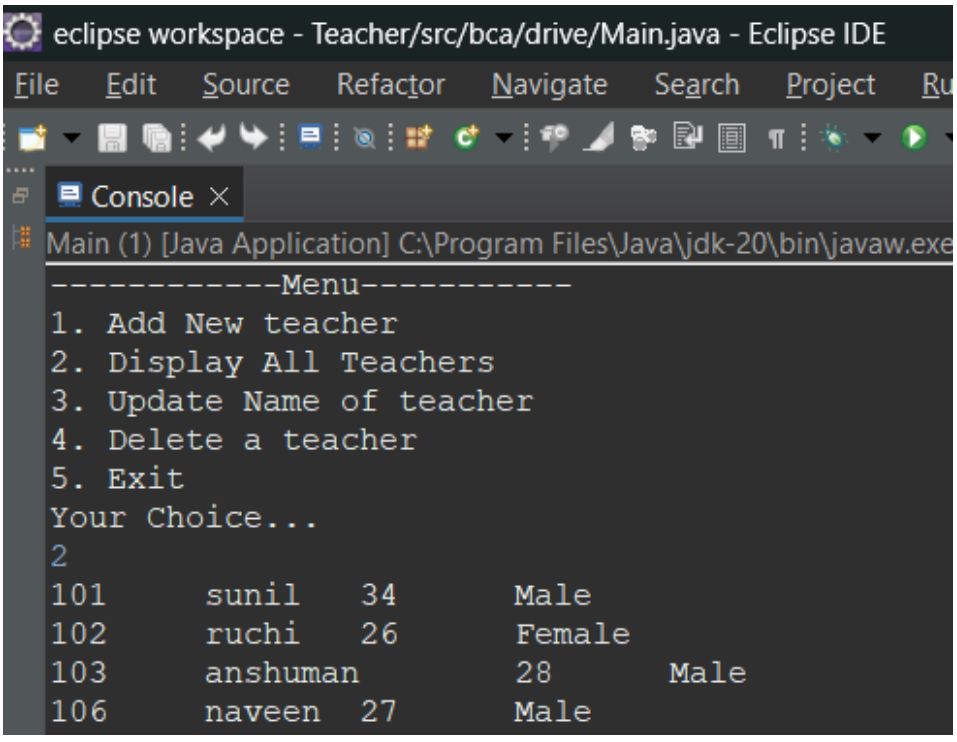
After Deletion:





Update operation

Before updation :



After updation: