

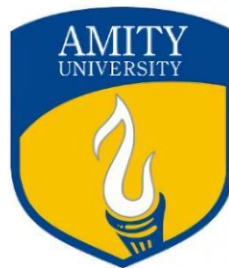
**AMITY UNIVERSITY, PATNA**

**AMITY INSTITUTE OF INFORMATION TECHNOLOGY**

**Advanced Java Lab**

**LAB FILE - 2**

**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:

## **Objective**

The objective of this project is to create a Java program in Eclipse that connects to a database using JDBC (Java Database Connectivity) and allows users to perform CRUD (Create, Read, Update, Delete) operations on an IPL\_TEAMS table. The program will present users with a menu containing seven options: insertion, read, update team's name, update team's captain, update team's coach, deletion of a record, and exit. Users will be able to interact with the program through the console to manipulate the team's records stored in the database.

## **Detailed Requirements:**

**1. Database Connectivity** - The program must establish a connection to a relational database management system (RDBMS) using JDBC. Database connection parameters such as URL, username, and password should be configurable and provided as constants or properties in the program.

**2. Menu Options** - The program should display a menu with options like Insert, Read, Update team's name, Update team's captain, Update team's coach, Deletion and Exit option. Users will input the corresponding option number to execute the desired operation.

**3. Insert Operation** - Upon selecting the insert option, users will be prompted to input the team's ID, name, city, captain, and coach through the keyboard. The program will then insert this data into the IPL\_TEAMS table in the database.

**4. Read Operation** - Selecting the read option will display all records from the IPL\_TEAMS table. The program should fetch and display the team's ID, name, city, captain, and coach for each record in the table.

**5. Update Operation** - The program should provide separate options to update the team's name, captain, and coach. Upon selecting any of these options, users will input the team ID whose details need to be updated, followed by the new value for the corresponding field. The program will then update the specified field for the given team ID in the database.

**6. Delete Operation** - Selecting the delete option will prompt users to input the team ID of the record they want to delete. The program will delete the record with the specified team ID from the IPL\_TEAMS table in the database.

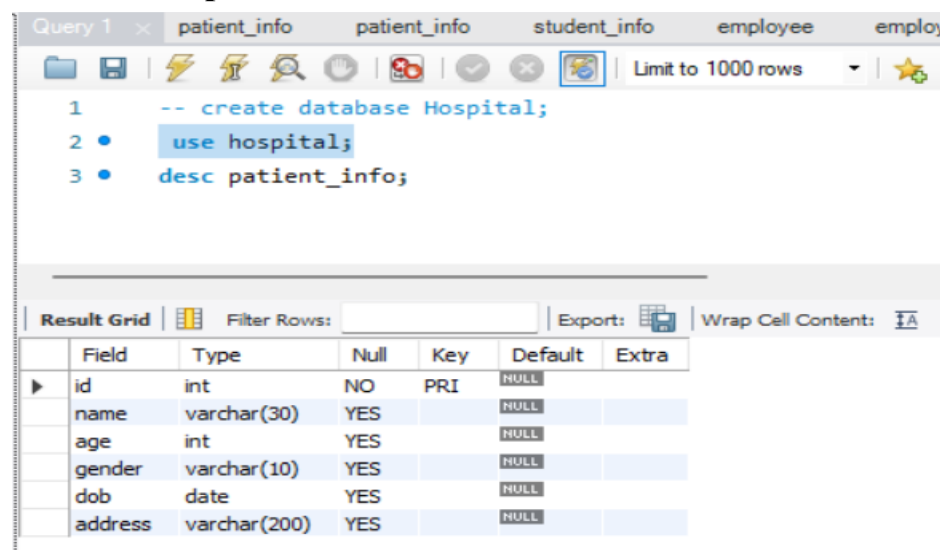
**7. Error Handling** - The program should handle exceptions gracefully, displaying meaningful error messages to users in case of connection failures, SQL errors, or invalid input. It should also validate user inputs to prevent SQL injection attacks or database errors due to incorrect data formats.

**8. Exit** - Upon selecting the exit option, the program will terminate gracefully, closing any open resources and releasing database connections.

**Tools and Technologies used:** Eclipse IDE and MySQL orkbench 8.0 CE

## **Design**

### **Table description –**



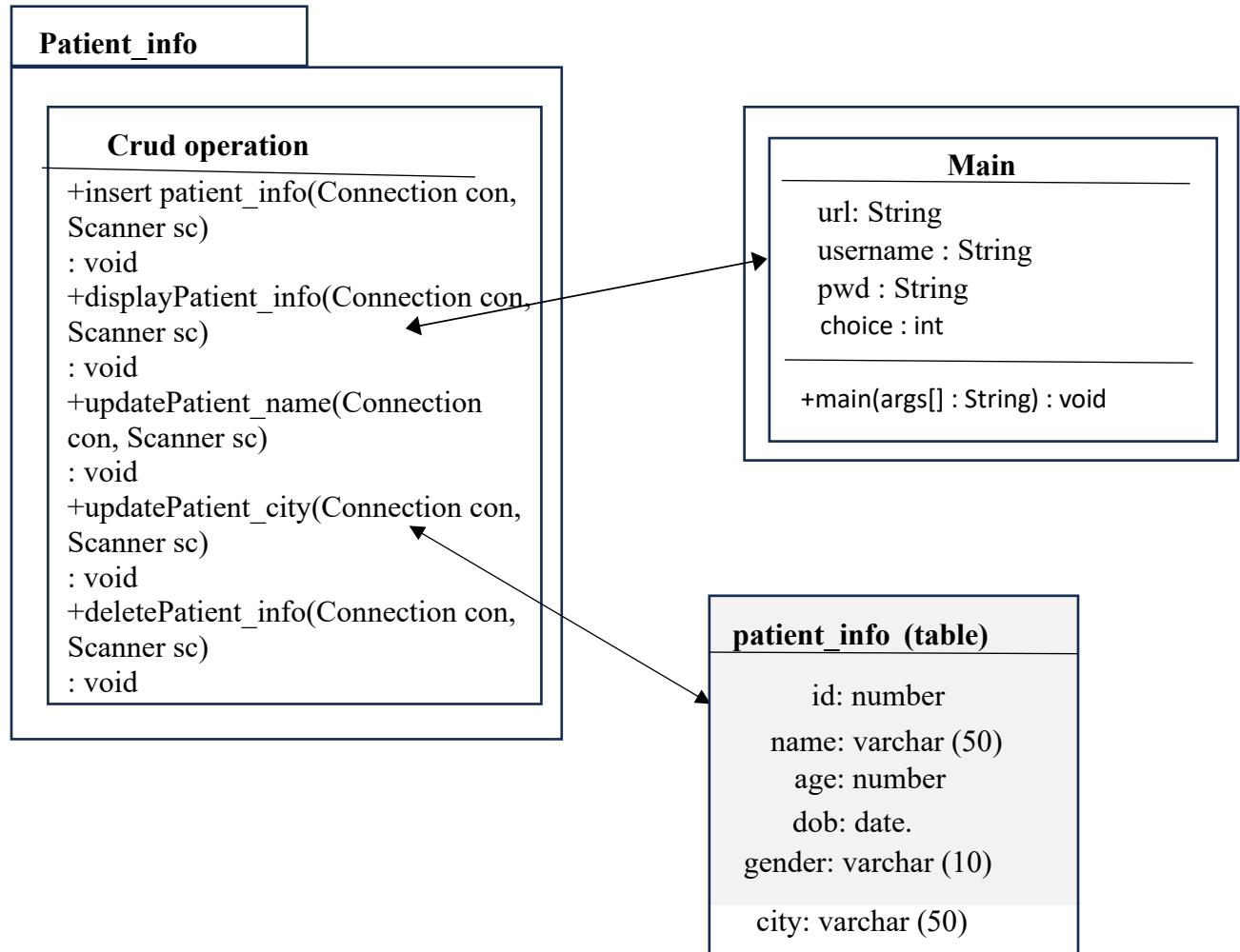
The screenshot shows the MySQL Workbench interface. The top pane displays three SQL queries:

```
1 -- create database Hospital;
2 use hospital;
3 desc patient_info;
```

The bottom pane shows the 'Result Grid' for the 'patient\_info' table. The table has the following structure:

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	
name	varchar(30)	YES		NULL	
age	int	YES		NULL	
gender	varchar(10)	YES		NULL	
dob	date	YES		NULL	
address	varchar(200)	YES		NULL	

**Class Diagram:** The class diagram for our project will be as follows -



# CODE

## **Patient info.java**

```
package hospital.details;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;

public class Patient_info {

    public Patient_info() {
        // TODO Auto-generated constructor stub
    }

    public void addPatient_info(Connection con, Scanner sc) throws SQLException
    {
        Statement st = con.createStatement();

        //read Patient details
        System.out.println("Enter Patient Id: ");
        int id = sc.nextInt();

        System.out.println("Enter Patient Name: ");
        String name = sc.next();

        System.out.println("Enter Patient Age: ");
        int age = sc.nextInt();
```

```
System.out.println("Enter Patient Gender: ");

String gender = sc.next();

System.out.println("Enter Patient Date of Birth (DOB) in format YYYY/MM/DD:");

String dob = sc.next();

System.out.println("Enter Patient City: ");

String city = sc.next();

//create sql squery string

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

//execute sql query

int rows = st.executeUpdate(query);

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

}

public void displayPatient_info(Connection con) throws SQLException

{

Statement st = con.createStatement();

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

System.out.println("Displaying all patient info");

while(rs.next()) {

    System.out.println(rs.getInt(1)+ "\t"+rs.getString(2)+ "\t"+
rs.getInt(3)+"\t"+rs.getString(4)+"\t"+rs.getString(5)+"\t"+rs.getString(6)
);

}

}
```

```

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

    Statement st = con.createStatement();

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

    int id = sc.nextInt();

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

    String name = sc.next();

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

    int rowsAffected = st.executeUpdate(query);

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

}

```

```

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

    Statement st = con.createStatement();

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

    int id = sc.nextInt();

    System.out.println("Enter Patient New City name: ");

    String city = sc.next();

    String query = String.format("update patient_info set city='%s' where id =
%d", city, id);

    int rowsAffected = st.executeUpdate(query);

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

}

```

```

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

    Statement st = con.createStatement();

```



```
System.out.println("Enter Patient ID: ");

int id = sc.nextInt();

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

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

}
```

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

```
// TODO Auto-generated method stub
```

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

```
String url = "jdbc:mysql://localhost:3306/hospital";
```

```
String username = "root";
```

```
String pwd = "Mysql@2024";
```

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

```
Scanner sc = new Scanner(System.in);
```

```
Patient_info pi = new Patient_info();
```

```
while(true) {
```

```
    menu();
```

```
    int choice = sc.nextInt();
```

```
    switch(choice) {
```

```
        case 1: pi.addPatient_info(con, sc);
```

```
            break;
```

```
        case 2: pi.displayPatient_info(con);
```

```
            break;
```

```
        case 3: pi.updatePatient_name(con, sc);
                break;
        case 4:
                pi.updatePatient_city(con, sc);
                break;

        case 5: pi.deletePatient_info(con, sc);
                break;

        case 6:
                System.out.println("Bye Bye ...");
                System.exit(0);

        default:
                System.out.println("Wrong Choice...");
    }

}

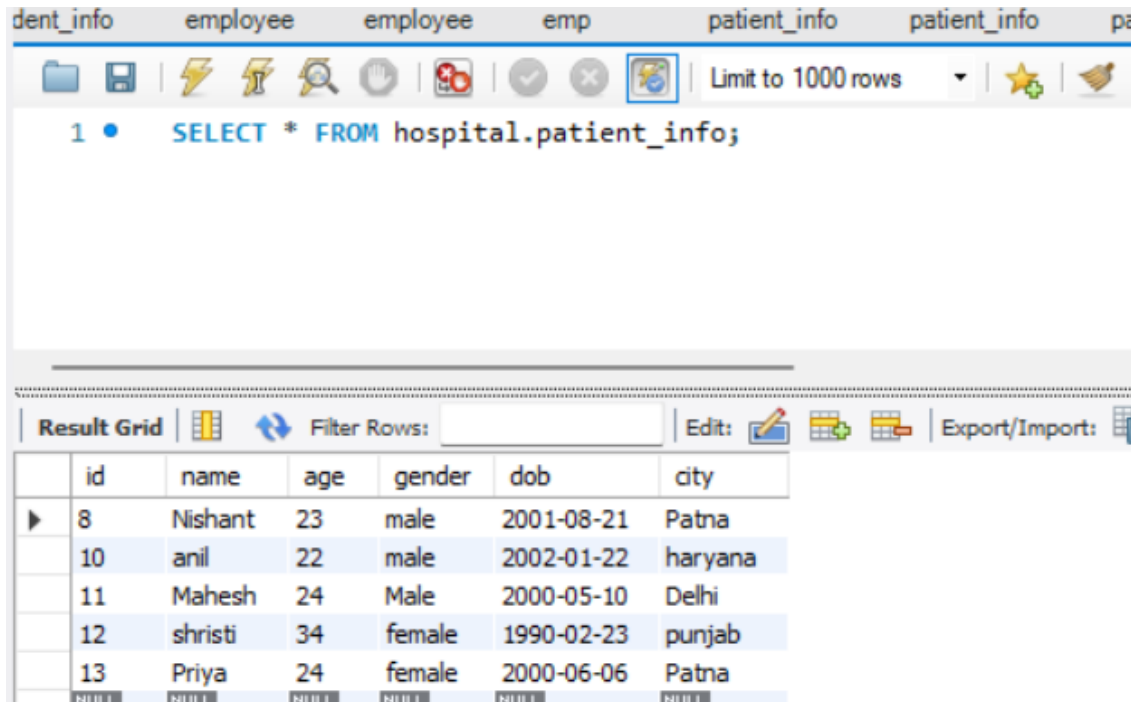
}

public static void menu() {
    System.out.println("-----Menu-----");
    System.out.println("1. Add New Patient");
    System.out.println("2. Display All Patients");
    System.out.println("3. Update Name of Patient");
    System.out.println("4. Update City of Patient");
    System.out.println("5. Delete a Patient details");
    System.out.println("6. Exit");
    System.out.println("Your Choice...");
}

}
```

# INPUT/OUTPUT

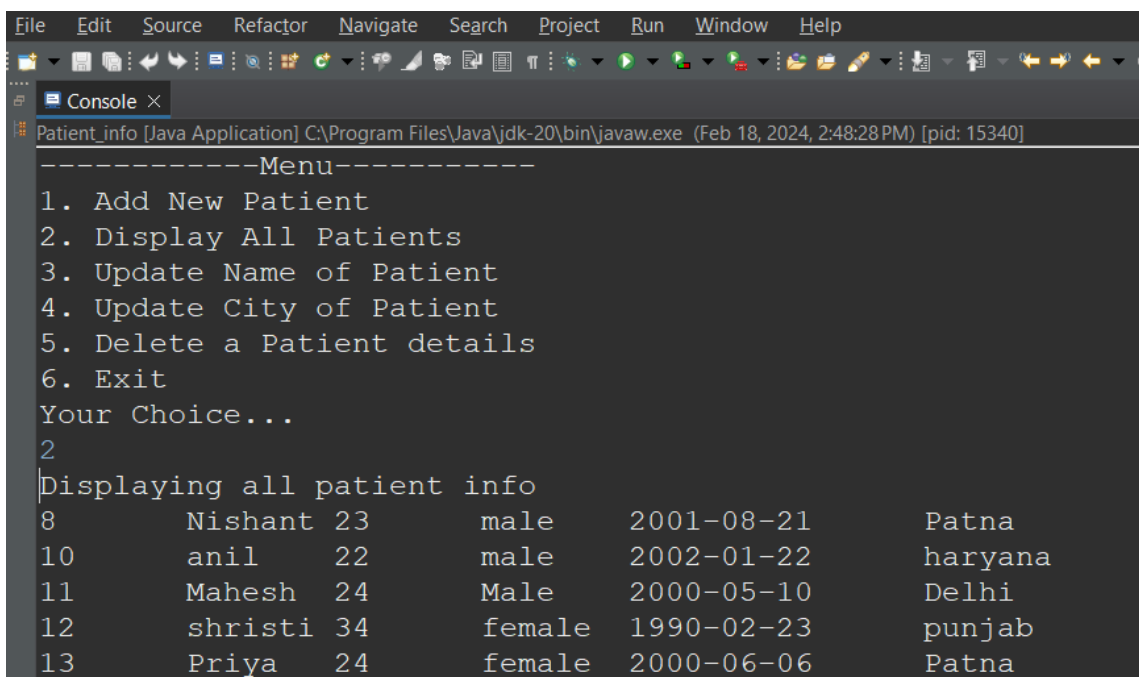
## Selecting the table



The screenshot shows a database management interface. At the top, there are tabs for different tables: 'dent\_info', 'employee', 'employee', 'emp', 'patient\_info', 'patient\_info', and 'pa'. Below the tabs is a toolbar with various icons. A SQL query is entered in the main area: `1 • SELECT * FROM hospital.patient_info;`. Below the query, there is a 'Result Grid' section. It includes a 'Filter Rows:' field and an 'Edit:' button. The result grid displays a table with the following data:

	id	name	age	gender	dob	city
▶	8	Nishant	23	male	2001-08-21	Patna
	10	anil	22	male	2002-01-22	haryana
	11	Mahesh	24	Male	2000-05-10	Delhi
	12	shruti	34	female	1990-02-23	punjab
	13	Priya	24	female	2000-06-06	Patna

## Display operation



The screenshot shows a Java application window titled 'Patient\_info [Java Application]'. The console output displays a menu and the results of a 'Display All Patients' operation. The menu is as follows:

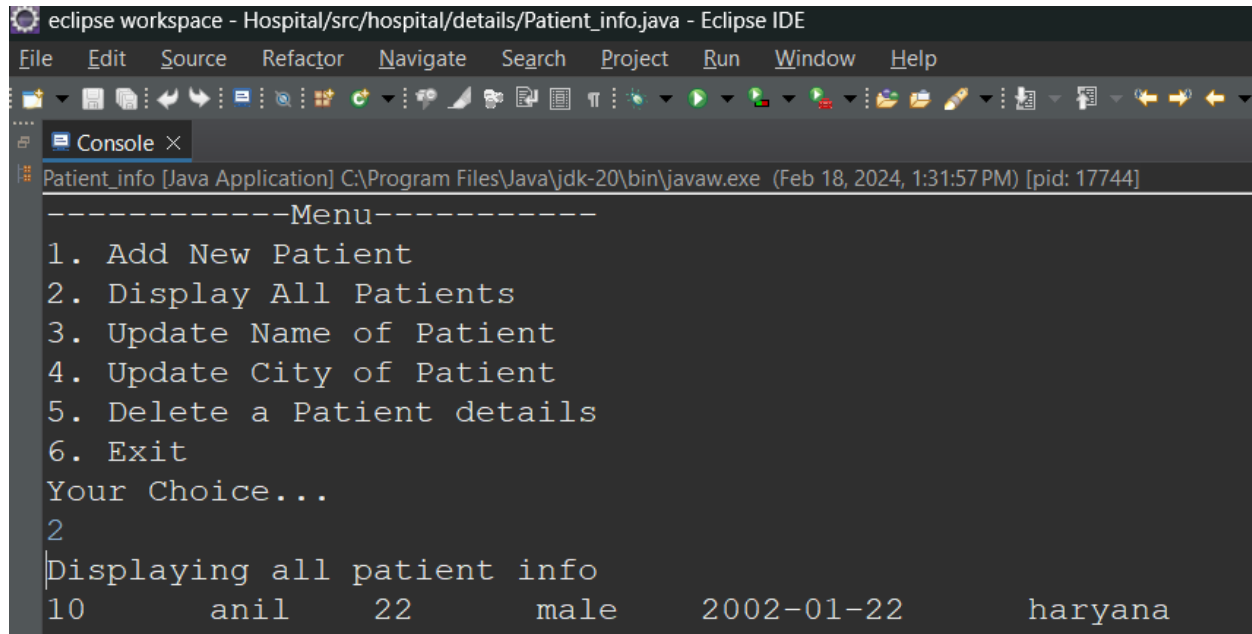
```
-----Menu-----
1. Add New Patient
2. Display All Patients
3. Update Name of Patient
4. Update City of Patient
5. Delete a Patient details
6. Exit
Your Choice...
2
```

The output then shows 'Displaying all patient info' followed by a table of patient details:

8	Nishant	23	male	2001-08-21	Patna
10	anil	22	male	2002-01-22	haryana
11	Mahesh	24	Male	2000-05-10	Delhi
12	shruti	34	female	1990-02-23	punjab
13	Priya	24	female	2000-06-06	Patna

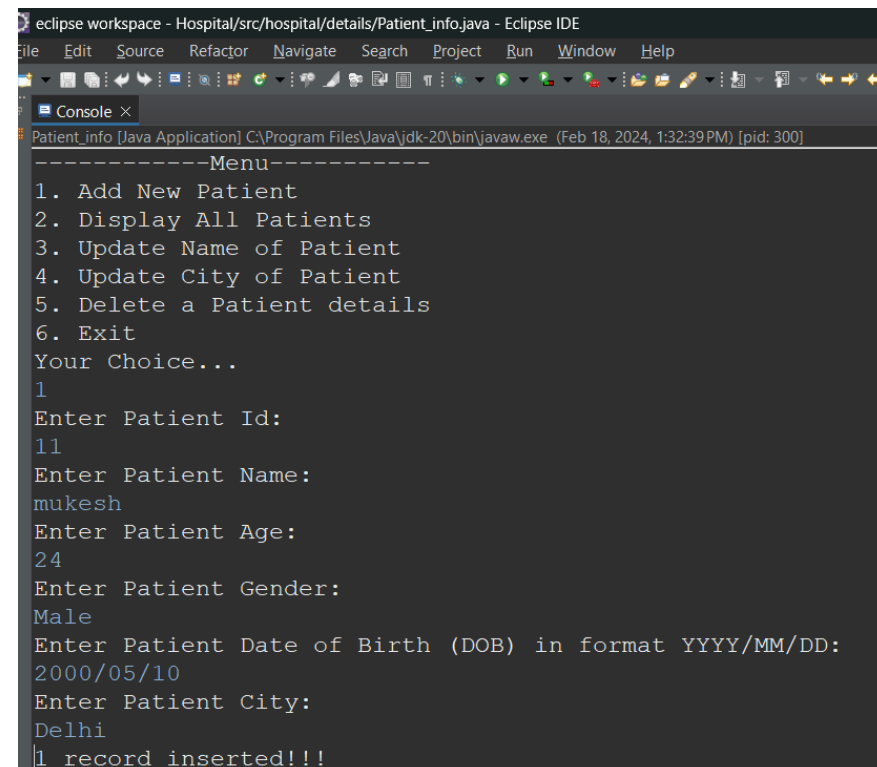
## Insert operation

Before insertion:



```
eclipse workspace - Hospital/src/hospital/details/Patient_info.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Patient_info [Java Application] C:\Program Files\Java\jdk-20\bin\javaw.exe (Feb 18, 2024, 1:31:57 PM) [pid: 17744]
-----Menu-----
1. Add New Patient
2. Display All Patients
3. Update Name of Patient
4. Update City of Patient
5. Delete a Patient details
6. Exit
Your Choice...
2
Displaying all patient info
10      anil      22      male      2002-01-22      haryana
```

After insertion:



```
eclipse workspace - Hospital/src/hospital/details/Patient_info.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Patient_info [Java Application] C:\Program Files\Java\jdk-20\bin\javaw.exe (Feb 18, 2024, 1:32:39 PM) [pid: 300]
-----Menu-----
1. Add New Patient
2. Display All Patients
3. Update Name of Patient
4. Update City of Patient
5. Delete a Patient details
6. Exit
Your Choice...
1
Enter Patient Id:
11
Enter Patient Name:
mukesh
Enter Patient Age:
24
Enter Patient Gender:
Male
Enter Patient Date of Birth (DOB) in format YYYY/MM/DD:
2000/05/10
Enter Patient City:
Delhi
1 record inserted!!!
```

```
eclipse workspace - Hospital/src/hospital/details/Patient_info.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Console X
Patient_info [Java Application] C:\Program Files\Java\jdk-20\bin\javaw.exe (Feb 18, 2024, 1:38:52 PM) [pid: 17448]
-----Menu-----
1. Add New Patient
2. Display All Patients
3. Update Name of Patient
4. Update City of Patient
5. Delete a Patient details
6. Exit
Your Choice...
2
Displaying all patient info
8      Nishant 23      male      2001-08-21      Patna
9      Ashutosh      20      male      2004-04-18      bihar
10     anil      22      male      2002-01-22      haryana
11     mukesh 24      Male      2000-05-10      Delhi
12     shristi 34      female    1990-02-23      punjab
13     Priya   24      female    2000-06-06      bihar
```

## Delete operation

Before Deletion:

```
eclipse workspace - Hospital/src/hospital/details/Patient_info.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Console X
Patient_info [Java Application] C:\Program Files\Java\jdk-20\bin\javaw.exe (Feb 18, 2024, 1:38:52 PM) [pid: 17448]
-----Menu-----
1. Add New Patient
2. Display All Patients
3. Update Name of Patient
4. Update City of Patient
5. Delete a Patient details
6. Exit
Your Choice...
2
Displaying all patient info
8      Nishant 23      male      2001-08-21      Patna
9      Ashutosh      20      male      2004-04-18      bihar
10     anil      22      male      2002-01-22      haryana
11     mukesh 24      Male      2000-05-10      Delhi
12     shristi 34      female    1990-02-23      punjab
13     Priya   24      female    2000-06-06      bihar
```

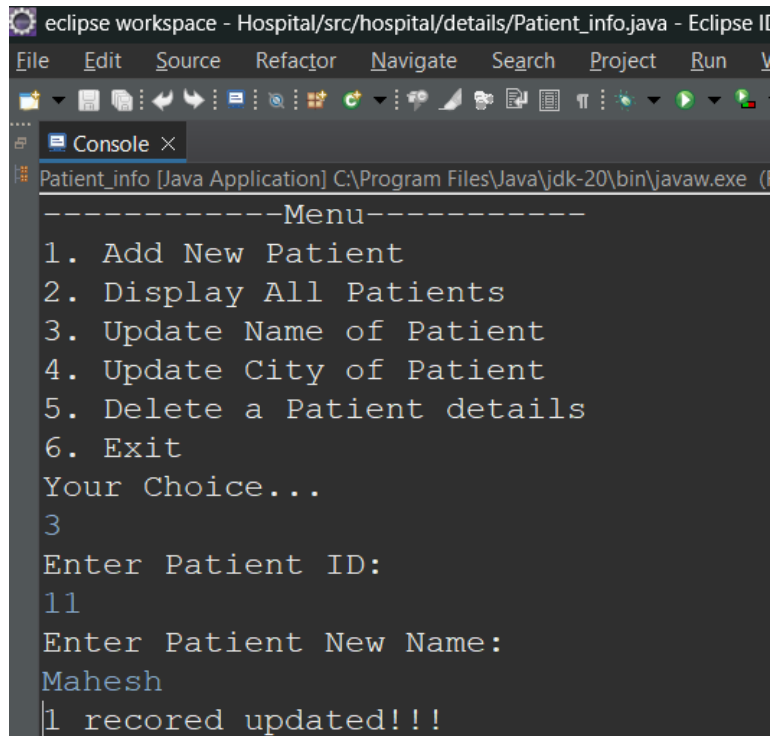
After Deletion:

```
eclipse workspace - Hospital/src/hospital/details/Patient_info.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Console ×
Patient_info [Java Application] C:\Program Files\Java\jdk-20\bin\javaw.exe (Feb 18, 2024, 1:43:51 PM) [pid: 2076]
-----Menu-----
1. Add New Patient
2. Display All Patients
3. Update Name of Patient
4. Update City of Patient
5. Delete a Patient details
6. Exit
Your Choice...
5
Enter Patient ID:
9
1 recored deleted!!!
-----Menu-----
1. Add New Patient
2. Display All Patients
3. Update Name of Patient
4. Update City of Patient
5. Delete a Patient details
6. Exit
Your Choice...
2
Displaying all patient info
8      Nishant 23      male      2001-08-21      Patna
10     anil    22      male      2002-01-22      haryana
11     Mahesh 24      Male      2000-05-10      Delhi
12     shristi 34      female    1990-02-23      punjab
13     Priya  24      female    2000-06-06      Patna
```

## Update operation

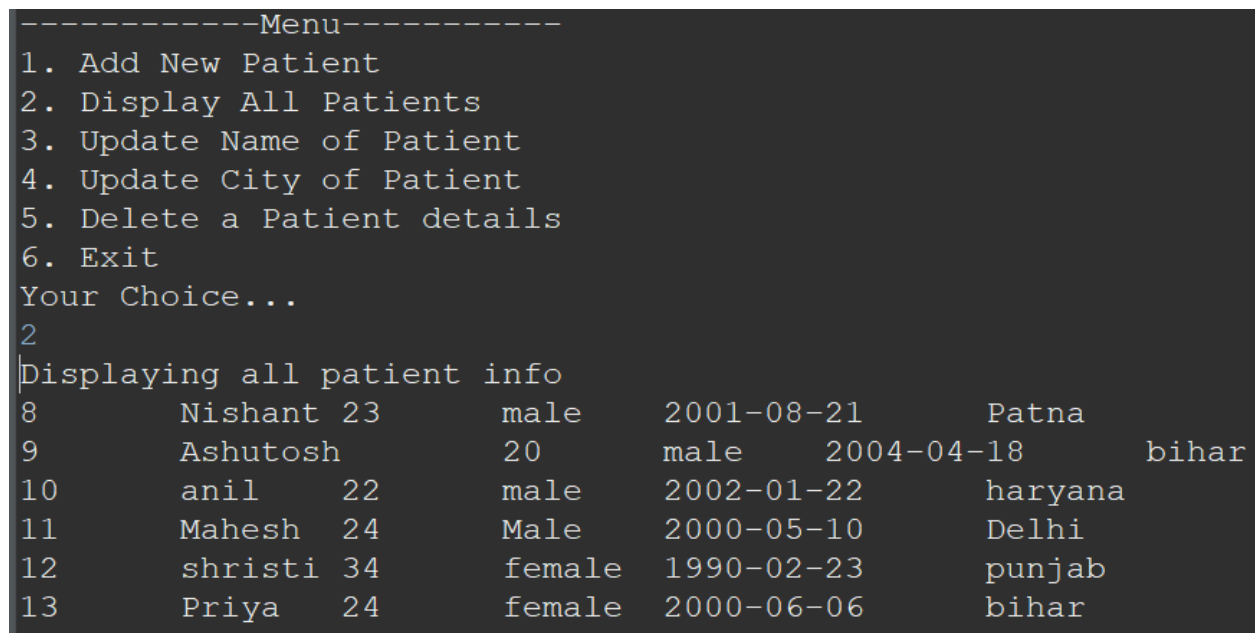
### Update name of patient

Before updation:



```
eclipse workspace - Hospital/src/hospital/details/Patient_info.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run View
-----Menu-----
1. Add New Patient
2. Display All Patients
3. Update Name of Patient
4. Update City of Patient
5. Delete a Patient details
6. Exit
Your Choice...
3
Enter Patient ID:
11
Enter Patient New Name:
Mahesh
1 recored updated!!!
```

After updation:



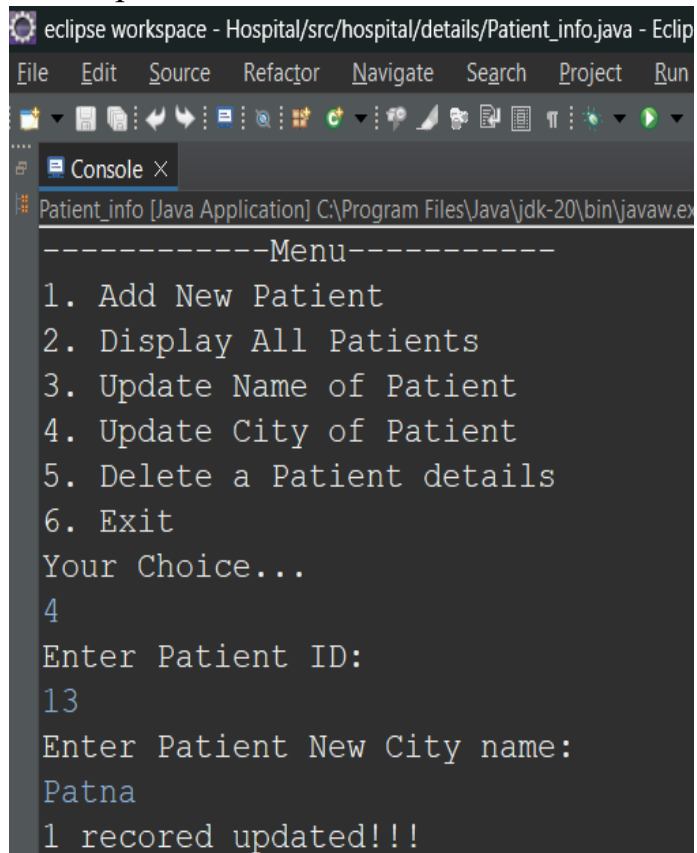
```
-----Menu-----
1. Add New Patient
2. Display All Patients
3. Update Name of Patient
4. Update City of Patient
5. Delete a Patient details
6. Exit
Your Choice...
2
Displaying all patient info
8      Nishant 23      male      2001-08-21      Patna      bihar
9      Ashutosh 20      male      2004-04-18      bihar
10     anil     22      male      2002-01-22      haryana
11     Mahesh  24      Male      2000-05-10      Delhi
12     shrusti 34      female    1990-02-23      punjab
13     Priya   24      female    2000-06-06      bihar
```

## Update city of Patient

Before updation:

```
-----Menu-----
1. Add New Patient
2. Display All Patients
3. Update Name of Patient
4. Update City of Patient
5. Delete a Patient details
6. Exit
Your Choice...
2
Displaying all patient info
8      Nishant 23      male    2001-08-21      Patna
9      Ashutosh      20      male    2004-04-18      bihar
10     anil      22      male    2002-01-22      haryana
11     Mahesh  24      Male    2000-05-10      Delhi
12     shristi 34      female  1990-02-23      punjab
13     Priya   24      female  2000-06-06      bihar
```

After updation:



The screenshot shows the Eclipse IDE interface with the console window open. The console displays the application's output after the city of patient 13 has been updated. The menu is shown, and option 4 (Update City of Patient) is selected. The user enters patient ID 13 and the new city name Patna. The application confirms that 1 record has been updated.

```
eclipse workspace - Hospital/src/hospital/details/Patient_info.java - Eclipse
File Edit Source Refactor Navigate Search Project Run
-----Menu-----
1. Add New Patient
2. Display All Patients
3. Update Name of Patient
4. Update City of Patient
5. Delete a Patient details
6. Exit
Your Choice...
4
Enter Patient ID:
13
Enter Patient New City name:
Patna
1 record updated!!!
```



-----Menu-----

1. Add New Patient
2. Display All Patients
3. Update Name of Patient
4. Update City of Patient
5. Delete a Patient details
6. Exit

Your Choice...

2

Displaying all patient info

8	Nishant	23	male	2001-08-21	Patna
10	anil	22	male	2002-01-22	haryana
11	Mahesh	24	Male	2000-05-10	Delhi
12	shristi	34	female	1990-02-23	punjab
13	Priya	24	female	2000-06-06	Patna