LAB 1

Keshav Kant Mishra

A45304821002

BCA VI ‘A’

Submitted To: Dr. Naveen Kumar Singh

CRUD OPERATIONS

Problem Description

The goal of this project is to implement a basic console-based application that performs Create, Read, Update, and Delete (CRUD) operations on a database, specifically focusing on managing employee records. The application will provide an interactive interface for users to manipulate the employee data stored in a relational database management system (RDBMS) such as MySQL, PostgreSQL, or SQLite.

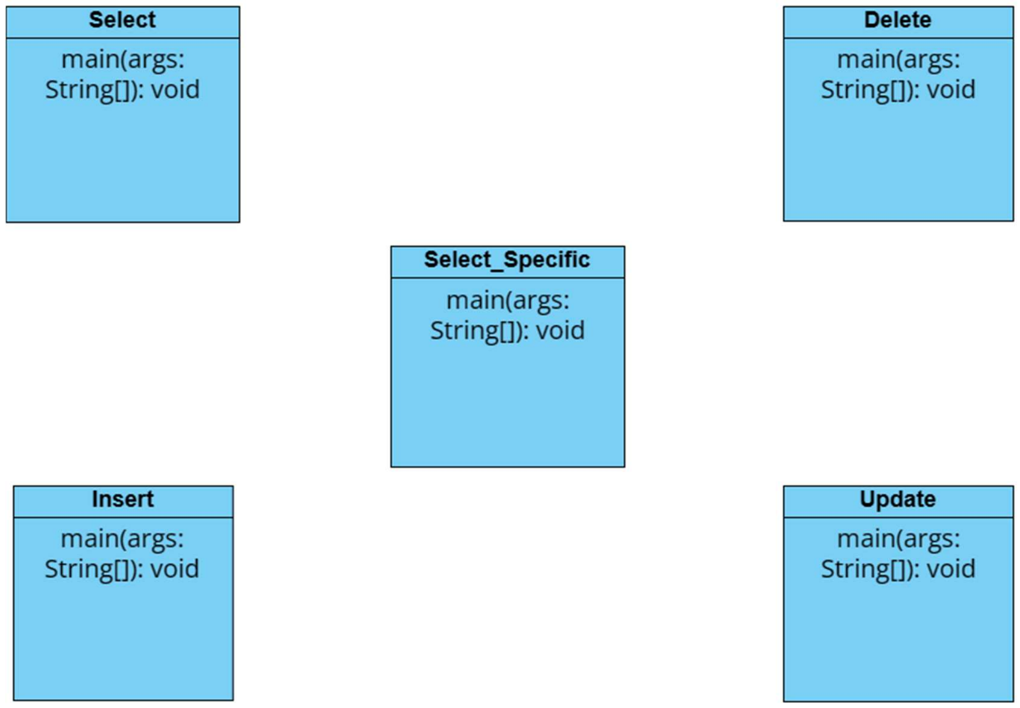
Functional Requirements:

1. **Create (Insertion):**
    - The application must allow users to add new employee records to the database. Each employee record consists of the following fields: ID, First Name, Last Name, Age, and Date of Birth (DOB). The application should prompt the user to enter these details.
2. **Read (Selection):**
    - The application should be capable of displaying employee records. It must support two types of read operations:
        - **View All:** Display all employee records stored in the database.
        - **View Specific:** Prompt the user to enter an employee ID and display the corresponding employee's details if found.
3. **Update:**
    - The application must enable users to update existing employee records. Users should be able to specify an employee ID and then choose which details to update (e.g., name, age, and/or DOB). The application should then prompt the user for the new values.
4. **Delete:**
    - Users should be able to delete an employee record from the database by specifying the employee's ID. The application should confirm the deletion.

Design Description

The design of the CRUD operations application for managing Employee records encompasses several critical components, including system architecture, database design, class design, user interaction flow, and technical considerations. This structured approach ensures a well-organized and maintainable application.

Class Design

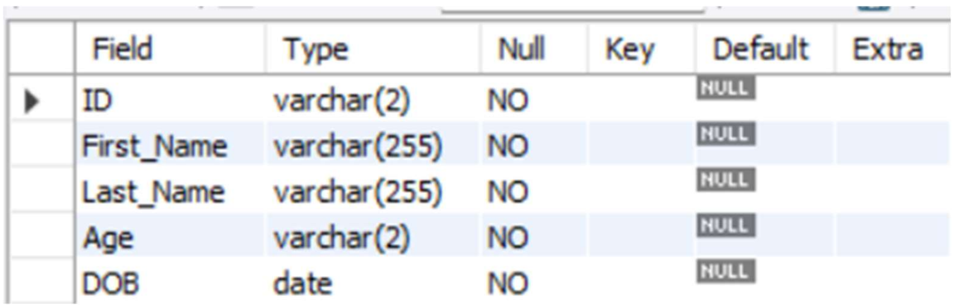

Select
main(args: String[]): void
Delete
main(args: String[]): void
Select_Specific
main(args: String[]): void
Insert
main(args: String[]): void
Update
main(args: String[]): void

1. **Delete**
    - **Attributes: None (uses local variables and system resources)**
    - **Methods:**
        - **+ main(args: String[]): void**
2. **Select**
    - **Attributes: None (uses local variables and system resources)**
    - **Methods:**
        - **+ main(args: String[]): void**
3. **Select_Specific**
    - **Attributes: None (uses local variables and system resources)**
    - **Methods:**
        - **+ main(args: String[]): void**
4. **Insert**
    - **Attributes: None (uses local variables and system resources)**
    - **Methods:**
        - **+ main(args: String[]): void**
5. **Update**
    - **Attributes: None (uses local variables and system resources)**
    - **Methods:**
        - **+ main(args: String[]): void**

Database Design

A crucial part of the application is its interaction with a database to store, retrieve, update, and delete student records. The database schema is designed as follows:

Table Schema: Emp1



	Field	Type	Null	Key	Default	Extra
▶	ID	varchar(2)	NO		NULL	
	First_Name	varchar(255)	NO		NULL	
	Last_Name	varchar(255)	NO		NULL	
	Age	varchar(2)	NO		NULL	
	DOB	date	NO		NULL	

1. **ID**
   - **Type:** VARCHAR(2)
   - **Constraints:** Primary Key
   - **Description:** Acts as the unique identifier for each employee. The choice of VARCHAR(2) suggests an intention to use alphanumeric IDs, though the length of 2 characters may be limiting for larger organizations. It's essential for uniquely identifying records in the Employee table.

2. **First_Name**
   - **Type:** VARCHAR(255)
   - **Constraints:** Not Null
   - **Description:** Represents the first name of the employee. The VARCHAR type with a length of 255 characters is used to accommodate first names of varying lengths and complexities, catering to a wide range of names.

3. **Last_Name**
   - **Type:** VARCHAR(255)
   - **Constraints:** Not Null
   - **Description:** Stores the last name or family name of the employee. Similar to **First_Name**, using VARCHAR(255) allows for a broad spectrum of last names to be stored, including those with hyphens, spaces, or apostrophes.

4. **Age**
    - **Type:** VARCHAR(2)
    - **Constraints:** Not Null
    - **Description:** Contains the age of the employee.

5. **DOB (Date of Birth)**
    - **Type:** DATE
    - **Constraints:** Not Null
    - **Description:** Stores the student's date of birth. The DATE type ensures that the data is stored in a standardized format (YYYY-MM-DD), facilitating easy sorting, querying, and age calculation based on this field.

Code

1. Insert Class

```
import java.sql.*;
import java.util.Scanner;
public class Insert
{
	public static void main(String args[]) throws SQLException,
ClassNotFoundException
	{
		Scanner sc = new Scanner(System.in);
		Class.forName("com.mysql.cj.jdbc.Driver");
		System.out.print("Enter The Details of the Employee\nID = ");
		String id = sc.next();
		System.out.print("First Name = ");
		String fname = sc.next();
		System.out.print("Last Name = ");
		String lname = sc.next();
		System.out.print("Age = ");
		String age = sc.next();
		System.out.print("DOB (YYYY-MM-DD) = ");
		String date = sc.next();


		Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/keshav","root","gautam74
88");

		String query = "Insert into Emp1 values (? , ? , ? , ? ,?)";
		PreparedStatement pst = con.prepareStatement(query);
		pst.setString(1, id);
		pst.setString(2, fname);
		pst.setString(3, lname);
		pst.setString(4, age);
		pst.setString(5, date);

		int affected_rows = pst.executeUpdate();

		if (affected_rows > 0)
		{
			System.out.println("Data Inserted");
		}
		sc.close();
		con.close();
	}
}
```

2. Select Class

```
import java.sql.*;

public class Select
{
	public static void main(String args[]) throws SQLException,
ClassNotFoundException
	{
		Class.forName("com.mysql.cj.jdbc.Driver");

		Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/keshav","root","gautam74
88");

		Statement st = con.createStatement();

		ResultSet rs = st.executeQuery("select * from Emp1");
		System.out.println("Roll\tFirst Name\tLast Name\tAge\t\tDOB");
		while (rs.next())
		{

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

		con.close();
	}
}
```

3. <u>Select Specific Class</u>

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;

public class Select_Specific
{
	public static void main(String[] args) throws SQLException ,
ClassNotFoundException
	{
		Scanner sc = new Scanner(System.in);
		Class.forName("com.mysql.cj.jdbc.Driver");
		System.out.println("Enter The ID of the Employee");
		String roll = sc.next();
		Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/keshav","root","gautam74
88");
		String query = "select * from Emp1 where id = ?";
		PreparedStatement pst = con.prepareStatement(query);
		pst.setString(1, roll);

		ResultSet rs = pst.executeQuery();
		if (rs.next())
		{
			System.out.println("Roll\tFirst Name\tLast Name\tAge\t\tDOB");

	System.out.println(rs.getString(1)+"\t"+rs.getString(2)+"\t\t"+rs.getString
(3)+"\t"+rs.getString(4)+"\t\t"+rs.getDate(5));
		}
		else
		{
			System.out.println("No Data with this ID");
		}
		sc.close();
		con.close();
	}
}
```

4. Update Class

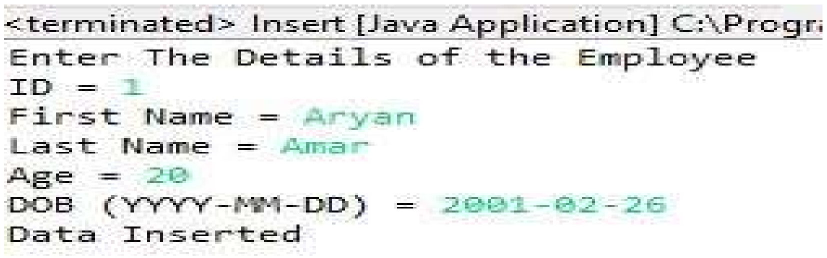
```
import java.sql.*;
import java.util.Scanner;

public class Update
{
	public static void main(String args[]) throws SQLException
,ClassNotFoundException
	{
		Scanner sc = new Scanner(System.in);

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

		Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/keshav","root","gautam74
88");

		System.out.println("Enter The ID of the Employee Whose Data needs to
be Updated");
		String id = sc.next();

		String query = "select * from Emp1 where id = ?";
		PreparedStatement pst1 = con.prepareStatement(query);
		pst1.setString(1, id);
		ResultSet rs = pst1.executeQuery();
		if (rs.next())
		{
			System.out.print("Enter First Name : ");
			String fname = sc.next();
			System.out.print("Enter Last Name : ");
			String lname = sc.next();
			System.out.print("Enter Age : ");
			String age = sc.next();
			System.out.print("Date Of Birth (YYYY-MM-DD) : ");
			String Date = sc.next();
			String querty = "Update Employee set First_Name = ? , Last_Name
= ? , Age = ? , Dob = ? where ID = ?";
			PreparedStatement pst = con.prepareStatement(querty);
			pst.setString(1, fname);
			pst.setString(2, lname);
			pst.setString(3, age);
			pst.setString(4, Date);
			pst.setString(5, id);

			int affected_rows = pst.executeUpdate();

			if (affected_rows > 0)
			{
				System.out.println("Data Updated");
			}
		}
		else
		{
			System.out.println("No Data with this ID");
		}
		sc.close();
		con.close();
	}
}
```

5. Delete Class

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

public class Delete
{
	public static void main(String args[]) throws SQLException, 
ClassNotFoundException
	{
		Scanner sc = new Scanner(System.in);
		Class.forName("com.mysql.cj.jdbc.Driver");

		Connection con = 
DriverManager.getConnection("jdbc:mysql://localhost:3306/keshav","root","gautam74
88");

		System.out.println("Enter The ID to be deleted");
		String r = sc.next();

		String query = "Delete from Emp1 where id = ?";
		PreparedStatement pst =  con.prepareStatement(query);
		pst.setString(1, r);
		int affected_rows = pst.executeUpdate();

		if (affected_rows > 0)
		{
			System.out.println("Data Deleted");
		}
		else
		{
			System.out.println("No Data To Be Deleted");
		}
		sc.close();
		con.close();
	}
}
```

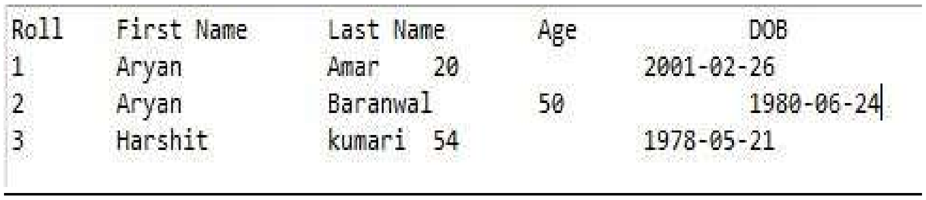
Input/Output

Insert Class



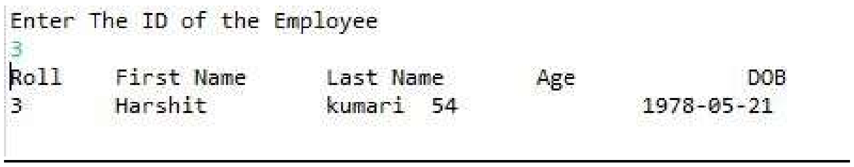
```
<terminated> Insert [Java Application] C:\Progr
Enter The Details of the Employee
ID = 1
First Name = Aryan
Last Name = Amar
Age = 20
DOB (YYYY-MM-DD) = 2001-02-26
Data Inserted
```

Select Class



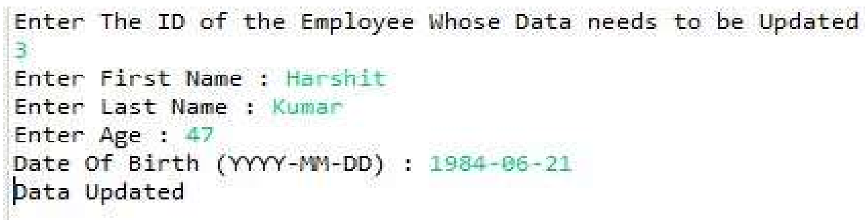
```
Roll    First Name      Last Name       Age             DOB
1       Aryan           Amar    20              2001-02-26
2       Aryan           Baranwal        50              1980-06-24
3       Harshit         kumari  54              1978-05-21
```

Select Specific Class



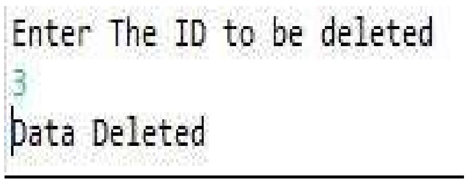
```
Enter The ID of the Employee
3
Roll    First Name      Last Name       Age             DOB
3       Harshit         kumari  54              1978-05-21
```

Update Class



```
Enter The ID of the Employee Whose Data needs to be Updated
3
Enter First Name : Harshit
Enter Last Name : Kumar
Enter Age : 47
Date Of Birth (YYYY-MM-DD) : 1984-06-21
Data Updated
```

Delete Class



```
Enter The ID to be deleted
3
Data Deleted
```