National University of Computer & Emerging Sciences, Karachi

Computer Science Department

Fall 2024, Lab Manual - 08

Course Code: SL3001	**Course: Software Development and construction**
Instructor:	**Yasir Arfat**

Lab # 08

Objective

To understand the fundamentals of JDBC (Java Database Connectivity) and how it enables Java applications to interact with various databases. This session will cover the architecture, key components, and basic operations.

What is JDBC?

- JDBC is an API (Application Programming Interface) that allows Java applications to connect and interact with databases.
- It provides methods for querying and updating data in a database, allowing developers to write portable code that can work with different database systems.

Prerequisites

Oracle SQL Developer: We will use Oracle SQL Developer to manage our Oracle database.

IntelliJ IDE: Our Java development will be conducted using IntelliJ.

JDBC Driver: Ensure you have the ojdbc11.jar driver included in your IntelliJ project for Oracle database connectivity. You can download from https://www.oracle.com/database/technologies/appdev/jdbc-downloads.html

JDBC Architecture

JDBC consists of two main layers:

1. **JDBC API**: Provides the application-to-JDBC Manager connection.
2. **JDBC Driver API**: Enables the JDBC Manager to connect to the database.

JDBC Driver Types:

- **Type 1: JDBC-ODBC Bridge Driver**: Uses ODBC drivers to connect to the database.
- **Type 2: Native-API Driver**: Uses native libraries of the database.
- **Type 3: Network Protocol Driver**: Converts JDBC calls into the database-specific protocol.
- **Type 4: Thin Driver**: Pure Java driver that communicates directly with the database.

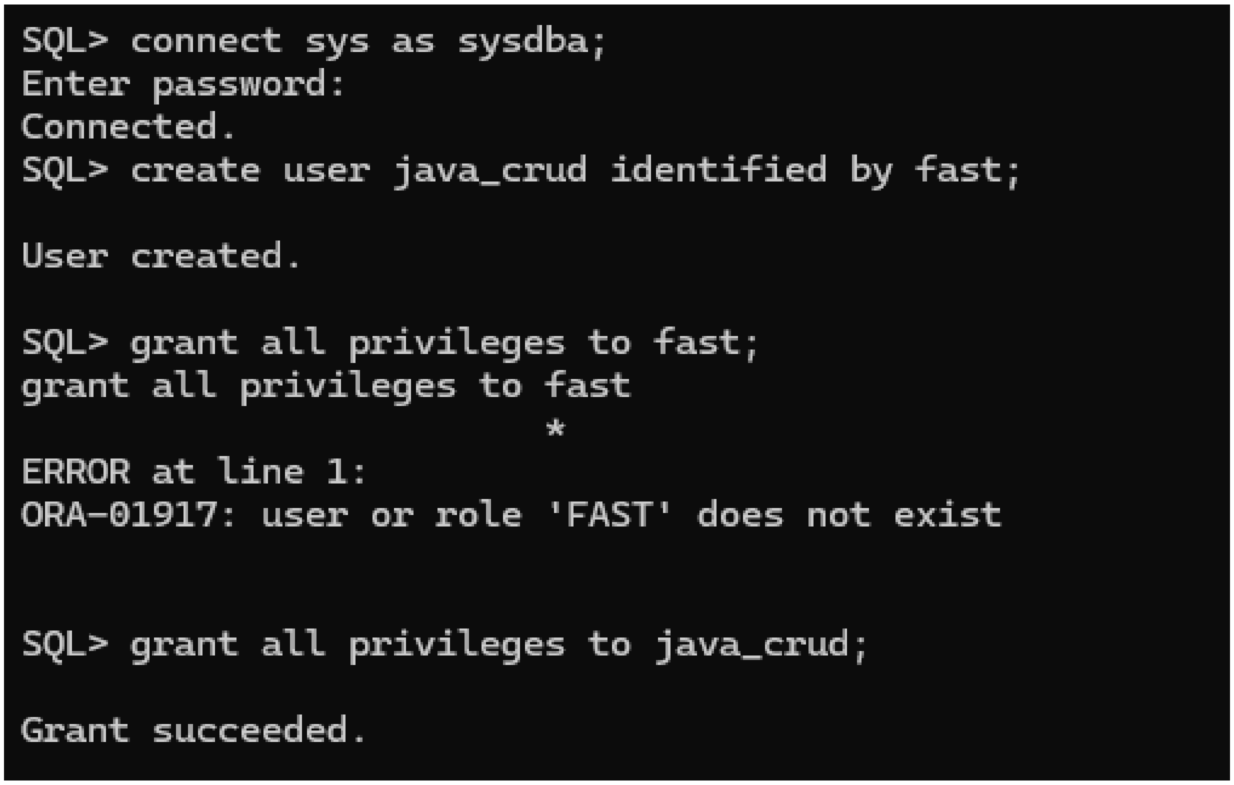
Key Components of JDBC

- **DriverManager**: Manages the list of database drivers.
- **Connection**: Represents a session with a specific database.
- **Statement**: Used to execute SQL queries.
- **ResultSet**: Represents the result set of a query.
- **SQLException**: Handles database access errors.

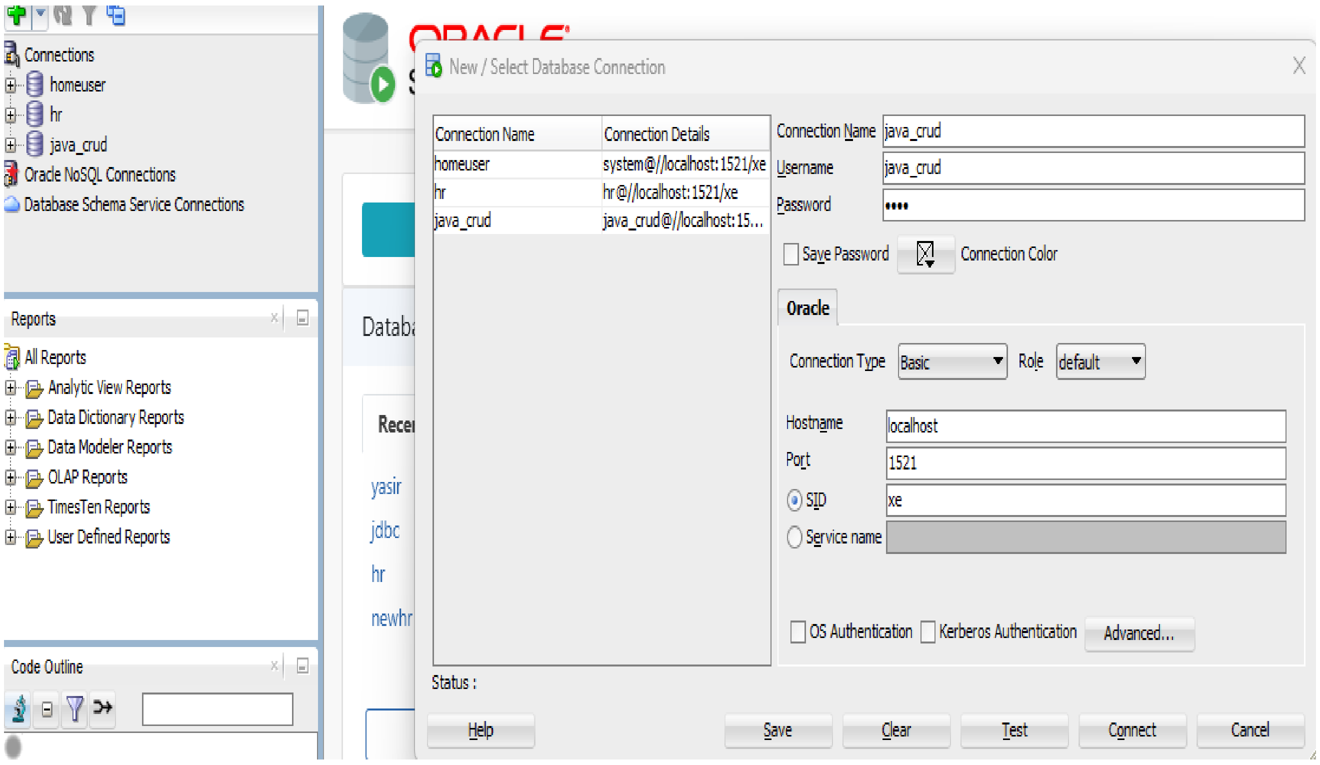
JDBC Project with CMD

- ➢ First, we will create new user in our Oracle SQL Developer
- ➢ Open Run SQL Command Line and do the following to create new user

- Now Open SQL Developer and Create new Connection



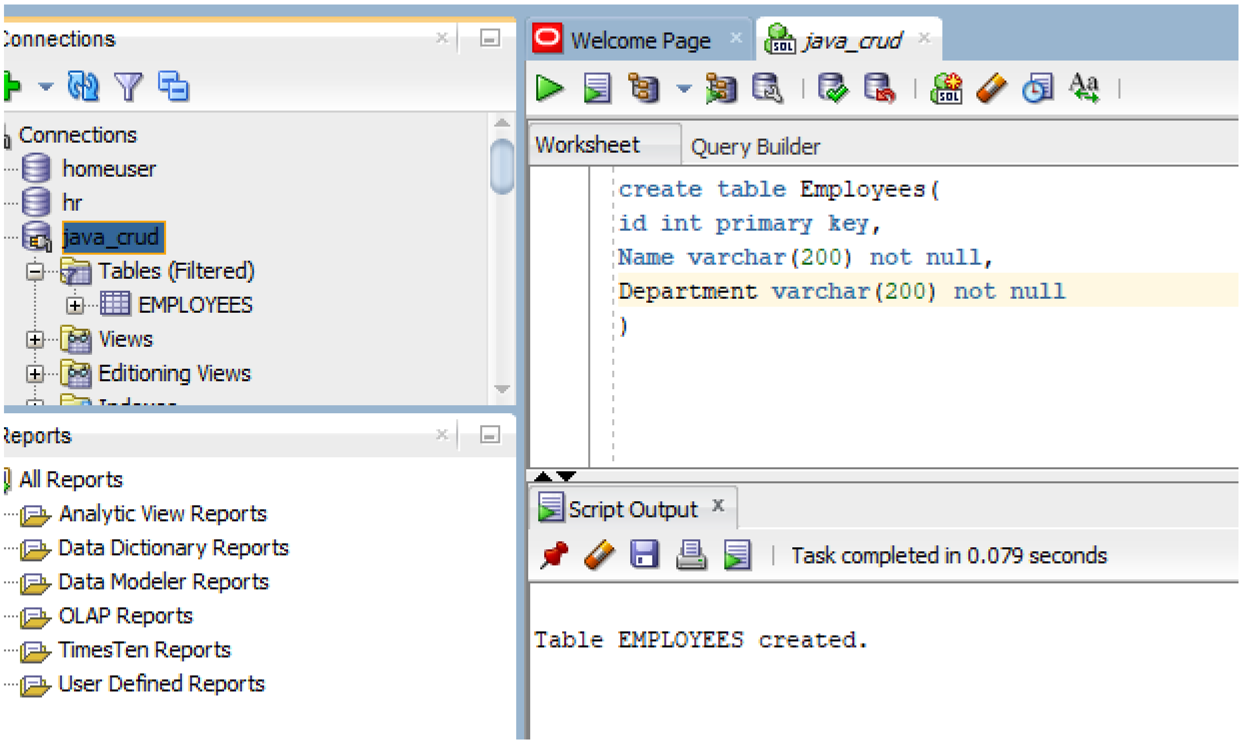
```
SQL> connect sys as sysdba;
Enter password:
Connected.
SQL> create user java_crud identified by fast;

User created.

SQL> grant all privileges to fast;
grant all privileges to fast
                        *
ERROR at line 1:
ORA-01917: user or role 'FAST' does not exist


SQL> grant all privileges to java_crud;

Grant succeeded.
```



Connection Name	Connection Details
homeuser	system@//localhost:1521/xe
hr	hr@//localhost:1521/xe
java_crud	java_crud@//localhost:15...

Connection Name: java_crud
Username: java_crud
Password: ••••
Save Password
Connection Color
Oracle
Connection Type: Basic
Role: default
Hostname: localhost
Port: 1521
SID: xe
Service name:
OS Authentication
Kerberos Authentication
Status :

- ➢ Now Create a Table Employees in the connection you created

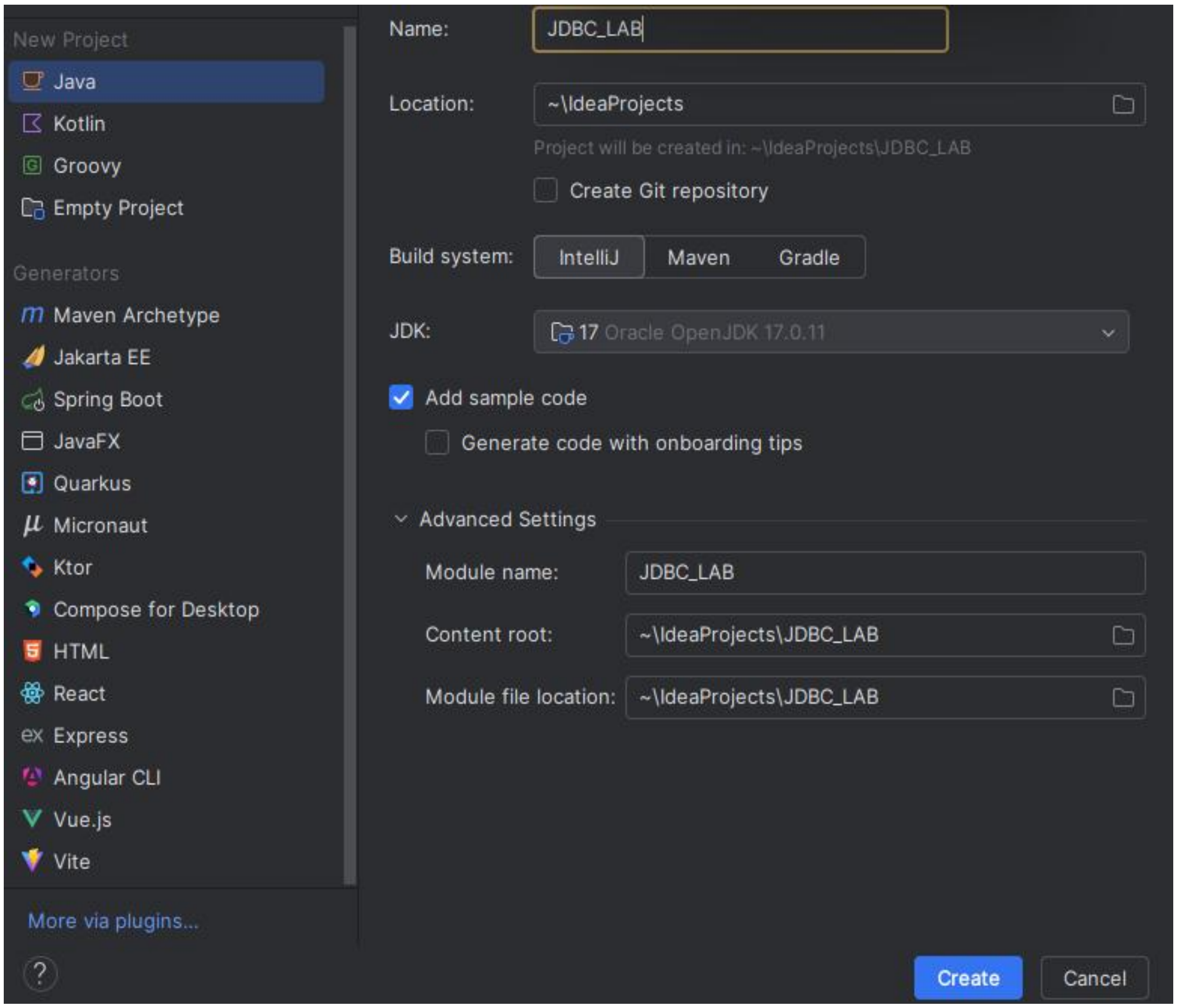


```
create table Employees(
id int primary key,
Name varchar(200) not null,
Department varchar(200) not null
)
```

Task completed in 0.079 seconds

```
Table EMPLOYEES created.
```

- Now Create a New Project in **IntelliJ** ide and Name it as JDBC_LAB



Name: JDBC_LAB

Location: ~\IdeaProjects

Project will be created in: ~\IdeaProjects\JDBC_LAB

☐ Create Git repository

Build system: [✓ IntelliJ] Maven Gradle

JDK: 17 Oracle OpenJDK 17.0.11

✓ Add sample code

☐ Generate code with onboarding tips

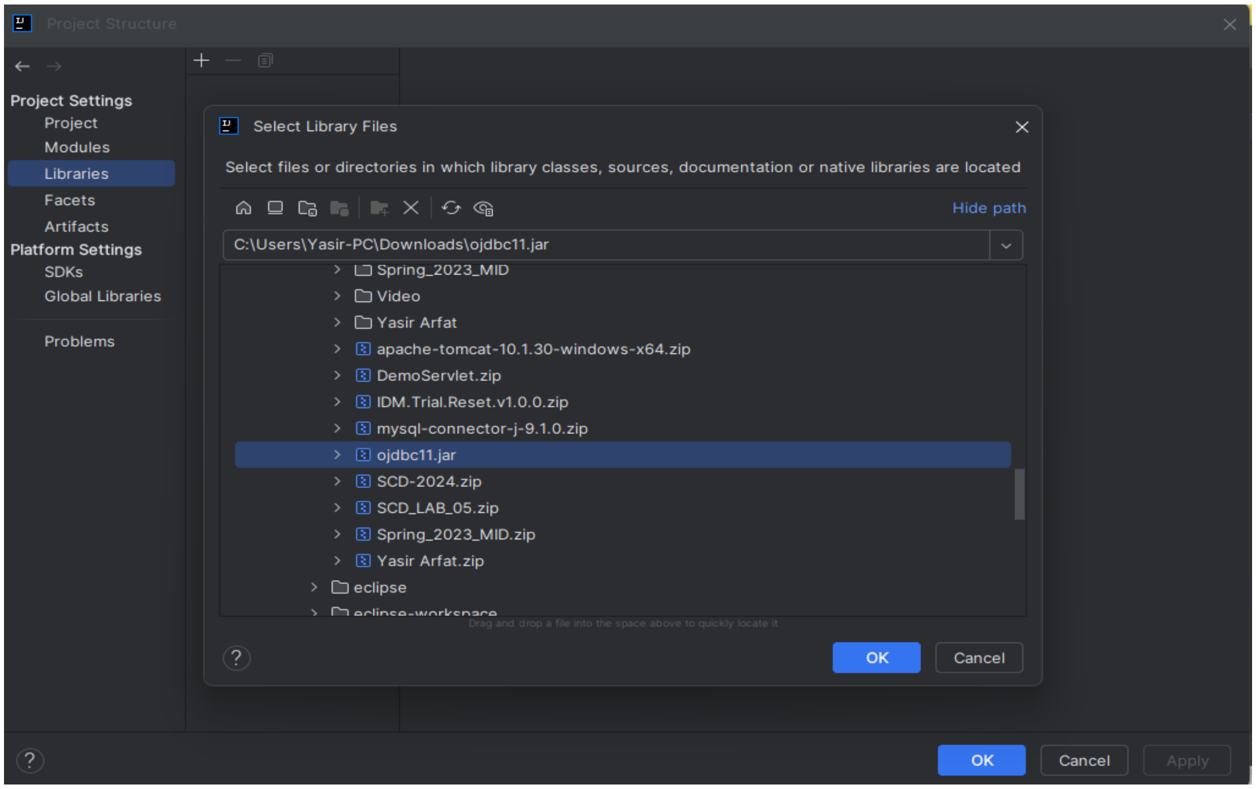
Advanced Settings

Module name: JDBC_LAB

Content root: ~\IdeaProjects\JDBC_LAB

Module file location: ~\IdeaProjects\JDBC_LAB

- Now we will add the ojdbc11 driver to our project
- For the click on File -> project structure -> libraries the click the + icon there and select java after that browser to your ojdbc11 select path and click apply and ok



Select files or directories in which library classes, sources, documentation or native libraries are located

C:\Users\Yasir-PC\Downloads\ojdbc11.jar

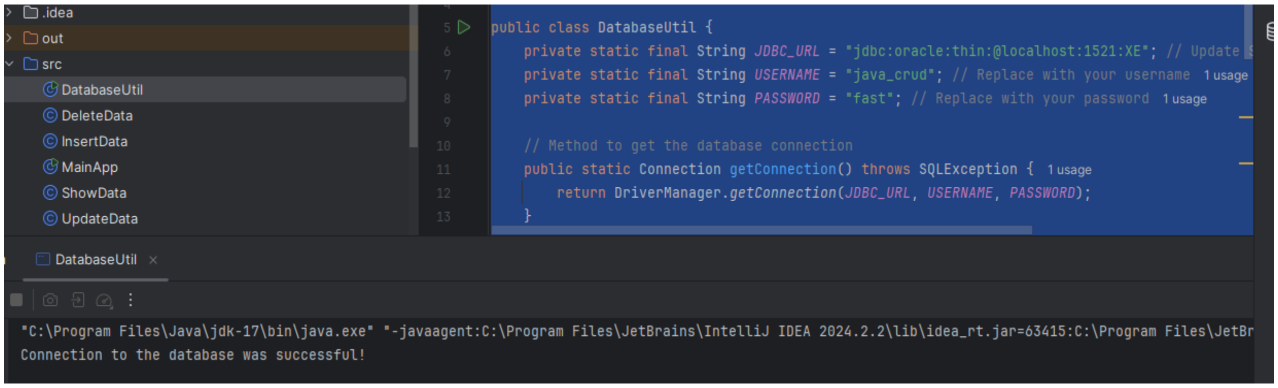
```
Spring_2023_MID
Video
Yasir Arfat
apache-tomcat-10.1.30-windows-x64.zip
DemoServlet.zip
IDM.Trial.Reset.v1.0.0.zip
mysql-connector-j-9.1.0.zip
ojdbc11.jar
SCD-2024.zip
SCD_LAB_05.zip
Spring_2023_MID.zip
Yasir Arfat.zip
eclipse
eclipse-workspace
```

➢ After this you will see we will add the DatabaseUtil class our project for connection with oracle database

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseUtil {
    private static final String JDBC_URL =
"jdbc:oracle:thin:@localhost:1521:XE"; // Update SID if necessary
    private static final String USERNAME = "java_crud"; // Replace with your
username
    private static final String PASSWORD = "fast"; // Replace with your
password

    // Method to get the database connection
    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(JDBC_URL, USERNAME, PASSWORD);
    }

    public static void main(String[] args) {
        Connection connection = null;
        try {
            connection = getConnection();
            if (connection != null) {
                System.out.println("Connection to the database was
successful!");
            }
        } catch (SQLException e) {
            System.out.println("Failed to connect to the database.");
```

```java
            e.printStackTrace();
        } finally {
            // Close the connection if it was established
            if (connection != null) {
                try {
                    connection.close();
                } catch (SQLException e) {
                    System.out.println("Failed to close the connection.");
                    e.printStackTrace();
                }
            }
        }
    }
}
```

➢ Run this to verify you are successfully connected to database



```java
public class DatabaseUtil {
    private static final String JDBC_URL = "jdbc:oracle:thin:@localhost:1521:XE"; // Update
    private static final String USERNAME = "java_crud"; // Replace with your username  1 usage
    private static final String PASSWORD = "fast"; // Replace with your password  1 usage

    // Method to get the database connection
    public static Connection getConnection() throws SQLException {  1 usage
        return DriverManager.getConnection(JDBC_URL, USERNAME, PASSWORD);
    }
```

```
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\lib\idea_rt.jar=63415:C:\Program Files\JetBr
Connection to the database was successful!
```

➢ Now add insertData.java code

```java
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class InsertData {
    public void insertEmployee(int u_id, String u_name, String u_department)
{



        String sql = "INSERT INTO employees (id, name, department) VALUES (?,
?, ?)";

        try (Connection connection = DatabaseUtil.getConnection();
             PreparedStatement preparedStatement =
connection.prepareStatement(sql)) {
```

```java
            preparedStatement.setInt(1, u_id);
            preparedStatement.setString(2, u_name);
            preparedStatement.setString(3, u_department);

            int rowsAffected = preparedStatement.executeUpdate();
            System.out.println(rowsAffected + " row(s) inserted.");
        } catch (SQLException e) {
            System.out.println("Connection or insert failed!");
            e.printStackTrace();
        }
    }
}
```

➢ Now lets add showData.java Code

```java
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class ShowData {
    public void displayEmployees() {
        try (Connection connection = DatabaseUtil.getConnection();
             Statement statement = connection.createStatement()) {

            String sql = "SELECT * FROM employees";
            ResultSet resultSet = statement.executeQuery(sql);

            System.out.println("ID\tName\tDepartment");
            System.out.println("---------------------------");
            while (resultSet.next()) {
                int id = resultSet.getInt("id");
                String name = resultSet.getString("name");
                String department = resultSet.getString("department");
                System.out.println(id + "\t" + name + "\t" + department);
            }
        } catch (SQLException e) {
            System.out.println("Connection or query failed!");
            e.printStackTrace();
        }
    }
}
```

➢ Now add UpdateData.java Code

```java
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
```

```java
public class UpdateData {
    public void updateEmployee(int id, String newName, String newDepartment)
{
        String sql = "UPDATE employees SET name = ?, department = ? WHERE id
= ?";

        try (Connection connection = DatabaseUtil.getConnection();
             PreparedStatement preparedStatement =
connection.prepareStatement(sql)) {

            preparedStatement.setString(1, newName);
            preparedStatement.setString(2, newDepartment);
            preparedStatement.setInt(3, id);

            int rowsAffected = preparedStatement.executeUpdate();
            System.out.println(rowsAffected + " row(s) updated.");
        } catch (SQLException e) {
            System.out.println("Connection or update failed!");
            e.printStackTrace();
        }
    }
}
```

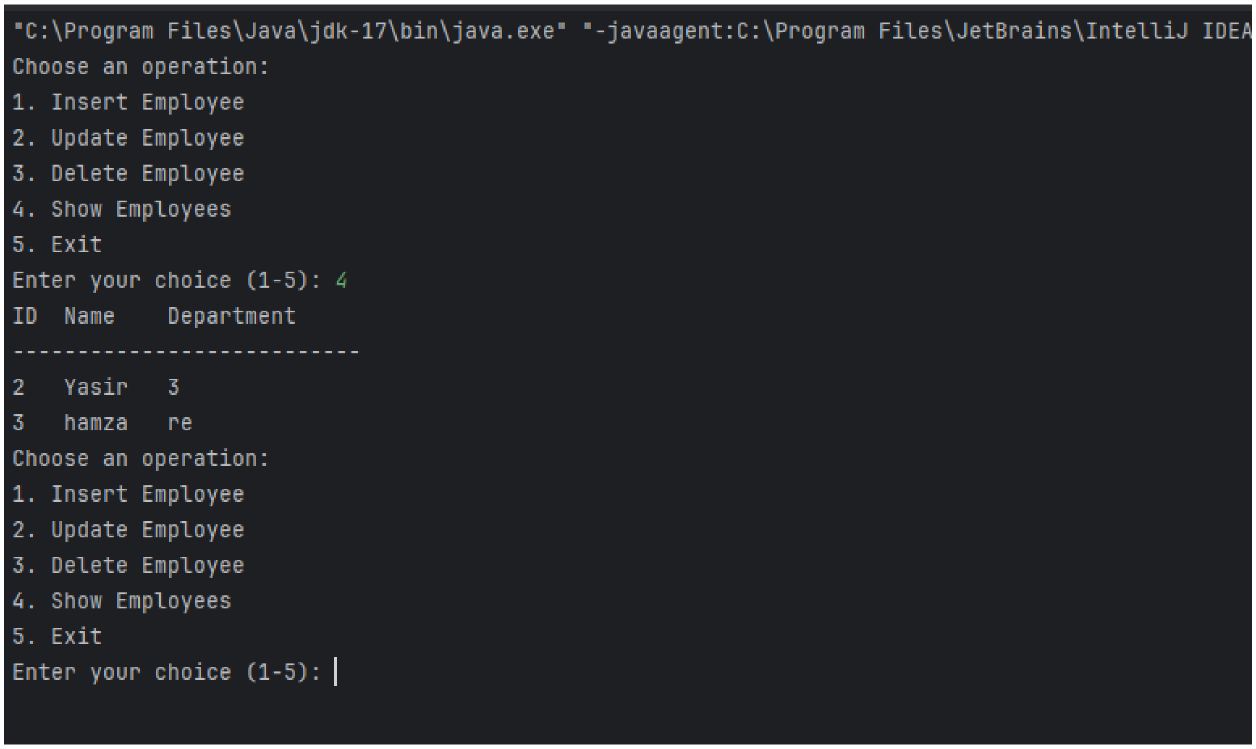
➢ Now add DeleteData.java code

```java
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;


public class DeleteData {
    public void deleteEmployee(int id) {
        String sql = "DELETE FROM employees WHERE id = ?";

        try (Connection connection = DatabaseUtil.getConnection();
             PreparedStatement preparedStatement =
connection.prepareStatement(sql)) {

            preparedStatement.setInt(1, id);

            int rowsAffected = preparedStatement.executeUpdate();
            System.out.println(rowsAffected + " row(s) deleted.");
        } catch (SQLException e) {
            System.out.println("Connection or delete failed!");
            e.printStackTrace();
        }
    }
}
```

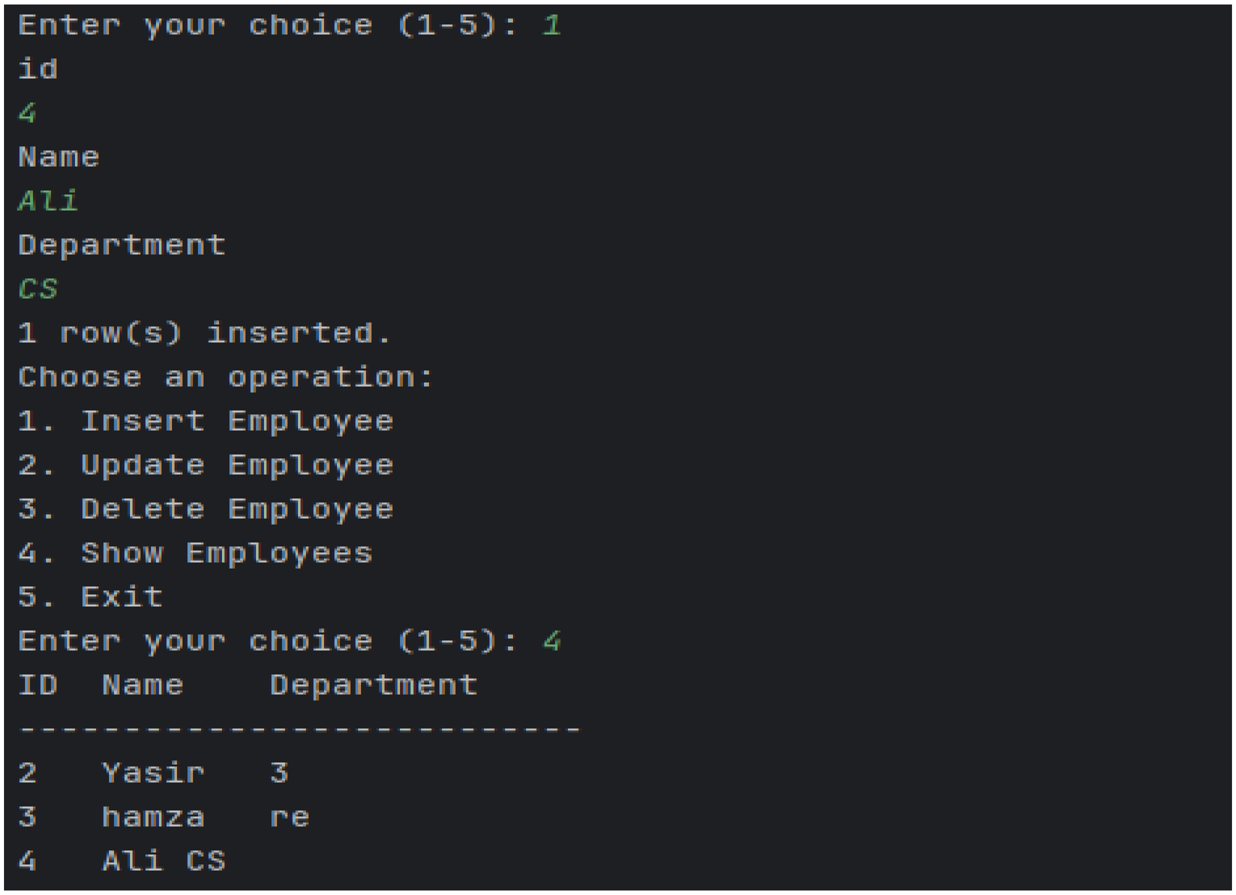
➢ Now Add MainApp.java code

```java
import java.util.Scanner;

public class MainApp {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        InsertData insertData = new InsertData();
        UpdateData updateData = new UpdateData();
        DeleteData deleteData = new DeleteData();
        ShowData showData = new ShowData();

        boolean continueRunning = true;

        while (continueRunning) {
            // Display the menu
            System.out.println("Choose an operation:");
            System.out.println("1. Insert Employee");
            System.out.println("2. Update Employee");
            System.out.println("3. Delete Employee");
            System.out.println("4. Show Employees");
            System.out.println("5. Exit");
            System.out.print("Enter your choice (1-5): ");

            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume the newline

            switch (choice) {
                case 1:
                    System.out.println("id");
                    int u_id = scanner.nextInt();

                    System.out.println("Name");
                    String u_name = scanner.next();

                    System.out.println("Department");
                    String u_department = scanner.next();
                    insertData.insertEmployee(u_id,u_name,u_department);
                    break;
                case 2:
                    System.out.print("Enter the ID of the employee to update: 
");
                    int updateId = scanner.nextInt();
                    scanner.nextLine(); // Consume the newline
                    System.out.print("Enter the new name: ");
                    String newName = scanner.nextLine();
                    System.out.print("Enter the new department: ");
                    String newDepartment = scanner.nextLine();
                    updateData.updateEmployee(updateId, newName, 
newDepartment);
                    break;
                case 3:
                    System.out.print("Enter the ID of the employee to delete: 
");
                    int deleteId = scanner.nextInt();
                    deleteData.deleteEmployee(deleteId);
                    break;
                case 4:
                    showData.displayEmployees();
                    break;
                case 5:
                    continueRunning = false; // Exit the loop
                    break;
                default:
                    System.out.println("Invalid choice! Please enter a number
between 1 and 5.");
            }
        }

        System.out.println("Exiting the program. Goodbye!");
        scanner.close();
    }
}
```

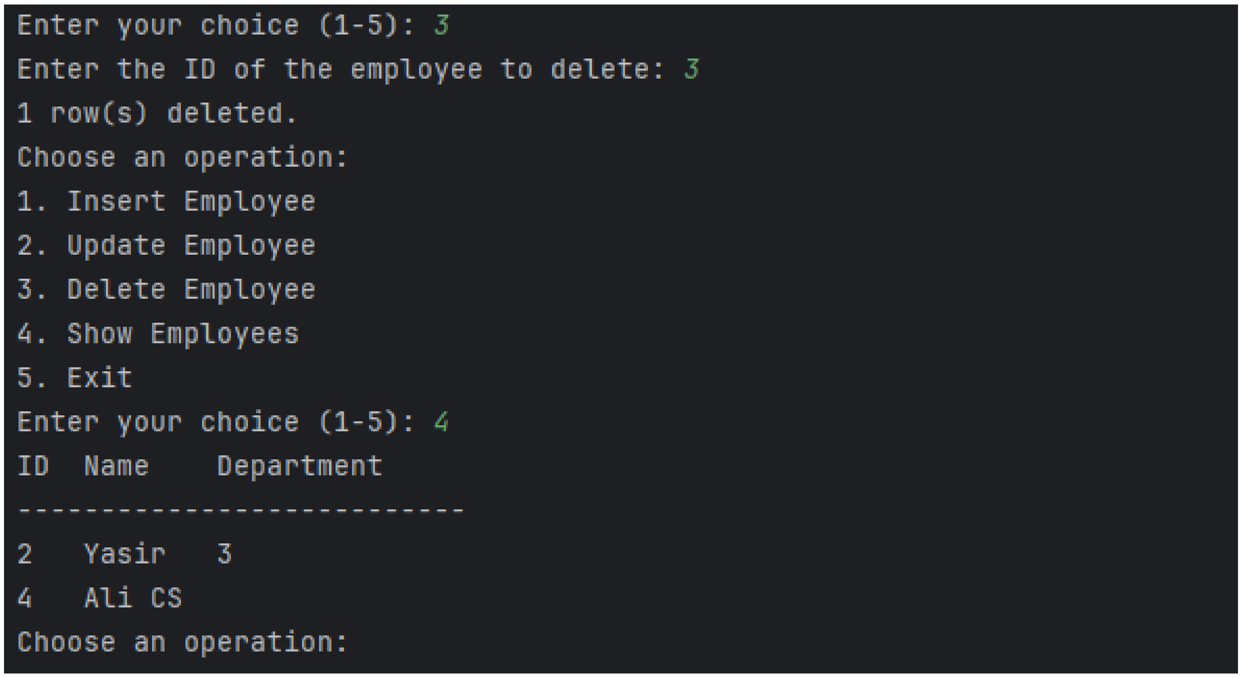
- Now Lets run the MainApp.java Code



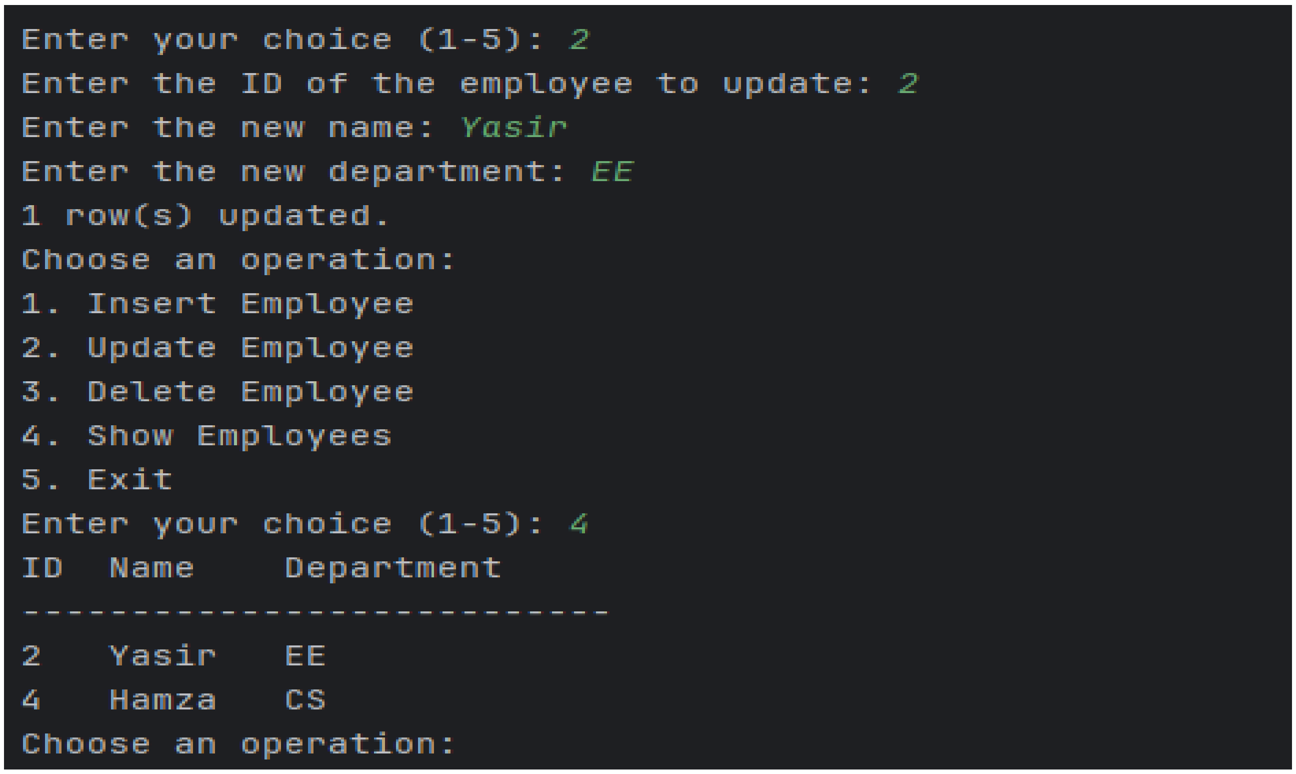
```
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
Choose an operation:
1. Insert Employee
2. Update Employee
3. Delete Employee
4. Show Employees
5. Exit
Enter your choice (1-5): 4
ID  Name    Department
---------------------------
2   Yasir   3
3   hamza   re
Choose an operation:
1. Insert Employee
2. Update Employee
3. Delete Employee
4. Show Employees
5. Exit
Enter your choice (1-5): 
```



```
Enter your choice (1-5): 1
id
4
Name
Ali
Department
CS
1 row(s) inserted.
Choose an operation:
1. Insert Employee
2. Update Employee
3. Delete Employee
4. Show Employees
5. Exit
Enter your choice (1-5): 4
ID  Name    Department
---------------------------
2   Yasir   3
3   hamza   re
4   Ali CS
```



```
Enter your choice (1-5): 3
Enter the ID of the employee to delete: 3
1 row(s) deleted.
Choose an operation:
1. Insert Employee
2. Update Employee
3. Delete Employee
4. Show Employees
5. Exit
Enter your choice (1-5): 4
ID  Name    Department
---------------------------
2   Yasir   3
4   Ali CS
Choose an operation:
```



```
Enter your choice (1-5): 2
Enter the ID of the employee to update: 2
Enter the new name: Yasir
Enter the new department: EE
1 row(s) updated.
Choose an operation:
1. Insert Employee
2. Update Employee
3. Delete Employee
4. Show Employees
5. Exit
Enter your choice (1-5): 4
ID  Name    Department
---------------------------
2   Yasir   EE
4   Hamza   CS
Choose an operation:
```

JDBC Project with Swing

Now we will create a swing Application with JDBC

Create a New Project and Name it as Jdbcswing

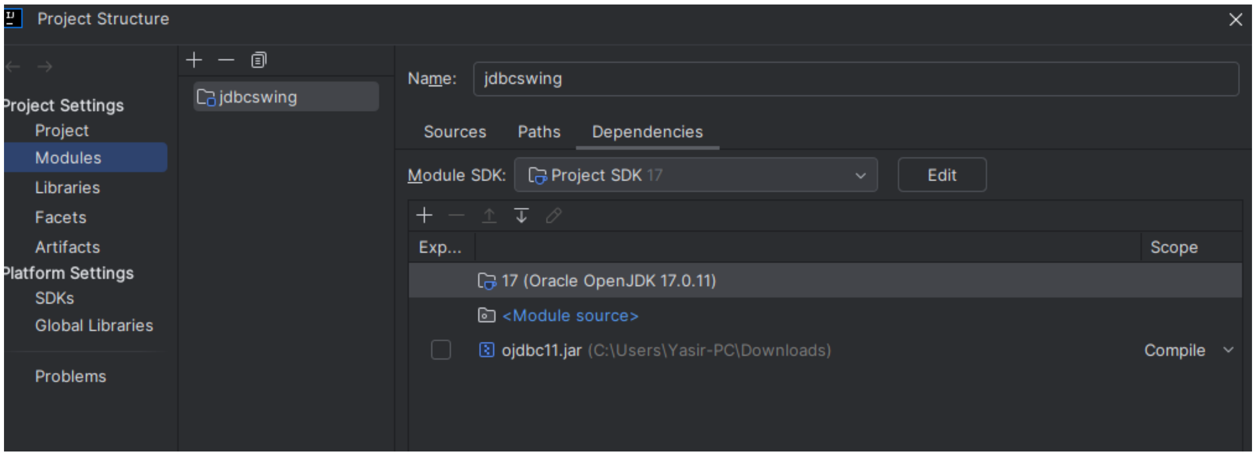
Set up a project structure and choose a location for your project.

 Add ojdbc11.jar to the Project:

Right-click on your project, go to "Modules" > "Dependencies."

Click the "+" button to add the JAR file and choose "JARs or directories."

Navigate to the location where you downloaded ojdbc11.jar and add it.



Name: jdbcswing

Module SDK: Project SDK 17

Exp...		Scope
	17 (Oracle OpenJDK 17.0.11)	
	<Module source>	
☐	ojdbc11.jar (C:\Users\Yasir-PC\Downloads)	Compile

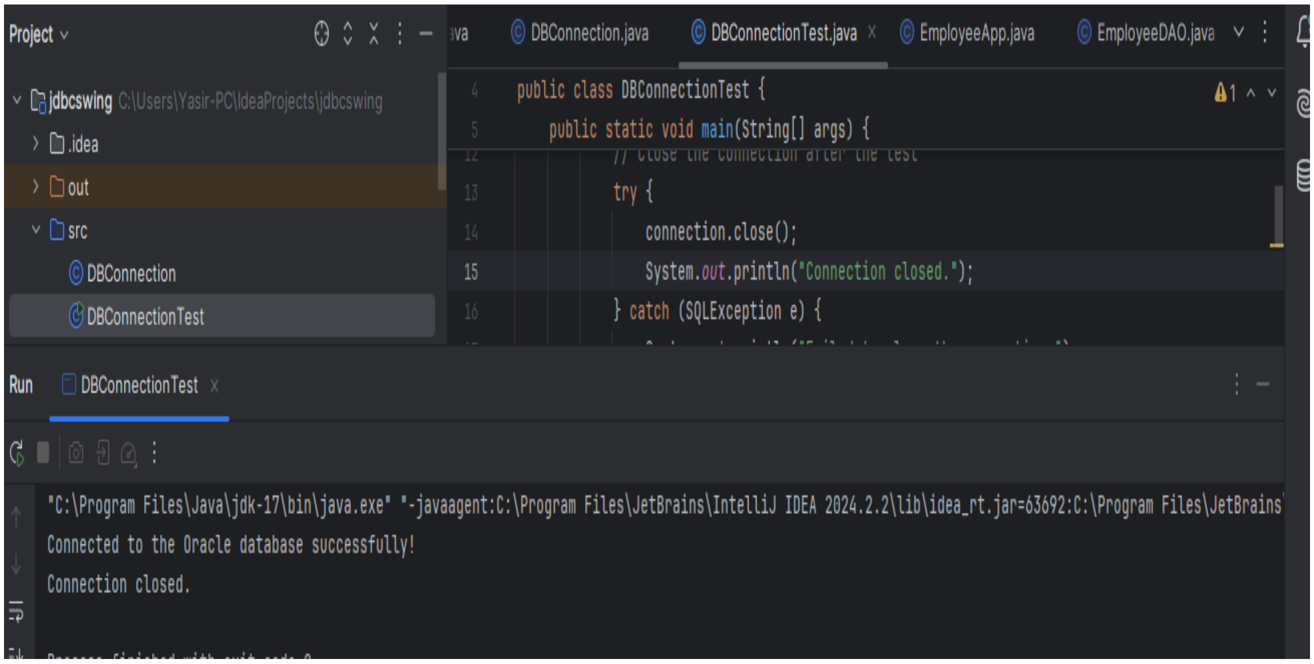
➢ Now add the DBConnection.java code into your project

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnection {
    private static final String URL = "jdbc:oracle:thin:@localhost:1521:xe";
// Update with your DB URL
    private static final String USERNAME = "java_crud"; // Update with your
DB username
    private static final String PASSWORD = "fast"; // Update with your DB
password

    public static Connection getConnection() {
        try {
            // Load Oracle JDBC Driver
            Class.forName("oracle.jdbc.driver.OracleDriver");

            // Establish connection to the database
            return DriverManager.getConnection(URL, USERNAME, PASSWORD);
        } catch (ClassNotFoundException e) {
            System.out.println("Oracle JDBC Driver not found.");
            e.printStackTrace();
        } catch (SQLException e) {
            System.out.println("Connection failed.");
            e.printStackTrace();
        }
        return null;
    }
}
```

➢ Now add the DBConnectionTest To test if you are connected to database or not

```java
import java.sql.Connection;
import java.sql.SQLException;

public class DBConnectionTest {
    public static void main(String[] args) {
        // Get the connection from the DBConnection class
        Connection connection = DBConnection.getConnection();

        if (connection != null) {
            System.out.println("Connected to the Oracle database
successfully!");

            // Close the connection after the test
            try {
                connection.close();
                System.out.println("Connection closed.");
            } catch (SQLException e) {
                System.out.println("Failed to close the connection.");
                e.printStackTrace();
            }
        } else {
            System.out.println("Failed to connect to the Oracle database.");
        }
    }
}
```

➢ Now lets Run this code to see if we are connected to database or not



```java
public class DBConnectionTest {
    public static void main(String[] args) {
            // Close the connection after the test
            try {
                connection.close();
                System.out.println("Connection closed.");
            } catch (SQLException e) {
```

```
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\lib\idea_rt.jar=63692:C:\Program Files\JetBrains
Connected to the Oracle database successfully!
Connection closed.

Process finished with exit code 0
```

➢ Now add EmployeApp.java Code

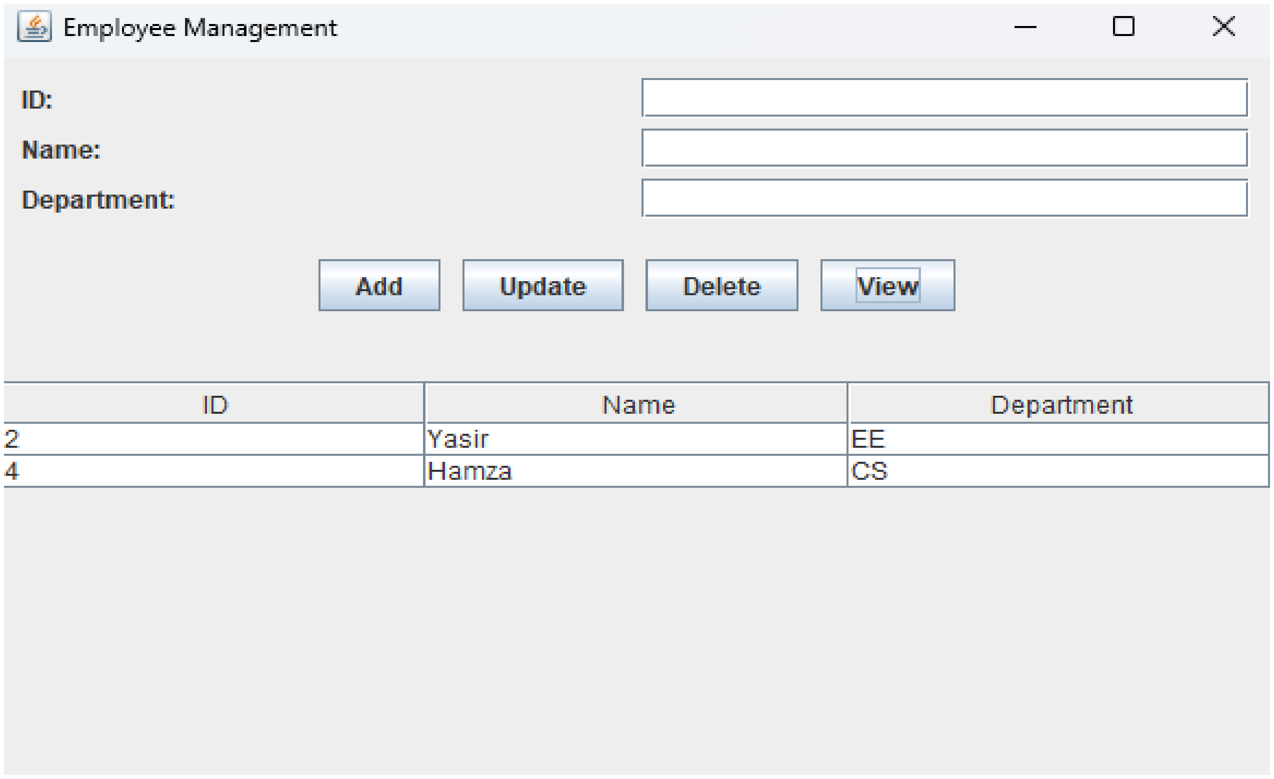
```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;
import javax.swing.table.DefaultTableModel;

public class EmployeeApp extends JFrame {
    private JTextField idField, nameField, departmentField;
    private JButton addButton, updateButton, deleteButton, viewButton;
    private JTable employeeTable;
    private EmployeeDAO employeeDAO;

    public EmployeeApp() {
        employeeDAO = new EmployeeDAO();

        // Set up the frame
        setTitle("Employee Management");
        setSize(600, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        // Panel for input fields
        JPanel inputPanel = new JPanel(new GridLayout(3, 2, 5, 5));
        inputPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10,
10));
        inputPanel.add(new JLabel("ID:"));
        idField = new JTextField();
        inputPanel.add(idField);
        inputPanel.add(new JLabel("Name:"));
        nameField = new JTextField();
        inputPanel.add(nameField);
        inputPanel.add(new JLabel("Department:"));
        departmentField = new JTextField();
        inputPanel.add(departmentField);
        add(inputPanel, BorderLayout.NORTH);

        // Panel for buttons
        JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 10,
10));
        addButton = new JButton("Add");
        updateButton = new JButton("Update");
        deleteButton = new JButton("Delete");
        viewButton = new JButton("View");
        buttonPanel.add(addButton);
        buttonPanel.add(updateButton);
        buttonPanel.add(deleteButton);
        buttonPanel.add(viewButton);
        add(buttonPanel, BorderLayout.CENTER);

        // Panel for the table
        JPanel tablePanel = new JPanel(new BorderLayout());
        employeeTable = new JTable();
        tablePanel.add(new JScrollPane(employeeTable), BorderLayout.CENTER);
        tablePanel.setPreferredSize(new Dimension(600, 200));
        add(tablePanel, BorderLayout.SOUTH);

        // Add action listeners to the buttons
        addButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                addEmployee();
            }
        });

        updateButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                updateEmployee();
            }
        });

        deleteButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                deleteEmployee();
            }
        });

        viewButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                viewEmployees();
            }
        });

        // Add mouse listener for table row selection
        employeeTable.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                int row = employeeTable.getSelectedRow();
                if (row != -1) { // Ensure a row is selected
                    idField.setText(employeeTable.getValueAt(row,
0).toString());
                    nameField.setText(employeeTable.getValueAt(row,
1).toString());
                    departmentField.setText(employeeTable.getValueAt(row,
2).toString());
                }
            }
        });

        // Show the frame
        setVisible(true);
    }

    private void addEmployee() {
        int id = Integer.parseInt(idField.getText());
        String name = nameField.getText();
        String department = departmentField.getText();
        if (employeeDAO.insertEmployee(id, name, department)) {
            JOptionPane.showMessageDialog(this, "Employee added 
successfully!");
            clearFields();
            viewEmployees();
        } else {
            JOptionPane.showMessageDialog(this, "Error adding employee.");
        }
    }

    private void updateEmployee() {
        String idText = idField.getText().trim();
        String name = nameField.getText().trim();
        String department = departmentField.getText().trim();

        if (idText.isEmpty() || name.isEmpty() || department.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Please fill in all 
fields.");
            return;
        }

        int id;
        try {
            id = Integer.parseInt(idText);
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "Invalid ID format.");
            return;
        }

        if (employeeDAO.updateEmployee(id, name, department)) {
            JOptionPane.showMessageDialog(this, "Employee updated 
successfully!");
            clearFields();
            viewEmployees();
        } else {
            JOptionPane.showMessageDialog(this, "Error updating employee. 
Please ensure the ID exists.");
        }
    }

    private void deleteEmployee() {
        String idText = idField.getText().trim();

        if (idText.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Please enter an ID.");
            return;
        }

        int id;
        try {
            id = Integer.parseInt(idText);
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "Invalid ID format.");
            return;
        }
        if (employeeDAO.deleteEmployee(id)) {
            JOptionPane.showMessageDialog(this, "Employee deleted
successfully!");
            clearFields();
            viewEmployees();
        } else {
            JOptionPane.showMessageDialog(this, "Error deleting employee.
Please ensure the ID exists.");
        }
    }

    private void viewEmployees() {
        List<String[]> employees = employeeDAO.getAllEmployees();
        String[] columnNames = {"ID", "Name", "Department"};
        DefaultTableModel model = new DefaultTableModel(columnNames, 0);

        for (String[] employee : employees) {
            model.addRow(employee);
        }

        employeeTable.setModel(model);
    }

    private void clearFields() {
        idField.setText("");
        nameField.setText("");
        departmentField.setText("");
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new EmployeeApp());
    }
}
```

- Now we will add EmployeeDAO.java

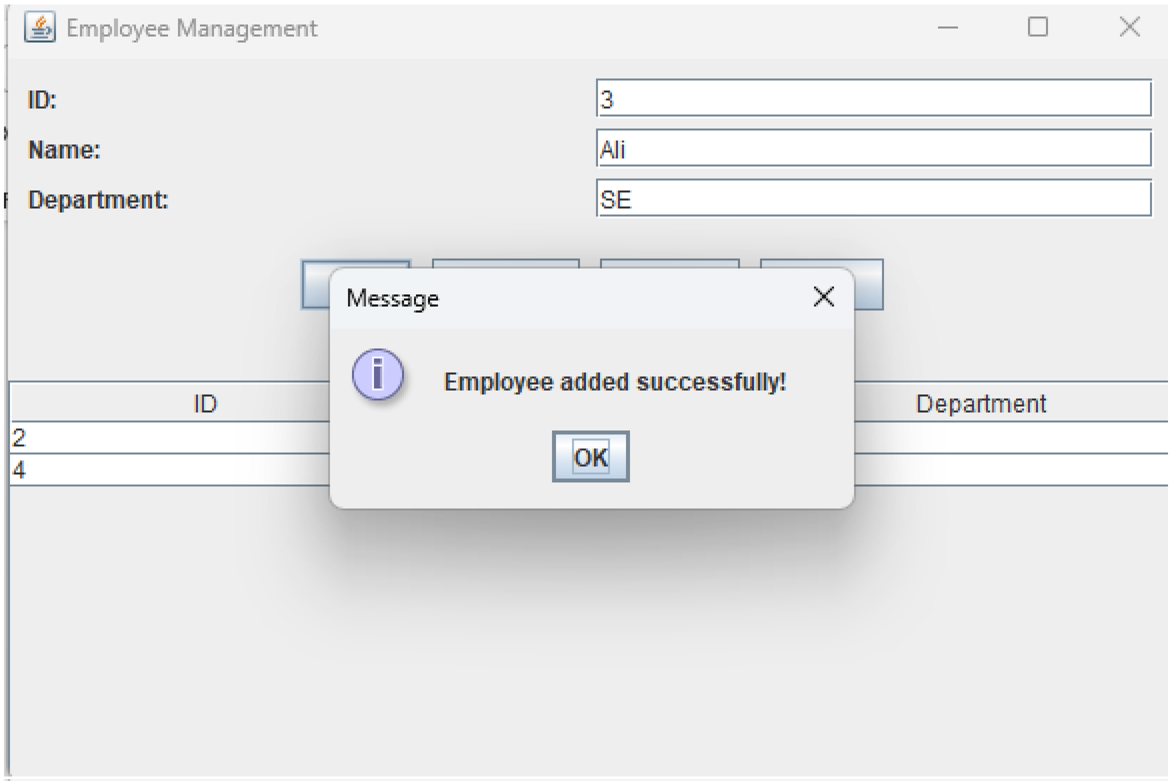
```
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class EmployeeDAO {
    private Connection connection;

    public EmployeeDAO() {
        connection = DBConnection.getConnection();
    }

    // Insert a new employee
    public boolean insertEmployee(int id, String name, String department) {
        String query = "INSERT INTO Employees (ID, Name, Department) VALUES
(?, ?, ?)";
        try (PreparedStatement stmt = connection.prepareStatement(query)) {
```

```java
            stmt.setInt(1, id);
            stmt.setString(2, name);
            stmt.setString(3, department);
            stmt.executeUpdate();
            return true;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    // Update an existing employee
    public boolean updateEmployee(int id, String name, String department) {
        String query = "UPDATE Employees SET Name = ?, Department = ? WHERE
ID = ?";
        try (PreparedStatement stmt = connection.prepareStatement(query)) {
            stmt.setString(1, name);
            stmt.setString(2, department);
            stmt.setInt(3, id);
            stmt.executeUpdate();
            return true;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    // Delete an employee
    public boolean deleteEmployee(int id) {
        String query = "DELETE FROM Employees WHERE ID = ?";
        try (PreparedStatement stmt = connection.prepareStatement(query)) {
            stmt.setInt(1, id);
            stmt.executeUpdate();
            return true;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    // Fetch all employees
    public List<String[]> getAllEmployees() {
        List<String[]> employees = new ArrayList<>();
        String query = "SELECT * FROM Employees";
        try (Statement stmt = connection.createStatement();
             ResultSet rs = stmt.executeQuery(query)) {
            while (rs.next()) {
                String[] employee = {
                        String.valueOf(rs.getInt("ID")),
                        rs.getString("Name"),
                        rs.getString("Department")
                };
                employees.add(employee);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return employees;
    }
}
```

Now Run the EmployeApp.java Code



ID	Name	Department
2	Yasir	EE
4	Hamza	CS



Employee Management

ID: 3

Name: Ali

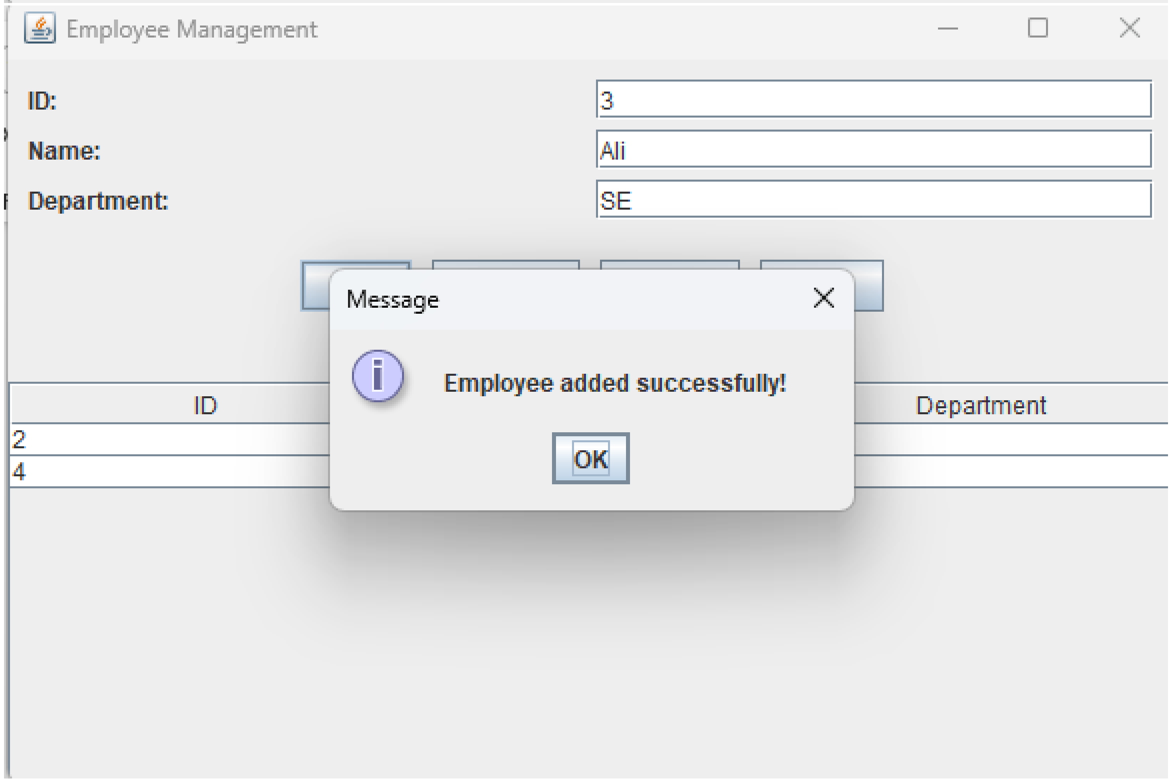
Department: SE

Message

Employee added successfully!

OK

ID	Department
2	
4	



Employee Management

ID: 3

Name: Ali

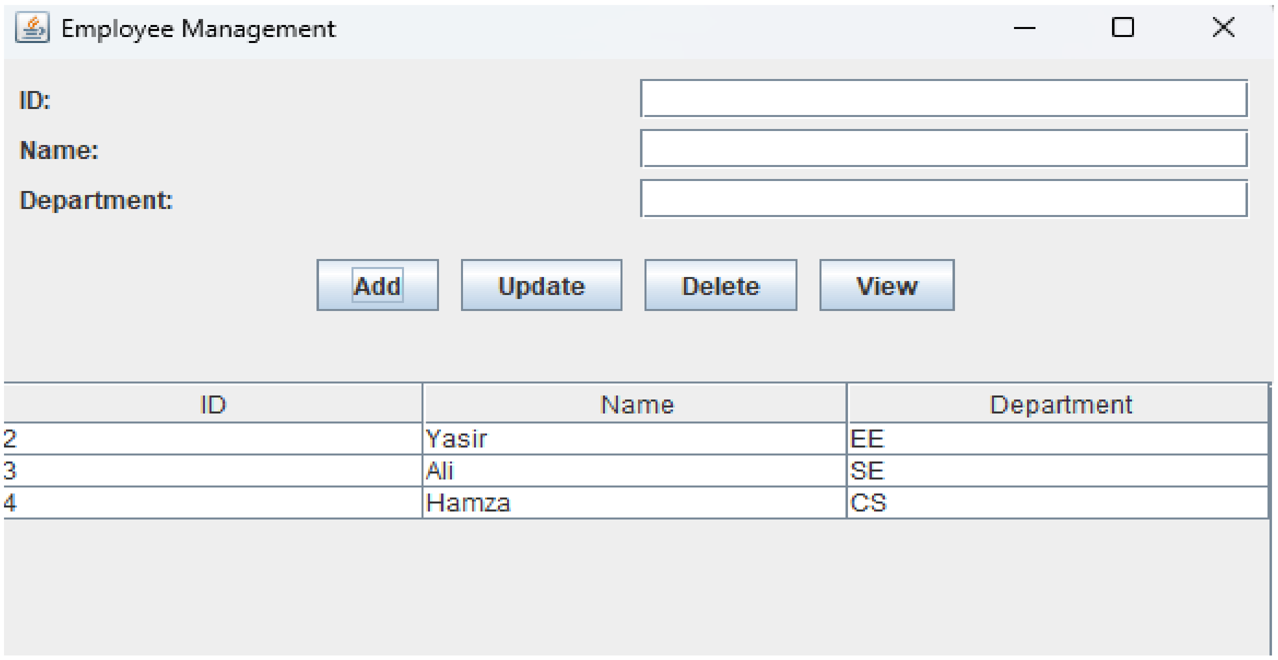
Department: SE

Message

Employee added successfully!

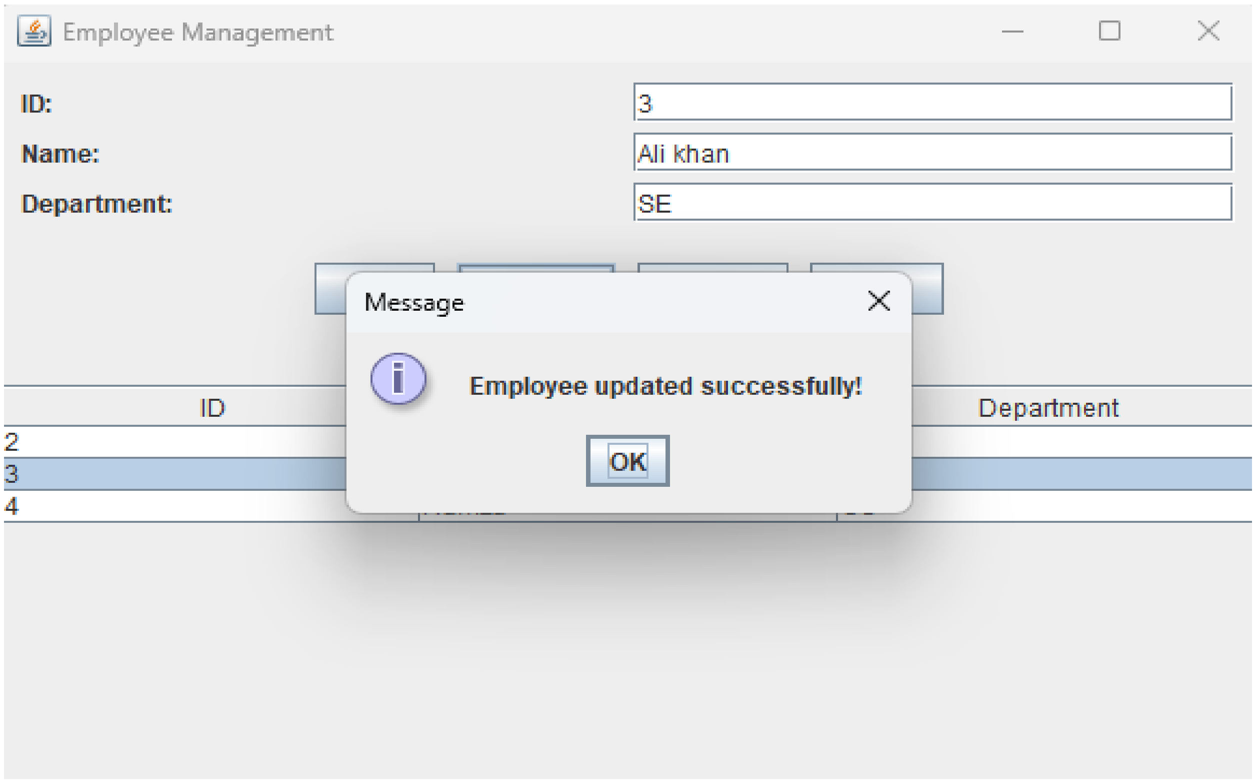
OK

ID	Department
2	
4	



ID:
Name:
Department:

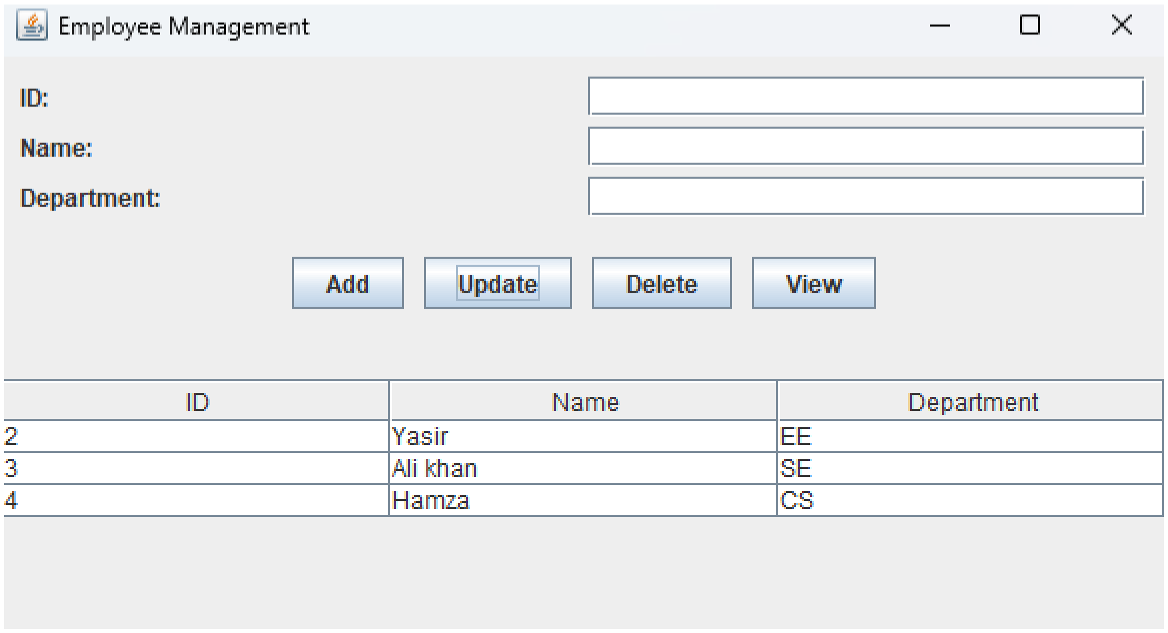
ID	Name	Department
2	Yasir	EE
3	Ali	SE
4	Hamza	CS



ID: 3
Name: Ali khan
Department: SE

Employee updated successfully!

ID	Department
2	
3	
4	

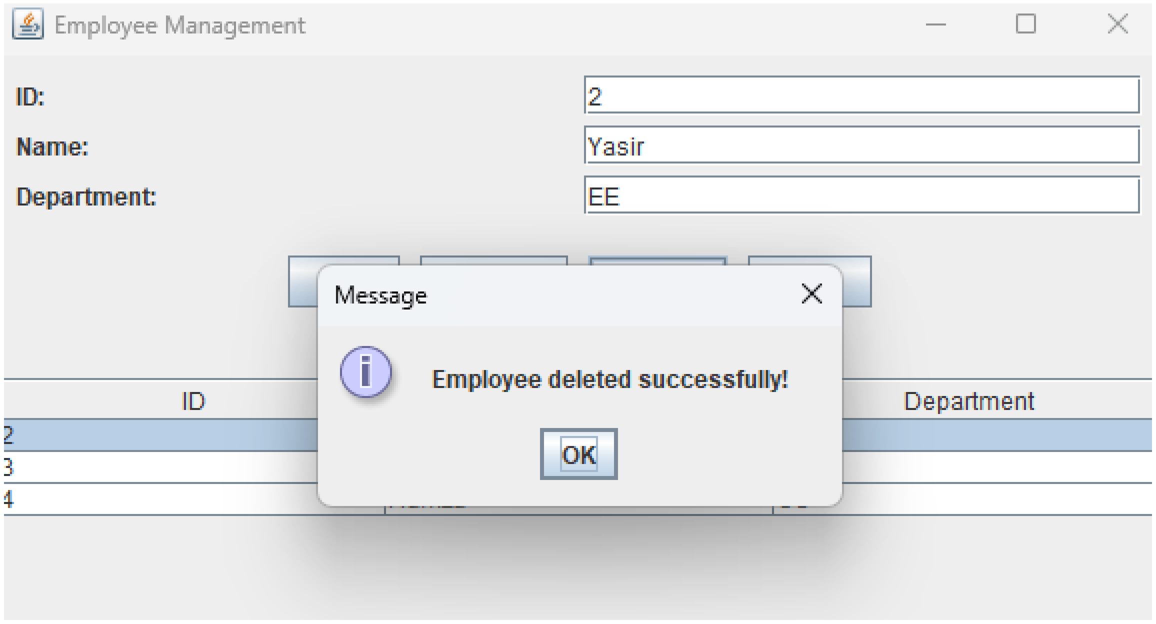

Employee Management

ID:

Name:

Department:

ID	Name	Department
2	Yasir	EE
3	Ali khan	SE
4	Hamza	CS


Employee Management

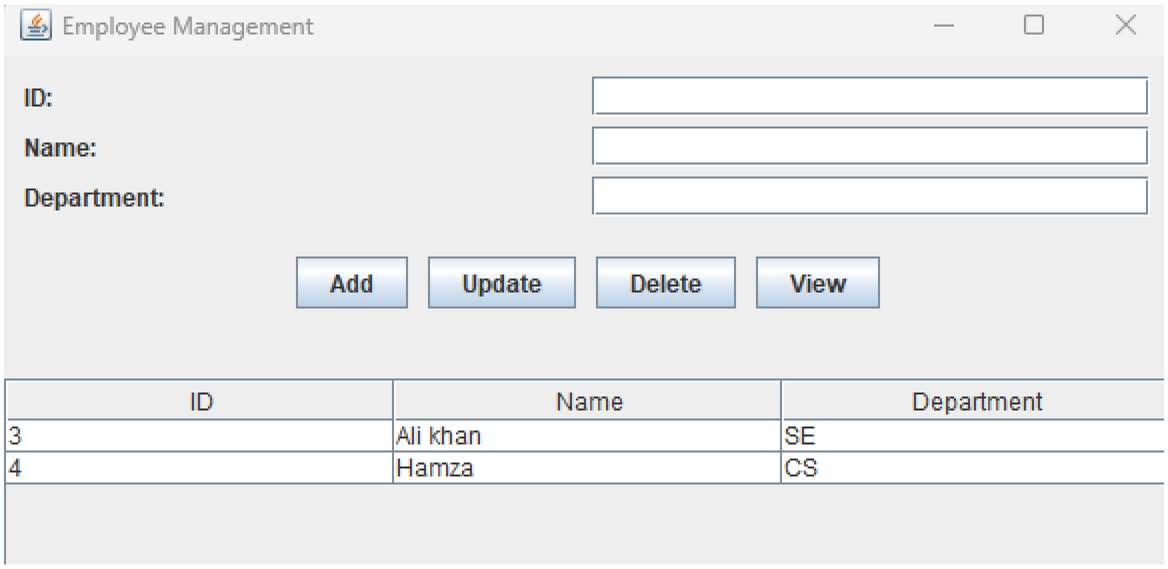
ID: 2

Name: Yasir

Department: EE

Message

Employee deleted successfully!



ID	Name	Department
3	Ali khan	SE
4	Hamza	CS

a