Week 1 Assignment -2 Java



Given:

public class TaxUtil {

   double rate = 0.15;

   public double calculateTax(double amount) {

       return amount \* rate;

   }

}

Would you consider the method calculateTax() a 'pure function'? Why or why not?

If you claim the method is NOT a pure function, please suggest a way to make it pure.

**Ans)** i)The method calculateTax() is not a pure function as it uses the instance variable rate, which exists outside the method.

ii)As a result, the output can change even if the input (amount) stays the same depending on the value of rate.

iii)So if we want to make it a pure function, we can remove the dependency on external state and pass rate directly

2)

What will be the output for following code?

class Super

{

static void show()

{

System.out.println("super class show method");

}

static class StaticMethods

{

void show()

{

System.out.println("sub class show method");

}

}

public static void main(String[]args)

{

Super.show();

new Super.StaticMethods().show();

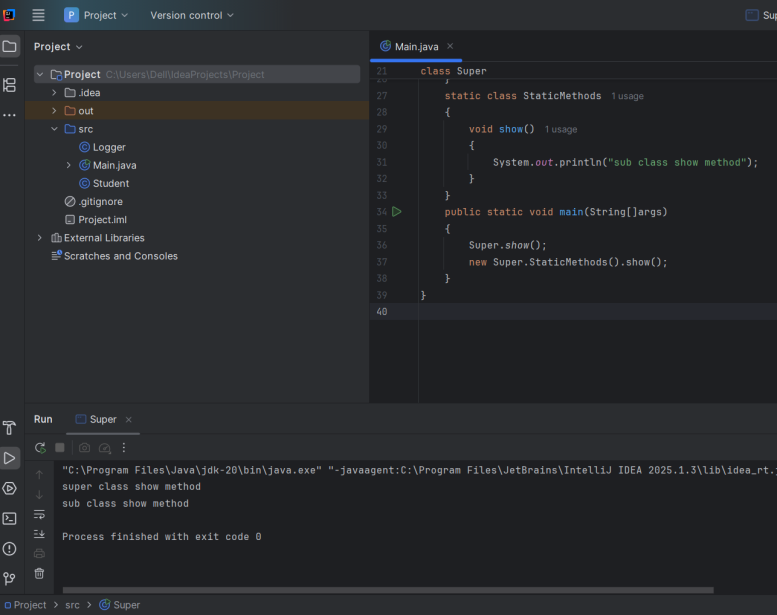
}

}

**Output :**

Super class show method

Sub class show method



3)

What will be the output for the following code?

class Super

{

int num=20;

public void display()

{

System.out.println("super class method");

}

}

public class ThisUse extends Super

{

int num;

public ThisUse(int num)

{

this.num=num;

}

public void display()

{

System.out.println("display method");

}

public void Show()

{

this.display();

display();

System.out.println(this.num);

System.out.println(num);

}

public static void main(String[]args)

{

ThisUse o=new ThisUse(10);

o.show();

}

}

**Ans)** o/p: display method  
 display method  
 10  
 10

4) What is the singleton design pattern? Explain with a coding example.

**Ans)** The Singleton design pattern ensures that a class has only one instance and provides a global point of access to it.

Example:

public class Singleton {

private static Singleton instance;

private Singleton() {

System.out.println("Singleton instance created");

}

public static Singleton getInstance() {

if (instance == null) {

instance = new Singleton();

}

return instance;

}

public void showMessage() {

System.out.println("Hello from Singleton!");

}

}

public class Main {

public static void main(String[] args) {

Singleton obj1 = Singleton.getInstance();

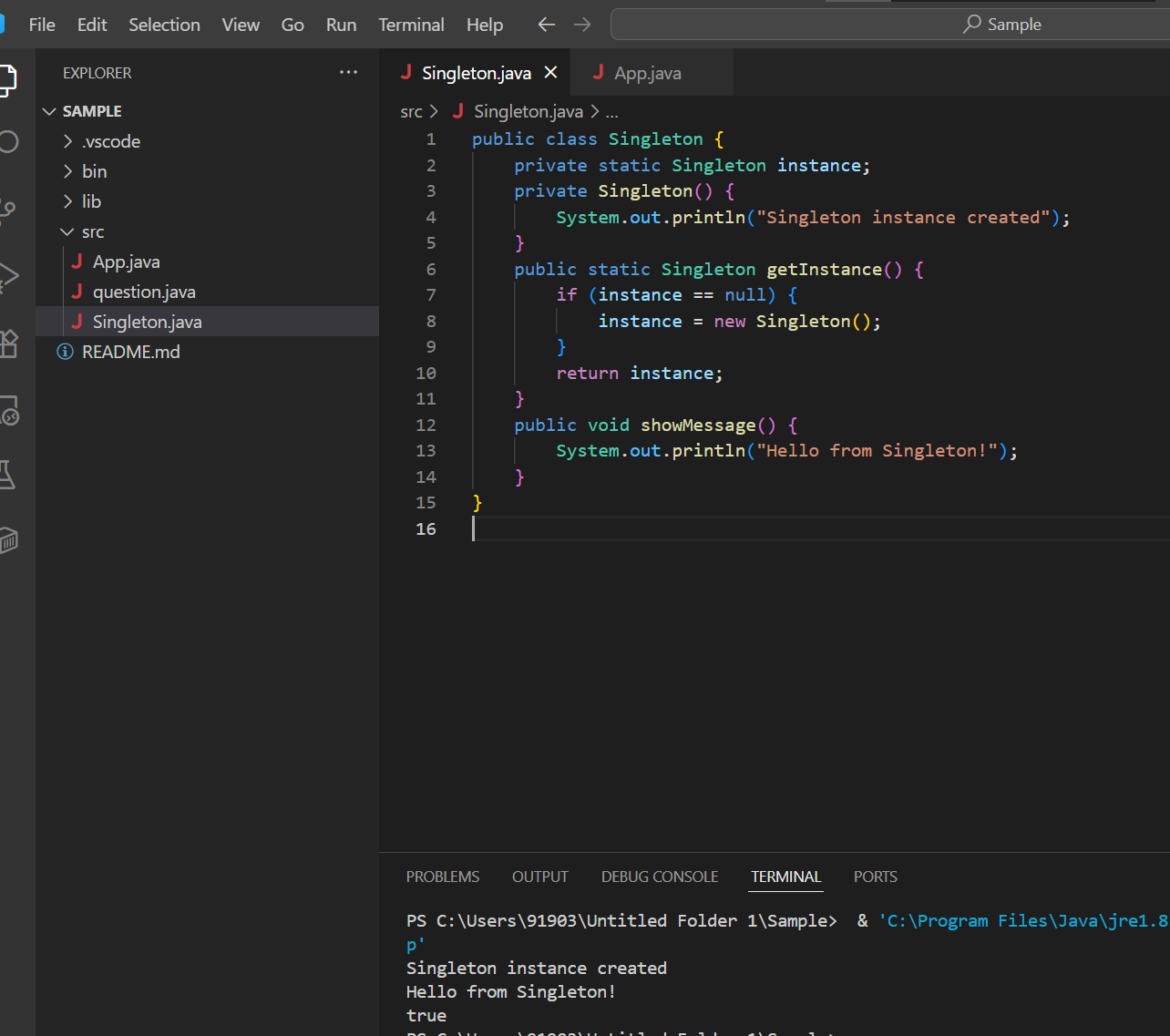
Singleton obj2 = Singleton.getInstance();

obj1.showMessage();

System.out.println(obj1 == obj2); // Output: true

}

}



5) How do we make sure a class is encapsulated? Explain with a coding example.

Ans) Encapsulation is the concept of wrapping data (variables) and methods that operate on that data into a single unit . To make sure the class is encapsulated we make the variables private and getters and setters as public.

Example-

Main.java

public class Main {

    public static void main(String[] args) {

        Bankaccount account = new Bankaccount("Alice", 1000.0);

        System.out.println("Account Holder: " + account.getAccountHolder());

        System.out.println("Balance: " + account.getBalance());

        account.deposit(500);

        System.out.println("After deposit: " + account.getBalance());

        account.withdraw(200);

        System.out.println("After withdrawal: " + account.getBalance());

    }

}

Bankaccount.java

public class Bankaccount {

    // Private fields - encapsulated

    private String accountHolder;

    private double balance;

    public Bankaccount(String accountHolder, double balance) {

        this.accountHolder = accountHolder;

        this.balance = balance;

    }

    public String getAccountHolder() {

        return accountHolder;

    }

    public double getBalance() {

        return balance;

    }

    public void deposit(double amount) {

        if (amount > 0) {

            balance += amount;

        } else {

            System.out.println("Invalid deposit amount.");

        }

    }

    public void withdraw(double amount) {

        if (amount > 0 && amount <= balance) {

            balance -= amount;

        } else {

            System.out.println("Invalid withdrawal amount or insufficient balance.");

        }

    }

}

Output

A screen shot of a computer

AI-generated content may be incorrect.

6)

Perform CRUD operation using ArrayList collection in an EmployeeCRUD class for the below Employee

class Employee{

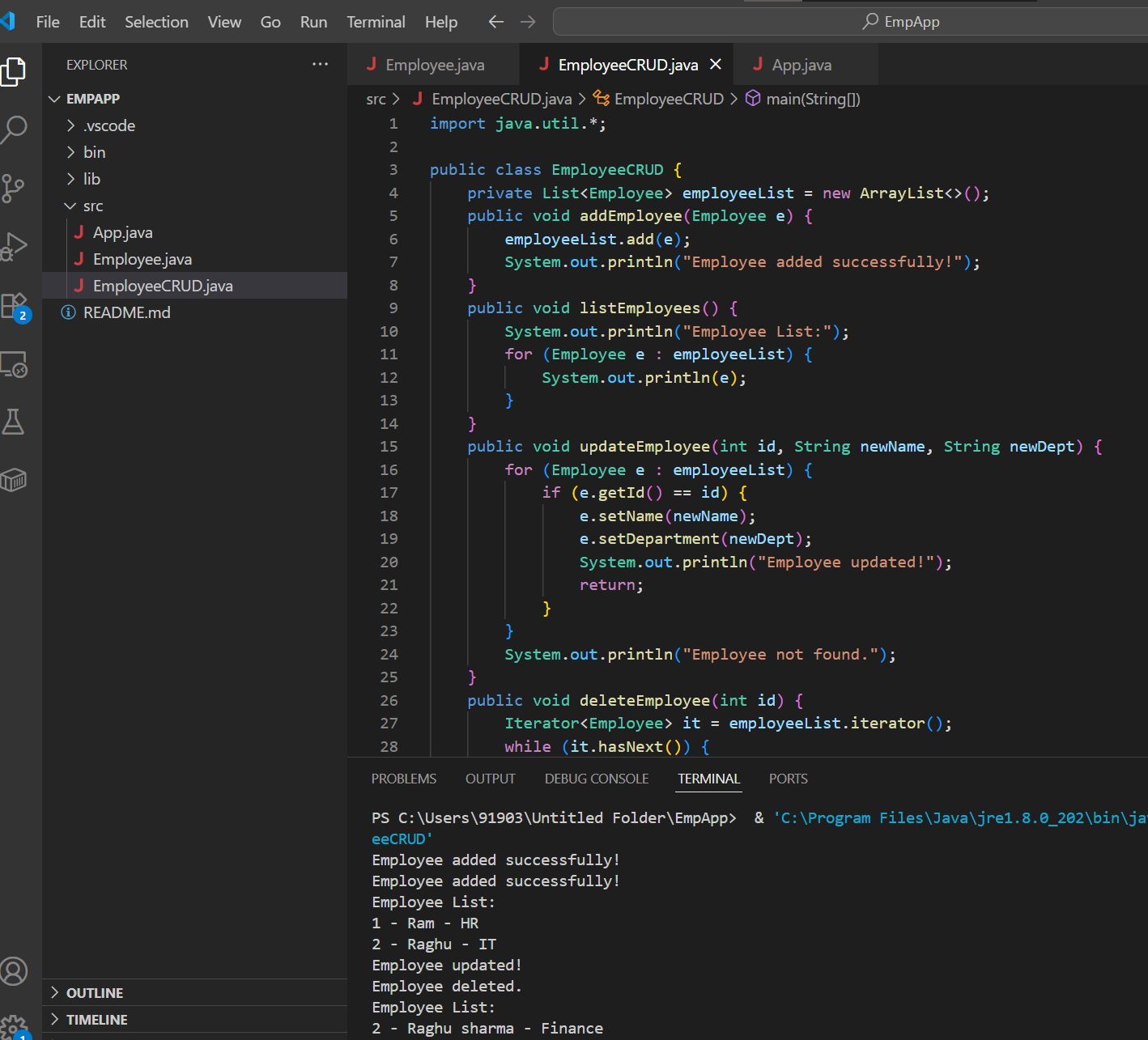
private int id;

private String name;

private String department;

}

**Ans)**



import java.util.\*;

public class EmployeeCRUD {

    private List<Employee> employeeList = new ArrayList<>();

    public void addEmployee(Employee e) {

        employeeList.add(e);

        System.out.println("Employee added successfully!");

    }

    public void listEmployees() {

        System.out.println("Employee List:");

        for (Employee e : employeeList) {

            System.out.println(e);

        }

    }

    public void updateEmployee(int id, String newName, String newDept) {

        for (Employee e : employeeList) {

            if (e.getId() == id) {

                e.setName(newName);

                e.setDepartment(newDept);

                System.out.println("Employee updated!");

                return;

            }

        }

        System.out.println("Employee not found.");

    }

    public void deleteEmployee(int id) {

        Iterator<Employee> it = employeeList.iterator();

        while (it.hasNext()) {

            Employee e = it.next();

            if (e.getId() == id) {

                it.remove();

                System.out.println("Employee deleted.");

                return;

            }

        }

        System.out.println("Employee not found.");

    }

    public static void main(String[] args) {

        EmployeeCRUD crud = new EmployeeCRUD();

        crud.addEmployee(new Employee(1, "Ram", "HR"));

        crud.addEmployee(new Employee(2, "Raghu", "IT"));

        crud.listEmployees();

        crud.updateEmployee(2, "Raghu sharma", "Finance");

        crud.deleteEmployee(1);

        crud.listEmployees();

    }

}

7) Perform CRUD operation using JDBC in an EmployeeJDBC class for the below Employee

class Employee{

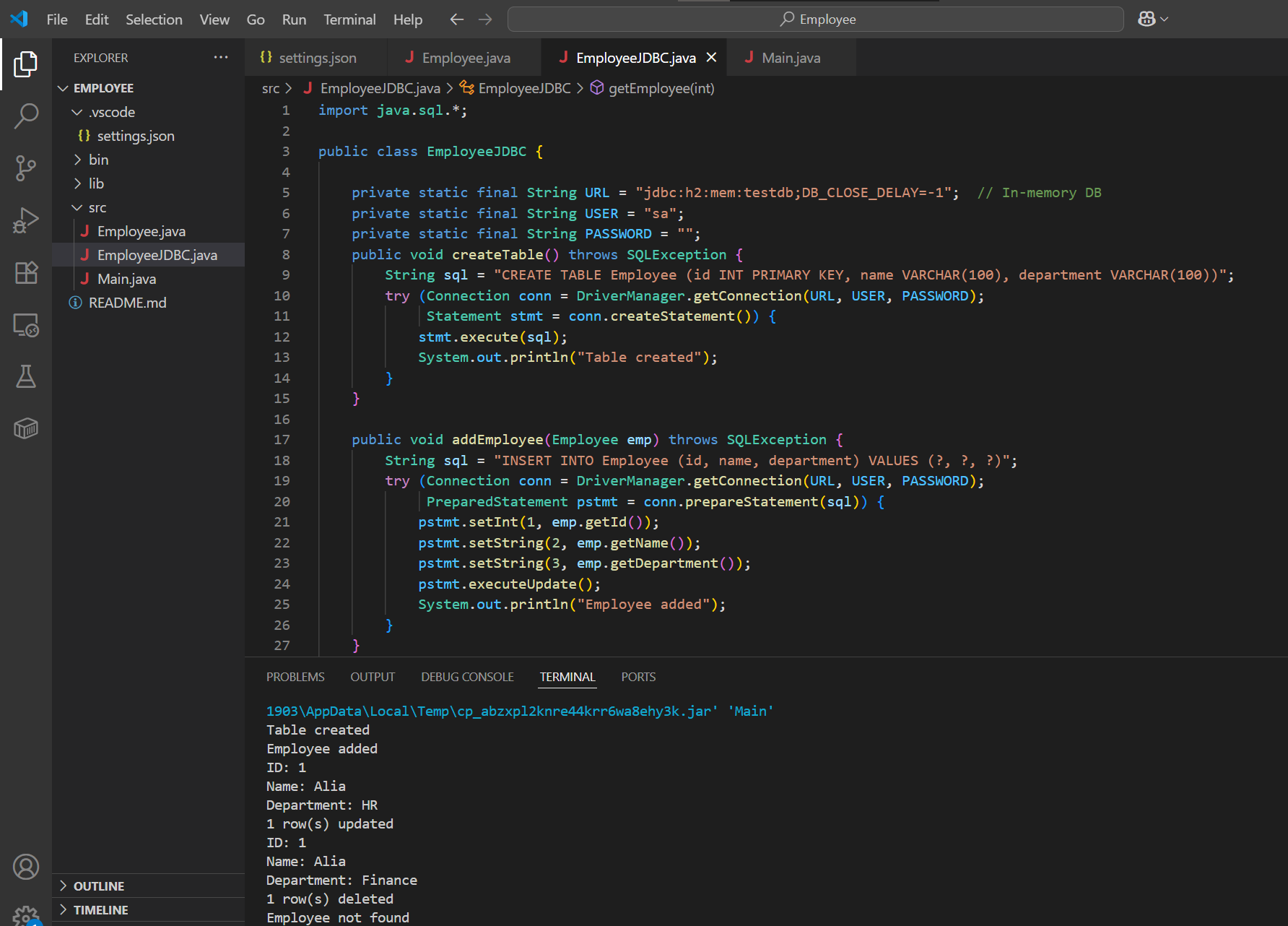
private int id;

private String name;

private String department;

}

**Ans)**



**EmployeeJDBC.java**

import java.sql.\*;

public class EmployeeJDBC {

private static final String URL = "jdbc:h2:mem:testdb;DB\_CLOSE\_DELAY=-1"; // In-memory DB

private static final String USER = "sa";

private static final String PASSWORD = "";

public void createTable() throws SQLException {

String sql = "CREATE TABLE Employee (id INT PRIMARY KEY, name VARCHAR(100), department VARCHAR(100))";

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

Statement stmt = conn.createStatement()) {

stmt.execute(sql);

System.out.println("Table created");

}

}

public void addEmployee(Employee emp) throws SQLException {

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

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

PreparedStatement pstmt = conn.prepareStatement(sql)) {

pstmt.setInt(1, emp.getId());

pstmt.setString(2, emp.getName());

pstmt.setString(3, emp.getDepartment());

pstmt.executeUpdate();

System.out.println("Employee added");

}

}

public void getEmployee(int id) throws SQLException {

String sql = "SELECT \* FROM Employee WHERE id = ?";

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

PreparedStatement pstmt = conn.prepareStatement(sql)) {

pstmt.setInt(1, id);

ResultSet rs = pstmt.executeQuery();

if (rs.next()) {

System.out.println("ID: " + rs.getInt("id"));

System.out.println("Name: " + rs.getString("name"));

System.out.println("Department: " + rs.getString("department"));

} else {

System.out.println("Employee not found");

}

}

}

public void updateEmployee(Employee emp) throws SQLException {

String sql = "UPDATE Employee SET name = ?, department = ? WHERE id = ?";

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

PreparedStatement pstmt = conn.prepareStatement(sql)) {

pstmt.setString(1, emp.getName());

pstmt.setString(2, emp.getDepartment());

pstmt.setInt(3, emp.getId());

int updated = pstmt.executeUpdate();

System.out.println(updated + " row(s) updated");

}

}

public void deleteEmployee(int id) throws SQLException {

String sql = "DELETE FROM Employee WHERE id = ?";

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

PreparedStatement pstmt = conn.prepareStatement(sql)) {

pstmt.setInt(1, id);

int deleted = pstmt.executeUpdate();

System.out.println(deleted + " row(s) deleted");

}

}

}

**Employee.java**

public class Employee {

    private int id;

    private String name;

    private String department;

    public Employee() {}

    public Employee(int id, String name, String department) {

        this.id = id;

        this.name = name;

        this.department = department;

    }

    public int getId() { return id; }

    public void setId(int id) { this.id = id; }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public String getDepartment() { return department; }

    public void setDepartment(String department) { this.department = department; }

}

**Main.java**

public class Main {

public static void main(String[] args) {

EmployeeJDBC empJDBC = new EmployeeJDBC();

try {

empJDBC.createTable();

Employee emp1 = new Employee(1, "Alia", "HR");

empJDBC.addEmployee(emp1);

empJDBC.getEmployee(1);

emp1.setName("Alia");

emp1.setDepartment("Finance");

empJDBC.updateEmployee(emp1);

empJDBC.getEmployee(1);

empJDBC.deleteEmployee(1);

empJDBC.getEmployee(1);

} catch (Exception e) {

e.printStackTrace(); } }}