JAVA ASSIGNMENT



**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**

It is not considered as a pure function.

Function depends on instance variable which is outside the method so if rate changes for same input(amount) it gives different result.

public class TaxUtil {

   public double calculateTax(double amount, double rate) {

        return amount \* rate;

    }

}

**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)**

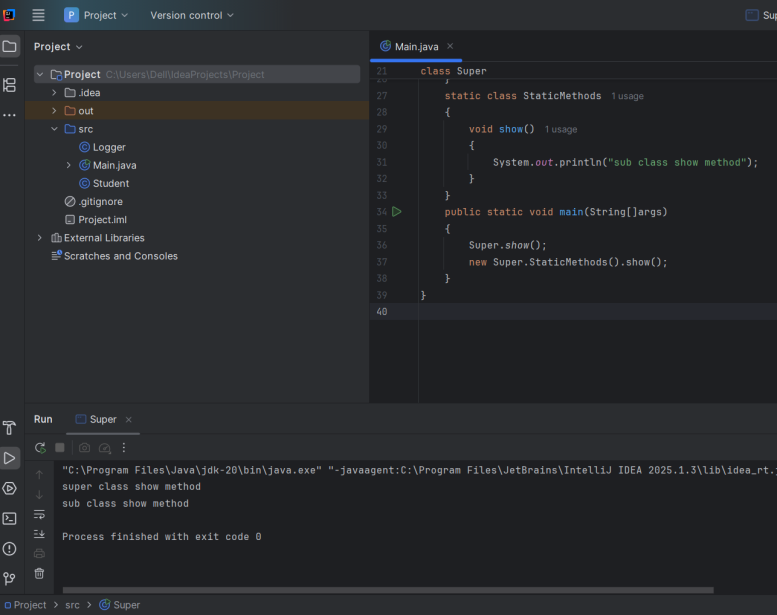
**{**

**Super.show();**

**new Super.StaticMethods().show();**

**}**

**}**



**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)**

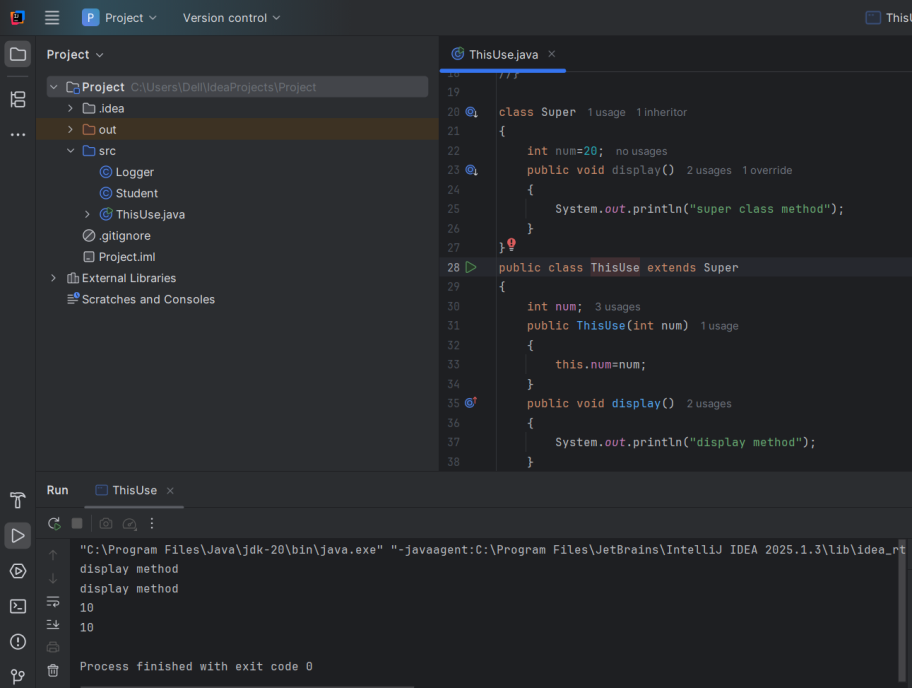
**{**

**ThisUse o=new ThisUse(10);**

**o.show();**

**}**

**}**



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

Singleton design pattern ensures that only one instance of the class is created and shared globally.

public class Singleton {

private static Singleton instance;

private Singleton() {

System.out.printf("Logger is created!\n");

}

public static Singleton getInstance() {

if (instance == null) {

instance = new Singleton();

}

return instance;

}

public void log(String message) {

System.out.println("Log:"+message);

}

}

public class Main {

public static void main(String[] args) {

Singleton log1= Singleton. getInstance();

log1.log("log1 created!");

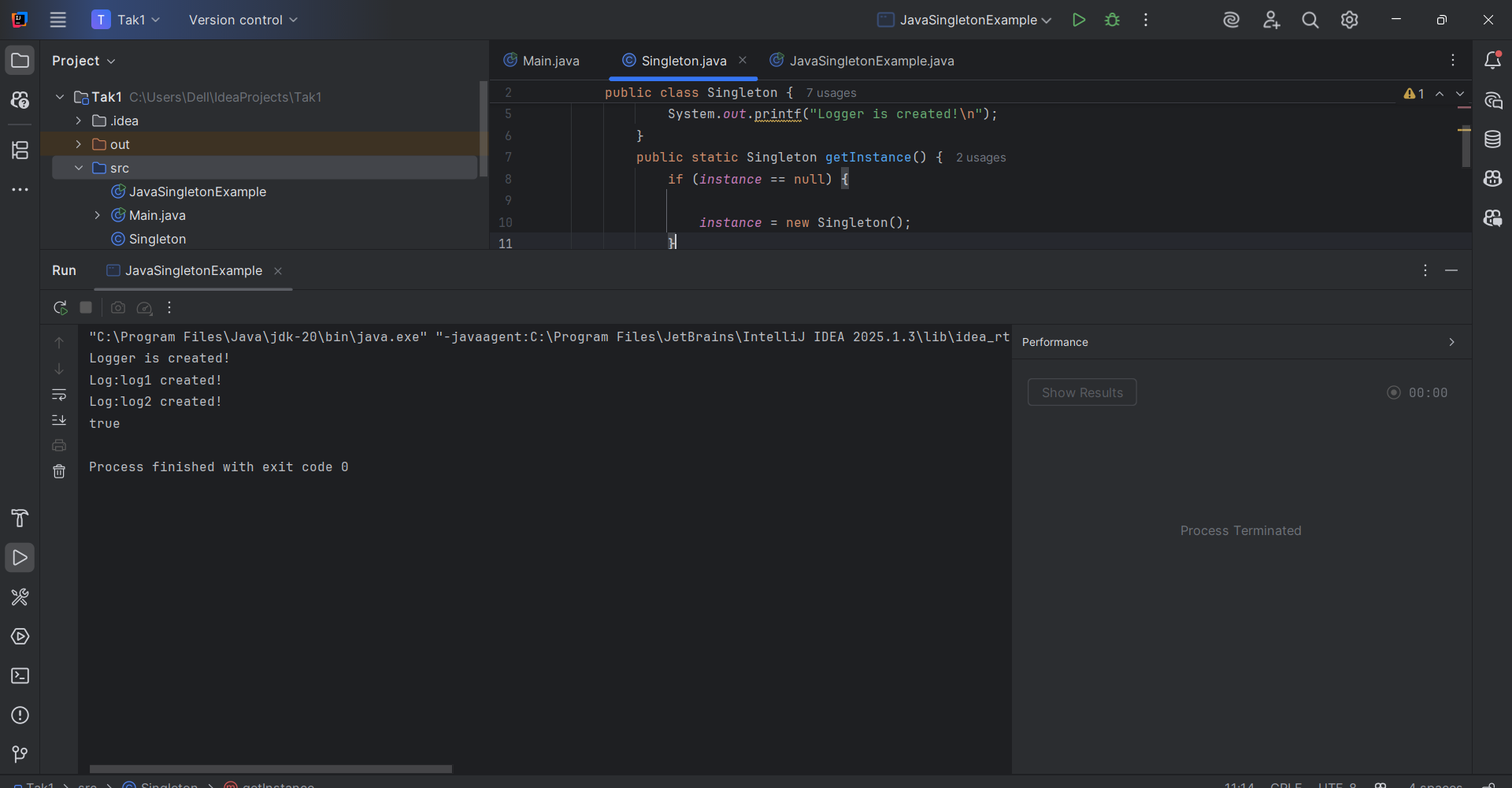
Singleton log2= Singleton.getInstance();

log2.log("log2 created!");

System.out.println(log1==log2);

}

}



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

To make sure the class is encapsulated we make the variables private and getters and setters as public.

public class Student {

private String name;

private int age;

public String getName(){

return name;

}

public void setName(String name){

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

if(age>0 && age<100){

this.age = age;

}

else{

System.out.println("Invalid Age");

}

}

}

public class Main {

public static void main(String[] args) {

Student student = new Student();

student.setAge(20);

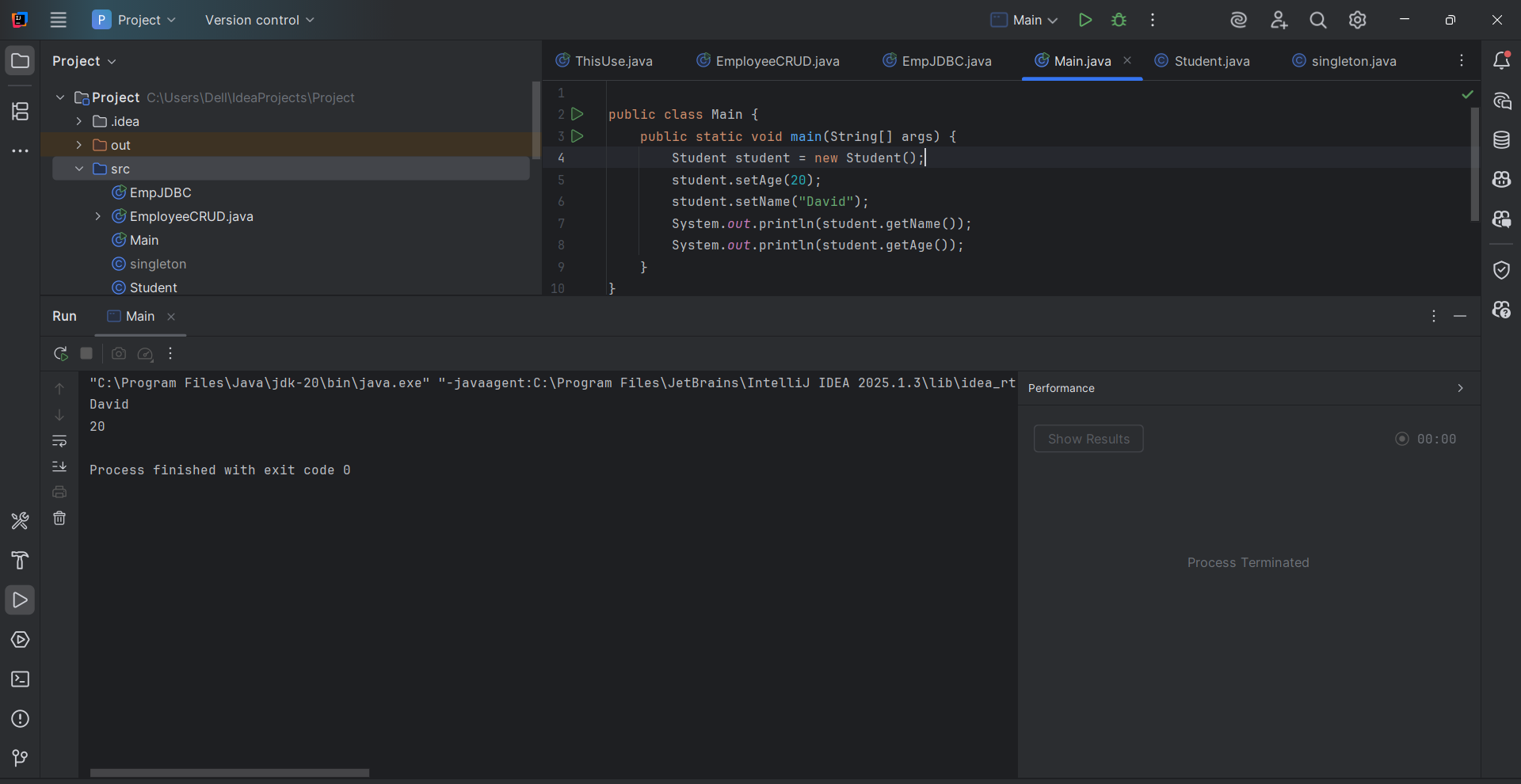
student.setName("David");

System.out.println(student.getName());

System.out.println(student.getAge());

}

}



**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;**

**}**

class Employee {

private int id;

private String name;

private String department;

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

this.id = id;

this.name = name;

this.department = department;

}

public int getId() { return id; }

public String getName() { return name; }

public String getDepartment() { return department; }

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

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

@Override

public String toString() {

return id + " | " + name + " | " + department;

}

}

class EmployeeCRUD {

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

//Create

public void addEmployee(Employee emp) {

employees.add(emp);

}

// Read

public Employee getEmployee(int id) {

for (Employee emp : employees) {

if (emp.getId() == id) return emp;

}

return null;

}

// Update

public boolean updateEmployee(int id, String name, String department) {

Employee emp = getEmployee(id);

if (emp != null) {

emp.setName(name);

emp.setDepartment(department);

return true;

}

return false;

}

// Delete

public boolean deleteEmployee(int id) {

Employee emp = getEmployee(id);

return emp != null && employees.remove(emp);

}

// DisplayAll

public void displayAll() {

for (Employee emp : employees) {

System.out.println(emp);

}

}

}

public class Main {

public static void main(String[] args) {

EmployeeCRUD crud = new EmployeeCRUD();

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

crud.addEmployee(new Employee(2, "Sita", "Marketing"));

System.out.println("All Employees:");

crud.displayAll();

System.out.println("\nUpdating Employee with ID 2:");

boolean updated = crud.updateEmployee(2, "Bob", "DevOps");

System.out.println("Update successful: " + updated);

crud.displayAll();

System.out.println("\nDeleting Employee with ID 1:");

boolean deleted = crud.deleteEmployee(1);

System.out.println("Delete successful: " + deleted);

crud.displayAll();

System.out.println("\nGetting Employee with ID 3:");

Employee emp = crud.getEmployee(3);

if (emp != null) {

System.out.println("Found: " + emp);

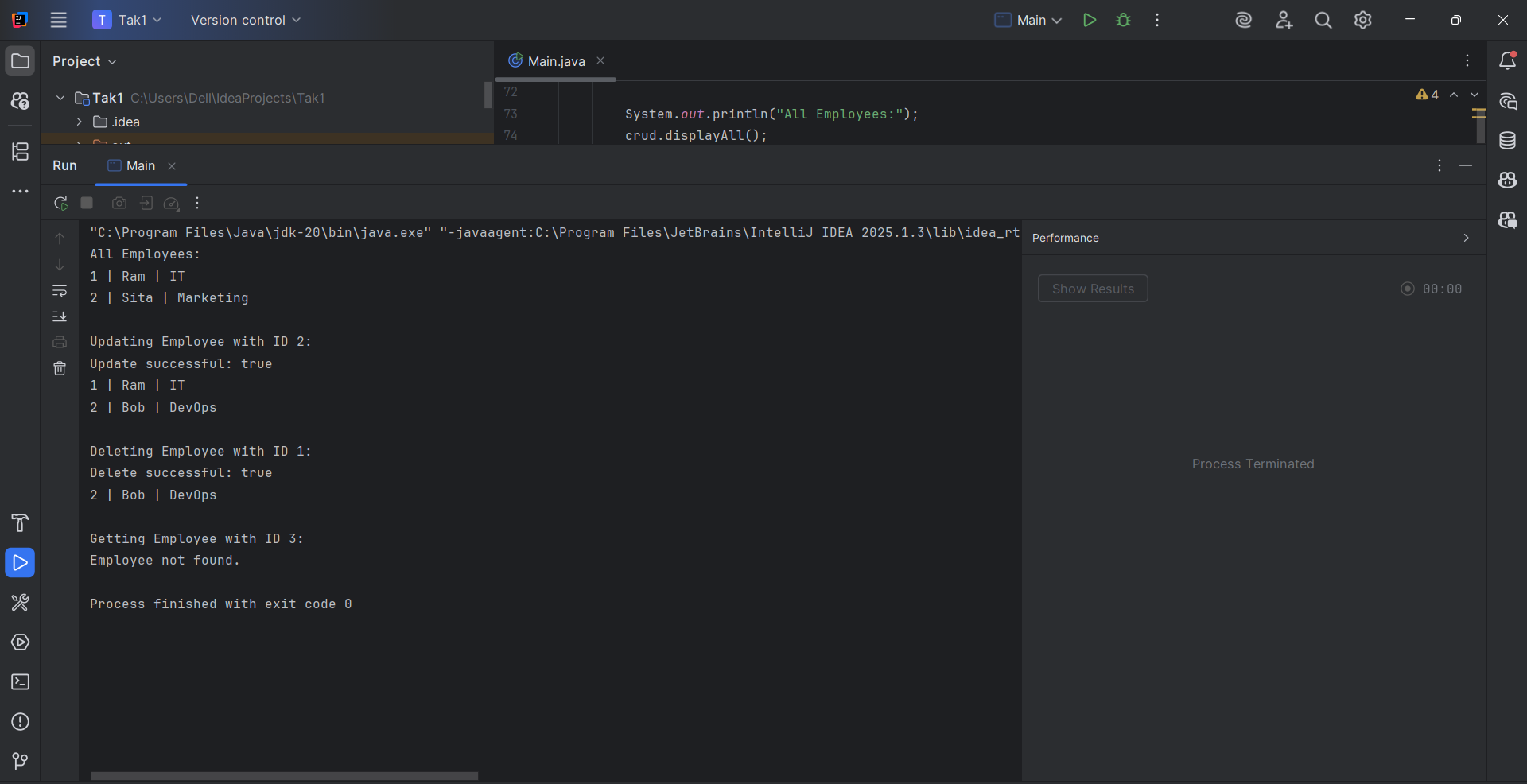
} else {

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

}

}

}



**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;**

**}**

import java.sql.\*;

public class EmpJDBC {

static final String DB\_URL = "jdbc:mysql://localhost:3306/company";

static final String USER = "root";

static final String PASS = "abcde";

public void addEmployee(int id, String name, String dept) {

String query = "INSERT INTO employee (id, name, department) VALUES (?, ?, ?)";

try (Connection conn = DriverManager.getConnection(DB\_URL, USER, PASS);

PreparedStatement stmt = conn.prepareStatement(query)) {

stmt.setInt(1, id);

stmt.setString(2, name);

stmt.setString(3, dept);

stmt.executeUpdate();

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

} catch (SQLException e) {

System.err.println("Error adding employee: " + e.getMessage());

System.err.println("SQLState: " + e.getSQLState());

}

}

public void displayEmployees() {

String query = "SELECT id, name, department FROM employee";

try (Connection conn = DriverManager.getConnection(DB\_URL, USER, PASS);

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(query)) {

while (rs.next()) {

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

", Name: " + rs.getString("name") +

", Dept: " + rs.getString("department"));

}

} catch (SQLException e) {

System.err.println("Error displaying employees: " + e.getMessage());

System.err.println("SQLState: " + e.getSQLState());

}

}

public void deleteEmployee(int id) {

String query = "DELETE FROM employee WHERE id=?";

try (Connection conn = DriverManager.getConnection(DB\_URL, USER, PASS);

PreparedStatement stmt = conn.prepareStatement(query)) {

stmt.setInt(1, id);

int rows = stmt.executeUpdate();

if (rows > 0) {

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

} else {

System.out.println("Employee with ID " + id + " not found.");

}

} catch (SQLException e) {

System.err.println("Error deleting employee: " + e.getMessage());

System.err.println("SQLState: " + e.getSQLState());

}

}

public void updateEmployee(int id, String name, String dept) {

String query = "UPDATE employee SET name=?, department=? WHERE id=?";

try (Connection conn = DriverManager.getConnection(DB\_URL, USER, PASS);

PreparedStatement stmt = conn.prepareStatement(query)) {

stmt.setString(1, name);

stmt.setString(2, dept);

stmt.setInt(3, id);

int rows = stmt.executeUpdate();

if (rows > 0) {

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

} else {

System.out.println("Employee with ID " + id + " not found.");

}

} catch (SQLException e) {

System.err.println("Error updating employee: " + e.getMessage());

System.err.println("SQLState: " + e.getSQLState());

}

}

public static void main(String[] args) {

try {

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

} catch (ClassNotFoundException e) {

System.err.println("MySQL JDBC Driver not found.");

return;

}

EmpJDBC obj = new EmpJDBC();

obj.addEmployee(1, "Raju", "Finance");

obj.addEmployee(2, "Ram", "HR");

obj.addEmployee(3, "Sita", "IT");

System.out.println("Employees:");

obj.displayEmployees();

obj.updateEmployee(1, "Roopa", "Marketing");

System.out.println("Employees after update:");

obj.displayEmployees();

obj.deleteEmployee(1);

System.out.println("Employees after deletion:");

obj.displayEmployees();

}

}

o/p:

