## Week 1 | Assignment 2 | Core Java | By: Sanya Dureja

### [Github Link](https://github.com/sanyadureja/sanya-dureja-assignments/tree/feature-java/Week%201/Assignment%202)

**Q1.**

Given:

public class TaxUtil {

double rate = 0.15;

public double calculateTax(double amount) {

return amount \* rate;

}

}

1. Would you consider the method calculateTax() a 'pure function'? Why or why not?
2. If you claim the method is NOT a pure function, please suggest a way to make it pure.

**Ans 1.**

1. No, method calculateTax() is not a pure function because it depends on the instance variable rate, which is external to the method and can change, breaking the pure function rules.
2. Way to make the method calculateTax() pure is as follows:  
   - Make rate a local variable or pass it as a parameter.

**Code - Modified: Pure version**public class TaxUtil {

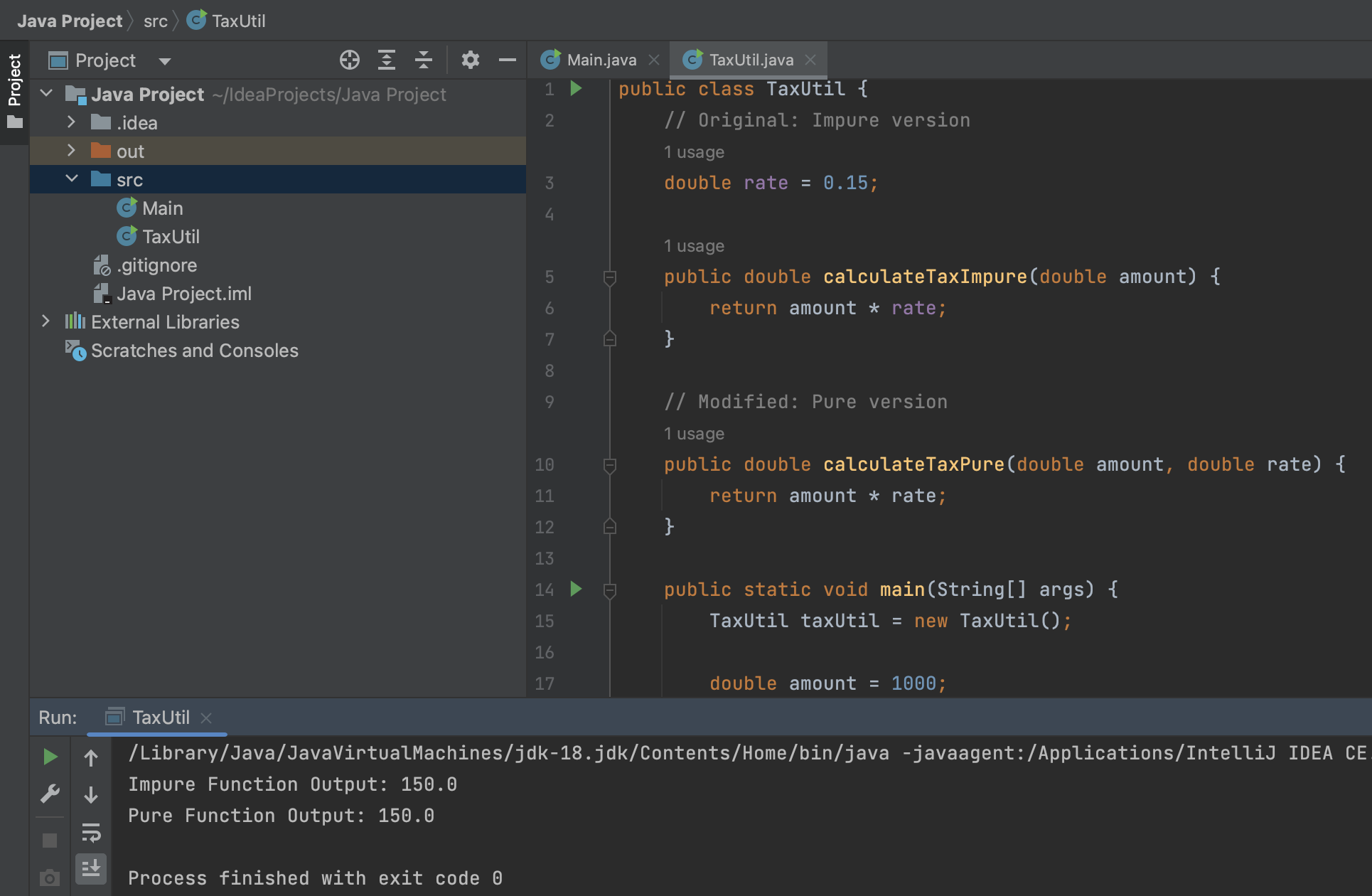
public double calculateTax(double amount, double rate) {

return amount \* rate;

}

}

**O/p**

****

**Q2.**

What will be the output for the 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)

{

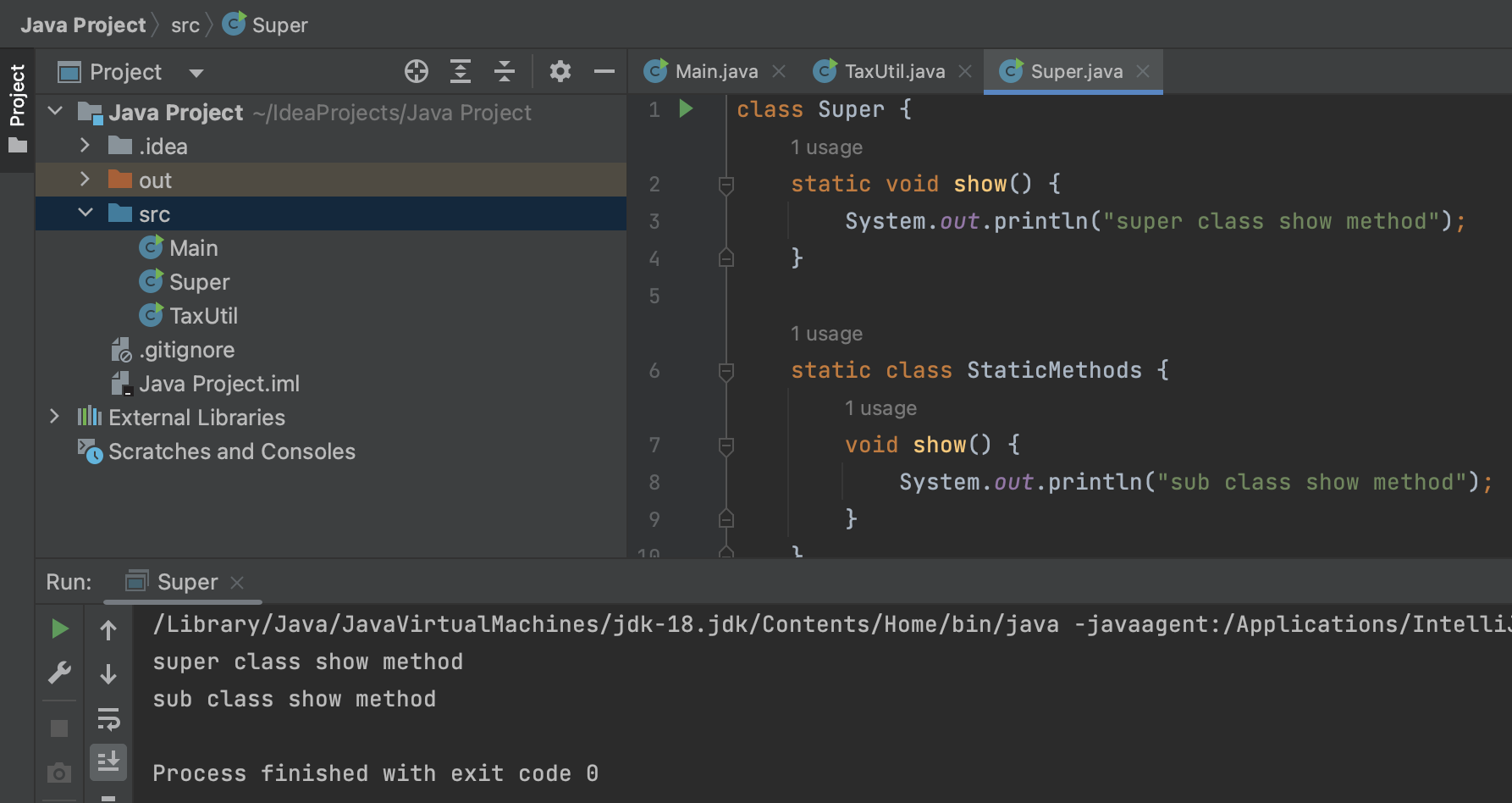
Super.show();

new Super.StaticMethods().show();

}

}

**Ans 2.**

**O/p**

**Q3.**

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);

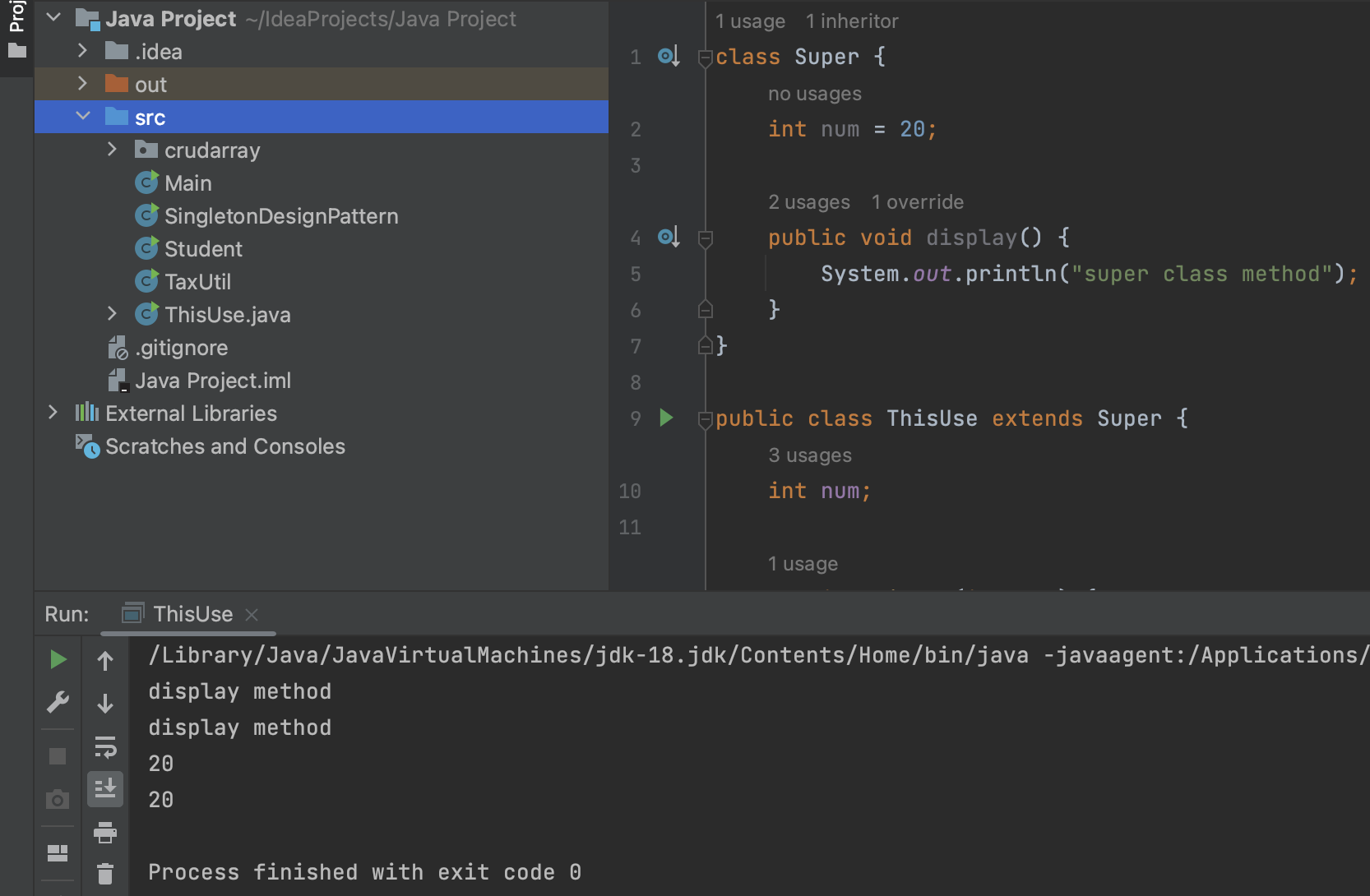
o.show();

}

}

**Ans 3.**

**O/p**

****

**Q4.**

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

**Ans 4.**

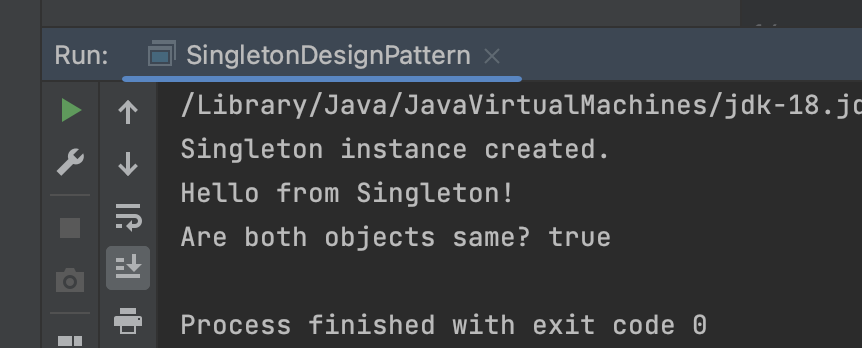
Singleton Design Pattern

* The Singleton Design Pattern ensures that a class has only one instance and provides a global point of access to it.
* It is commonly used when exactly one object is needed to coordinate actions across a system.

**Coding Example**



**O/p**

****

**Q5.** How do we make sure a class is encapsulated? Explain with a coding example.

**Ans 5.**

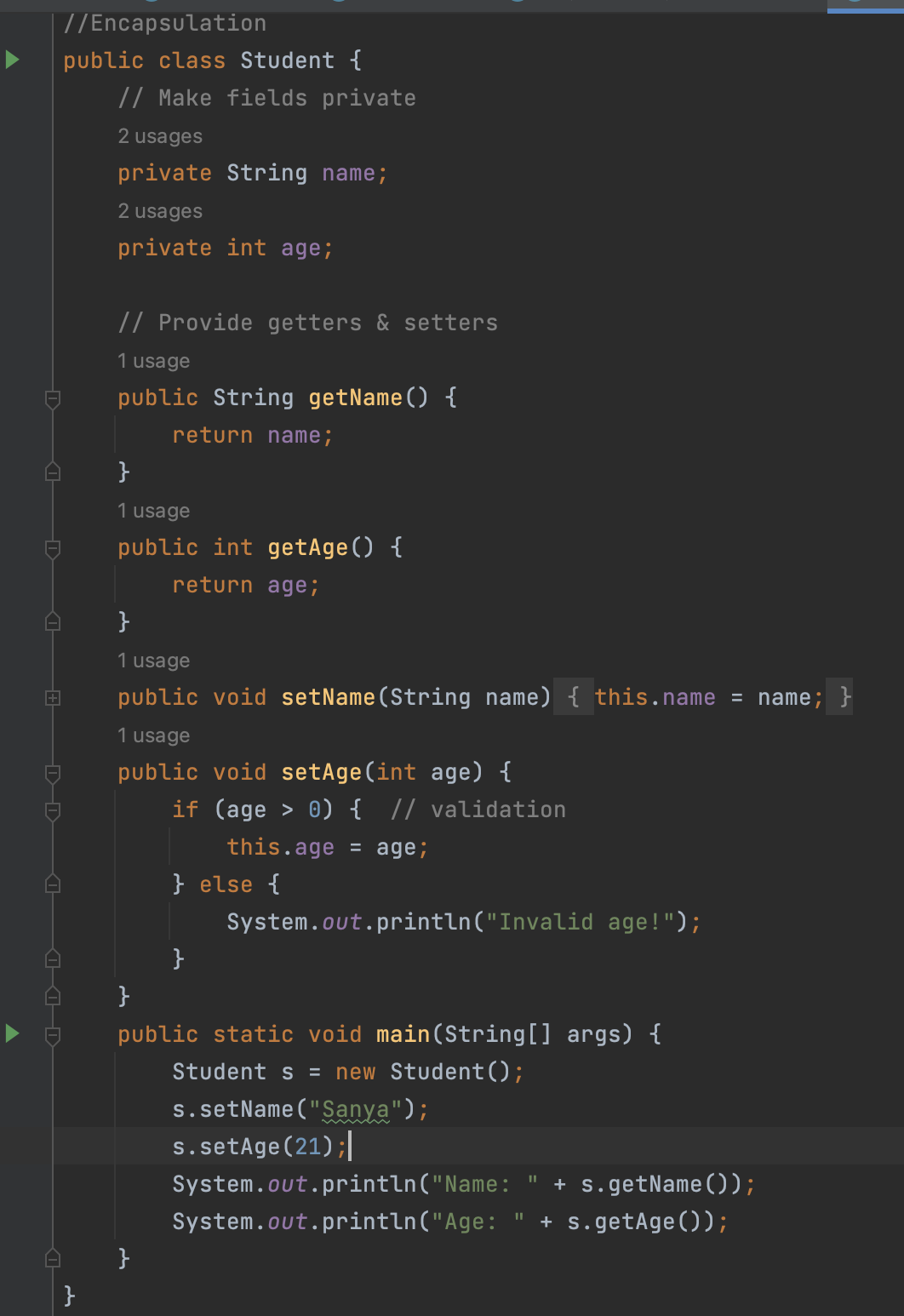
**Encapsulation**

* It is one of the fundamental principles of OOP (Object-Oriented Programming).
* It means hiding the internal details of an object and exposing only what’s necessary using methods (getters/setters).
* It helps in data protection, control, and modularity.

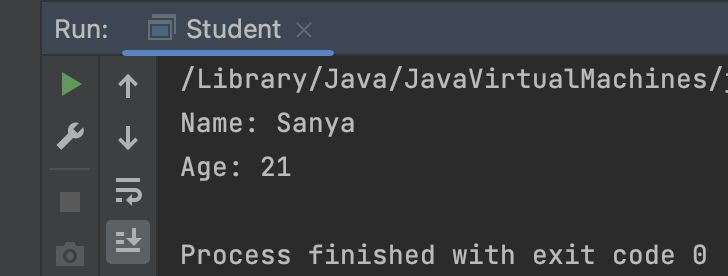
**Steps to ensure a class is encapsulated?**

* Make all data members private (access modifier).
* Provide public getter and setter methods to access/update private fields.
* Optionally, add validation in setters to control changes.

**Coding Example**



**O/p**



**Q6.**

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

class Employee{

private int id;

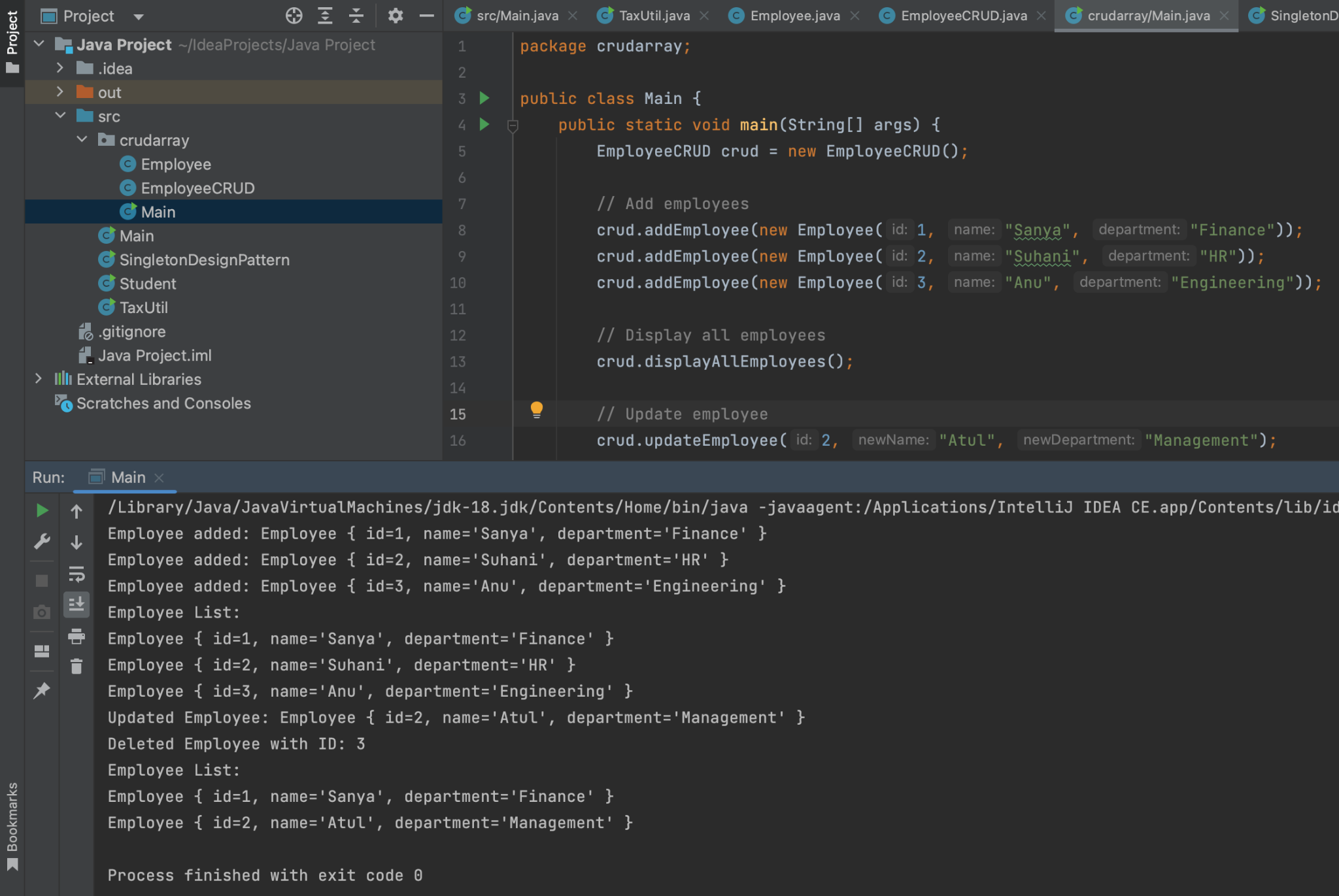
private String name;

private String department;

}

**Ans 6.**

**O/p**

****