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

[Github Link]

Q1.

Given:

```
public class TaxUtil {
  double rate = 0.15;

  public double calculateTax(double amount) {
     return amount * rate;
  }
}
```

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

Ans 1.

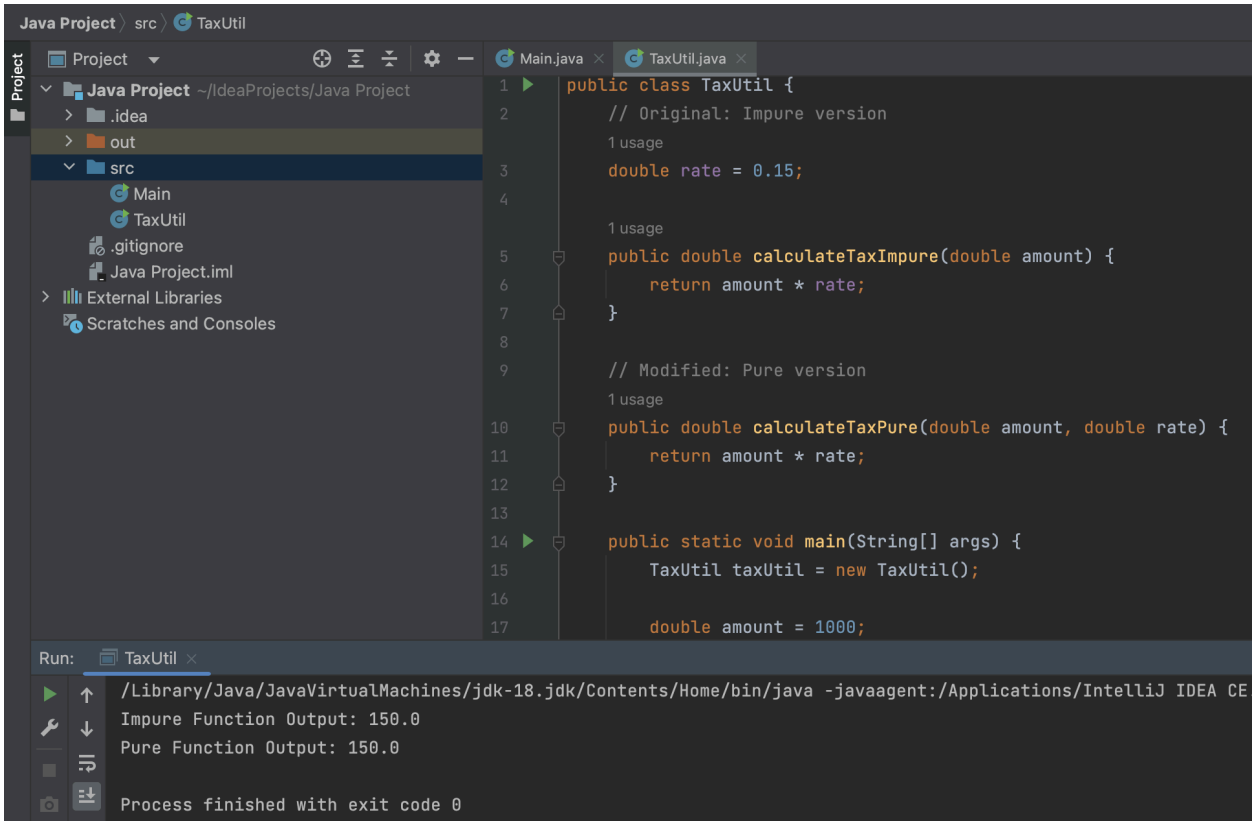
a) No, method calculateTax() is not a pure function because it depends on the instance variable rate, which is external to the method and can change, breaking the pure function rules.

b) 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


Java Project 〉 src 〉 TaxUtil
Project
Main.java
TaxUtil.java
Java Project ~/IdeaProjects/Java Project
.idea
out
src
Main
TaxUtil
.gitignore
Java Project.iml
External Libraries
Scratches and Consoles
public class TaxUtil {
// Original: Impure version
1 usage
double rate = 0.15;
1 usage
public double calculateTaxImpure(double amount) {
return amount * rate;
}
// Modified: Pure version
1 usage
public double calculateTaxPure(double amount, double rate) {
return amount * rate;
}
public static void main(String[] args) {
TaxUtil taxUtil = new TaxUtil();
double amount = 1000;
Run:
TaxUtil
/Library/Java/JavaVirtualMachines/jdk-18.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE
Impure Function Output: 150.0
Pure Function Output: 150.0
Process finished with exit code 0

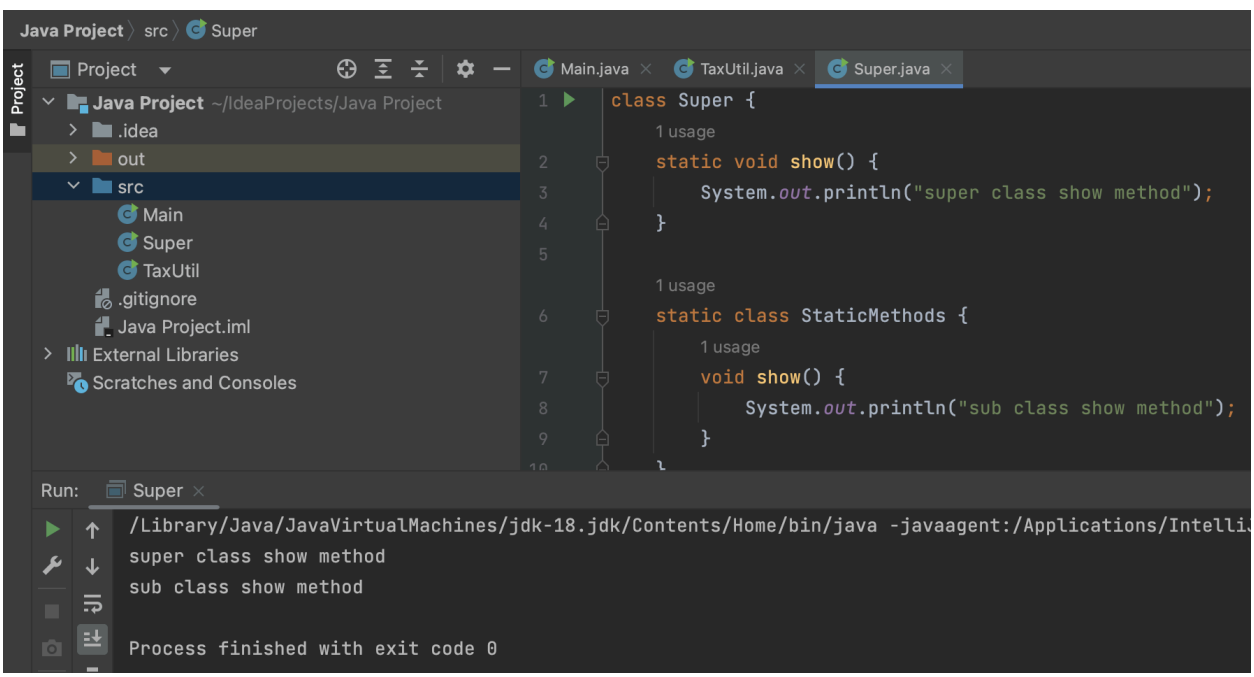
Q2.

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

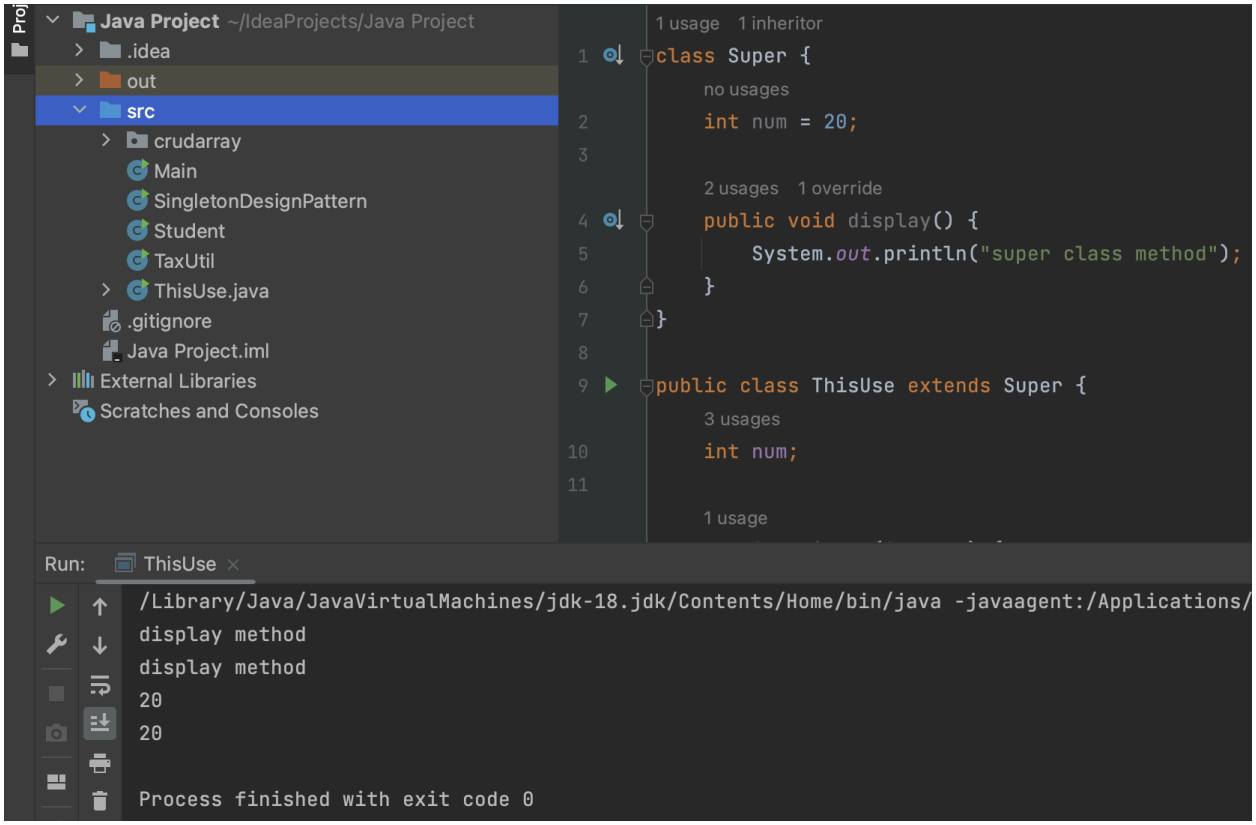


Q3.

```
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


Java Project ~/IdeaProjects/Java Project
.idea
out
src
crudarray
Main
SingletonDesignPattern
Student
TaxUtil
ThisUse.java
.gitignore
Java Project.iml
External Libraries
Scratches and Consoles
1 usage 1 inheritor
class Super {
no usages
int num = 20;
2 usages 1 override
public void display() {
System.out.println("super class method");
}
}
public class ThisUse extends Super {
3 usages
int num;
1 usage
Run: ThisUse
/Library/Java/JavaVirtualMachines/jdk-18.jdk/Contents/Home/bin/java -javaagent:/Applications/
display method
display method
20
20
Process finished with exit code 0

Q4.

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

Ans 4.

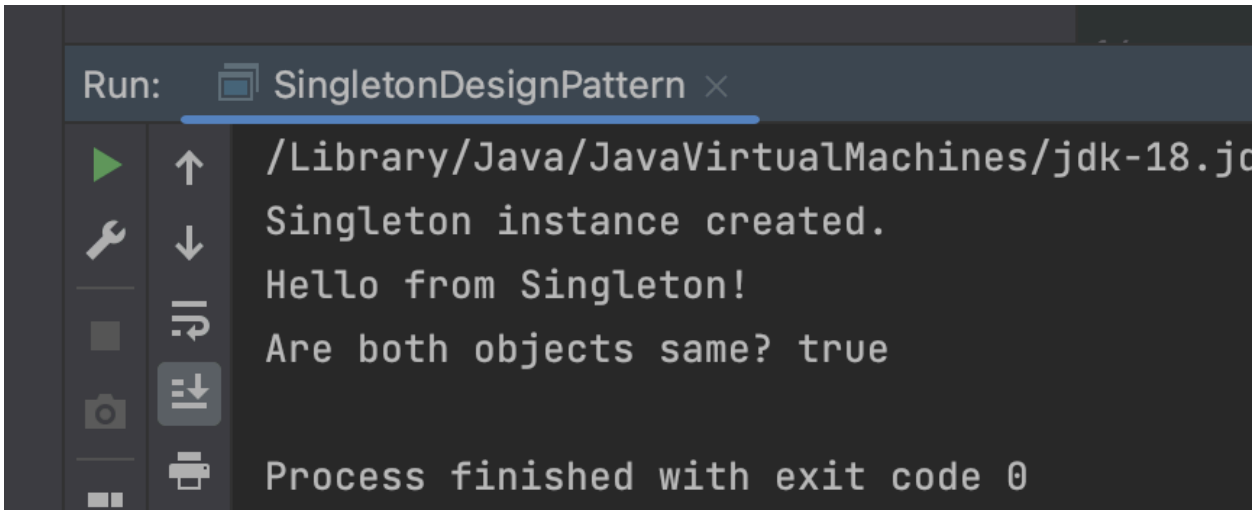
Singleton Design Pattern

- 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

```java
public class SingletonDesignPattern {
    // Private static variable of the same class
    3 usages
    private static SingletonDesignPattern instance;
    1 usage
    private SingletonDesignPattern() {
        System.out.println("Singleton instance created.");
    }
    // Public static method to provide access to the instance
    2 usages
    public static SingletonDesignPattern getInstance() {
        if (instance == null) {
            instance = new SingletonDesignPattern(); // Lazy initialization
        }
        return instance;
    }

    1 usage
    public void showMessage() {
        System.out.println("Hello from Singleton!");
    }

    public static void main(String[] args) {
        // singleton instance
        SingletonDesignPattern obj1 = SingletonDesignPattern.getInstance();
        SingletonDesignPattern obj2 = SingletonDesignPattern.getInstance();

        obj1.showMessage();

        // Verifying both objects
        System.out.println("Are both objects same? " + (obj1 == obj2));
    }
}
```

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

```java
//Encapsulation
public class Student {
    // Make fields private
    2 usages
    private String name;
    2 usages
    private int age;

    // Provide getters & setters
    1 usage
    public String getName() {
        return name;
    }
    1 usage
    public int getAge() {
        return age;
    }
    1 usage
    public void setName(String name) { this.name = name; }
    1 usage
    public void setAge(int age) {
        if (age > 0) {  // validation
            this.age = age;
        } else {
            System.out.println("Invalid age!");
        }
    }
    public static void main(String[] args) {
        Student s = new Student();
        s.setName("Sanya");
        s.setAge(21);
        System.out.println("Name: " + s.getName());
        System.out.println("Age: " + s.getAge());
    }
}
```

O/p

```
Run:  Student
/Library/Java/JavaVirtualMachines/
Name: Sanya
Age: 21

Process finished with exit code 0
```

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