

Please make sure to save/push all your code in the branch feature-java created in the previous week assignment as part of your github repo rg-assignments

Please share your output screenshots in the assignment document along with the github link for each question. Provide an explanation wherever possible as part of your response :-)

1)

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.

The method calculateTax() is not a pure function.

A pure function must satisfy two conditions:

- Given the same inputs, it always returns the same output.
- It has no side effects (does not modify global state, perform I/O, etc.).

The method uses the instance variable rate, which can be changed outside the method (e.g., via a setter or direct access). Therefore, the output of calculateTax() depends not only on the input amount, but also on the mutable internal state rate. This violates the first rule of pure functions: the same input must always yield the same output.

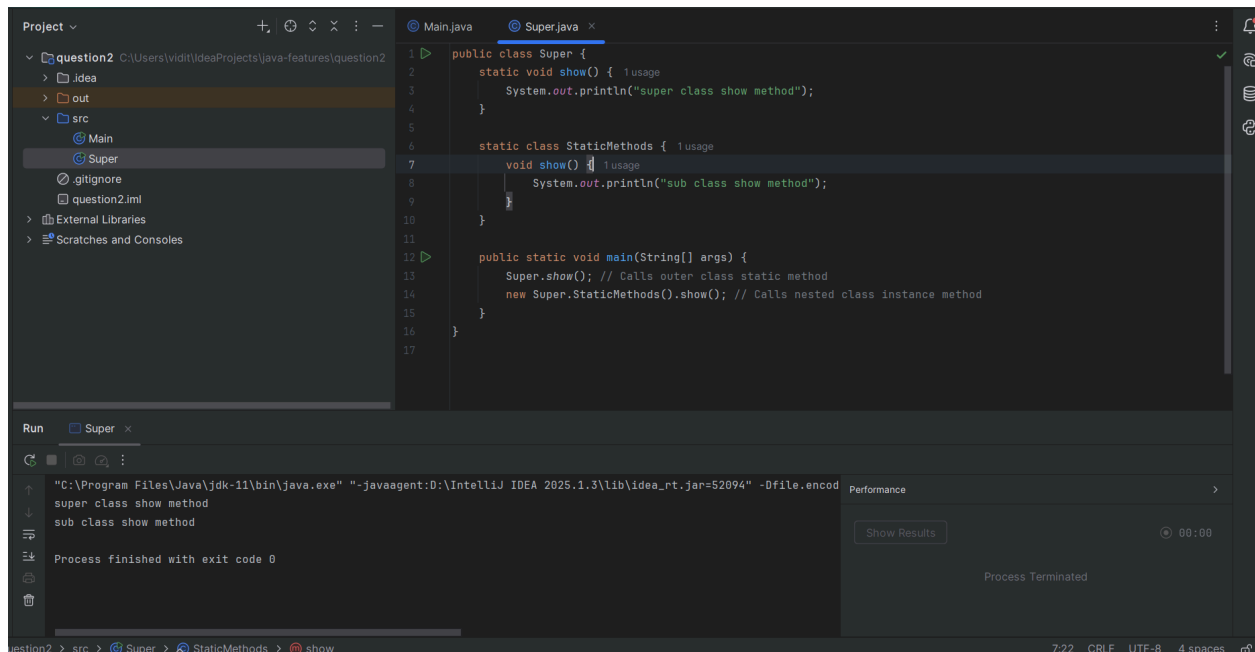
If the tax rate is fixed and should never change then it will be a pure function.

```
public class TaxUtil {  
    private static final double RATE = 0.15;  
  
    public static double calculateTax(double amount) {  
        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();
}
}
```

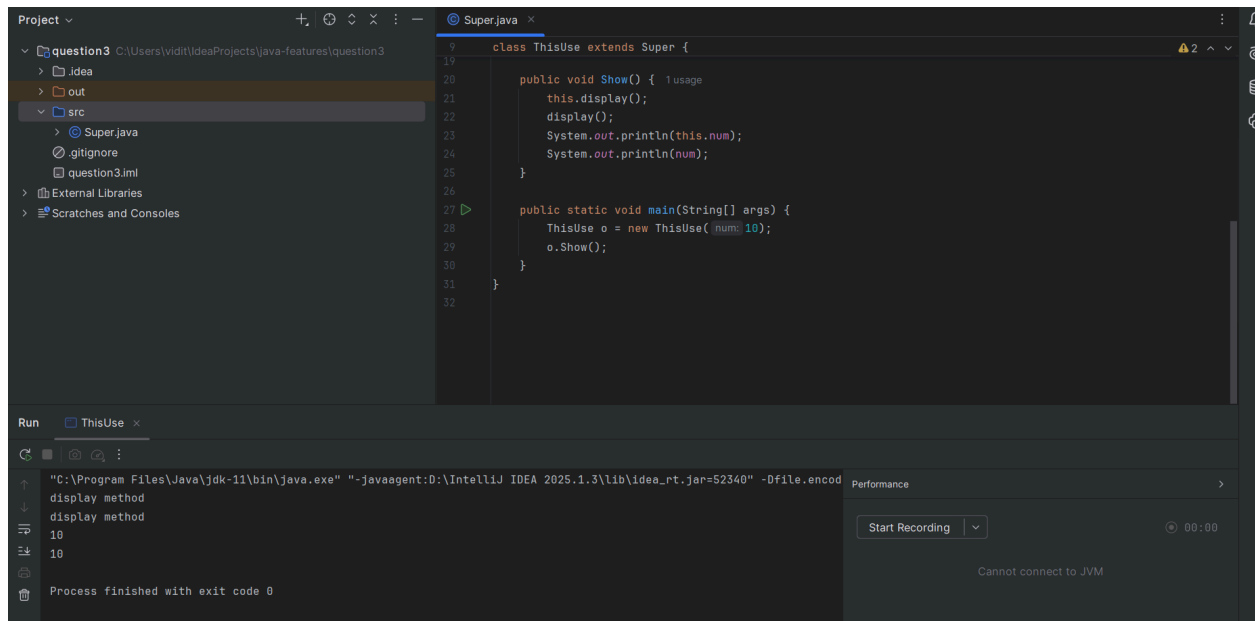


3)

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.

The Singleton Pattern ensures that a class has only one instance and provides a global point of access to it. It is mainly used to limit object creation, ensuring shared access to a resource across the applications. This class has a private constructor (so new objects can't be created from outside).

The screenshot displays the IntelliJ IDEA IDE with a project named 'question4'. The 'src' directory contains a file named 'Singleton.java'. The code in 'Singleton.java' implements the Singleton pattern:

```

1 public class Singleton {
2     // private constructor
3     private Singleton() {}
4
5     // public static method to get the instance
6     public static Singleton getInstance() { 2 usages
7         if (singleInstance == null) {
8             singleInstance = new Singleton();
9         }
10        return singleInstance;
11    }
12
13    public void showMessage() { 1 usage
14        System.out.println("Hello from Singleton!");
15    }
16
17    public static void main(String[] args) {
18        // get Singleton instances
19        Singleton s1 = Singleton.getInstance();
20        Singleton s2 = Singleton.getInstance();
21    }
22 }

```

The 'Run' tab at the bottom shows the execution output:

```

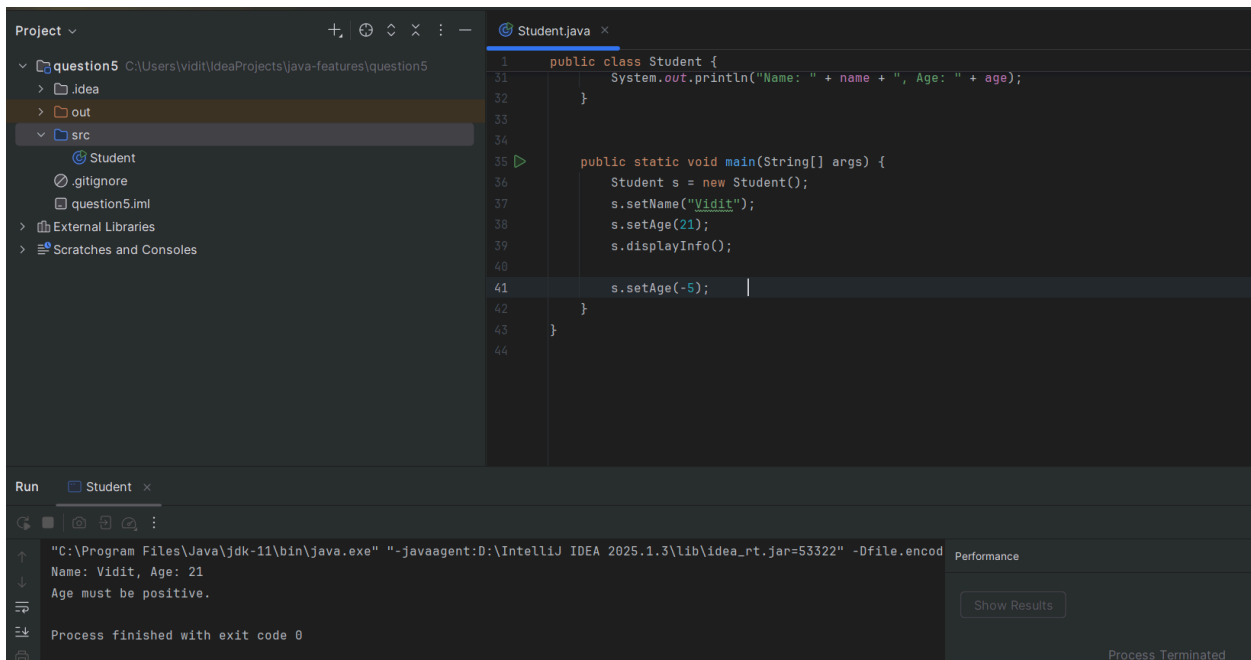
Singleton instance created
Hello from Singleton!
true
Process finished with exit code 0

```

The 'Performance' tab shows 'Process Terminated'.

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

It is object oriented programming principle where Data (fields) and code (methods) are wrapped together as a single unit, and direct access to some components is restricted.



```
1 public class Student {
2     private String name;
3     private int age;
4
5     public void setName(String name) {
6         this.name = name;
7     }
8
9     public void setAge(int age) {
10        this.age = age;
11    }
12
13    public void displayInfo() {
14        System.out.println("Name: " + name + ", Age: " + age);
15    }
16
17    public static void main(String[] args) {
18        Student s = new Student();
19        s.setName("Vidit");
20        s.setAge(21);
21        s.displayInfo();
22
23        // s.setAge(-5);
24    }
25 }
```

Run Student

"C:\Program Files\Java\jdk-11\bin\java.exe" "-javaagent:D:\IntelliJ IDEA 2025.1.3\lib\idea_rt.jar=53322" -Dfile.encoding=UTF-8

Name: Vidit, Age: 21

Age must be positive.

Process finished with exit code 0

Show Results

Process Terminated

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;
}
```

```
"C:\Program Files\Java\jdk-11\bin\java.exe" "-javaagent:D:\IntelliJ IDEA 2025.1.3\lib\idea_rt.jar=53542" -Dfi
Employee added.
Employee added.

All Employees:
ID: 101, Name: Alice, Dept: HR
ID: 102, Name: Bob, Dept: Finance

Updating Bob's department to Marketing...
Employee updated.

All Employees After Update:
ID: 101, Name: Alice, Dept: HR
ID: 102, Name: Bob, Dept: Marketing

Deleting Employee with ID 101...
Employee removed.

All Employees After Deletion:
ID: 102, Name: Bob, Dept: Marketing
```

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;  
}
```

```
"C:\Program Files\Java\jdk-11\bin\java.exe" "-javaagent:D:\IntelliJ IDEA 2025.1.3\lib\idea_rt.jar=57159" -Dfile.encoding=UTF-8  
=== CREATE ===  
Employee [id=1, name=Alice, department=Engineering]  
Employee [id=2, name=Bob, department=HR]  
  
=== READ (single) ===  
Employee [id=1, name=Alice, department=Engineering]  
  
=== UPDATE ===  
Employee [id=1, name=Alice, department=Engineering]  
Employee [id=2, name=Bob, department=Finance]  
  
=== DELETE ===  
Employee [id=2, name=Bob, department=Finance]  
  
Process finished with exit code 0
```

Performance

Show Results