Core Java Assignment

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.

```
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.

Aans.) github link – https://github.com/Harshit-Raj-14/PayPal-RG-Assignment-Harshit-Raj/blob/main/Week%202/Java%20Core/TaxUtil.java

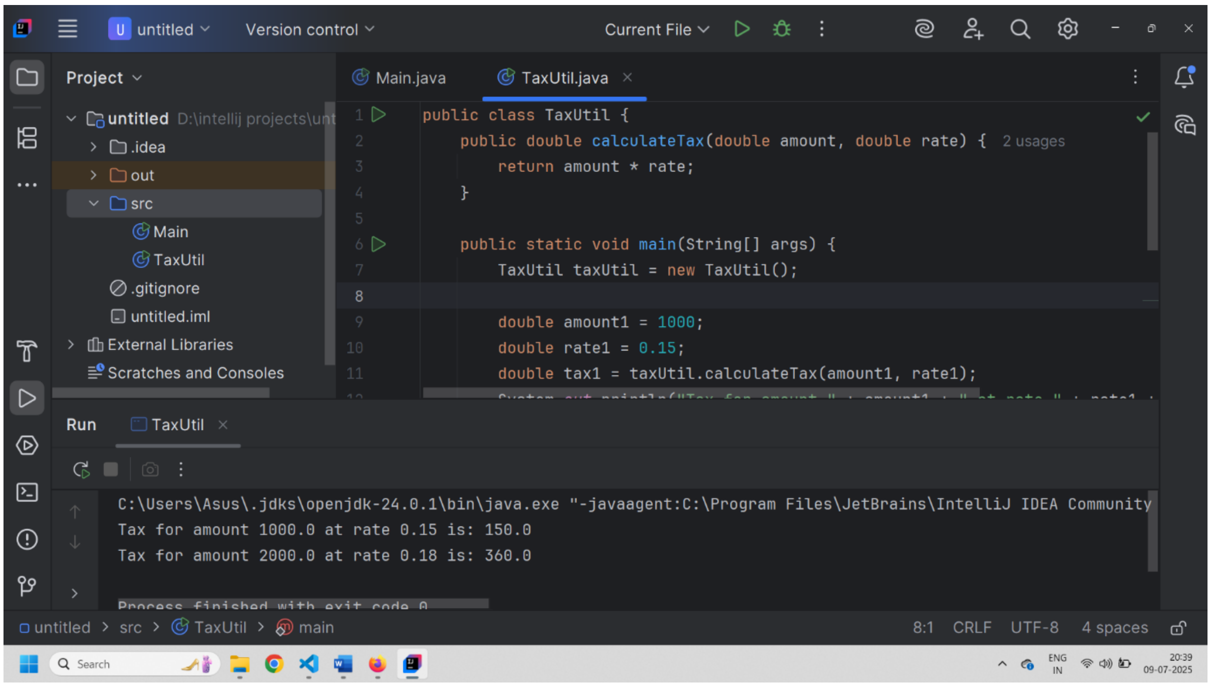
A *pure function* is a function that always returns the **same output for the same input**. And has **no side effects** (it doesn't change any state outside the function, like modifying a global variable or class property).

calculateTax() is not a pure function because it depends on the instance variable rate, which can **change** if another part of the code modifies it.

To make it pure: Pass rate as a parameter instead of using the instance variable.

Pure version:

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



```java
public class TaxUtil {
    public double calculateTax(double amount, double rate) {
        return amount * rate;
    }

    public static void main(String[] args) {
        TaxUtil taxUtil = new TaxUtil();

        double amount1 = 1000;
        double rate1 = 0.15;
        double tax1 = taxUtil.calculateTax(amount1, rate1);
```

```
C:\Users\Asus\.jdks\openjdk-24.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community
Tax for amount 1000.0 at rate 0.15 is: 150.0
Tax for amount 2000.0 at rate 0.18 is: 360.0

Process finished with exit code 0
```

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){

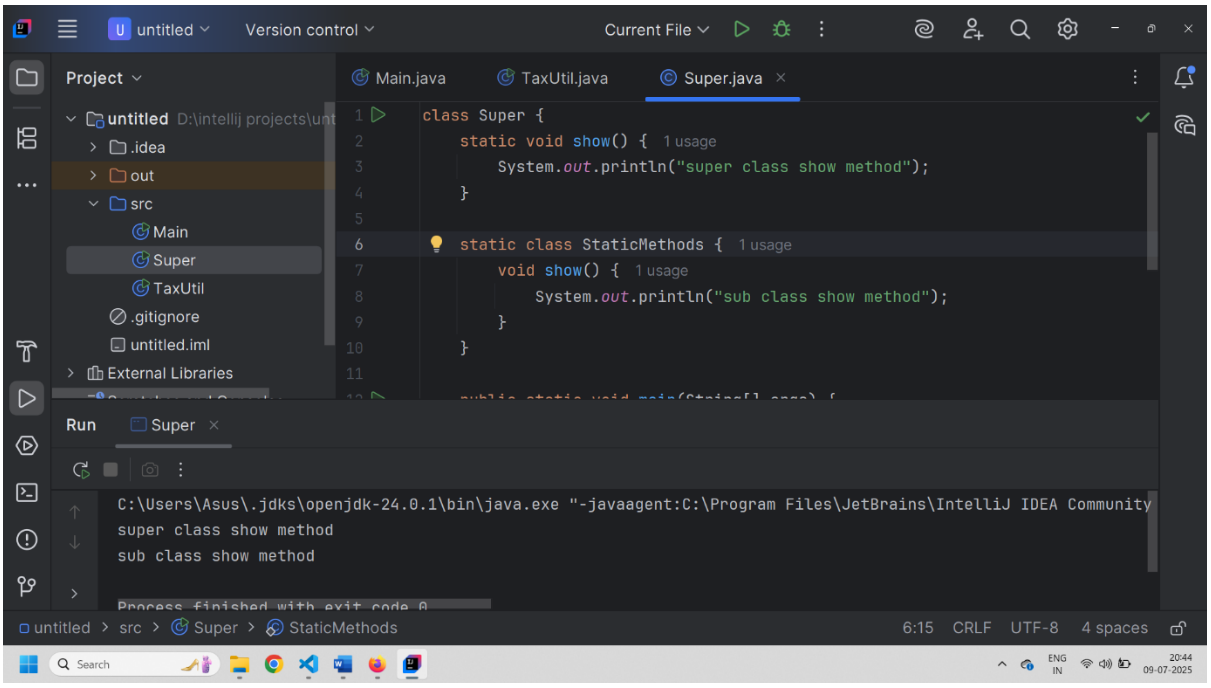
Super.show();

new Super.StaticMethods().show();

}

}

Github Link - https://github.com/Harshit-Raj-14/PayPal-RG-Assignment-Harshit-Raj/blob/main/Week%202/Java%20Core/Super.java



```
class Super {
    static void show() {  1 usage
        System.out.println("super class show method");
    }

    static class StaticMethods {  1 usage
        void show() {  1 usage
            System.out.println("sub class show method");
        }
    }

    public static void main(String[] args) {
```

```
C:\Users\Asus\.jdks\openjdk-24.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community
super class show method
sub class show method

Process finished with exit code 0
```

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;

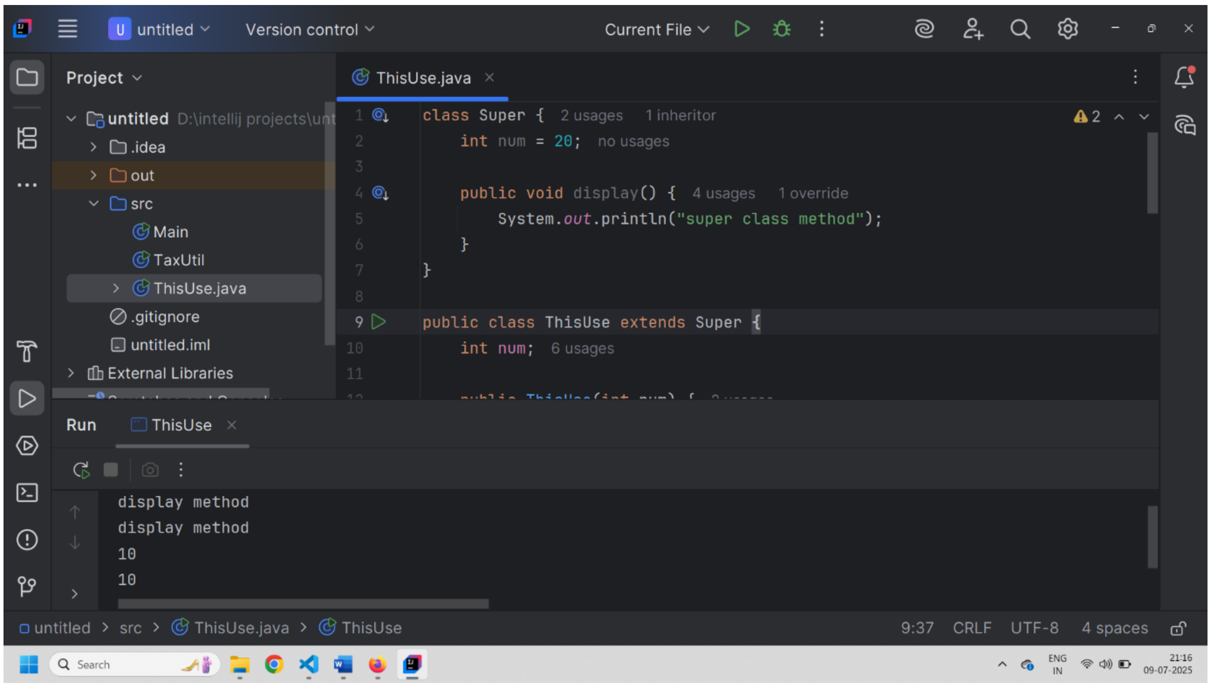
public ThisUse(int num){

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();

}

}
```

Github Link – https://github.com/Harshit-Raj-14/PayPal-RG-Assignment-Harshit-Raj/blob/main/Week%202/Java%20Core/ThisUse.java



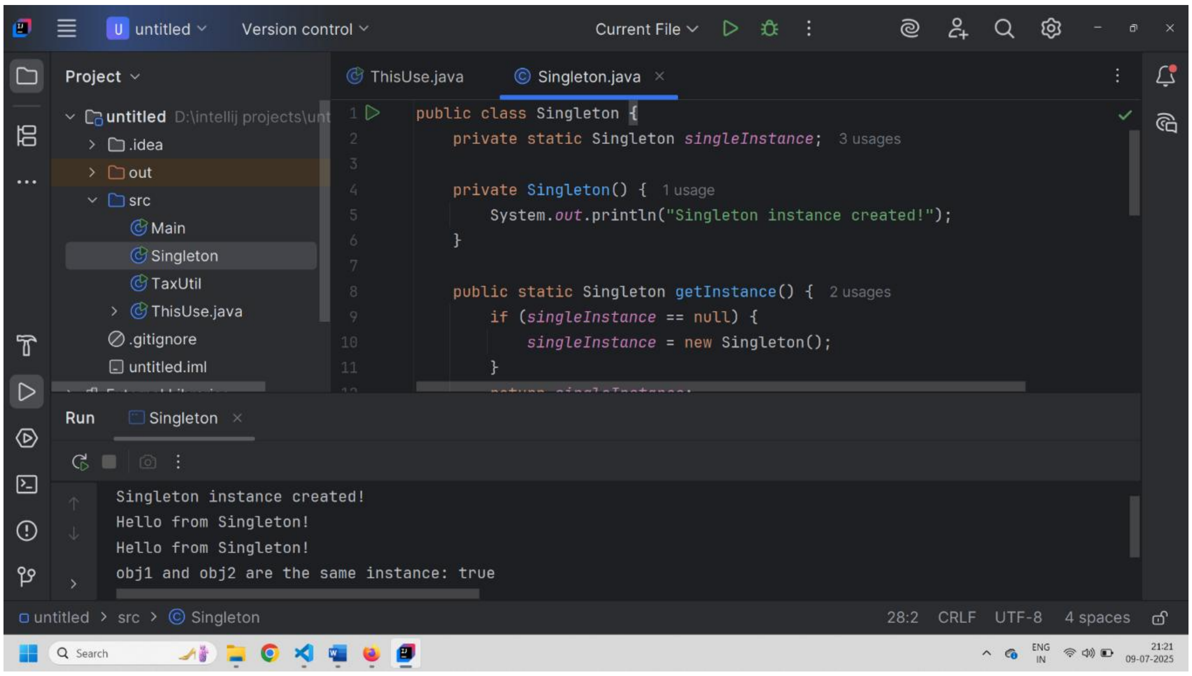
```
class Super {
    int num = 20;

    public void display() {
        System.out.println("super class method");
    }
}

public class ThisUse extends Super {
    int num;
```

```
display method
display method
10
10
```

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

Singleton is a creational design pattern that lets us ensure that a class has only one instance, while providing a global access point to this instance.

Github link - https://github.com/Harshit-Raj-14/PayPal-RG-Assignment-Harshit-Raj/blob/main/Week%202/Java%20Core/Singleton.java



```java
public class Singleton {
    private static Singleton singleInstance;  3 usages

    private Singleton() {  1 usage
        System.out.println("Singleton instance created!");
    }

    public static Singleton getInstance() {  2 usages
        if (singleInstance == null) {
            singleInstance = new Singleton();
        }
```

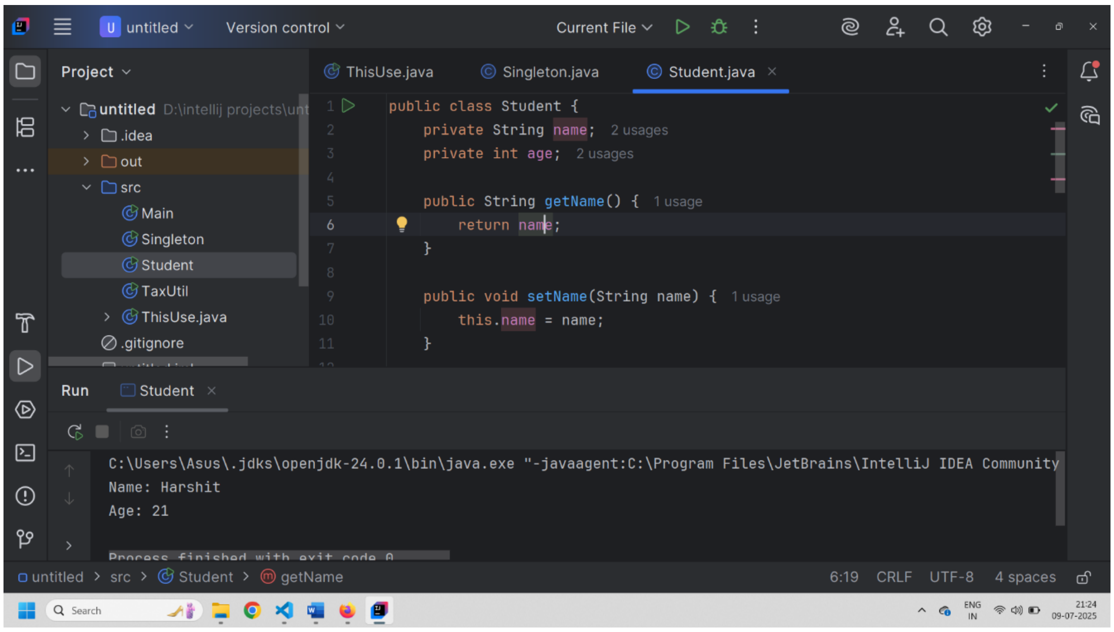
```
Singleton instance created!
Hello from Singleton!
Hello from Singleton!
obj1 and obj2 are the same instance: true
```

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

Ans.)

Make all member variables private and provide public getter and setter methods to control access. This ensures fields cannot be accessed directly and allows validation or logic during read/write operations.

Github Link – https://github.com/Harshit-Raj-14/PayPal-RG-Assignment-Harshit-Raj/blob/main/Week%202/Java%20Core/Student.java



```java
public class Student {
    private String name;
    private int age;

    public String getName() {
        return name;
    }

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

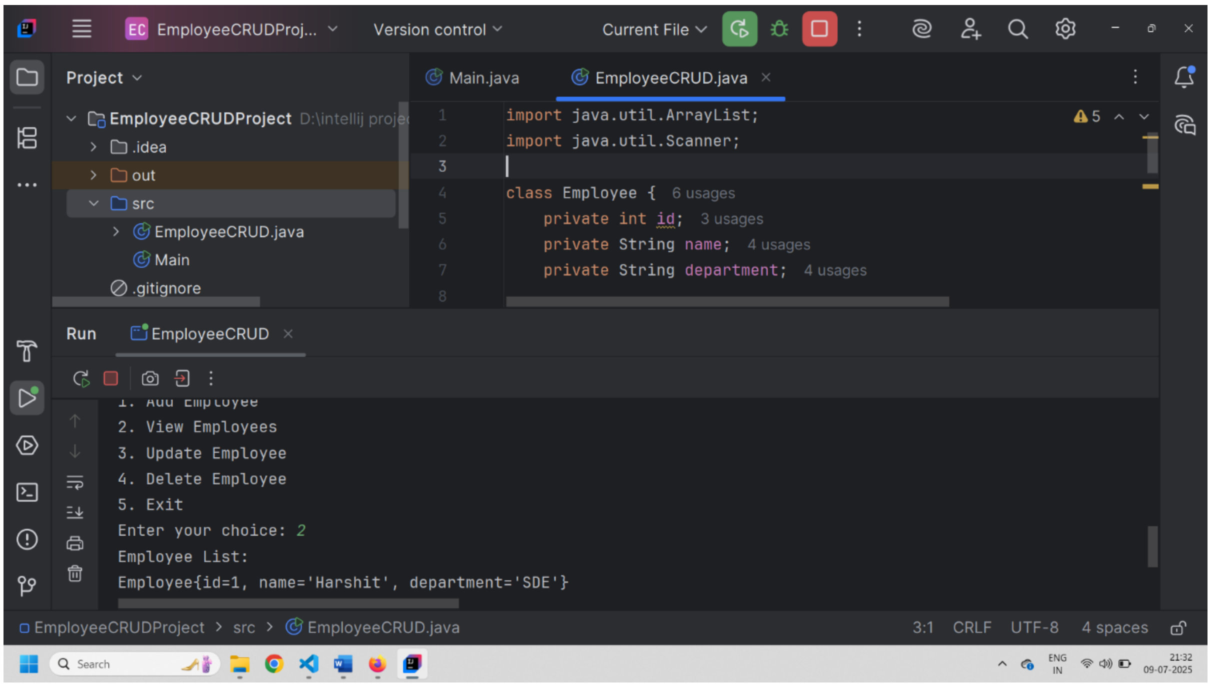
```
C:\Users\Asus\.jdks\openjdk-24.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community
Name: Harshit
Age: 21

Process finished with exit code 0
```

6)

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

```java
class Employee{
        private int id;
        private String name;
        private String department;
}
```

Github Link – https://github.com/Harshit-Raj-14/PayPal-RG-Assignment-Harshit-Raj/blob/main/Week%202/Java%20Core/EmployeeCRUD.java



```
import java.util.ArrayList;
import java.util.Scanner;

class Employee {
    private int id;
    private String name;
    private String department;
```

```
1. Add Employee
2. View Employees
3. Update Employee
4. Delete Employee
5. Exit
Enter your choice: 2
Employee List:
Employee{id=1, name='Harshit', department='SDE'}
```

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

class Employee{

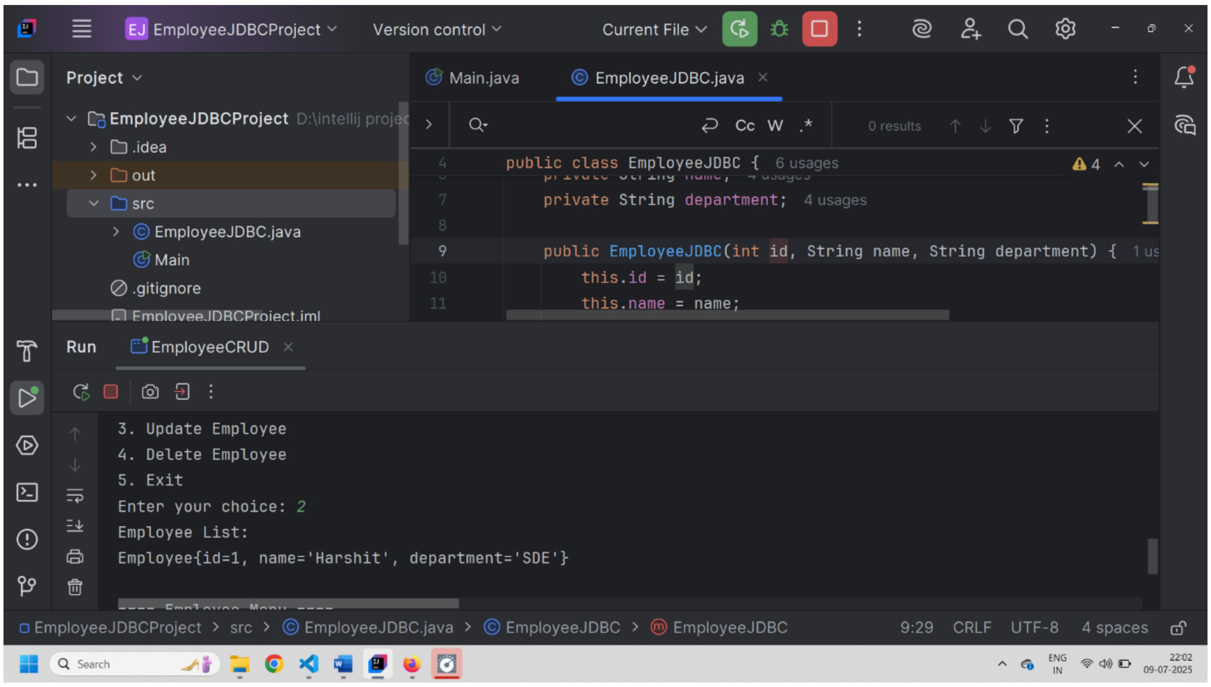
private int id;

private String name;

private String department;

}

Github Link – https://github.com/Harshit-Raj-14/PayPal-RG-Assignment-Harshit-Raj/blob/main/Week%202/Java%20Core/EmployeeJDBC.java



```java
public class EmployeeJDBC {  6 usages
    private String name;  4 usages
    private String department;  4 usages

    public EmployeeJDBC(int id, String name, String department) {  1 us
        this.id = id;
        this.name = name;
```

```
3. Update Employee
4. Delete Employee
5. Exit
Enter your choice: 2
Employee List:
Employee{id=1, name='Harshit', department='SDE'}

---- Employee Menu ----
```