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?

Answer: No, it is not a pure function since it depends on an instance variable ‘rate’ which may change dynamically. Ideally, a pure function should not have any side effects and should always return same output for same input. Here, output varies with outer variable ‘rate’, hence this isn’t pure function

If you claim the method is NOT a pure function, please suggest a way to make it pure.
We can provide rate as an argument to the function, so its output depends only on provided parameters

```
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();
}
}
```

Output:
super class show method
sub class show method

Super.show() calls the static method show() of the Super class.
Super.StaticMethods().show() creates an instance of the static nested class StaticMethods and calls its non-static show() method

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();
}
}
```

Output:
In main method, show is missspelled as ‘show’ in all lowercase, hence it gives error -

```
ThisUse.java:29: error: cannot find symbol
    o.show();
     ^
  symbol:   method show()
  location: variable o of type ThisUse
```

Assuming this typo is fixed, to o.Show(), output is -

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

Even though, ThisUse extends Super class, it has its own definitions for method Show which overrides the super class Show. Also, it has own definition for variable num which hides the num from Super.
this.display() and display() both call the overridden method in ThisUse, so it prints "display method" twice.
Later, this.num and num both referr to the child class's variable, which has value 10.
If we wanted 20 (super class value), we’d need to specifically put super.num

4) What is the singleton design pattern? Explain with a coding example.
The Singleton Design Pattern ensures that only one instance of a class is created and provides a global access point to that instance.
It is used when a single shared resource or manager is required across the application like an IPL points table manager.
In the IPL, only one official points table should track wins, draws, and team rankings. If multiple instances were allowed, different parts of the application might show inconsistent standings.

Full Example Code link -
https://github.com/OmkarPh/rg-assignments/blob/master/week1/assignment2_snippets/IPLPointsTableManager.java

5) How do we make sure a class is encapsulated? Explain with a coding example.
Encapsulation is the bundling of data (attributes) and methods (functions) that operate on that data within a single unit. It prevents external code from being concerned with the internal workings of an object.

Full example -
Consider an IPL team entity where teamName and points are private variable which can't be accessed directly. Access is provided only through methods which we can specify public/protected/private as per need. This keeps the data safe and prevents misuse.

Code link -
https://github.com/OmkarPh/rg-assignments/blob/master/week1/assignment2_snippets/IPLTeam.java

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;
}
```

Code link -
https://github.com/OmkarPh/rg-assignments/blob/master/week1/assignment2_snippets/EmployeeCRUD.java

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;
}
```

Code link -
https://github.com/OmkarPh/rg-assignments/blob/master/week1/assignment2_snippets/EmployeeJDBC.java