**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 :-)**



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.

1. Not a pure function because it depends on an external variable and the function might produce a different output if that external variable is changed.

To make it pure, move the rate variable to inside the function or substitute the constant value in place of 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();

}

}

1. super class show method

sub class 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();

}

}

1. This should throw a compilation error as java function names are case sensitive (show != Show)

If we consider Show as show, the output will be:

display method

display method

10

10

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

A. If a class is allowed to have only one instance, such pattern is called singleton design pattern.

Example: [rg-assignments/week-1/assg-2/hello-world-java/src/main/java/in/ganeshkalyan/SingletonBiscuit.java at main · ganeshkalyank/rg-assignments](https://github.com/ganeshkalyank/rg-assignments/blob/main/week-1/assg-2/hello-world-java/src/main/java/in/ganeshkalyan/SingletonBiscuit.java)

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

A. If we bundle data members and methods that operate on them together in a class and make the data members private, then the class is encapsulated.

Example: [rg-assignments/week-1/assg-2/hello-world-java/src/main/java/in/ganeshkalyan/EncapsulatedChocolate.java at main · ganeshkalyank/rg-assignments](https://github.com/ganeshkalyank/rg-assignments/blob/main/week-1/assg-2/hello-world-java/src/main/java/in/ganeshkalyan/EncapsulatedChocolate.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;

}

1. Code : [rg-assignments/week-1/assg-2/hello-world-java/src/main/java/in/ganeshkalyan/EmployeeCRUD.java at main · ganeshkalyank/rg-assignments](https://github.com/ganeshkalyank/rg-assignments/blob/main/week-1/assg-2/hello-world-java/src/main/java/in/ganeshkalyan/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;

}

1. Code: [rg-assignments/week-1/assg-2/hello-world-java/src/main/java/in/ganeshkalyan/EmployeeJDBC.java at main · ganeshkalyank/rg-assignments](https://github.com/ganeshkalyank/rg-assignments/blob/main/week-1/assg-2/hello-world-java/src/main/java/in/ganeshkalyan/EmployeeJDBC.java)