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.

No, the method calculateTax() is not a pure function because a pure function always produces the same output for the same given input (or arguments) whereas in this case the output of the method calculateTax() will change every time the value of ‘rate’ changes (as it is not constant and can be changed anytime) even if the amount (input) remains the same.

One way to make it a pure function is to make the variable ‘rate’ constant:

```
public class TaxUtil {

  private static final double RATE = 0.15;

  public 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();
}
}
```

OUTPUT:

```
super class show method
sub class show method
```

Explanation:

Super.show() calls the static method show() defined in the class Super which prints "super class show method"
and new Super.StaticMethods().show(); creates an object of the static class StaticMethods and then calls its instance method show() which prints "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();
}
}
```

OUTPUT:

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

Explanation:

this.display(); & display(); both of them calls the overridden method in ThisUse class. Similarly this.num and num both refers to the same variable that is passed on to ThisUse and not the Super class. To use the variable in Super class, we have to use ‘super.num’.

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

The Singleton Pattern ensures that only one instance of a class is created and shared throughout the application. This means that no matter how many times the class is requested, it will always return the same object.

```java
public class Singleton {
    // Private static instance of the class
    private static Singleton instance = new Singleton();

    // Private constructor to prevent instantiation from outside
    private Singleton() {
        System.out.println("Singleton instance created");
    }

    // Public static method to provide access to the instance
    public static Singleton getInstance() {
        return instance;
    }

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

    public static void main(String[] args) {
        Singleton obj1 = Singleton.getInstance();
        Singleton obj2 = Singleton.getInstance();

        obj1.showMessage();

        // Check if both references point to the same object
        System.out.println("Same instance? " + (obj1 == obj2)); // true
    }
}
```

OUTPUT:

```
PS D:\Downloads\Aditya Garg - RG Assignment\Java\Week 1 Assignment 2\Question 4> javac Singleton.java
PS D:\Downloads\Aditya Garg - RG Assignment\Java\Week 1 Assignment 2\Question 4> java Singleton.java
Singleton instance created
Hello from Singleton!
Same instance? true
```

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

Encapsulation is defined as the wrapping up of data under a single unit. It can be achieved by declaring all the variables in the class as private and writing public methods in the class to set and get the values of variables.

CODE:

```java
class Human
{
    private int age;
    private String name;
    public int getAge()
    {
        return age;
    }
    public void SetAge(int age)
    {
        this.age=age;
    }
    public String getName()
    {
        return name;
    }
    public void setName(String name)
    {
        this.name=name;
    }
}
public class Encapsulation {
    public static void main(String[] args)
    {
        Human obj=new Human();
        obj.SetAge(21);
        obj.setName("Aditya");
        System.out.println(obj.getName()+" : "+obj.getAge());
    }
}
```

OUTPUT:

```
PS D:\Downloads\Aditya Garg - RG Assignment\Java\Week 1 Assignment 2\Question 5> javac Encapsulation.java
PS D:\Downloads\Aditya Garg - RG Assignment\Java\Week 1 Assignment 2\Question 5> java Encapsulation
Aditya : 21
```

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;
}
```

```
===== Employee Management System =====
1. Add Employee
2. Display All Employees
3. Update Employee
4. Delete Employee
5. Exit
Enter your choice (1-5): 1
Enter Employee ID: 1342
Enter Name: Aditya
Enter Department: IT
Employee added successfully.
```

```
===== Employee Management System =====
1. Add Employee
2. Display All Employees
3. Update Employee
4. Delete Employee
5. Exit
Enter your choice (1-5): 2
Employee List:
Employee { ID = 1342, Name = Aditya, Department = IT }
```

```
===== Employee Management System =====
1. Add Employee
2. Display All Employees
3. Update Employee
4. Delete Employee
5. Exit
Enter your choice (1-5): 3
Enter Employee ID to update: 1342
Enter New Name: Aditya Garg
Enter New Department: IT
Employee updated successfully.

===== Employee Management System =====
1. Add Employee
2. Display All Employees
3. Update Employee
4. Delete Employee
5. Exit
Enter your choice (1-5): 2
Employee List:
Employee { ID = 1342, Name = Aditya Garg, Department = IT }
```

```
===== Employee Management System =====
1. Add Employee
2. Display All Employees
3. Update Employee
4. Delete Employee
5. Exit
Enter your choice (1-5): 4
Enter Employee ID to delete: 1342
Employee deleted successfully.

===== Employee Management System =====
1. Add Employee
2. Display All Employees
3. Update Employee
4. Delete Employee
5. Exit
Enter your choice (1-5): 2
 No employees to display.
```

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;
}
```

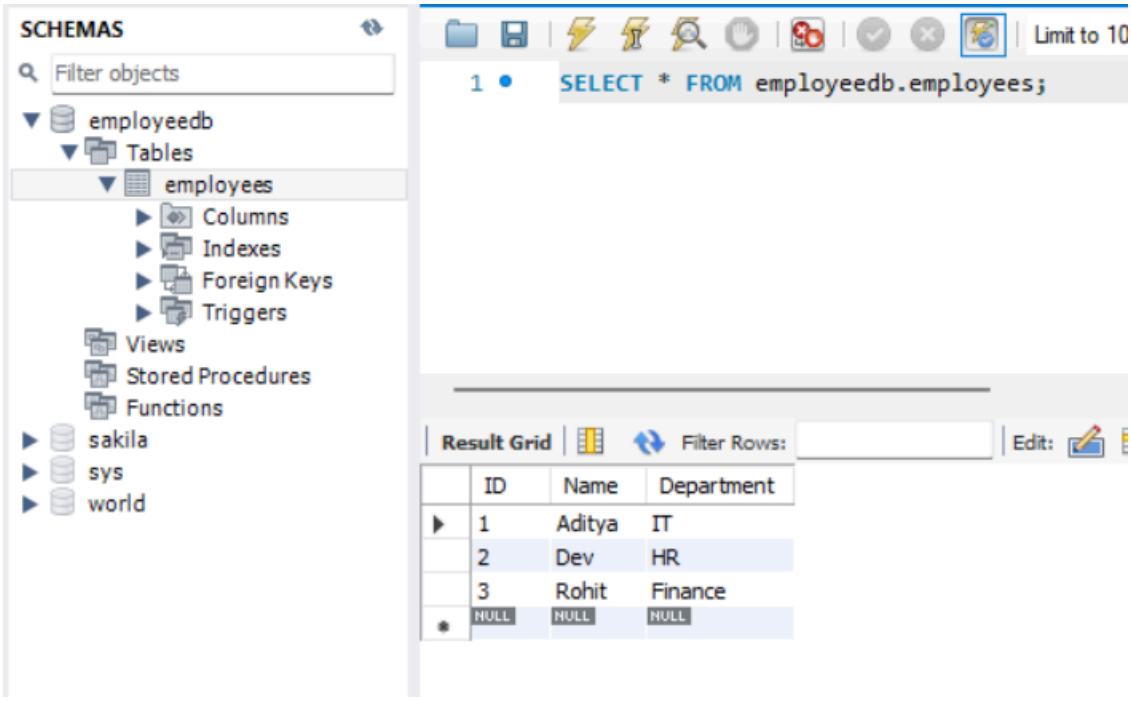
```
===== Employee CRUD Menu =====
1. Add Employee
2. View All Employees
3. Update Employee
4. Delete Employee
5. Exit
Enter your choice: 1
Enter employee name: Ram
Enter department: Sales
 Employee added.
```

```
===== Employee CRUD Menu =====
1. Add Employee
2. View All Employees
3. Update Employee
4. Delete Employee
5. Exit
Enter your choice: 2

 Employee List:
ID: 1, Name: Aditya, Department: IT
ID: 2, Name: Dev, Department: HR
ID: 3, Name: Ram, Department: Sales
```

```
===== Employee CRUD Menu =====
1. Add Employee
2. View All Employees
3. Update Employee
4. Delete Employee
5. Exit
Enter your choice: 3
Enter employee ID to update: 3
Enter new name: Rohit
Enter new department: Finance
Employee updated.
```

DATABASE:



```
===== Employee CRUD Menu =====
1. Add Employee
2. View All Employees
3. Update Employee
4. Delete Employee
5. Exit
Enter your choice: 4
Enter employee ID to delete: 3
Employee deleted.

===== Employee CRUD Menu =====
1. Add Employee
2. View All Employees
3. Update Employee
4. Delete Employee
5. Exit
Enter your choice: 2

 Employee List:
ID: 1, Name: Aditya, Department: IT
ID: 2, Name: Dev, Department: HR
```