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, calculateTax() is not a pure function because it uses the instance variable rate, which makes the output dependent on the external state.

To make it pure, we can pass rate as a parameter so the method depends only on its inputs:

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

Now, the method always gives the same output for the same inputs and has no side effects making it a **pure function**

2. **What will be the output for the 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();}}
```

```java
public 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)
        {
            Super.show(); // Calls static method in outer class
            new Super.StaticMethods().show(); // Calls instance method of static inner class
        }

}
```

Run Super

```
C:\Users\bhimk\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:D:\IntelliJ IDEA 2025.1.3\lib\idea_rt.
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();
}}

```java
public class ThisUse extends Super {
    int num;  3 usages

    public ThisUse(int num) {  1 usage
        this.num = num;
    }

    public void display() {  2 usages
        System.out.println("display method");
    }

    public void show() {  1 usage
        this.display(); // calls display() from ThisUse class
        display();      // same as above
        System.out.println(this.num); // prints 10
        System.out.println(num);      // same as above
    }

    public static void main(String[] args) {
        ThisUse o = new ThisUse( num: 10);
        o.show();
    }
}
```

Run ThisUse

```
C:\Users\bhimk\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:D:\IntelliJ IDEA 2025.1.3\lib\idea
display method
display method
10
10

Process finished with exit code 0
```

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

The Singleton Design Pattern ensures that a class has only one instance and provides a global access point to that instance. It's used when exactly one object is needed to coordinate actions across the system like a single database connection, logger, or configuration manager.

```
public class Singleton {

    //  Create a private static instance of the class
    private static Singleton instance;

    //  Make the constructor private to prevent instantiation
    private Singleton() {
        System.out.println("Singleton instance created!");
    }

    // Provide a public static method to get the instance
    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton(); // Lazy initialization
        }
        return instance;
    }

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

Usage in main() Method:

```
public class Main {
    public static void main(String[] args) {
        // Only one instance will be created
        Singleton obj1 = Singleton.getInstance();
        Singleton obj2 = Singleton.getInstance();

        obj1.showMessage();

        // Verify both references point to the same object
        System.out.println(obj1 == obj2);  // Output: true
    }
}
```

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

Encapsulation is the process of hiding the internal details of a class and restricting direct access to them. It allows controlled access through getter and setter methods. This is one of the key principles of Object-Oriented Programming (OOP).

To Achieve Encapsulation:

1. Declare class variables private.
2. Provide public getter and setter methods to access and modify them.

```
public class Student {

        //  Private fields (data hiding)

          private String name;

          private int age;

          // Public getters and setters for controlled access

          public String getName() {

              return name;

          }

          public void setName(String name) {

              this.name = name;

          }

          public int getAge() {

              return age;

          }

          public void setAge(int age) {

              if (age > 0) {  // example of validation

                  this.age = age;

              }  }}
```

Usage in MAIN class:

```
public class Main {
    public static void main(String[] args) {
        Student s = new Student();
        s.setName("Sneha");
        s.setAge(21);
        System.out.println("Name: " + s.getName());
        System.out.println("Age: " + s.getAge());
    }}
```

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;
```

Output:

```
C:\Users\bhimk\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:D:\IntelliJ IDEA 2025.1.3\lib\idea_
Employee added successfully.
Employee added successfully.

All Employees:
Employee { ID: 1, Name: Sneha, Department: IT }
Employee { ID: 2, Name: Abhinav, Department: HR }

Updating Employee with ID 2:
Employee updated.

All Employees after update:
Employee { ID: 1, Name: Sneha, Department: IT }
Employee { ID: 2, Name: Abhinav Sharma, Department: Finance }

Deleting Employee with ID 1:
Employee deleted.

All Employees after deletion:
Employee { ID: 2, Name: Abhinav Sharma, Department: Finance }

Process finished with exit code 0
```

Employe.java:

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

    public Employee(int id, String name, String department) {
        this.id = id;
        this.name = name;
        this.department = department;
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public String getDepartment() {
        return department;
    }

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

    public void setDepartment(String department) {
        this.department = department;
    }

    @Override
    public String toString() {
        return "Employee { ID: " + id + ", Name: " + name + ", Department: " + department + " }";
    }
}
```

EmployeeCRUD.java:

Emplo... Version co... Employe...

Employee.java EmployeeCRUD.java

```java
import java.util.ArrayList;

public class EmployeeCRUD {
    private final ArrayList<Employee> employeeList = new ArrayList<>();  6 usages
    public void addEmployee(Employee e) {  2 usages
        employeeList.add(e);
        System.out.println("Employee added successfully.");}
    public void viewEmployees() {  3 usages
        if (employeeList.isEmpty()) {
            System.out.println("No employees found.");
        } else {
            for (Employee e : employeeList) {
                System.out.println(e);
            }
        }}
    public void updateEmployee(int id, String newName, String newDept) {  1 usage
        for (Employee e : employeeList) {
            if (e.getId() == id) {
                e.setName(newName);
                e.setDepartment(newDept);
                System.out.println("Employee updated.");
                return;
            }}
        System.out.println("Employee not found.");}
    public void deleteEmployee(int id) {  1 usage
        for (Employee e : employeeList) {
            if (e.getId() == id) {
                employeeList.remove(e);
                System.out.println("Employee deleted.");
                return;
            }}
        System.out.println("Employee not found.");}

    public static void main(String[] args) {
        EmployeeCRUD crud = new EmployeeCRUD();
        crud.addEmployee(new Employee( id: 1,  name: "Sneha",  department: "IT"));
        crud.addEmployee(new Employee( id: 2,  name: "Abhinav",  department: "HR"));

        System.out.println("\nAll Employees:");
        crud.viewEmployees();

        System.out.println("\nUpdating Employee with ID 2:");
        crud.updateEmployee( id: 2,  newName: "Abhinav Sharma",  newDept: "Finance");

        System.out.println("\nAll Employees after update:");
        crud.viewEmployees();

        System.out.println("\nDeleting Employee with ID 1:");
        crud.deleteEmployee( id: 1);

        System.out.println("\nAll Employees after deletion:");
        crud.viewEmployees();
    }
}
```

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.java

```java
public class Employee {  7 usages
    private int id;  3 usages
    private String name;  3 usages
    private String department;  3 usages

    public Employee(int id, String name, String department) {  2 usages
        this.id = id;
        this.name = name;
        this.department = department;
    }

    // Getters and Setters
    public int getId() { return id; }  2 usages
    public String getName() { return name; }  no usages
    public String getDepartment() { return department; }  no usages

    public void setId(int id) { this.id = id; }  no usages
    public void setName(String name) { this.name = name; }  1 usage
    public void setDepartment(String department) { this.department = department; }  1 u
}
```

EmployeeCRUD.java

```java
ployee.java    EmployeeCRUD.java ×

import java.sql.*;
import java.util.Scanner;

public class EmployeeCRUD {
    static final String DB_URL = "jdbc:mysql://localhost:3306/company";  1 usage
    static final String USER = "root";  1 usage
    static final String PASS = "root";  1 usage

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASS)) {
            System.out.println("Connected to database ");

            Statement stmt = conn.createStatement();

            // Create
            System.out.println("Inserting an employee...");
            String insert = "INSERT INTO employees (id, name, department) VALUES (3, 'Rahul', 'Finance')";
            stmt.executeUpdate(insert);

            // Read
            System.out.println("Reading employees:");
            ResultSet rs = stmt.executeQuery( sql: "SELECT * FROM employees");
            while (rs.next()) {
                System.out.printf("ID: %d, Name: %s, Department: %s%n",
                        rs.getInt( columnLabel: "id"), rs.getString( columnLabel: "name"), rs.getString( columnLabel: "department"));
            }

            // Update
            String update = "UPDATE employees SET department='Marketing' WHERE id=3";
            stmt.executeUpdate(update);
            System.out.println("Updated employee department.");

            // Delete
            String delete = "DELETE FROM employees WHERE id=3";
            stmt.executeUpdate(delete);
            System.out.println("Deleted employee.");

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

Run EmployeeCRUD

```
C:\Users\bhimk\.jdks\openjdk-21.0.1\bin\java.exe "-javaagen
Connected to database
Inserting an employee...
Reading employees:
ID: 1, Name: Sneha, Department: IT
ID: 2, Name: Abhinav, Department: HR
ID: 3, Name: Rahul, Department: Finance
Updated employee department.
Deleted employee.

Process finished with exit code 0
```