**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.

**Answer:**

No, calculateTax is not a pure function because the value depends not just on the amount, but also on the rate, which is set to 0.15. If the rate is changed, the final ans will also change.

We can make it a pure function but giving rate as a parameter along with amount.

public 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();

}

}

**Answer:**

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();

}

}

**Answer:**

display method

display method

10

10

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

The Singleton design pattern is when a class has only one instance and provides a global access point to it.

import java.io.\*;

class Singleton {

private static Singleton instance;

private Singleton()

{

System.out.println("Singleton created.");

}

public static Singleton getInstance()

{

if (instance == null)

instance = new Singleton();

return instance;

}

public static void testMethod()

{

System.out.println("This is a Singleton class.");

}

}

class Main {

public static void main(String[] args)

{

Singleton s = Singleton.getInstance();

s.testMethod();

}

}

<https://github.com/Jeshma-Ullas/rg-assignments/blob/main/singleton.java>

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

We can declare all the variables in the class as private and write public methods in the class to set and get the values of variables. This makes the class encapsulated.

Example:

class Details {

private String name;

private String email;

public String getName() {

return name;

}

public String getEmail() {

return email;

}

public void setName(String name) {

this.name = name;

}

public void setEmail(String email) {

this.email = email;

}

}

public class Encapsulation {

public static void main(String[] args) {

Details d = new Details();

d.setName("Jeshma Ullas");

d.setEmail("jeshmarajiullas@gmail.com");

System.out.println("Name: " + d.getName());

System.out.println("Email: " + d.getEmail());

}

}

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;

}

import java.util.\*;

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 getDept() {

        return department;

    }

    public void setName(String *name*) {

        this.name = *name*;

    }

    public void setDept(String *department*) {

        this.department = *department*;

    }

    public String details() {

        return "ID: " + id + " Name: " + name + " Dept: " + department;

    }

}

public class EmployeeCRUD {

    private ArrayList<Employee> employees = new ArrayList<>();

*//Create*

    public void createEmployee(int *id*, String *name*, String *department*) {

        employees.add(new Employee(*id*, *name*, *department*));

        System.out.println("Employee " + *id* + " added");

    }

*//Read*

    public void readEmployee() {

        System.out.println("Reading employees...");

        if(employees.isEmpty()) {

            System.out.println("No employees");

            return;

        }

        for(Employee e : employees) {

            System.out.println(e.details());

        }

    }

*//Update*

    public void updateEmployee(int *id*, String *name*, String *department*) {

        for(Employee e : employees) {

            if(e.getId() == *id*) {

                e.setName(*name*);

                e.setDept(*department*);

                System.out.println("Employee " + *id* + " updated");

                return;

            }

        }

        System.out.println("Employee not found");

    }

*//Delete*

    public void deleteEmployee(int *id*) {

        for(Employee e : employees) {

            if(e.getId() == *id*) {

                employees.remove(e);

                System.out.println("Employee " + *id* + " deleted");

                return;

            }

        }

        System.out.println("Employee not found");

    }

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

        EmployeeCRUD employee = new EmployeeCRUD();

        employee.createEmployee(1, "Jeshma Ullas", "Software Development");

        employee.createEmployee(2, "ABC", "AI");

        employee.readEmployee();

        employee.updateEmployee(2, "DEF", "Data");

        employee.readEmployee();

        employee.deleteEmployee(2);

        employee.readEmployee();

    }

}



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;

}

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 getDept() {

        return department;

    }

    public void setName(String *name*) {

        this.name = *name*;

    }

    public void setDept(String *department*) {

        this.department = *department*;

    }

}

public class EmployeeJDBC {

    private static final String URL = "jdbc:mysql://127.0.0.1:3306/my-db";

    private static final String USER = "root";

    private static final String PASSWORD = "password";

*// Create*

    public void createEmployee(Employee *emp*) {

        String query = "INSERT INTO Employee (id, name, department) VALUES (?, ?, ?)";

        try {

            Class.forName("com.mysql.cj.jdbc.Driver");

            try (Connection con = DriverManager.getConnection(URL, USER, PASSWORD);

                 PreparedStatement stmt = con.prepareStatement(query)) {

                stmt.setInt(1, *emp*.getId());

                stmt.setString(2, *emp*.getName());

                stmt.setString(3, *emp*.getDept());

                int rows = stmt.executeUpdate();

                System.out.println("Employee inserted: " + rows + " row(s)");

            }

        } catch (SQLException | ClassNotFoundException *e*) {

            e.printStackTrace();

        }

    }

*// Read*

    public void readEmployees() {

        String query = "SELECT \* FROM Employee";

        try (Connection con = DriverManager.getConnection(URL, USER, PASSWORD);

             Statement stmt = con.createStatement();

             ResultSet rs = stmt.executeQuery(query)) {

            while (rs.next()) {

                System.out.println("ID: " + rs.getInt("id") +

                                   " Name: " + rs.getString("name") +

                                   " Dept: " + rs.getString("department"));

            }

        } catch (SQLException *e*) {

            e.printStackTrace();

        }

    }

*// Update*

    public void updateEmployee(int *id*, String *name*, String *department*) {

        String query = "UPDATE Employee SET name = ?, department = ? WHERE id = ?";

        try (Connection con = DriverManager.getConnection(URL, USER, PASSWORD);

             PreparedStatement stmt = con.prepareStatement(query)) {

            stmt.setString(1, *name*);

            stmt.setString(2, *department*);

            stmt.setInt(3, *id*);

            int rows = stmt.executeUpdate();

            System.out.println("Employee updated: " + rows + " row(s)");

        } catch (SQLException *e*) {

            e.printStackTrace();

        }

    }

*// Delete*

    public void deleteEmployee(int *id*) {

        String query = "DELETE FROM Employee WHERE id = ?";

        try (Connection con = DriverManager.getConnection(URL, USER, PASSWORD);

             PreparedStatement stmt = con.prepareStatement(query)) {

            stmt.setInt(1, *id*);

            int rows = stmt.executeUpdate();

            System.out.println("Employee deleted: " + rows + " row(s)");

        } catch (SQLException *e*) {

            e.printStackTrace();

        }

    }

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

        EmployeeJDBC ejdbc = new EmployeeJDBC();

        Employee e1 = new Employee(1, "Jeshma", "Software");

        Employee e2 = new Employee(2, "ABC", "AI");

        ejdbc.createEmployee(e1);

        ejdbc.createEmployee(e2);

        ejdbc.readEmployees();

        ejdbc.updateEmployee(2, "DEF", "Data");

        ejdbc.readEmployees();

        ejdbc.deleteEmployee(1);

        ejdbc.readEmployees();

    }

}