JAVA ASSIGNMENT

1. It is not considered as a pure function.

Function depends on instance variable which is outside the method so if rate changes for same input(amount) it gives different result.

public class TaxUtil {

   public double calculateTax(double amount, double rate) {

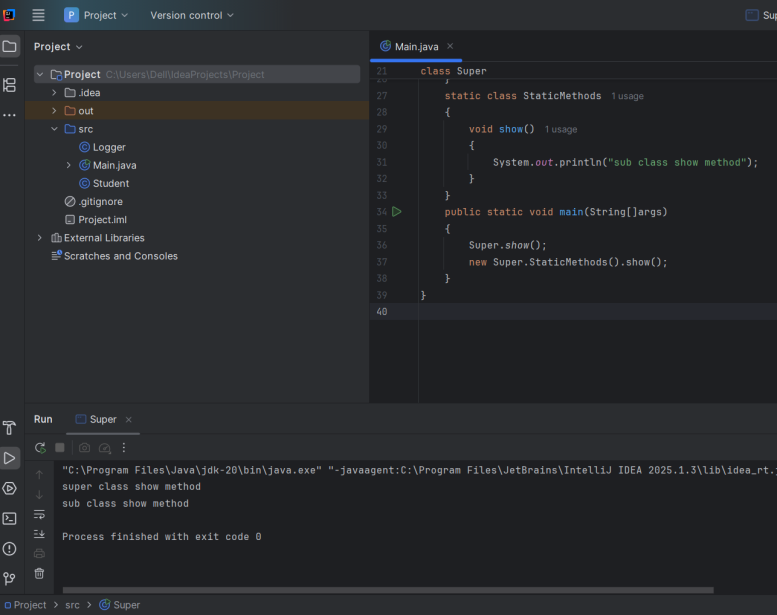
        return amount \* rate;

    }

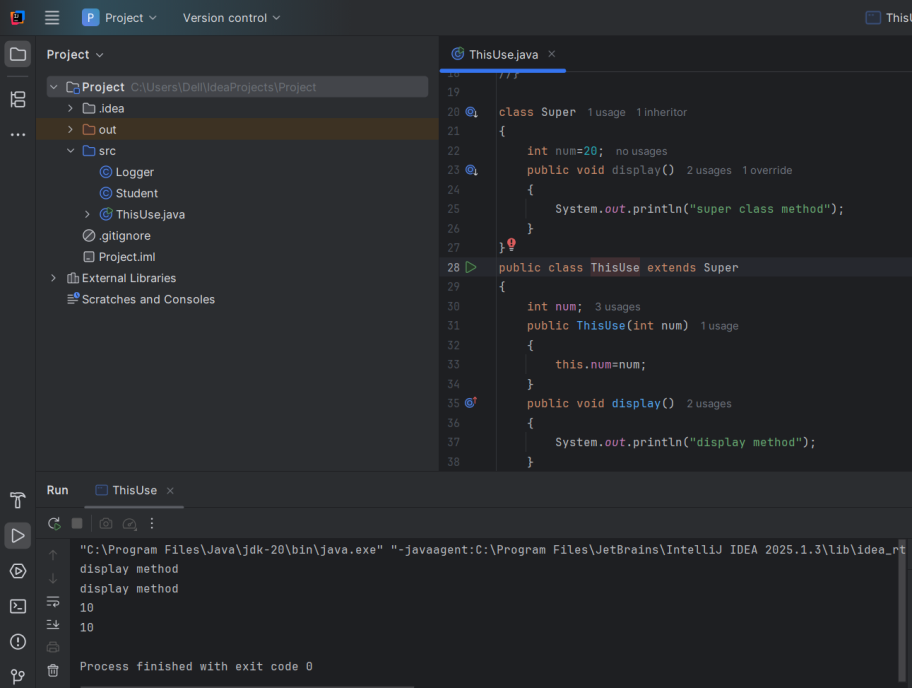
}

1. o/p: super class show method

sub class show method



1. o/p: display method  
    display method  
    10  
    10



1. Singleton design pattern ensures that only one instance of the class is created and shared globally.

public class Logger {

private static Logger instance;

private Logger() {

System.out.printf("Logger is created!\n");

}

public static Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

public void log(String message) {

System.out.println("Log:"+message);

}

}

public class Main {

public static void main(String[] args) {

Logger log1=Logger.getInstance();

log1.log("log1 created!");

Logger log2=Logger.getInstance();

log2.log("log2 created!");

System.out.println(log1==log2);

}

}

**5**. To make sure the class is encapsulated we make the variables private and getters and setters as public.

public class Student {

private String name;

private int age;

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 && age<100){

this.age = age;

}

else{

System.out.println("Invalid Age");

}

}

}

public class Main {

public static void main(String[] args) {

Student student = new Student();

student.setAge(20);

student.setName("David");

System.out.println(student.getName());

System.out.println(student.getAge());

}

}

**6**.

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 id + " | " + name + " | " + department;

}

}

class EmployeeCRUD {

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

//Create

public void addEmployee(Employee emp) {

employees.add(emp);

}

// Read

public Employee getEmployee(int id) {

for (Employee emp : employees) {

if (emp.getId() == id) return emp;

}

return null;

}

// Update

public boolean updateEmployee(int id, String name, String department) {

Employee emp = getEmployee(id);

if (emp != null) {

emp.setName(name);

emp.setDepartment(department);

return true;

}

return false;

}

// Delete

public boolean deleteEmployee(int id) {

Employee emp = getEmployee(id);

return emp != null && employees.remove(emp);

}

// DisplayAll

public void displayAll() {

for (Employee emp : employees) {

System.out.println(emp);

}

}

}

7.

import java.sql.\*;

public class EmpJDBC {

static final String DB\_URL = "jdbc:mysql://localhost:3306/company";

static final String USER = "root";

static final String PASS = "abcde";

public void addEmployee(int id, String name, String dept) {

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

try (Connection conn = DriverManager.getConnection(DB\_URL, USER, PASS);

PreparedStatement stmt = conn.prepareStatement(query)) {

stmt.setInt(1, id);

stmt.setString(2, name);

stmt.setString(3, dept);

stmt.executeUpdate();

System.out.println("Employee added successfully.");

} catch (SQLException e) {

System.err.println("Error adding employee: " + e.getMessage());

System.err.println("SQLState: " + e.getSQLState());

}

}

public void displayEmployees() {

String query = "SELECT id, name, department FROM employee";

try (Connection conn = DriverManager.getConnection(DB\_URL, USER, PASS);

Statement stmt = conn.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) {

System.err.println("Error displaying employees: " + e.getMessage());

System.err.println("SQLState: " + e.getSQLState());

}

}

public void updateEmployee(int id, String name, String dept) {

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

try (Connection conn = DriverManager.getConnection(DB\_URL, USER, PASS);

PreparedStatement stmt = conn.prepareStatement(query)) {

stmt.setString(1, name);

stmt.setString(2, dept);

stmt.setInt(3, id);

int rows = stmt.executeUpdate();

if (rows > 0) {

System.out.println("Employee updated successfully.");

} else {

System.out.println("Employee with ID " + id + " not found.");

}

} catch (SQLException e) {

System.err.println("Error updating employee: " + e.getMessage());

System.err.println("SQLState: " + e.getSQLState());

}

}

public void deleteEmployee(int id) {

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

try (Connection conn = DriverManager.getConnection(DB\_URL, USER, PASS);

PreparedStatement stmt = conn.prepareStatement(query)) {

stmt.setInt(1, id);

int rows = stmt.executeUpdate();

if (rows > 0) {

System.out.println("Employee deleted successfully.");

} else {

System.out.println("Employee with ID " + id + " not found.");

}

} catch (SQLException e) {

System.err.println("Error deleting employee: " + e.getMessage());

System.err.println("SQLState: " + e.getSQLState());

}

}

public static void main(String[] args) {

try {

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

} catch (ClassNotFoundException e) {

System.err.println("MySQL JDBC Driver not found.");

return;

}

EmpJDBC obj = new EmpJDBC();

obj.addEmployee(1, "Raju", "Finance");

obj.addEmployee(2, "Ram", "HR");

obj.addEmployee(3, "Sita", "IT");

System.out.println("Employees:");

obj.displayEmployees();

obj.updateEmployee(1, "Roopa", "Marketing");

System.out.println("Employees after update:");

obj.displayEmployees();

obj.deleteEmployee(2);

System.out.println("Employees after deletion:");

obj.displayEmployees();

}

}

o/p:

