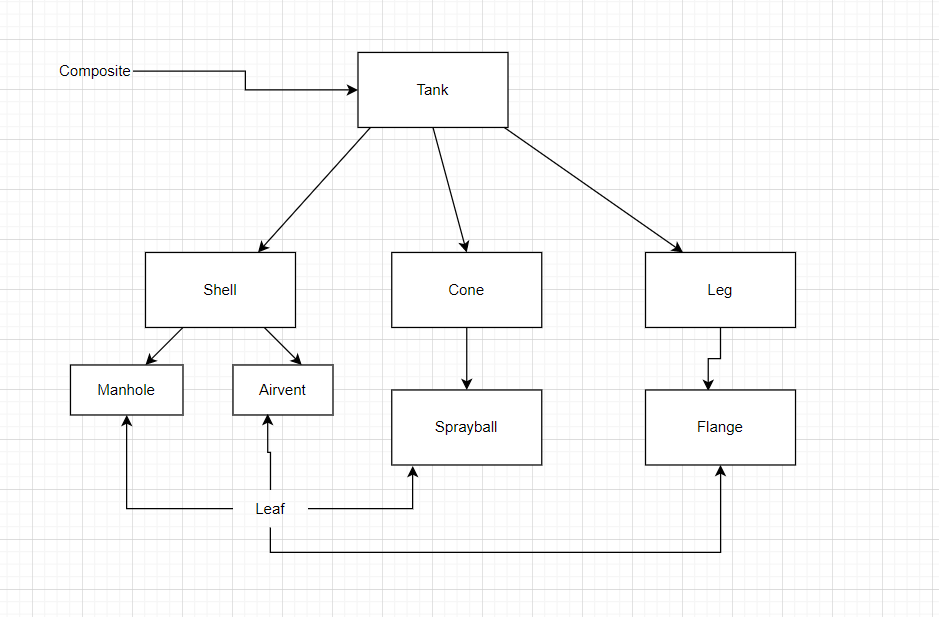
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Course Name: Design Patterns/Thinking LAB** | **EXPERIMENT NO. 6** | |
| **Course Code: 20CP210P**  **Faculty: Dr. Ketan Sabale** | **Branch: CSE** | **Semester: IV** |
| **Submitted by: Jangle Parth**  **Roll no: 22BCP083** | | | |

Objective: To familiarize students with standard Structural design patterns.

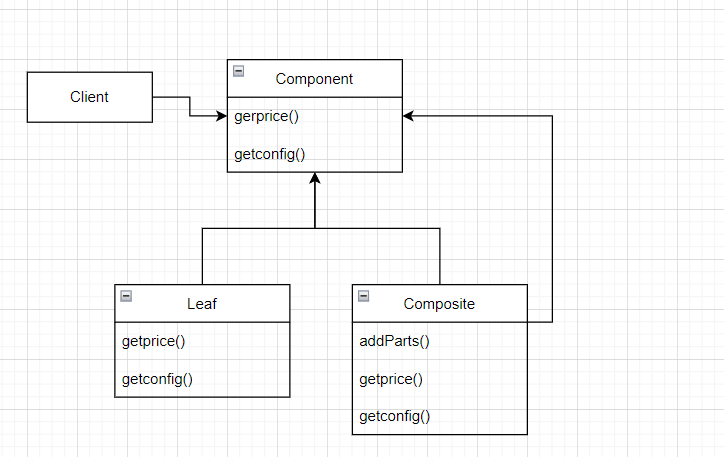
Experiment: Explain the Composite design pattern and write a program using any object-oriented programming language to demonstrate the working of builder design pattern.

**Theory:**

Composite Design pattern is a Structural Design Pattern. Everything in this world is made up of Something. Now this Could Be represented in a Structure for Example You want to make a Tank. Tank has Parts Like Shell, Cone, Leg etc. Shell IN turn has a Manhole, Airvent etc. In this Tree Like Structure the root i.e. The Tank is Called Composite and its part are Known as Leaf



**Flowchart:**



**Problem Statement Explanation:** we have a class Part that is a basic Object and two classes leaf and composite leaf is the smallest component and composite is a collection of leafs if we change something in leaf class it must reflect in composite and there must be a add and remove method in component to add leaf or remove leaf.

**Code:**

import java.util.ArrayList;

import java.util.List;

interface Part {

    int getprice();

    void getconfig();

}

class Leaf implements Part {

    String name;

    int dia;

    String material;

    Leaf(String name, int dia, String material) {

        this.name = name;

        this.dia = dia;

        this.material = material;

    }

    public int getprice() {

        double weight = dia \* dia \* 3.1415 \* 3;

        int price = (int) weight \* 370;

        return price;

    }

    public void getconfig() {

        System.out.println("Component name: " + name);

        System.out.println("Component Dia: " + dia);

        System.out.println("Component material: " + material);

        System.out.println("Component Price: " + getprice());

        System.out.println();

    }

}

class Composite implements Part {

    String name;

    List<Part> parts = new ArrayList<>();

    public Composite(String name) {

        super();

        this.name = name;

    }

    public void addParts(Part p) {

        parts.add(p);

    }

    public int getprice() {

        int price = 0;

        for (Part p : parts) {

            price = price + p.getprice();

        }

        int finalprice = (int) (price + 0.1 \* price);

        return finalprice;

    }

    public void getconfig() {

        System.out.println("The Product " + name + " Consist of Following Parts");

        System.out.println();

        for (Part p : parts) {

            p.getconfig();

        }

        System.out.println();

    }

}

public class compositedesignpatter {

    public static void main(String[] args) {

        Leaf Manhole = new Leaf("Manhole", 1500, "SS 304");

        Leaf Airvent = new Leaf("Airvent", 1500, "SS 304");

        Leaf Sprayball = new Leaf("Sprayball", 150, "SS 316");

        Leaf Leg = new Leaf("Leg", 150, "SS 316");

        Leaf Shell = new Leaf("Shell", 3000, "SS 304");

        Leaf Cone = new Leaf("Cone", 2000, "SS 304");

        Composite MilkStorageTank = new Composite("MilkStorageTank");

        Composite AcidStorageTank = new Composite("AcidStorageTank");

        Composite PectinMixingTank = new Composite("PectinMixingTank");

        Composite WaterStorageTank = new Composite("WaterStorageTank");

        MilkStorageTank.addParts(Manhole);

        MilkStorageTank.addParts(Airvent);

        AcidStorageTank.addParts(Airvent);

        AcidStorageTank.addParts(Sprayball);

        PectinMixingTank.addParts(Leg);

        PectinMixingTank.addParts(Shell);

        WaterStorageTank.addParts(Airvent);

        WaterStorageTank.addParts(Manhole);

        WaterStorageTank.addParts(Cone);

        WaterStorageTank.getconfig();

        WaterStorageTank.getprice();

        System.out.println("Individual Component");

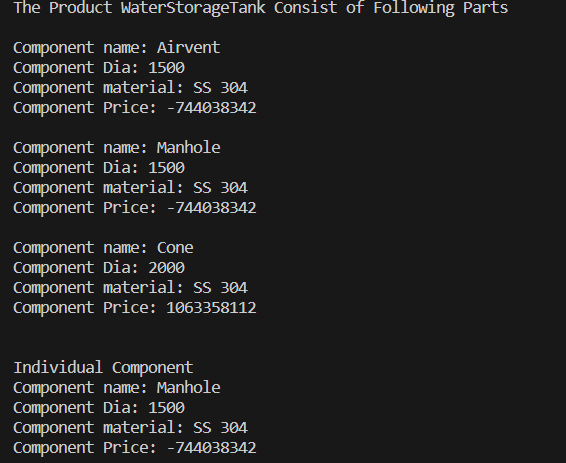
        Manhole.getconfig();

        Manhole.getprice();

    }

}

**Output:**

****