	Course Name: Design Patterns/Thinking LAB	**EXPERIMENT NO. 6**	
	Course Code: 20CP210P **Faculty: Dr. Ketan Sabale**	**Branch: CSE**	**Semester: IV**
Submitted by: Rhythm Shah **Roll no:** 22BCP071			

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 Composite design pattern.

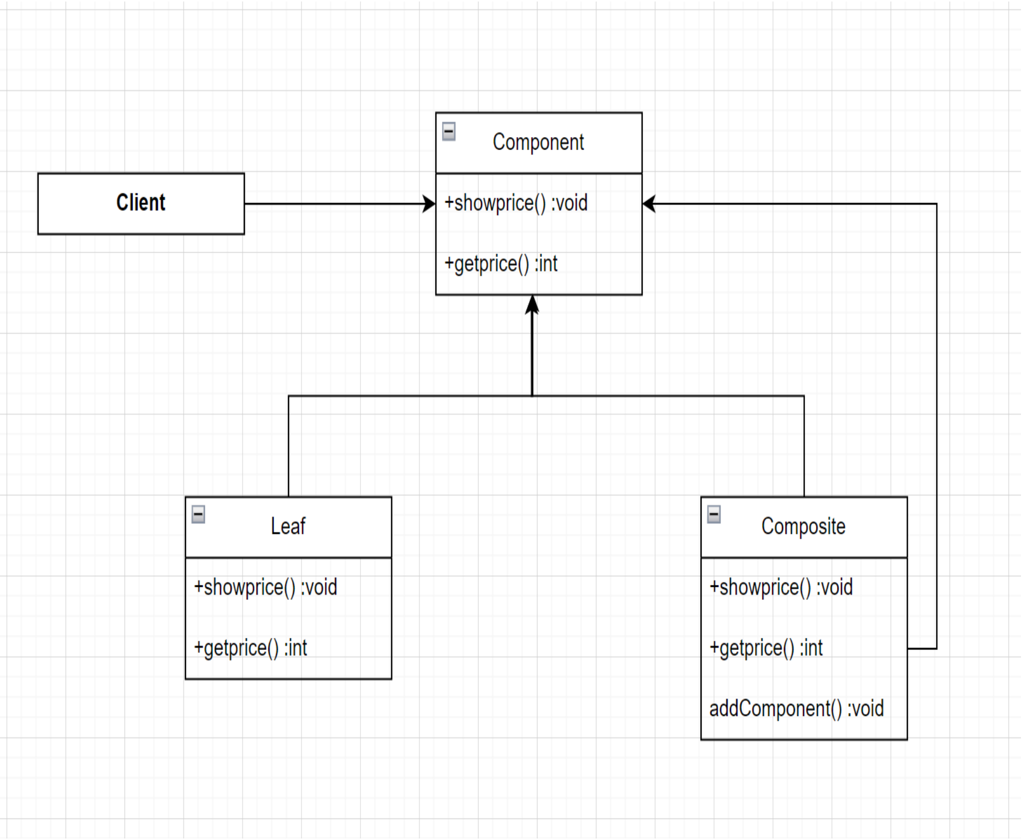
Theory:

It is a type of Structural Design pattern. This pattern lets you compose objects into tree structure and then work with these structure as if they were individual objects. There are components, and leaf of the tree. When an component of tree can not further be divided is called leaf and a common method can be implemented by creating an interface first and then implementing the methods in further components.

In this pattern, a "composite" object can contain other objects, which can be either individual objects or further composite objects. This creates a hierarchical structure where clients can treat individual objects and compositions of objects uniformly.

Implementation: - I have written a code in which I have taken computer as a component which is a common interface for my composite and leaf class. Class leaf and class composite has been implemented into component interface in which two methods are created showPrice() and getPrice().In composite class a method called addComponent() is made and at last in main method all the components are added according to composite and leaf.

UML Diagram:

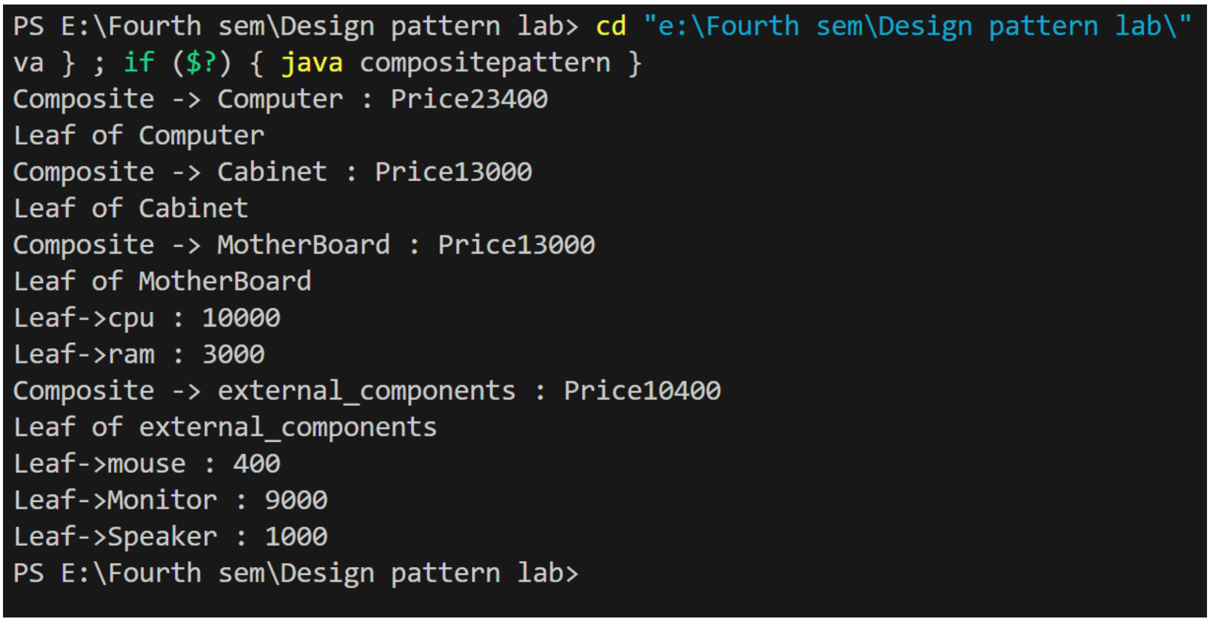


Code:

```
import java.util.ArrayList;
import java.util.List;

interface Component
{
   void showPrice();
   int getPrice();
}
class Leaf implements Component{
   int price;
   String name;
   Leaf(String name,int price)
   {
      this.name = name;
      this.price = price;
   }
   public void showPrice() {

      System.out.println("Leaf" + "->" +name+ " : " +price);
   }
   public int getPrice() {

      return price;
   }

}
```

```
class Composite implements Component{
    String name;
    List<Component> components = new ArrayList<>();

    public Composite(String name)
    {
        super();
        this.name = name;
    }

    public void addComponent(Component com)
    {
            components.add(com);
    }
    public int getPrice()
    {
        int p =0;
        for(Component c : components)
        {
            p += c.getPrice();
        }
        return p;
    }
    public void showPrice()
  {
     System.out.println("Composite -> " +name+" : Price" +getPrice());
     System.out.println("Leaf of " +name);
     for(Component c : components)
    {
        c.showPrice();
    }


  }
}
public class compositepattern {
    public static void main(String[] args)
    {
        Component speaker = new Leaf("Speaker", 1000);
        Component mouse = new Leaf("mouse", 400);
        Component monitor = new Leaf("Monitor", 9000);
        Component ram = new Leaf("ram", 3000);
        Component cpu = new Leaf("cpu", 10000);

        Composite external = new Composite("external_components");
        Composite cabinet = new Composite("Cabinet");
        Composite mb = new Composite("MotherBoard");

        Composite computer = new Composite("Computer");

        mb.addComponent(cpu);
        mb.addComponent(ram);

        external.addComponent(mouse);
        external.addComponent(monitor);
        external.addComponent(speaker);

        cabinet.addComponent(mb);
        computer.addComponent(cabinet);
        computer.addComponent(external);
        computer.showPrice();


    }
}
```

Output:



```
PS E:\Fourth sem\Design pattern lab> cd "e:\Fourth sem\Design pattern lab\"
va } ; if ($?) { java compositepattern }
Composite -> Computer : Price23400
Leaf of Computer
Composite -> Cabinet : Price13000
Leaf of Cabinet
Composite -> MotherBoard : Price13000
Leaf of MotherBoard
Leaf->cpu : 10000
Leaf->ram : 3000
Composite -> external_components : Price10400
Leaf of external_components
Leaf->mouse : 400
Leaf->Monitor : 9000
Leaf->Speaker : 1000
PS E:\Fourth sem\Design pattern lab>
```