	Course Name: Design Patterns/Thinking LAB	**EXPERIMENT NO. 10**	
	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 Decorator design pattern and write a program using any object-oriented programming language to demonstrate the working of Decorator design pattern.

Theory:-

Decorator is a structural design pattern that lets you attach new behaviors to objects by placing these objects inside special wrapper objects that contain the behaviors.
A user can extend an object's functionality using decorator patterns without changing the object's structural makeup. Therefore, nothing has changed from the original class.
As a structural pattern, the decorator design pattern offers an interface to the current class.
In order to implement the wrapper, the decorator design pattern makes use of abstract classes or interfaces with the composition.
Decorator design patterns generate decorator classes, which preserve the signature of the class methods while adding extra functionality to the original class.
Since decorator design patterns segregate functionality into classes with distinct areas of concern, they are most commonly used for adopting single responsibility principles.
From a structural perspective, the decorator design pattern is similar to the chain of responsibility pattern.

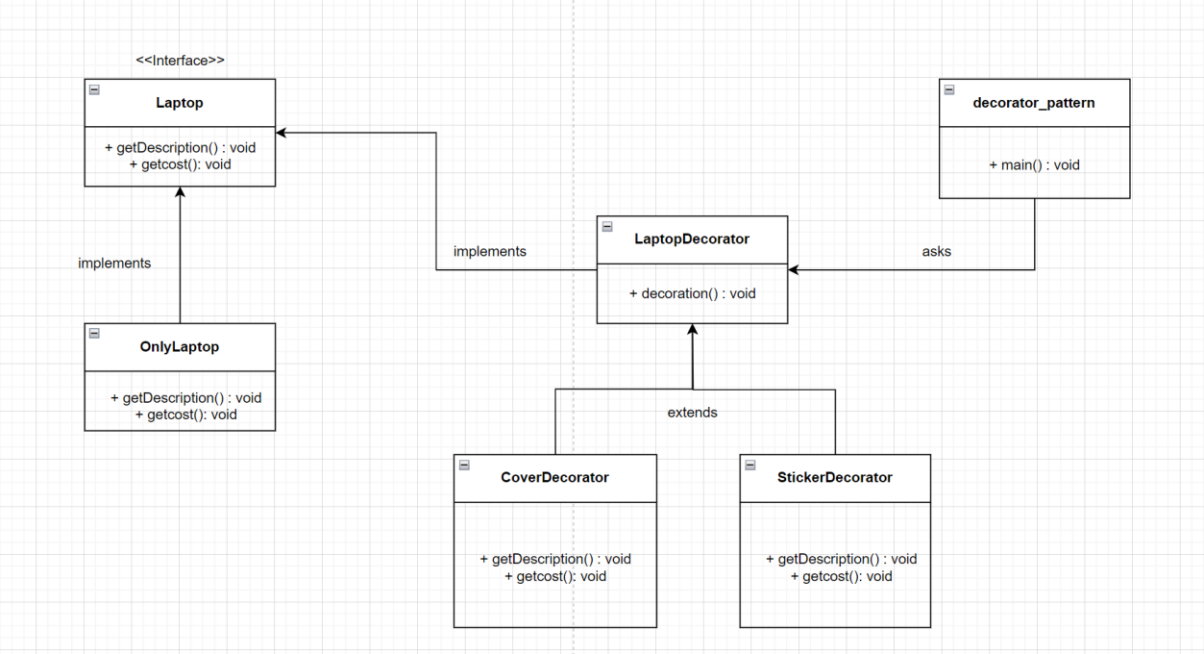
Implementation: -

Here I have implemented the pattern of Laptop that to decorate the laptop extra object need to be created. An interface Laptop is created in which OnlyLaptop implements Laptop and for that two method are created which are getDescription() and getCost().

Laptop can be decorated by adding stickers and taking cover for the laptop so for that two decorators are created cover and laptop which implements laptopdecoractor and LaptopDecoractor implements Laptop.

One main method to created objects and call the function.

UML Diagram: -



Code: -

```
interface Laptop {
    String getDescription();
    double getCost();
}


class OnlyLaptop implements Laptop {
```

```
    @Override
    public String getDescription() {
        return "OnlyLaptop";
    }

    @Override
    public double getCost() {
        return 70000;
    }
}

abstract class LaptopDecorator implements Laptop {
    protected Laptop decoratedLaptop;

    public LaptopDecorator(Laptop decoratedLaptop) {
        this.decoratedLaptop = decoratedLaptop;
    }

    @Override
    public String getDescription() {
        return decoratedLaptop.getDescription();
    }

    @Override
    public double getCost() {
        return decoratedLaptop.getCost();
    }
}


class CoverDecorator extends LaptopDecorator {
    public CoverDecorator(Laptop decoratedLaptop) {
        super(decoratedLaptop);
    }

    @Override
    public String getDescription() {
        return decoratedLaptop.getDescription() + ", Cover";
    }

    @Override
    public double getCost() {
        return decoratedLaptop.getCost() + 5000;
    }
}


class StickerDecorator extends LaptopDecorator {
    public StickerDecorator(Laptop decoratedLaptop) {
```

```
        super(decoratedLaptop);
    }

    @Override
    public String getDescription() {
        return decoratedLaptop.getDescription() + ", Stickers";
    }

    @Override
    public double getCost() {
        return decoratedLaptop.getCost() + 1000;
    }
}

public class decorator_pattern {
    public static void main(String[] args) {

        Laptop laptop = new OnlyLaptop();
        System.out.println("Description: " + laptop.getDescription());
        System.out.println("Cost: " + laptop.getCost());


        Laptop laptop_cover = new CoverDecorator(new OnlyLaptop());
        System.out.println("\nDescription: " + laptop_cover.getDescription());
        System.out.println("Cost: " + laptop_cover.getCost());

        Laptop laptop_cover_stickers = new StickerDecorator(new
CoverDecorator(new OnlyLaptop()));
        System.out.println("\nDescription: " +
laptop_cover_stickers.getDescription());
        System.out.println("Cost: " + laptop_cover_stickers.getCost());
    }
}
```

Output: -

```
PS E:\Fourth sem\Design pattern lab> cd "e:\Fourth sem
va } ; if ($?) { java decorator_pattern }
Description: OnlyLaptop
Cost: 70000.0

Description: OnlyLaptop, Cover
Cost: 75000.0

Description: OnlyLaptop, Cover, Stickers
Cost: 76000.0
PS E:\Fourth sem\Design pattern lab>
```