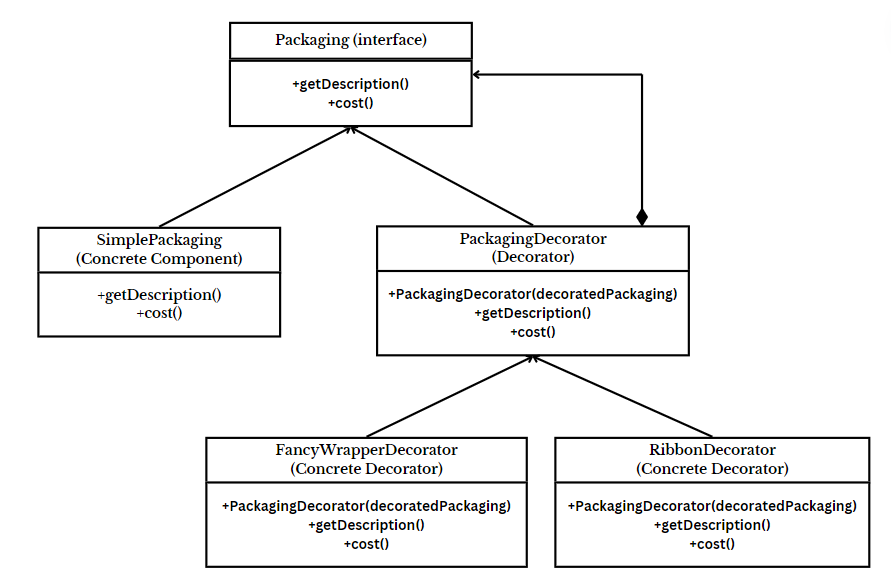
**Decorator Design Pattern**

**Assignment - 2**

Name - Heet Dobariya Roll No. - 22BCP177 Group - G5

* **Program :** Implement decorator design pattern for packaging example.
* **UML Diagram :**



* **Code :**

interface Packaging {

String getDescription();

double cost();

}

class SimplePackaging implements Packaging {

@Override

public String getDescription() {

return "Simple Packaging";

}

@Override

public double cost() {

return 5.0;

}

}

abstract class PackagingDecorator implements Packaging {

protected Packaging decoratedPackaging;

public PackagingDecorator(Packaging decoratedPackaging) {

this.decoratedPackaging = decoratedPackaging;

}

@Override

public String getDescription() {

return decoratedPackaging.getDescription();

}

@Override

public double cost() {

return decoratedPackaging.cost();

}

}

class FancyWrapperDecorator extends PackagingDecorator {

public FancyWrapperDecorator(Packaging decoratedPackaging) {

super(decoratedPackaging);

}

@Override

public String getDescription() {

return super.getDescription() + ", with Fancy Wrapper";

}

@Override

public double cost() {

return super.cost() + 10.0;

}

}

class RibbonDecorator extends PackagingDecorator {

public RibbonDecorator(Packaging decoratedPackaging) {

super(decoratedPackaging);

}

@Override

public String getDescription() {

return super.getDescription() + ", with Ribbon";

}

@Override

public double cost() {

return super.cost() + 7.0;

}

}

class PackagingDecoratorPatternExample {

public static void main(String[] args) {

// Create a simple packaging

Packaging packaging = new SimplePackaging();

System.out.println("Cost: " + packaging.cost() + ", Description: " + packaging.getDescription());

// Decorate the simple packaging with fancy wrapper

Packaging fancyPackaging = new FancyWrapperDecorator(packaging);

System.out.println("Cost: " + fancyPackaging.cost() + ", Description: " + fancyPackaging.getDescription());

// Decorate the simple packaging with ribbon

Packaging ribbonPackaging = new RibbonDecorator(packaging);

System.out.println("Cost: " + ribbonPackaging.cost() + ", Description: " + ribbonPackaging.getDescription());

}

}

* **Output :**

