```
package com.sxt.Io03;
/**
  * 模拟咖啡
  * 1.抽象组件：需要装饰的抽象对象（接口或者抽象类）
  * 2.具体组件：需要装饰的对象
  * 3.抽象装饰类：包含了对抽象组件的引用以及装饰着共有的方法
  * 4.具体装饰类：被装饰的对象
  * @author 江
  *
  */
public class DecorateTest02 {
          public static void main(String[] args) {
                    Drink coffee=new Coffee();
                    Drink suger=new Suger(coffee);
                    System.out.println(suger.info()+"--
>"+suger.cost());
                    Drink milk=new Milk(coffee);
                    System.out.println(milk.info()+"--
>"+milk.cost());
          }

}
//抽象组件
interface Drink{
          double cost();//费用
          String info();//说明
}
//具体组件
class Coffee implements Drink{
       private String name="原味咖啡";
```

```
        @Override
        public double cost() {
                return 10;
        }
        @Override
        public String info() {
                return name;
        }

}
//抽象装饰类
abstract class Decorate implements Drink{
        //对抽象组件的引用
        private Drink drink;
        public Decorate(Drink drink){
                this.drink=drink;
        }
        @Override
        public double cost() {
                return drink.cost();
        }
        @Override
        public String info() {
                return drink.info();
        }

}
//具体装饰类
class Milk extends Decorate{
```

```
        public Milk(Drink drink) {
                super(drink);
        }
        @Override
        public double cost() {
                return super.cost()*4;
        }
        @Override
        public String info() {
                return super.info()+"加入了牛奶";
        }
}
class Suger extends Decorate{
        public Suger(Drink drink) {
                super(drink);
        }
        @Override
        public double cost() {
                return super.cost()*3;
        }
        @Override
        public String info() {
                return super.info()+"加入了蔗糖";
        }
}
```