// Component Interface

interface Coffee {

String getDescription();

double cost();

}

// Concrete Component

class BasicCoffee implements Coffee {

@Override

public String getDescription() {

return "Basic Coffee";

}

@Override

public double cost() {

return 5.0;

}

}

// Decorator Abstract Class

abstract class CoffeeDecorator implements Coffee {

protected Coffee decoratedCoffee;

public CoffeeDecorator(Coffee decoratedCoffee) {

this.decoratedCoffee = decoratedCoffee;

}

public String getDescription() {

return decoratedCoffee.getDescription();

}

public double cost() {

return decoratedCoffee.cost();

}

}

// Concrete Decorators

class MilkDecorator extends CoffeeDecorator {

public MilkDecorator(Coffee decoratedCoffee) {

super(decoratedCoffee);

}

@Override

public String getDescription() {

return decoratedCoffee.getDescription() + ", Milk";

}

@Override

public double cost() {

return decoratedCoffee.cost() + 1.0;

}

}

class SugarDecorator extends CoffeeDecorator {

public SugarDecorator(Coffee decoratedCoffee) {

super(decoratedCoffee);

}

@Override

public String getDescription() {

return decoratedCoffee.getDescription() + ", Sugar";

}

@Override

public double cost() {

return decoratedCoffee.cost() + 0.5;

}

}

// Client code

public class DecoratorPatternDemo {

public static void main(String[] args) {

Coffee coffee = new BasicCoffee();

System.out.println(coffee.getDescription() + " $" + coffee.cost());

Coffee milkCoffee = new MilkDecorator(new BasicCoffee());

System.out.println(milkCoffee.getDescription() + " $" + milkCoffee.cost());

Coffee sugarMilkCoffee = new SugarDecorator(new MilkDecorator(new BasicCoffee()));

System.out.println(sugarMilkCoffee.getDescription() + " $" + sugarMilkCoffee.cost());

}

}