

Паттерн проектирования Decorator

Или почему `new BufferedReader(new InputStreamReader(...))` выглядит именно так

DESIGN PATTERNS SERIES

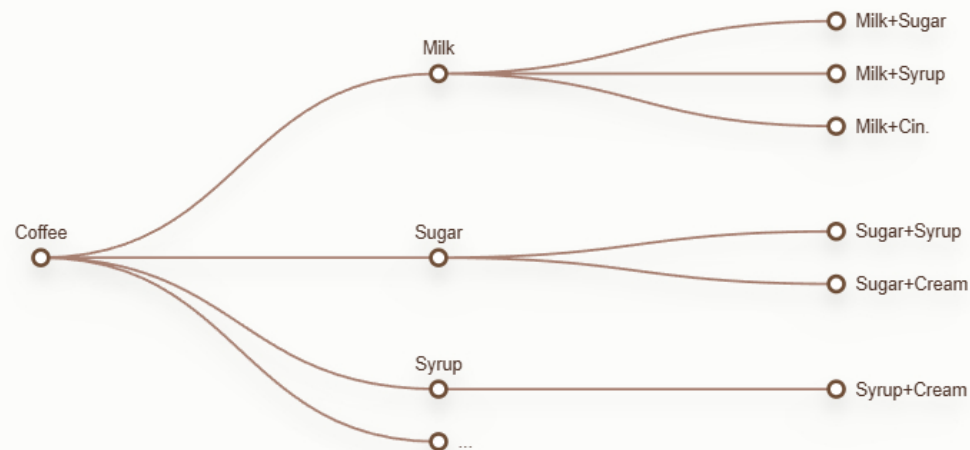
Проблема: Взрыв классов

02

Задача: Расчет стоимости кофе с добавками.

Проблема: Class Explosion («Взрыв» количества классов).

Минус: Сложно поддерживать, статичная структура.



"Class Explosion" Problem

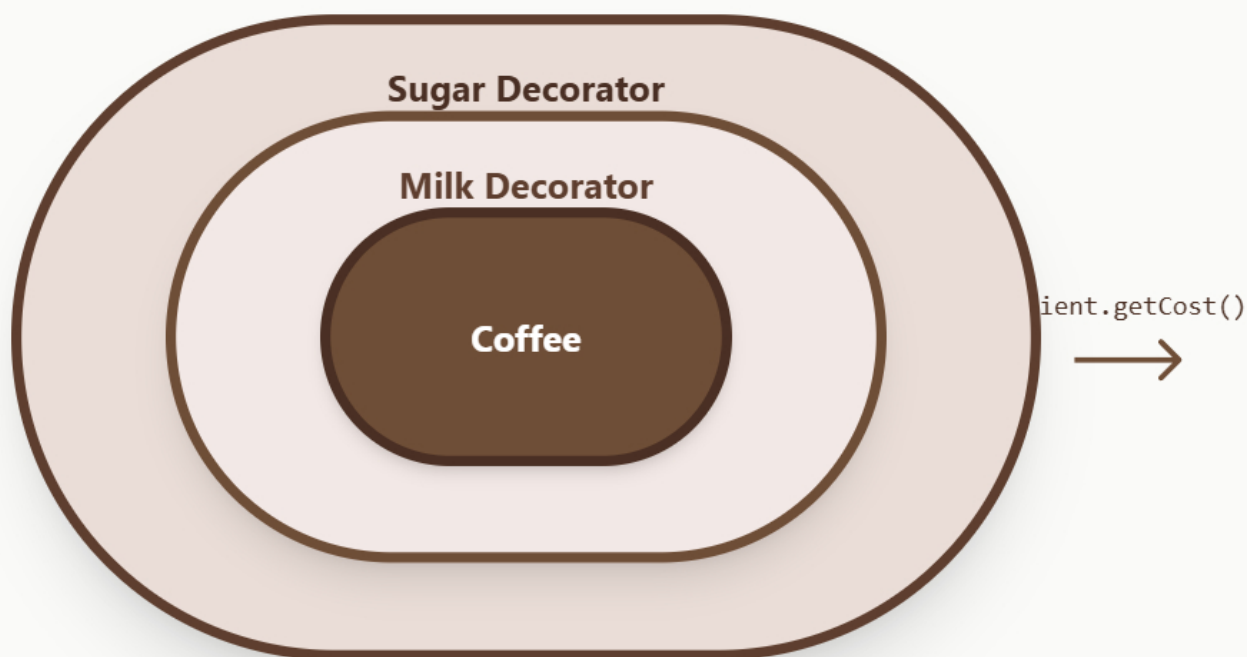
Решение — Обертки (Wrappers)

03

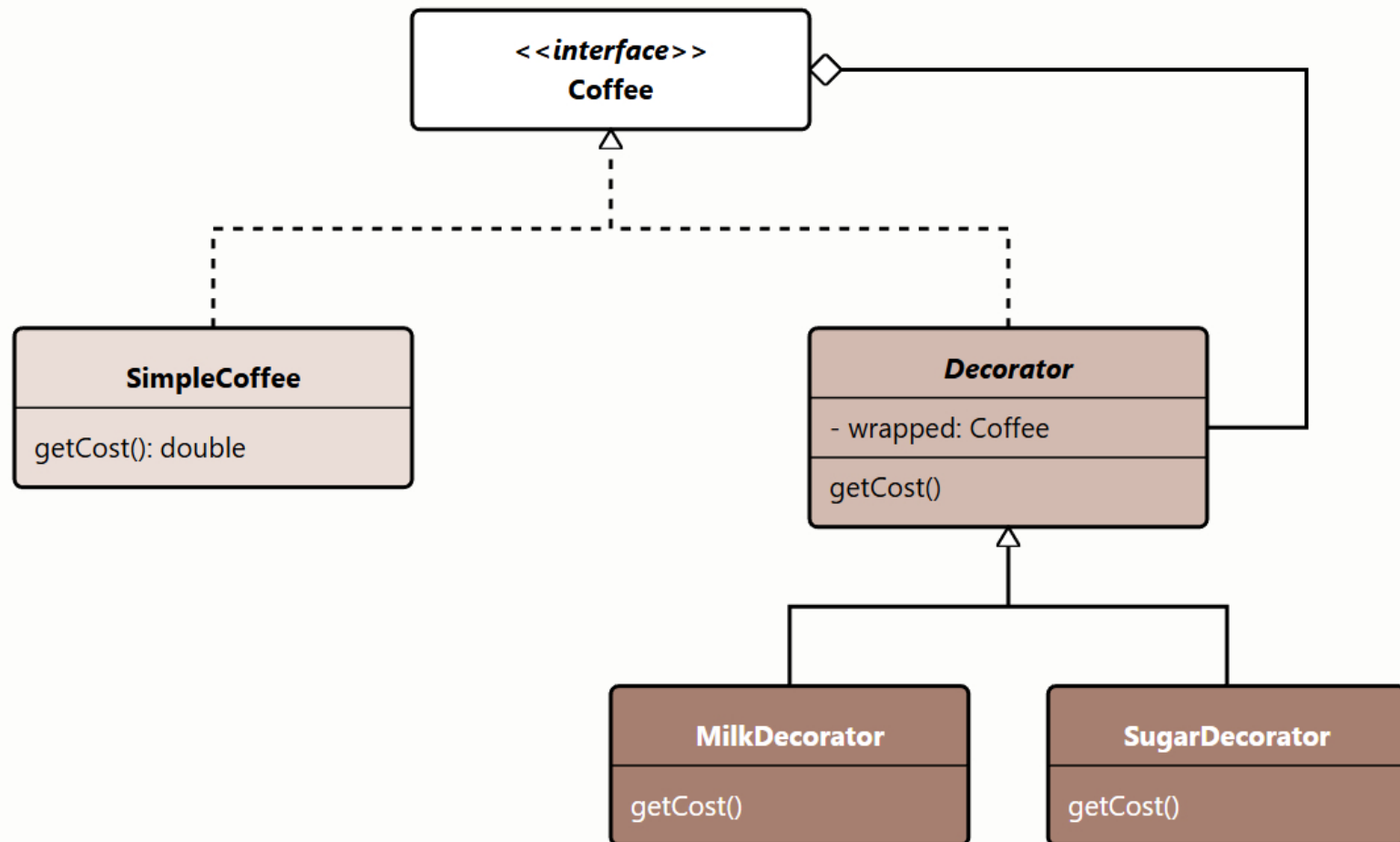
Принцип: Композиция вместо наследования.

Идея: Объект помещается внутрь другого объекта.

Важно: И «Обертка», и «Оборачиваемый объект» имеют один интерфейс.



Calls pass through wrappers, each adding their behavior.



● ● ● JAVA

```
// 1. Общий интерфейс
public interface Coffee {
    double getCost();
}

// 2. Базовый компонент
public class SimpleCoffee implements Coffee {
    public double getCost() { return 100; }
}

// 3. Декоратор
public class MilkDecorator implements Coffee {
    private Coffee tempCoffee; // Ссылка на объект внутри

    public MilkDecorator(Coffee newCoffee) {
        this.tempCoffee = newCoffee;
    }
}
```

● ● ● JAVA

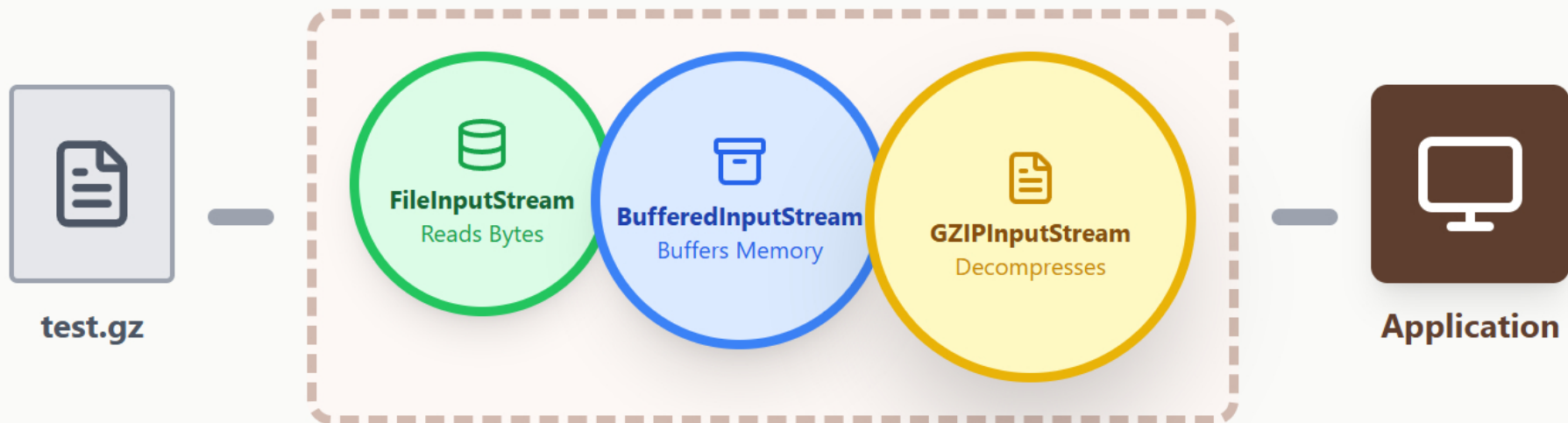
```
public static void main(String[] args) {  
    // 1. Просто кофе  
    Coffee myCoffee = new SimpleCoffee();  
  
    // 2. Добавили молоко и сахар (матрешка)  
    myCoffee = new SugarDecorator(  
        new MilkDecorator(myCoffee)  
    );  
  
    System.out.println(myCoffee.getCost());  
    // Результат: 100 (база) + 30 (молоко) + 10 (сахар) = 140  
}
```

Пример из жизни (Java I/O)

```
new GZIPInputStream(new BufferedInputStream(new FileInputStream("test.gz"...
```

07

```
new GZIPInputStream( new BufferedInputStream( new FileInputStream(...) ))
```



Плюсы и минусы

- + Гибкость выше, чем у наследования.
- + Динамическое добавление функций на лету (runtime).
- + SRP (Принцип единственной ответственности).
- Трудно отлаживать (много мелких объектов).
- Код создания объектов громоздкий (много new).

Декоратор — структурный паттерн.

Расширяет функционал без наследования.

Повсеместно используется в Java (I/O потоки).

Спасибо за внимание!