SOFTWARE ENGINEERING INSTITUTE
华东师范大学软件学院

Object oriented Analysis &Design
面向对象分析与设计

Lecture_18 Design Pattern-Composite

主讲: 陈小红

日期:

笑话-这是什么模式?

- 话说十年前，有一个暴发户，他家有三辆汽车——Benz 奔驰、Bmw 宝马、Audi奥迪，还雇了司机为他开车。不过，暴发户坐车时总是怪怪的：上 Benz 车后跟司机说“开奔驰车！ ”，坐上 Bmw 后他说“开宝马车！”，坐上Audi说“开奥迪车！”。你一定说：这人有病！直接说开车不就行了？！
- 而当把这个暴发户的行为放到我们程序设计中来时，会发现这是一个普遍存在的现象。
- 幸运的是，这种有病的现象在 OO（面向对象）语言中可以避免了。

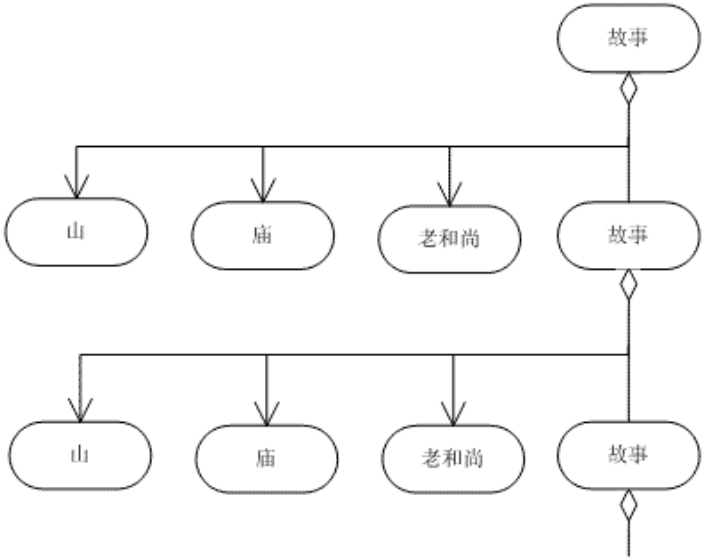
GoF设计模式的分类

- （1）Creational（创建型）5个
- （2）Structural（结构型）7个
- （3）Behavioral（行为型）11个

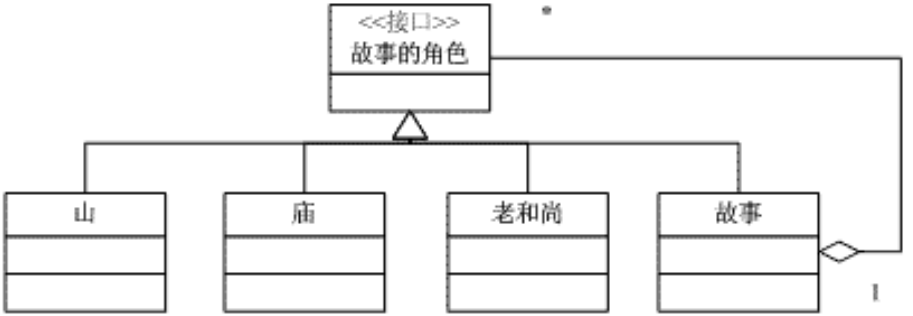
	创建型	结构型	行为型
类	Factory Method	Adapter_Class	Interpreter Template Method
对象	Abstract Factory Builder Prototype Singleton	Adapter_Object Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

介绍合成模式-----从和尚的故事谈起

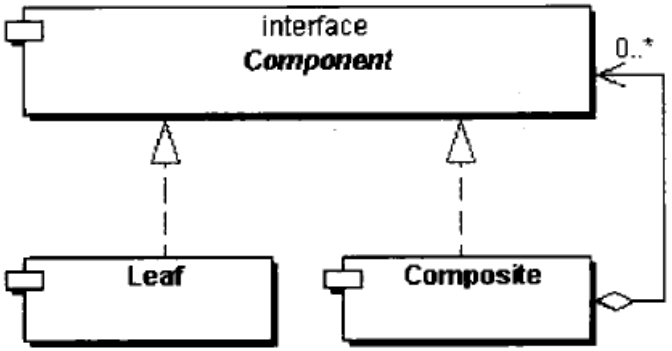
- 从前有座山，山里有个庙，庙里有个老和尚和一个小和尚。一天，老和尚在对小和尚讲故事，故事的内容是：
  - 从前有座山,山里有个庙，庙里有个老和尚和一个小和尚。一天，老和尚在对小和尚讲故事，故事的内容是：
    - 从前有座山，山里有个庙，…


故事
山
庙
老和尚
故事
山
庙
老和尚
故事

和尚的故事



合成模式-类图



•抽象构件（Component）角色：这是一个抽象角色，它给参与组合的对象规定一个接口。这个角色给出共有接口及其默认行为。

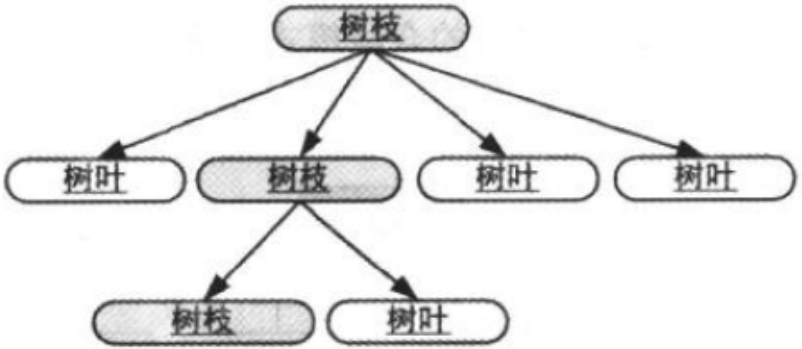
•树叶构件（Leaf）角色：代表参加组合的树叶对象。一个树叶对象没有下级子对象。

•树枝构件（Composite）角色：代表参加组合的有子对象的对象，并给出树枝构件对象的行为。

为什么要有一个公共接口?

- 客户代码在操作时需要有一致性，需要遍历。这些没有公共接口就不能实现。
- 比如:
- 目录可以复制，剪切...，也可以打开下一级，回到上一级。
- 它也规范了合成模式是同接口对象的合成。

典型的composite对象图



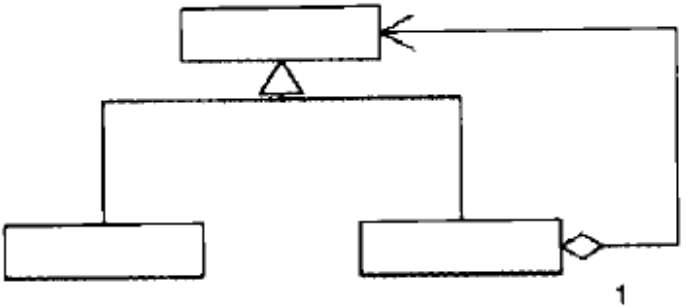
合成模式概述

- 一个类中能包含自身的集合，就叫合成模式.
- 常见的如目录下有目录和文件...
- 树是合成模式的其中的一种抽象.

Composite-合成模式

- 合成模式有时又叫做部分－整体模式（Part-Whole）。
- 合成模式将对象组织到树结构中，可以用来描述整体与部分的关系。
- 合成模式可以使客户端将单纯元素与复合元素同等看待。

树结构类图



树图中的两种结点

- 树枝节点和树叶节点。
  - 树枝节点可以有子节点，
  - 而一个树叶节点不可以有子节点。
  - 除了根节点外，其它节点有且只有一个父节点。

合成模式的实现

- 合成模式可以不提供父对象的管理方法，但合成模式必须在合适的地方提供子对象的管理方法（诸如：add、remove、getChild等）。
- 根据所实现接口的区别分为
  - 安全模式
  - 透明模式
- 这两个形式各有优缺点，需要根据软件的具体情况做出取舍决定。

透明方式I

- 在Component里面声明所有的用来管理子类对象的方法，包括add()、remove()，以及getChild()方法。
- 好处是所有的构件类都有相同的接口。
- 在客户端看来，树叶类对象与合成类对象的区别起码在接口层次上消失了，客户端可以同等同的对待所有的对象。

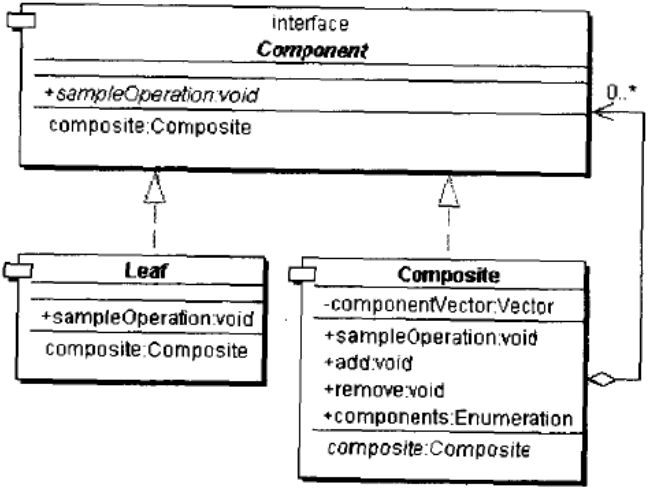
透明方式II

- 缺点是不够安全，因为树叶类对象和合成类对象在本质上是有区别的。树叶类对象不可能有下一个层次的对象，因此add()、remove()以及getChild()方法没有意义，是在编译时期不会出错，而只会在运行时期才会出错。

安全方式

- Composite类里面声明所有的用来管理子类对象的方法。
- 这样的做法是安全的做法，因为树叶类型的对象根本就没有管理子类对象的方法，因此，如果客户端对树叶类对象使用这些方法时，程序会在编译时期出错。
- 缺点是不够透明，因为树叶类和合成类将具有不同的接口。

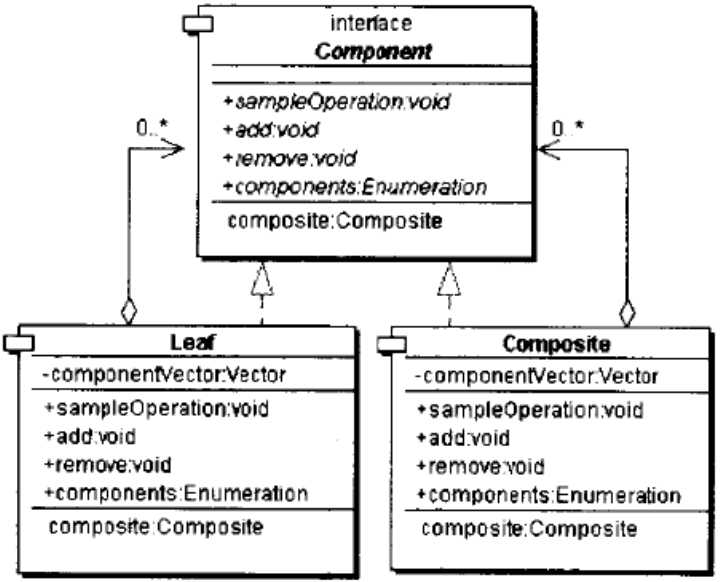
安全式合成模式的结构


interface
Component
+sampleOperation:void
composite:Composite
0..*
Leaf
+sampleOperation:void
composite:Composite
Composite
-componentVector:Vector
+sampleOperation:void
+add:void
+remove:void
+components:Enumeration
composite:Composite

三种角色

- 抽象构件(Component)角色：这是一个抽象角色，它给参加组合的对象定义出公共的接口及其默认行为，可以用来管理所有的子对象。
- 树叶构件（Leaf）角色：树叶对象是没有下级子对象的对象，定义出参加组合的原始对象的行为。
- 树枝构件（Composite）角色：代表参加组合的有下级子对象的对象。树枝对象给出所有的管理子对象的方法，如add（）、remove（）、getChild（）等。

透明式的合成模式结构


interface
Component
+sampleOperation:void
+add:void
+remove:void
+components:Enumeration
composite:Composite
0..*
0..*
Leaf
-componentVector:Vector
+sampleOperation:void
+add:void
+remove:void
+components:Enumeration
composite:Composite
Composite
-componentVector:Vector
+sampleOperation:void
+add:void
+remove:void
+components:Enumeration
composite:Composite

三个角色

- 抽象构件（Component）角色：这是一个抽象角色，它给参加组合的对象规定一个接口，规范共有的接口及默认行为。
- 树叶构件（Leaf）角色：代表参加组合的树叶对象，定义出参加组合的原始对象的行为。树叶类会给出add（）、remove（）以及getChild（）之类的用来管理子类对对象的方法的平庸实现。
- 树枝构件（Composite）角色：代表参加组合的有子对象的对象，定义出这样的对象的行为。

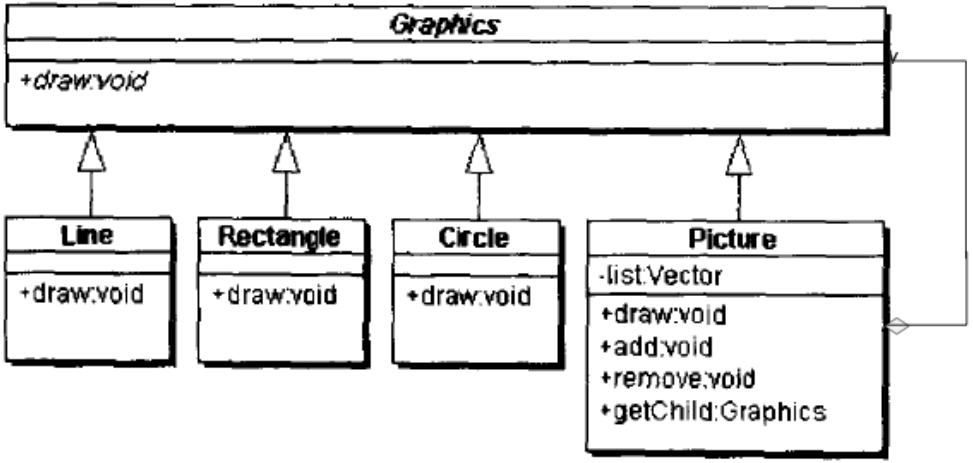
适用性

- 以下情况适用合成模式
  - 你想表示对象的部分-整体层次结构
  - 你希望用户忽略组合对象与单个对象的不同，用户将统一地使用组合结构中的所有对象。

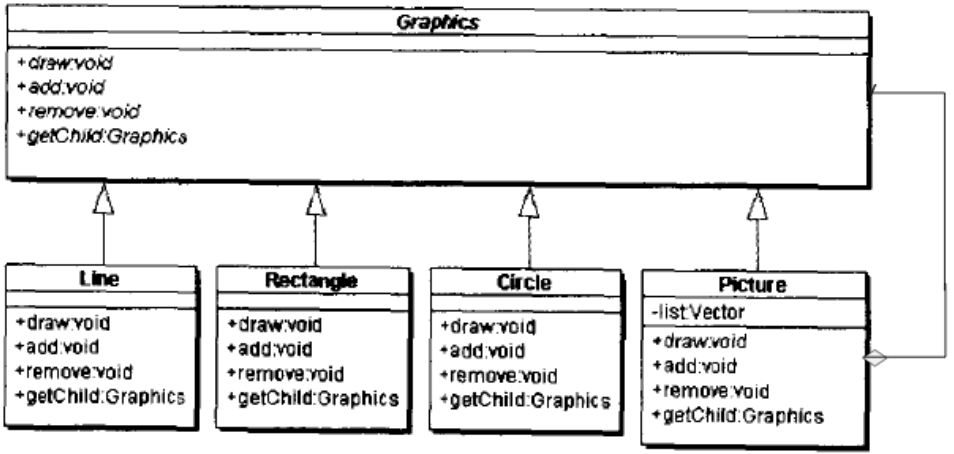
案例-一个绘图的例子

- 一个绘图系统给出各种工具用来描绘由线、长方形和圆形等基本图形组成的图形。
- 可以想象，设计应当包括Line、Rectangle和Circle等对象，而且每一个对象都应当配备有一个draw()方法，在调用时，会画出对象所代表的图形。

应用安全式合成模式


Graphics
+draw:void
Line
+draw:void
Rectangle
+draw:void
Circle
+draw:void
Picture
-list:Vector
+draw:void
+add:void
+remove:void
+getChild:Graphics

应用透明式的合成模式


Graphics
+draw:void
+add:void
+remove:void
+getChild:Graphics
Line
+draw:void
+add:void
+remove:void
+getChild:Graphics
Rectangle
+draw:void
+add:void
+remove:void
+getChild:Graphics
Circle
+draw:void
+add:void
+remove:void
+getChild:Graphics
Picture
-list:Vector
+draw:void
+add:void
+remove:void
+getChild:Graphics

The end

- Thank you for your attention to this lesson!