设计模式 - 建造者模式

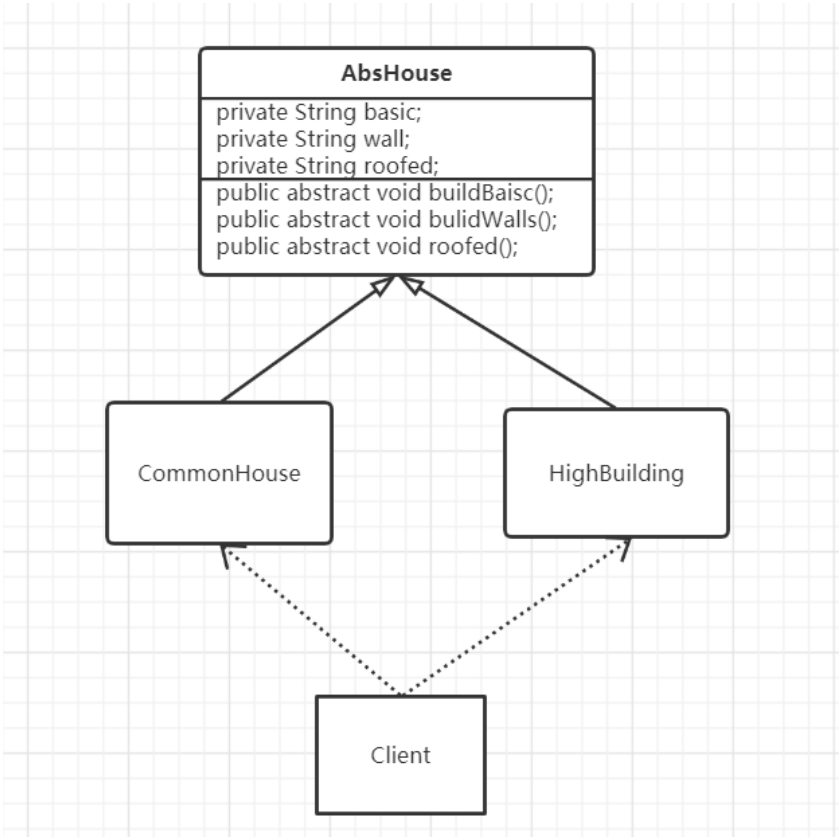
寂然

案例演示 - 工程项目

> 一，建造房屋的过程有筑地基、砌墙、封顶等操作
>
> 二，房子有各种各样的，比如普通砖瓦房，高楼，别墅，各种房子的过程虽然一样，但是要求不一样
>
> 三，请编写程序，完成需求

解决方案一

如何完成业务逻辑?



案例分析

完成了业务逻辑

优势

容易想到，容易理解

劣势

这种设计方案，把产品和创建产品的流程封装在了一起，耦合性是很高的

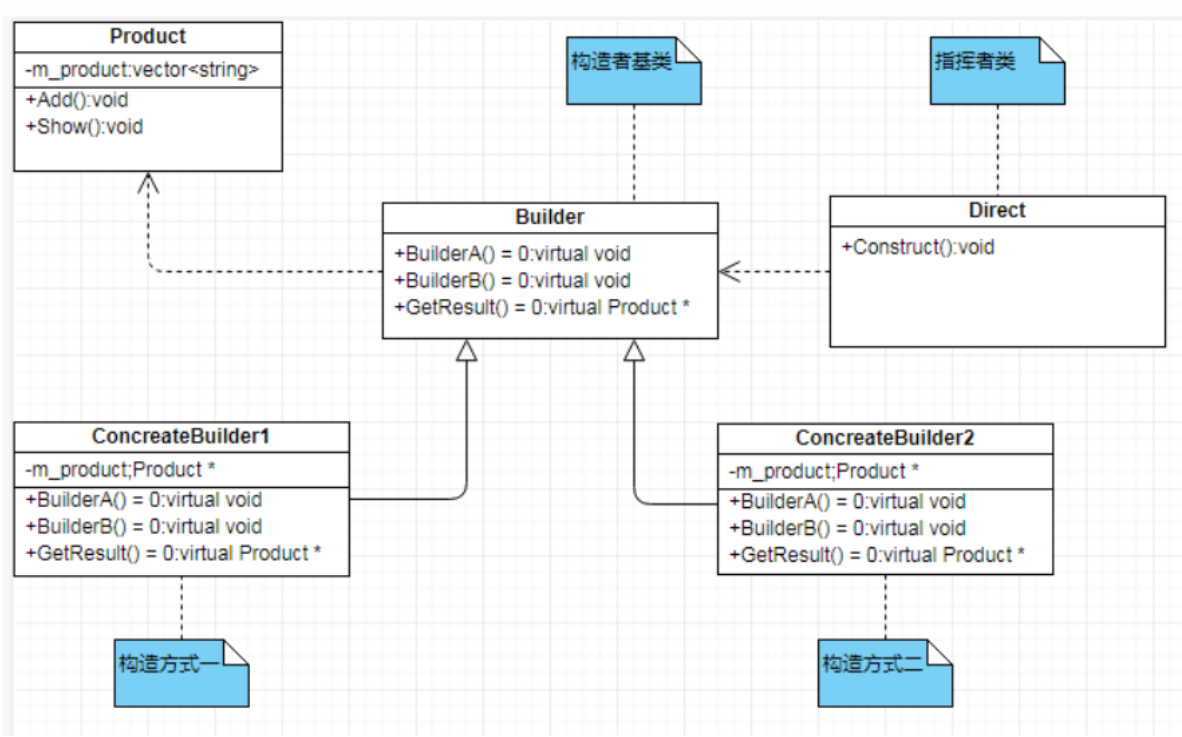
现在所有的创建细节都是暴露给客户端的，？？？

能不能内部定义创建的顺序以及过程？？？

基本介绍

> 建造者模式（Builder Pattern） 又叫生成器模式，是一种对象构建模式
>
> 它可以将复杂对象的建造过程抽象出来，使这个抽象过程的不同实现方法可以构造出不同表现的对象
>
> 建造者模式是一步一步创建一个复杂的对象，它允许用户只通过指定复杂对象的类型和内容就可以构建它
>
> 们， 用户不需要知道内部的具体构建细节

原理类图



建造者模式的四个角色

product：产品角色，一个具体的产品对象

Builder：抽象建造者，创建一个产品对象的各个部件指定的一个接口或者是抽象类

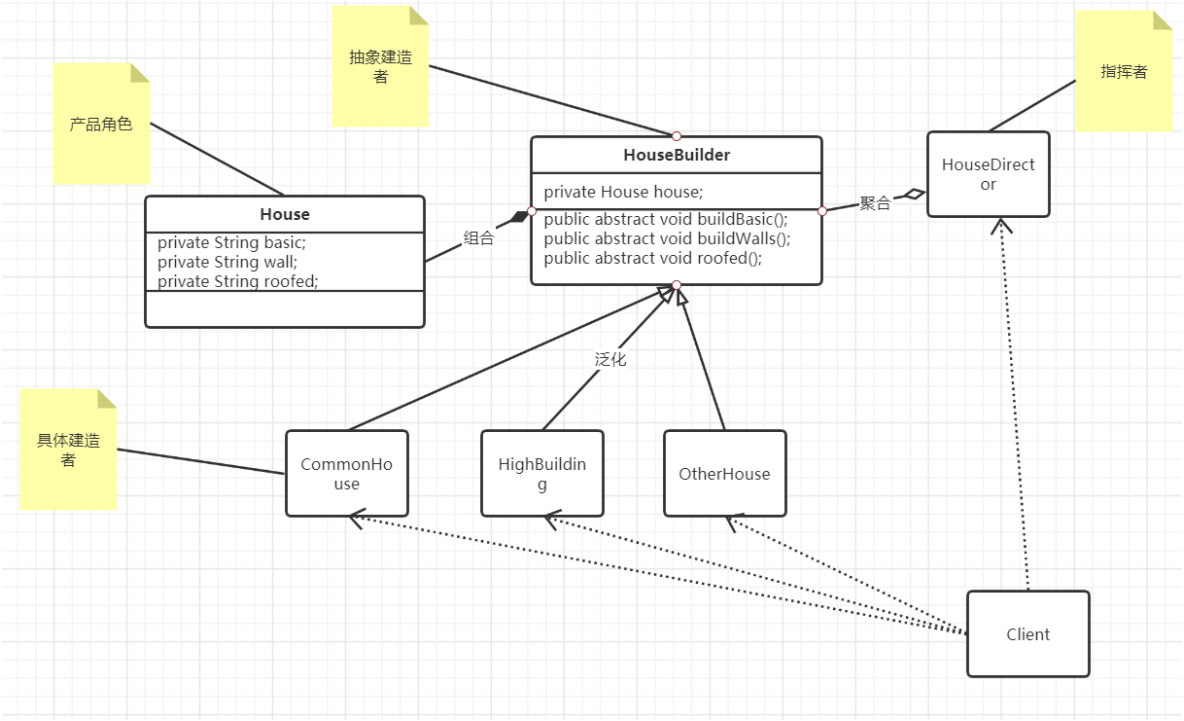
ConcreateBuilder：具体建造者，实现Buidler接口，构建和装配各个部件，来实现具体的细节

Director：指挥者，构建了一个使用Builder接口的对象，用来创建复杂的对象，主要有两个作用

- 隔离了客户与对象的生产过程
- 负责控制产品的生产过程，可以来定义具体的创建顺序

解决方案二 - 建造者模式

思路 - 类图



实操 - 代码演示

使用建造者模式

注意事项

优势

- 将产品本身与产品的创建过程解耦合
- 扩展性很高，很方便的增加或者替换具体建造者
- 方便用程序来控制产品的创建过程以及顺序
- 指挥者类针对抽象建造者编程，系统扩展方便，符合开闭原则

注意点

- 建造者模式创建的产品一般组成部分类似，如果产品差异性非常大，不适合了，，合理利用
- 如果产品内部变化复杂，**导致系统非常庞大**，权衡

抽象工厂模式 vs 建造者模式？

抽象工厂模式： 异地披萨订购，具有不同维度的产品组合，不需要关心构建过程，**关心的是什么样工厂去生产什**

么样的产品

建造者模式：按照指定的蓝图构建产品，**需要考虑组装的过程以及顺序** --》 指挥者

StringBuilder 源码分析

- Appendable ： 充当的就是抽象建造者的角色
- AbstractStringBuilder：其实已经是一个具体建造者了
- StringBuilder：充当了指挥者的角色，但是同时充当了具体建造者

不完全一样，很多个角色，思想是很相似的

XXXBuilder 建造者模式

下节预告

适配器模式