**来源：**[**https://blog.csdn.net/liang19890820/article/details/66974516**](https://blog.csdn.net/liang19890820/article/details/66974516)

# 概念

聚合：表示两个对象之间是整体和部分的弱关系，部分的生命周期可以超越整体。如电脑和鼠标。

组合：表示两个对象之间是整体和部分的**强**关系，部分的生命周期不能超越整体，或者说不能脱离整体而存在。组合关系的“部分”，是不能在整体之间进行共享的。

# 创建型模式

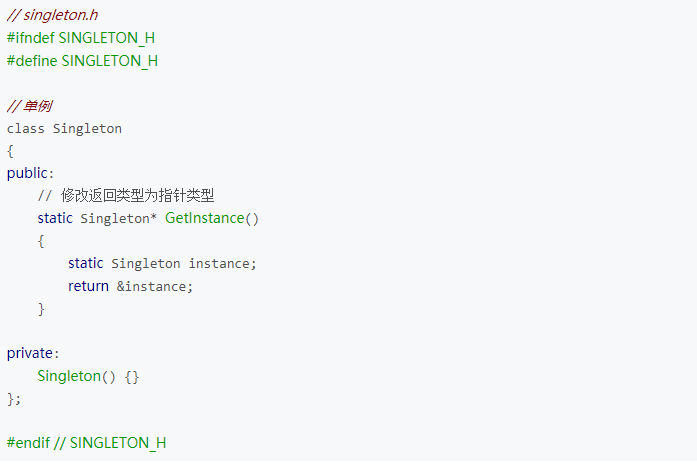
1. 单例模式

保证一个类仅有一个实例，并提供一个访问它的全局访问点。

生活举例：

比如小公司所有的考勤都要通过进门的打卡机进行打卡





1. 抽象工厂模式（Abstract Factory Pattern）

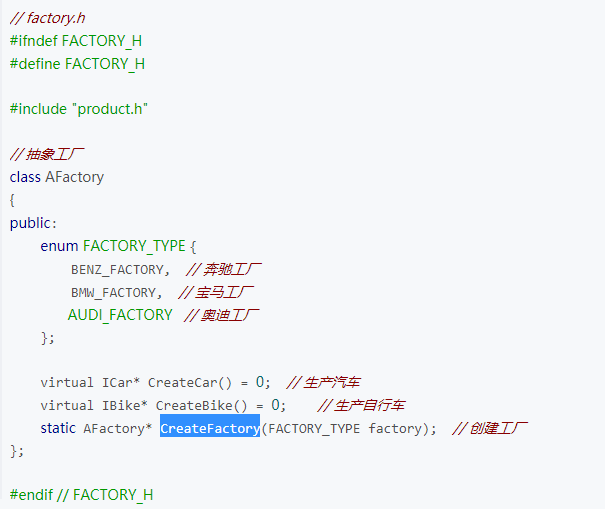
提供一个创建一系列相关或相互依赖对象的接口，而无需指定它们具体的类。

生活举例：

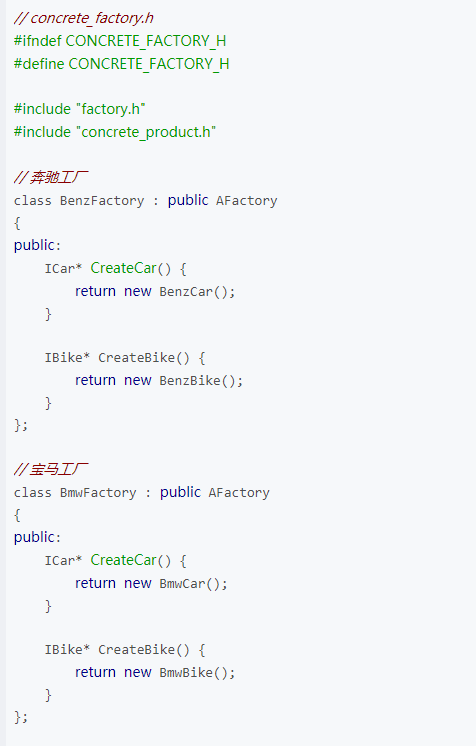
比如装修，需要走水电、安装地板、刷墙、吊顶、买家具等，如果通过装修公司，则只需要告诉他们我需要做什么，装修公司即可输出对应的操作，我直接面对的就是装修公司，后面的具体细节，我不需要关心；感觉有点像代理，这也确实是一个对外接口。



父类是可以调用派生类里面的接口的



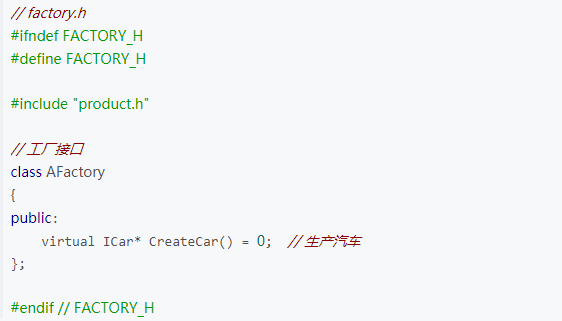


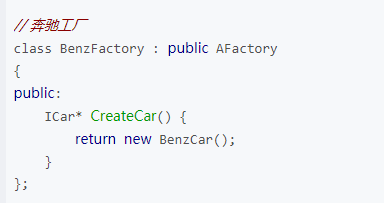


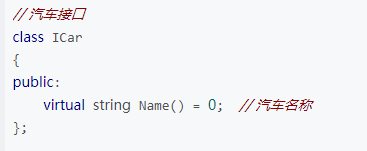
1. 工厂方法模式（Factory Method Pattern）

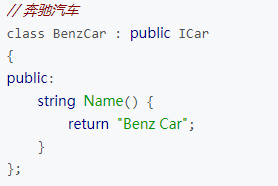
工厂方法模式（Factory Method Pattern）是一种常用的对象创建型设计模式，此模式的核心思想是封装类中不变的部分，提取其中个性化善变的部分为独立类，通过依赖注入以达到解耦、复用以及方便后期维护拓展的目的。











1. 建造者模式

将一个复杂对象的构建与它的表示分离，使得同样的构建过程可以创建不同的表示。





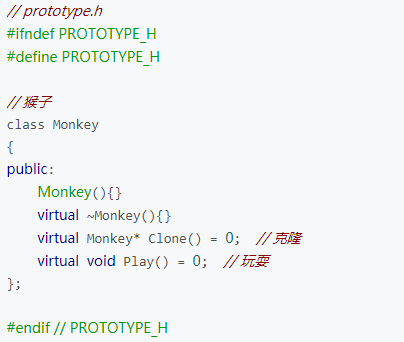




1. 原型模式

用原型实例指定创建对象的种类，并且通过拷贝这个原型来创建新的对象。







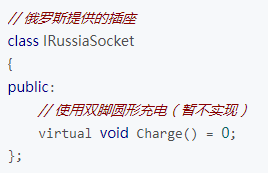


# 结构性模式

1. 适配器模式

将一个类的接口转换成客户希望的另外一个接口。Adapter 模式使得原本由于接口不兼容而不能一起工作的那些类可以一起工作。





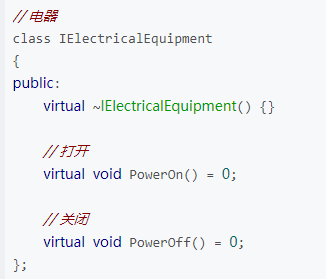




1. 桥接模式

将抽象部分与它的实现部分分离，使它们都可以独立地变化。







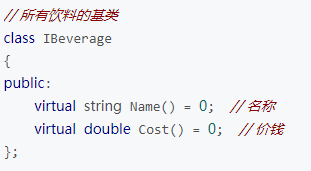




1. 装饰者模式

动态地给一个对象添加一些额外的职责。就扩展功能而言，它比生成子类方式更为灵活。











1. 组合模式

安全组合模式



* Component（抽象构件）：为叶子构件和容器构件对象定义接口，可以包含所有子类共有行为的声明和实现。在抽象构件中，声明了访问及管理子构件的接口（例如：Add()、Remove()、GetChild() 等）。
* Leaf（叶子构件）：叶子节点没有子节点。它实现了 Component 中定义的行为，对于访问及管理子构件的接口，可以通过异常等方式进行处理。
* Composite（容器构件）：容器节点包含子节点（可以是叶子构件，也可以是容器构件）。它提供了一个集合用于存储子节点，实现了 Component 中定义的行为，包括访问及管理子构件的接口，在其业务方法中可以递归调用其子节点的业务方法

适用场景：

* 表示对象的“整体-部分”层次结构（树形结构）
* 希望用户忽略组合对象与单个对象的不同，统一地使用组合结构中的所有对象。

1. 代理模式

为其他对象提供一个代理以控制对这个对象的访问