合成复用原则(Composite Reuse Principle, CRP)：尽量使用对象组合，而不是继承来达到复用的目的。

又称合成/聚合复用原则(Composite/Aggregate Reuse Principle，CARP)

合成复用原则就是在一个新的对象里通过关联关系（包括组合关系和聚合关系）来使用一些已有的对象，使之成为新对象的一部分；新对象通过委派调用已有对象的方法达到复用功能的目的。简言之：**复用时要尽量使用组合/聚合关系（关联关系），少用继承。**

为什么说要少用继承呢？

在面向对象设计中，可以通过两种方法在不同的环境中复用已有的设计和实现，即**通过组合/聚合关系或通过继承，但首先应该考虑使用组合/聚合**，组合/聚合可以使系统更加灵活，降低类与类之间的耦合度，一个类的变化对其他类造成的影响相对较少；**其次才考虑继承**，在使用继承时，需要严格遵循里氏代换原则，有效使用继承会有助于对问题的理解，降低复杂度，而滥用继承反而会增加系统构建和维护的难度以及系统的复杂度，因此需要慎重使用继承复用。

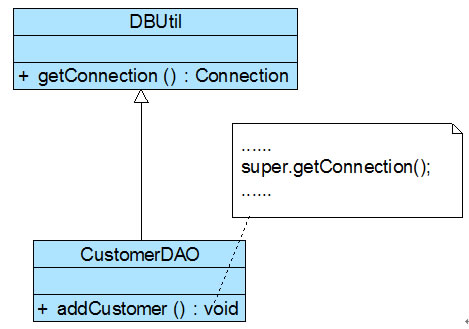
通过继承来进行复用的主要问题在于继承复用会破坏系统的封装性，因为继承会将基类的实现细节暴露给子类，由于基类的内部细节通常对子类来说是可见的，所以这种复用又称“白箱”复用，如果基类发生改变，那么子类的实现也不得不发生改变；从基类继承而来的实现是静态的，不可能在运行时发生改变，没有足够的灵活性；而且继承只能在有限的环境中使用（如类没有声明为不能被继承）。

由于组合或聚合关系可以将已有的对象（也可称为成员对象）纳入到新对象中，使之成为新对象的一部分，因此新对象可以调用已有对象的功能，这样做可以使得成员对象的内部实现细节对于新对象不可见，所以这种复用又称为“黑箱”复用，相对继承关系而言，其耦合度相对较低，成员对象的变化对新对象的影响不大，可以在新对象中根据实际需要有选择性地调用成员对象的操作；合成复用可以在运行时动态进行，新对象可以动态地引用与成员对象类型相同的其他对象。

一般而言，如果两个类之间是“Has-A”的关系应使用组合或聚合，如果是“Is-A”关系可使用继承。"Is-A"是严格的分类学意义上的定义，意思是**一个类是另一个类的"一种"**；而"Has-A"则不同，它表示**某一个角色具有某一项责任**。

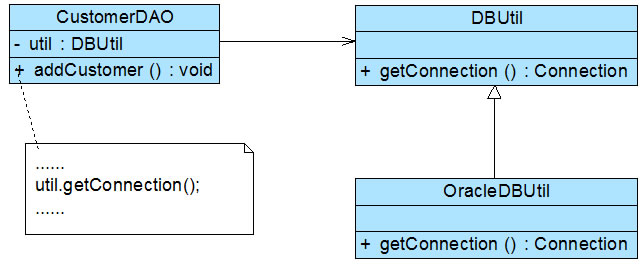
实例：

Sunny软件公司开发人员在初期的CRM系统设计中，考虑到客户数量不多，系统采用MySQL作为数据库，与数据库操作有关的类如CustomerDAO类等都需要连接数据库，连接数据库的方法getConnection()封装在DBUtil类中，由于需要重用DBUtil类的getConnection()方法，设计人员将CustomerDAO作为DBUtil类的子类，初始设计方案结构如图所示：



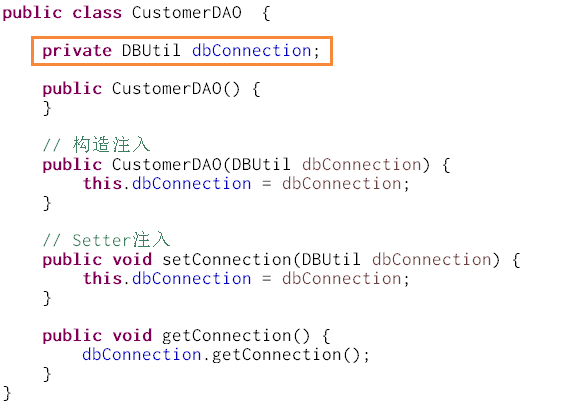
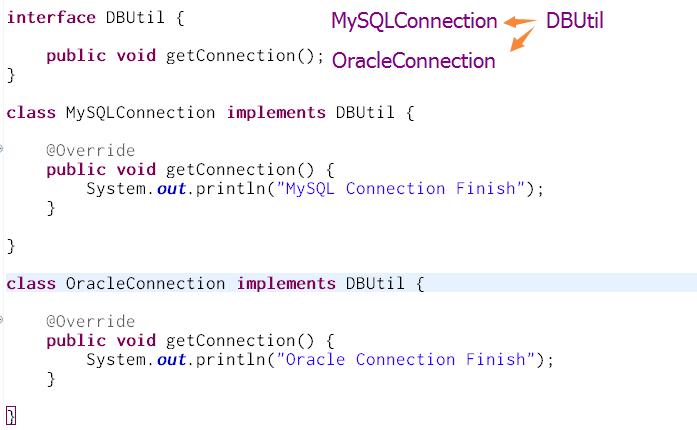
随着客户数量的增加，系统决定升级为Oracle数据库，因此需要增加一个新的OracleDBUtil类来连接Oracle数据库，由于在初始设计方案中CustomerDAO和DBUtil之间是继承关系，因此在更换数据库连接方式时需要修改CustomerDAO类的源代码，将CustomerDAO作为OracleDBUtil的子类，这将违反开闭原则。（当然也可以修改DBUtil类的源代码，同样会违反开闭原则。）

根据合成复用原则进行重构：



在图中可知，CustomerDAO和DBUtil之间的关系由**继承关系**变为**关联关系（对象关联）**，采用依赖注入的方式**将DBUtil对象注入到CustomerDAO中**，可以使用构造注入，也可以使用Setter注入。如果需要对DBUtil的功能进行扩展，可以通过其子类来实现，如通过子类OracleDBUtil来连接Oracle数据库。由于CustomerDAO针对DBUtil编程，根据里氏代换原则，DBUtil子类的对象可以覆盖DBUtil对象，只需在CustomerDAO中注入子类对象即可使用子类所扩展的方法。例如在CustomerDAO中注入OracleDBUtil对象，即可实现Oracle数据库连接，原有代码无须进行修改，而且还可以很灵活地增加新的数据库连接方式。

代码实现：



[面向对象设计原则之合成复用原则](http://blog.csdn.net/lovelion/article/details/7563441" \t "https://blog.csdn.net/LoveLion/article/details/_blank)