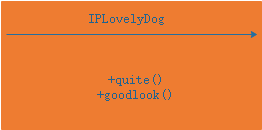
# 接口隔离原则

类不应该依赖它不需要的接口，类间的依赖关系应该建立在最小接口上。简单来说就是建立单一的接口，不要建立庞大臃肿的接口，接口粒度尽量细化，接口中同时存在的方法尽量少，保持接口的纯洁性。

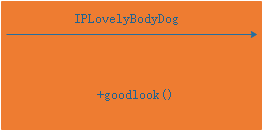
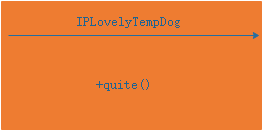
例如:

我们使用IPLovelyDog接口来形容一直可爱的狗，这只狗外形好看，并且安静。



但是并不是所有人都喜欢安静的狗狗，有的人喜欢活泼的，所以这个接口就不适合所有人。

如果把接口设计成



把可爱的狗狗接口拆分成外在与性格的可爱，这样喜欢安静狗狗的人就可以实现上面两个接口，而喜欢活泼的人则可以再建一个接口来描述狗狗活泼。这样设计接口会更加灵活或，增强了代码的复用性且更为灵活。

# 接口隔离原则总结

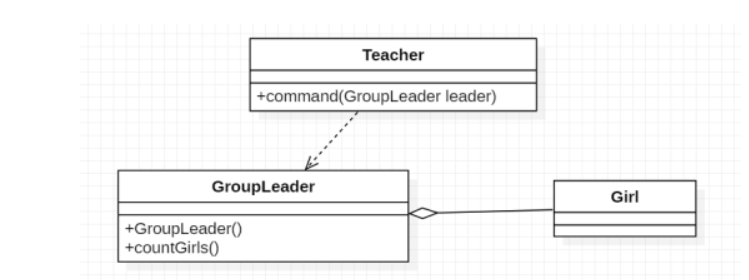
1. 接口尽量细化粒度，保持接口的纯洁性。
2. 接口要高内聚，减少对外交互。

# 迪米特法则

定义：一个软件内的实体应当尽可能少的与其他实体发生相互作用。迪米特法则初衷在于降低类之间的耦合。

例如:

每个体育老师上课前都要清点学生人数，教师一般不是自己数，而是委托班长等人去庆典，即教师通过下达命令给班长要求清点人数。



这样以来，教师与学生的耦合度就会降低。

# 迪米特法则总结：

类定义时尽量内联，少使用public 权限修饰符，尽量使用private,protected等修饰符

# 合成复用原则

通常类的服用分为继承复用和合成复用的特点，虽然继承复用简单且容易实现，但是有以下缺点:

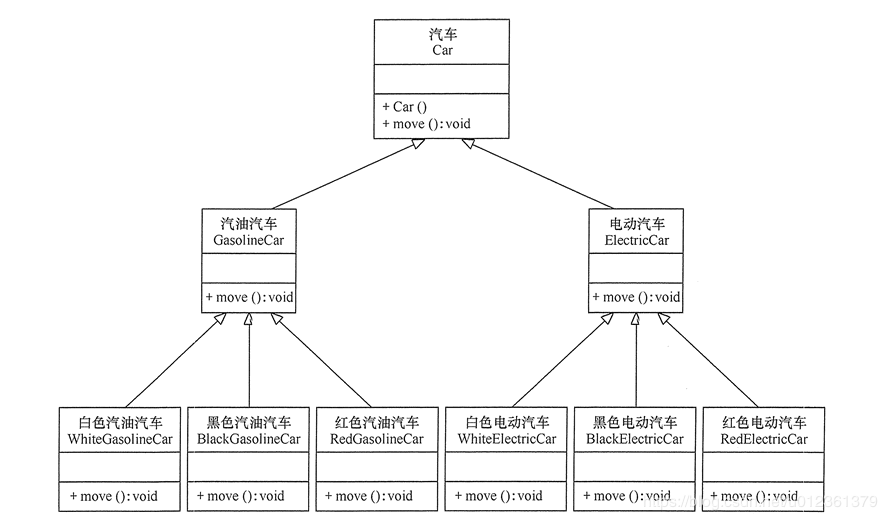
1. 破环了父类的封装性，父类细节完全暴露给子类。
2. 子类和父类的耦合度高，父类实现的任何改变都会导致子类改变，不利于扩展和维护。
3. 它限制了复用的灵活性。从父类继承而来的实现是静态的，在编译时已经定义，所以在运行时不可能发生变化。

解决方案:

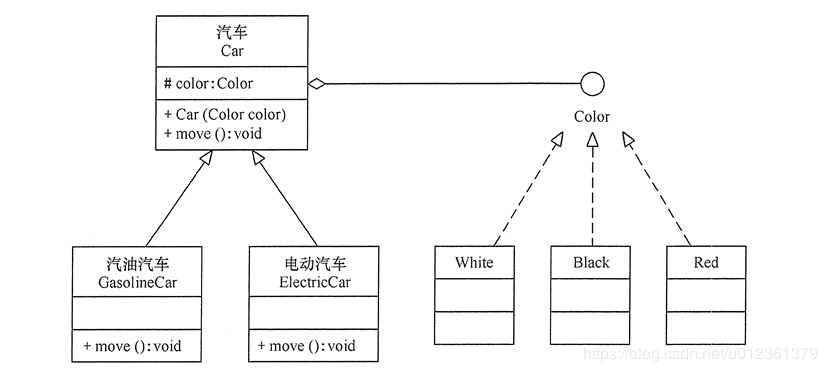
合成复用原则时通过将已有的对象纳入新对象中，作为新对象的成员来实现，新对象可以调用已有对象的功能从而达到复用。

例如：

分析：汽车按“动力源”划分可分为汽油汽车、电动汽车等；按“颜色”划分可分为白色汽车、黑色汽车和红色汽车等。如果同时考虑这两种分类，其组合就很多。图 1 所示是用继淨：关系实现的汽车分类的类图。



从图中可看出，继承可以产生很多子类，如果增加新的“动力源”或者新的“颜色”都要改动源码，这违背了开闭原则。如果采用采用继承和组合关系就可以很好解决上面问题。



采用组合或聚合复用时，可以将已有对象纳入新对象中，使之成为新对象的一部分，新对象可以调用已有对象的功能，它有以下优点：

1. 维持了类的封装性，因为成分对象的内部细节时新对象看不见的。
2. 新旧类之间耦合度降低。复用所需要的依赖较少，新对象存取对象的唯一方法时通过成分对象的接口。
3. 复用的灵活性高。这种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相同的对象。