里氏代换原则(Liskov Substitution Principle, LSP)：所有引用基类（父类）的地方必须能透明地使用其子类的对象。

来个严格表述版本：如果对每一个类型为S的对象o1，都有类型为T的对象o2，使得以T定义的所有程序P在所有的对象o1代换o2时，程序P的行为没有变化，那么类型S是类型T的子类型。

里氏代换原则告诉我们，在软件中将一个基类对象替换成它的子类对象，程序将不会产生任何错误和异常，反过来则不成立，如果一个软件实体使用的是一个子类对象的话，那么它不一定能够使用基类对象。

里氏代换原则是**实现开闭原则的重要方式之一**，由于使用基类对象的地方都可以使用子类对象，因此在程序中尽量使用基类类型来对对象进行定义，而在运行时再确定其子类类型，用子类对象来替换父类对象。

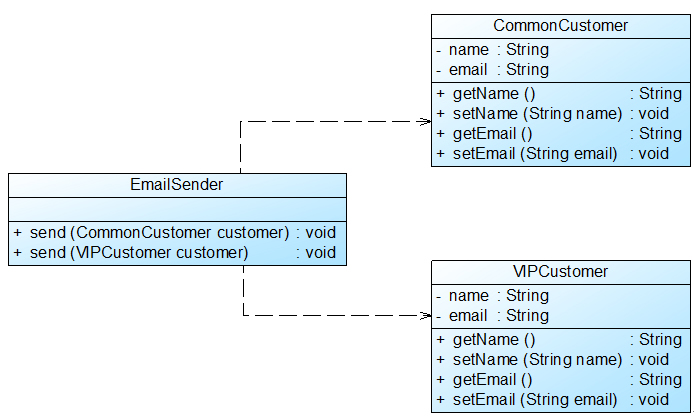
在使用里氏代换原则时需要注意如下问题：

(1)子类的所有方法必须在父类中声明，或子类必须实现父类中声明的所有方法。根据里氏代换原则，为了保证系统的扩展性，在程序中通常使用父类来进行定义，如果一个方法只存在子类中，在父类中不提供相应的声明，则无法在以父类定义的对象中使用该方法。

(2)  我们在运用里氏代换原则时，尽量把父类设计为抽象类或者接口，让子类继承父类或实现父接口，并实现在父类中声明的方法，运行时，子类实例替换父类实例，我们可以很方便地扩展系统的功能，同时无须修改原有子类的代码，增加新的功能可以通过增加一个新的子类来实现。里氏代换原则是开闭原则的具体实现手段之一。

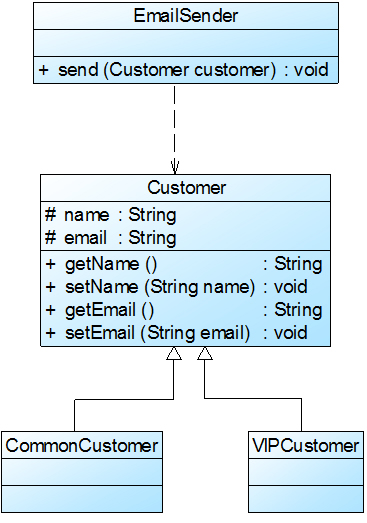
(3) Java语言中，在编译阶段，Java编译器会检查一个程序是否符合里氏代换原则，这是一个与实现无关的、纯语法意义上的检查，但Java编译器的检查是有局限的。

实例：客户(Customer)可以分为VIP客户(VIPCustomer)和普通客户(CommonCustomer)两类，系统需要提供一个发送Email的功能。



在对系统进行进一步分析后发现，无论是普通客户还是VIP客户，发送邮件的过程都是相同的，也就是说两个send()方法中的代码重复，而且在本系统中还将增加新类型的客户。为了让系统具有更好的扩展性，同时减少代码重复，使用里氏代换原则对其进行重构。

根据里氏代换原则进行重构：



增加一个新的抽象客户类Customer，而将CommonCustomer和VIPCustomer类作为其子类，邮件发送类EmailSender类针对抽象客户类Customer编程，根据里氏代换原则，能够接受基类对象的地方必然能够接受子类对象，因此将EmailSender中的send()方法的参数类型改为Customer，如果需要增加新类型的客户，只需将其作为Customer类的子类即可。

里氏代换原则是实现开闭原则的重要方式之一。在本实例中，在传递参数时使用基类对象，除此以外，在**定义成员变量、定义局部变量、确定方法返回类型时**都可使用里氏代换原则。针对基类编程，在程序运行时再确定具体子类。

趣味拓展：

里氏代换原则由2008年图灵奖得主、美国第一位计算机科学女博士Barbara Liskov教授和卡内基·梅隆大学Jeannette Wing教授于1994年提出。

[面向对象设计原则之里氏代换原则](http://blog.csdn.net/lovelion/article/details/7540445" \t "https://blog.csdn.net/LoveLion/article/details/_blank)