**面向对象的基本设计思想——SOLID原则**

**一、开闭原则（OCP）**

【定义】**软件实体应当对扩展开放，对修改关闭。**

【释义】当应用的需求改变时，在不修改软件实体的源代码或者二进制代码的前提下，可以扩展模块的功能（通常通过继承），使其满足新的需求。

【例1】Windows 的桌面主题设计：Windows 的主题是桌面背景图片、窗口颜色和声音等元素的组合。用户可以根据自己的喜爱更换自己的桌面主题，也可以从网上下载新的主题。这些主题有共同的特点，可以为其定义一个抽象类（Abstract Subject），而每个具体的主题（Specific Subject）是其子类。用户窗体可以根据需要选择或者增加新的主题，而不需要修改原代码，所以它是满足开闭原则的，其类图如图 1 所示。

【例2】手机外壳的设计：不同的手机外壳为不同的子类。

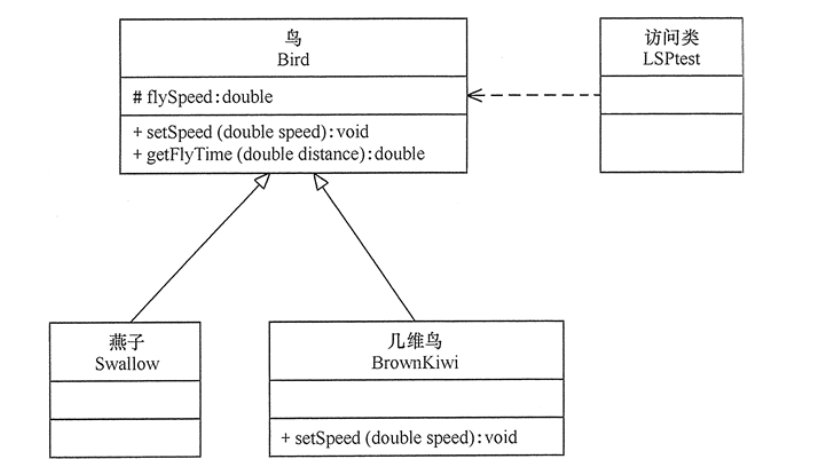
**二、里氏替换原则（LSP）**

【定义】**继承必须确保超类所拥有的性质在子类中仍然成立。**

【释义】子类可以扩展父类的功能，但不能改变父类原有的功能。里氏替换原是继承复用的基础，它反映了基类与子类之间的关系，是对开闭原则的补充，是对实现抽象化的具体步骤的规范。它是动作正确性的保证。即类的扩展不会给已有的系统引入新的错误，降低了代码出错的可能性。

【例1】里氏替换原则在“几维鸟不是鸟”实例中的应用：鸟一般都会飞行，如燕子的飞行速度大概是每小时 120 千米。但是新西兰的几维鸟由于翅膀退化无法飞行。假如要设计一个实例，计算这两种鸟飞行 300 千米要花费的时间。显然，拿燕子来测试这段代码，结果正确，能计算出所需要的时间；但拿几维鸟来测试，结果会发生“除零异常”或是“无穷大”，明显不符合预期，其类图如图所示。

程序运行错误的原因是：几维鸟类重写了鸟类的 setSpeed(double speed) 方法，这违背了里氏替换原则。正确的做法是：取消几维鸟原来的继承关系，定义鸟和几维鸟的更一般的父类，如动物类，它们都有奔跑的能力。几维鸟的飞行速度虽然为 0，但奔跑速度不为 0，可以计算出其奔跑 300 千米所要花费的时间。



**三、依赖倒置原则（DIP）**

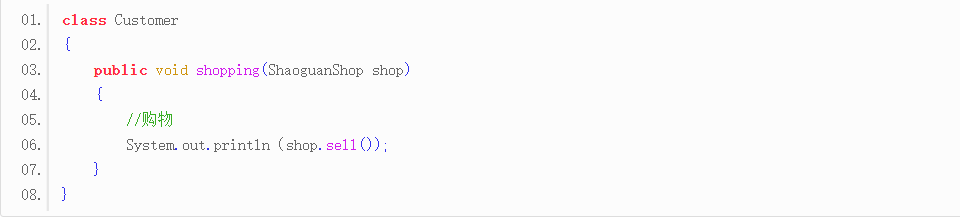
【定义】**高层模块不应该依赖低层模块，两者都应该依赖其抽象；抽象不应该依赖细节，细节应该依赖抽象。**

【核心思想】要面向接口编程，不要面向实现编程。依赖倒置原则的目的是通过要面向接口的编程来降低类间的耦合性。每个类尽量提供接口或抽象类，或者两者都具备。变量的声明类型尽量是接口或者是抽象类。**任何类都不应该从具体类派生**。

【释义】由于在软件设计中，细节具有多变性，而抽象层则相对稳定，因此以抽象为基础搭建起来的架构要比以细节为基础搭建起来的架构要稳定得多。这里的抽象指的是接口或者抽象类，而细节是指具体的实现类。

【作用】依赖倒置原则可以降低类间的耦合性，提高系统的稳定性，减少并行开发引起的风险，提高代码的可读性和可维护性。

【例1】顾客购物程序： 本程序反映了 “顾客类”与“商店类”的关系。商店类中有sell()方法，顾客类通过该方法购物以下代码定义了顾客类通过韶关网店 ShaoguanShop 购物



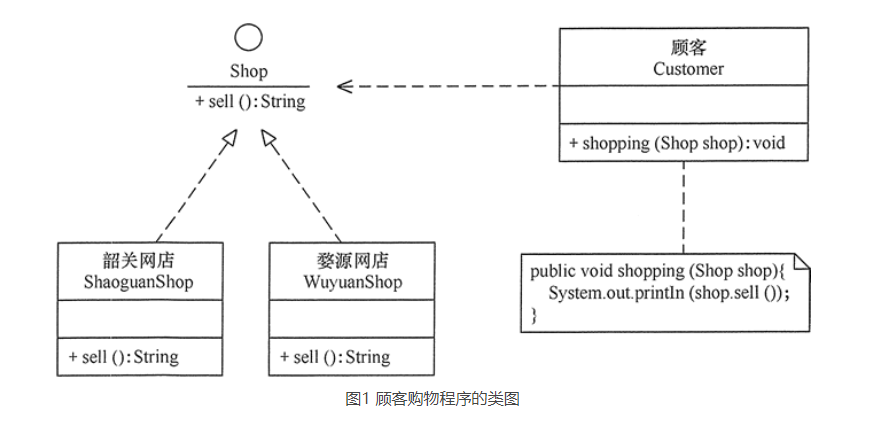
但是，这种设计存在缺点，如果该顾客想从另外一家商店（如婺源网店 WuyuanShop）购物，就要将该顾客的代码修改如下：



顾客每更换一家商店，都要修改一次代码，这明显违背了开闭原则。存在以上缺点的原因是：顾客类设计时同具体的商店类绑定了，这违背了依赖倒置原则（**模块之间耦合度太高**）。解决方法是：定义“婺源网店”和“韶关网店”的共同接口 Shop，顾客类面向该接口编程，其代码修改如下：



这样，不管顾客类 Customer 访问什么商店，或者增加新的商店，都不需要修改原有代码了，其类图如图所示：

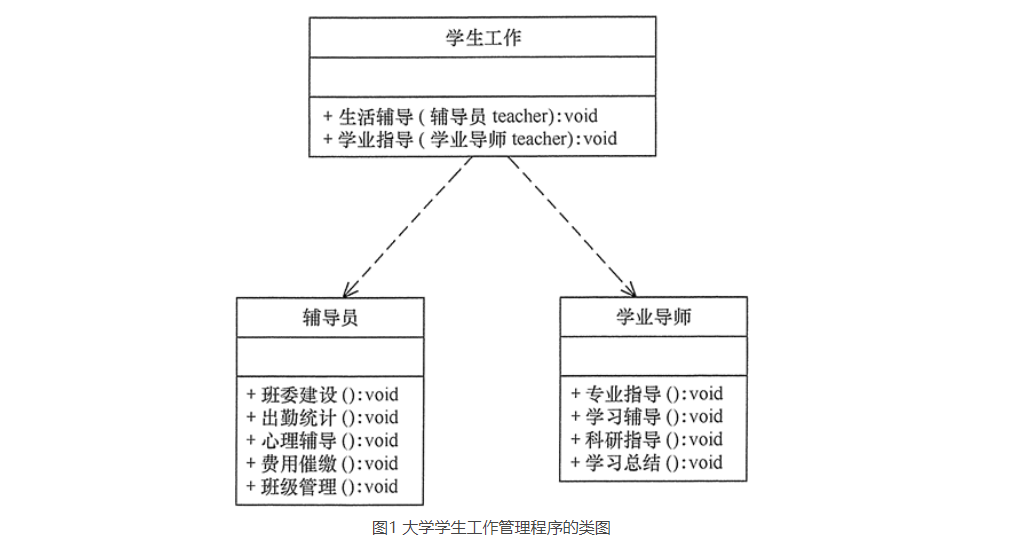


**四、单一职责原则（SRP）**

【定义】**一个类应该有且仅有一个引起它变化的原因，否则类应该被拆分。**

【释义】当客户端需要该对象的某一个职责时，不得不将其他不需要的职责全都包含进来，从而造成冗余代码或代码的浪费。同时，**当需要修改一个职责的代码时，往往容易导致另一个职责出现问题。**因此单一职责原则的核心就是控制类的粒度大小、将对象解耦、提高其内聚性，使得**一个类/接口/方法只负责一项职责或职能**。

【例1】大学学生工作管理程序：大学学生工作主要包括学生生活辅导和学生学业指导两个方面的工作，其中生活辅导主要包括班委建设、出勤统计、心理辅导、费用催缴、班级管理等工作，学业指导主要包括专业引导、学习辅导、科研指导、学习总结等工作。如果将这些工作交给一位老师负责显然不合理，正确的做 法是生活辅导由辅导员负责，学业指导由学业导师负责，其类图如图 所示。



**五、接口隔离原则（ISP）**

【定义1】**客户端不应该被迫依赖于它不使用的方法。**

【定义2】**一个类对另一个类的依赖应该建立在最小的接口上。**

【释义】接口隔离原则和单一职责都是为了提高类的内聚性、降低它们之间的耦合性，体现了封装的思想，但两者是不同的：**单一职责原则注重的是职责，而接口隔离原则注重的是对接口依赖的隔离**。单一职责原则主要是约束类，它针对的是程序中的实现和细节；接口隔离原则主要约束接口，主要针对抽象和程序整体框架的构建。

【例1】学生成绩管理程序：学生成绩管理程序一般包含插入成绩、删除成绩、修改成绩、计算总分、计算均分、打印成绩信息、査询成绩信息等功能，如果将这些功能全部放到一个接口中显然不太合理，正确的做法是将它们分别放在输入模块、统计模块和打印模块等 3 个模块中，其类图如图 1 所示。

