5、依赖倒转原则：

定义：高层模块不应该依赖于低层模块。二者都应该依赖于抽象。

抽象不应该依赖于细节。细节应该依赖于抽象。

核心思想：解耦——通过引入抽象层（接口或抽象类），打破高层模块对具体低层模块实现的直接依赖，使得系统更灵活、更易扩展、更易测试。

应用：在校园卡管理系统中实现的数据库（Database）是一个抽象类，他下面封装了两个子类——学生和管理员类，这两个均放在别的cpp文件中，他们才是具体的实现，而dadtabase只提供虚函数当接口；还有就是sqlite里的查询等操作都是面向接口（那几个含有exec()调用的操作函数）编程，而非面向直接使用sql语句本身的实现编程。

1. 合成复用原则:

定义：尽量使用对象组合（Composition）或聚合（Aggregation），而不是继承（Inheritance）来达到复用的目的。

核心思想：优先通过“拥有”关系（组合/聚合）来复用已有组件的功能，而不是通过“是”关系（继承）。继承会带来强耦合，破坏封装性。

应用：例如对于低耦合的实现，我们的管理员类也要在学生使用校园卡的同时增删查改数据，但我们在一些管理员独有的数据项就不需要改变，减少工作量以及他和学生类的绑定度。针对“高复用”的原则，消费记录变动可以被任何需要该操作的模块使用（用户界面、管理员界面、定时任务、API接口等），只需传入userId和cardId以及新余额。它不依赖于具体的调用者身份。