实验十三 设计模块（二）

实验目的：

面向对象设计原则

软件设计原则与校园卡管理系统的应用

在设计和开发校园卡管理系统时，应用软件设计原则可以提高系统的可维护性、可扩展性和稳定性。下面将结合校园卡管理系统的需求，论述几个关键的设计原则，并说明如何在项目中应用它们。

1. 里氏替换原则（Liskov Substitution Principle，LSP）

原则：子类对象应该可以替换父类对象，而不会导致程序错误。

应用：在校园卡管理系统中，不同类型的用户（如学生、教师、管理员）可以有不同的账户信息查询和管理方式。遵循里氏替换原则，我们可以设计一个通用的用户基类，然后具体的学生类、教师类和管理员类继承该基类，确保在需要使用用户对象的地方，可以用任何具体用户类型替换。

举例：在用户信息查询功能中，调用的接口应能处理所有用户类型，无论是学生、教师还是管理员。

2. 单一职责原则（Single Responsibility Principle，SRP）

原则：一个类应该只有一个引起它变化的原因。

应用：在校园卡管理系统中，将用户管理和账户管理分开，以确保每个类只负责一个特定的功能。这样，如果用户信息的管理方式改变，不会影响账户管理的逻辑，反之亦然。

举例：用户登录和信息修改功能由UserManager类负责，而账户查询和充值功能由AccountManager类负责。

3. 开闭原则（Open/Closed Principle，OCP）

原则：软件实体应当对扩展开放，对修改关闭。

应用：在校园卡管理系统中，通过接口和抽象类来实现对新功能的扩展，而不需要修改已有的代码。比如，如果需要添加新的用户类型，只需实现新的子类，而不需要修改现有的用户管理逻辑。

举例：在系统中定义一个通用的User接口，不同的用户类型（学生、教师、管理员）实现这个接口，当需要添加新的用户类型时，只需实现新的子类。

4. 迪米特法则（Law of Demeter，LoD）

原则：一个对象应当尽可能少地了解其他对象。

应用：在校园卡管理系统中，减少对象之间的直接依赖关系，通过中介者或封装方法来简化对象之间的交互。

举例：用户类只需要调用账户管理类的接口，而不直接访问账户的具体实现细节，这样可以减少耦合，提高系统的可维护性。

5. 依赖倒转原则（Dependency Inversion Principle，DIP）

原则：高层模块不应依赖低层模块，二者都应依赖其抽象；抽象不应依赖细节，细节应依赖抽象。

应用：在校园卡管理系统中，使用依赖注入和接口来实现高层模块与低层模块的解耦。这样，当低层模块发生变化时，高层模块不需要做相应的修改。

举例：账户管理模块和数据库操作模块应通过接口进行交互，而不是直接依赖具体的数据库实现。

6. 合成复用原则（Composition over Inheritance，CoI）

原则：优先使用对象组合而不是类继承来实现功能复用。

应用：在校园卡管理系统中，使用组合来实现不同功能模块的复用。例如，账户管理类可以通过组合使用支付处理类、数据库操作类等，而不是通过继承来实现这些功能。

举例：充值功能可以通过组合一个支付处理类来实现，而不是在账户管理类中直接实现所有支付逻辑。