实验十三 设计模块（二）

实验目的：

面向对象设计原则

实验内容：

论述利斯科夫替换原则（里氏代换原则）、单一职责原则、开闭原则、德（迪）米特法则、依赖倒转原则、合成复用原则，结合自己的实践项目举例说明如何应用 （保存到每个小组选定的协作开发平台上，以组为单位）。

### **利斯科夫替换原则（里氏代换原则）**

****定义****：  
利斯科夫替换原则（Liskov Substitution Principle, LSP）是面向对象设计中的一个基本原则，它指出子类对象必须能够替换其基类对象，且程序的功能不会受到影响。

****应用****：  
在实践项目中，当设计类的继承关系时，确保子类不会破坏父类所建立的契约。例如，在一个动物分类的系统中，猫（子类）应当可以替换猫科动物（基类）而不影响系统的正确性。如果猫类添加了一个与猫科动物不符的特性（如飞翔），则会违反此原则。

### **单一职责原则**

****定义****：  
单一职责原则（Single Responsibility Principle, SRP）规定一个类应该只有一个引起它变化的原因。

****应用****：  
在项目中，将功能划分到不同的类中，每个类只负责一个特定的职责。例如，在电商系统中，可以将订单处理和用户账户管理分为两个类，每个类只处理与其职责相关的功能。

### **开闭原则**

****定义****：  
开闭原则（Open-Closed Principle, OCP）规定软件实体（类、模块、函数等）应该对扩展开放，对修改封闭。

****应用****：  
在项目中，当需要添加新功能时，通过扩展现有系统来实现，而不是修改现有代码。例如，在支付系统中，如果需要添加新的支付方式，可以创建新的支付类来实现，而不是修改现有的支付类。

### **德（迪）米特法则**

****定义****：  
德米特法则（Law of Demeter, LOD）也被称为最少知识原则（Principle of Least Knowledge），它要求一个对象应该对其他对象有尽可能少的了解。

****应用****：  
在项目中，尽量降低类之间的耦合度。通过创建接口或使用数据传输对象（DTOs）来隐藏类的内部实现细节，使得一个类只需要知道与它紧密相关的其他类。

### **依赖倒转原则**

****定义****：  
依赖倒转原则（Dependency Inversion Principle, DIP）要求高层模块不应该依赖于低层模块，它们都应该依赖于抽象。抽象不应该依赖于细节，细节应该依赖于抽象。

****应用****：  
在项目中，通过定义接口或抽象类来解耦高层模块和低层模块之间的依赖关系。这样，当低层模块发生变化时，只需要修改实现该接口的类，而不需要修改高层模块的代码。

### **合成复用原则**

****定义****：  
合成复用原则（Composite/Aggregate Reuse Principle, CARP）要求尽量使用对象组合（has-a）或对象聚合（contains-a）的方式来实现代码复用，而不是使用继承（is-a）关系。

****应用****：  
在项目中，当需要实现某些功能时，优先考虑使用已有的对象进行组合或聚合，而不是通过继承来扩展类的功能。这样可以降低类之间的耦合度，提高系统的灵活性和可维护性。例如，在图形编辑软件中，可以使用不同的图形对象（如圆形、矩形等）进行组合来创建复杂的图形。

项目跟踪，建立能反映项目及小组每个人工作的进度、里程碑、工作量的跟踪图或表，将其保存到每个小组选定的协作开发平台上，每周更新。

