实验十三 设计模块（二）

郑家彤 202100032018

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）。

1.里氏替换原则（Liskov Substitution Principle, LSP）表明子类应该能够无缝地替换其父类，而不影响程序的行为。这意味着子类应当遵循父类的契约，不扩展而不是改变接口或添加新行为，以确保调用父类的客户端代码不会因替换而受影响。

例如，一个图形类如果定义了Draw方法，所有子类如圆形、正方形都应遵循并正确实现Draw`而不改变其含义，客户端代码无需关心具体图形类型。

2.单一职责原则（Single Responsibility Principle, SRP）强调每个类或模块应有且仅有一个职责。这意味着一个类应专注于一个功能，变更原因只有一个，修改不会影响其他功能，易于理解、测试和维护。例如，一个类专门负责数据库访问，不应该同时处理业务逻辑。

比如，一个员工管理类负责处理员工信息和工资计算，当工资规则变化，可能影响员工信息处理，最好拆分为员工信息管理和工资管理`两个类。

3.开闭原则（Open/Closed for Extension, OCP）和闭合修改）建议软件实体对扩展开放，对修改关闭。这意味着设计应允许新增功能而无需修改现有代码。例如，通过接口和抽象类实现扩展新功能，不改变已有类。

比如，定义服务接口，具体服务实现邮件服务，新增短信服务`时，客户端代码无需修改。

4.依赖倒转原则（Dependence Inversion, DI）鼓励依赖抽象而非具体。模块依赖抽象，不依赖具体实现，让它们可交换。例如，服务的消费者依赖接口，而不是具体服务类，通过DI容器注入具体实现。

例如，日志库使用数据库操作，不依赖MySQL操作，而是依赖数据库操作`接口，通过DI注入具体实现。

5.合成复用原则（Composition Over Inheritance）偏好组合而非继承，因为组合更灵活。组合允许动态改变行为，继承固定静态，组合通过添加或替换组件。比如，一个类聚合多个对象，而非从基类继承，通过组合不同对象来扩展功能。

例如，构建一个交通工具类，通过组合轮子类自行车、汽车、飞机等，而非直接继承交通工具，增加功能。

通过这些原则，设计更灵活、易于维护，易于测试，适应未来变化，且清晰，提高系统的整体健壮。

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。