**实验十二 设计模块（一）**

**夏思晓 202100031020**

**一、实验目的：**

1. 培养设计原则实践的能力

2. 学习依赖注入（dependency injection）

**二、实验内容：**

1. 参考教材6.2，结合项目的进程和开发历程，从设计原则的几个方面，组员对负责设计的模块进行评估，思考存在的问题和解决方案。

参考教材6.2，从设计原则的模块化、接口、信息隐藏、增量式开发、抽象和通用性6个角度，对负责设计的模块进行评估，总结存在的问题及解决方案如下：

**（1）模块化**

问题：模块之间的耦合度过高，导致修改一个模块时，需要同时修改多个相关模块；部分模块功能划分不够清晰，存在功能冗余或功能缺失。

解决方案：① 重新梳理模块间的依赖关系，降低模块间的耦合度。可以通过引入事件、消息队列等机制，实现模块间的松耦合通信。② 对模块功能进行细分和重构，确保每个模块都承担明确的职责，避免功能冗余和缺失。

**（2）接口**

问题：接口定义不够清晰，导致实现者难以理解接口的预期行为；接口变更频繁，影响实现者的开发工作。

解决方案：① 对接口进行详细的文档说明，包括接口的目的、输入参数、输出参数、异常处理等，确保实现者能够准确理解接口的预期行为。② 引入接口版本控制机制，当接口发生变更时，通过版本号进行区分，减少对实现者的影响。

**（3）信息隐藏**

问题：部分模块的内部实现细节暴露给外部调用者，导致模块难以维护和扩展；缺少对外部调用者的访问控制，存在安全隐患。

解决方案：① 对模块的内部实现细节进行封装，只暴露必要的接口给外部调用者。同时，提供完善的错误处理和异常机制，确保模块的稳定性和可维护性。② 引入访问控制机制，如权限管理、身份验证等，确保外部调用者只能访问其有权访问的模块和资源。

**（4）增量式开发**

问题：项目开发过程中缺乏明确的阶段划分和里程碑，导致项目进度难以控制；增量开发过程中，新旧代码之间的集成和测试存在困难。

解决方案：① 制定详细的项目开发计划，明确阶段划分和里程碑，确保项目进度可控。同时，定期进行项目评审和风险评估，及时调整开发计划。② 引入持续集成和持续测试机制，确保新旧代码之间的集成和测试能够自动化进行。同时，建立代码审查和质量保证流程，确保代码质量和可维护性。

**（5）抽象**

问题：部分代码重复度较高，缺乏抽象和复用；抽象层次不够清晰，导致代码难以理解和维护。

解决方案：① 对重复的代码进行抽象和封装，形成可复用的组件或库。同时，建立代码库和组件库，方便团队成员共享和使用。② 明确抽象层次和粒度，确保代码结构清晰、易于理解。同时，引入设计模式等思想，提高代码的可维护性和可扩展性。

**（6）通用性**

问题：系统功能过于定制化，难以适应不同行业或企业的需求；系统扩展性不足，难以应对未来业务的变化和发展。

解决方案：① 在设计系统时，充分考虑不同行业或企业的需求差异，采用可配置化、插件化等机制，提高系统的通用性和可定制性。② 引入微服务架构等思想，将系统拆分为多个独立的服务模块，降低模块间的耦合度。同时，建立统一的API接口和数据规范，方便系统扩展和集成。

2. 阅读下面DI资料（或查阅其它相关资料），学习依赖注入技术。

[Dependency injection - Wikipedia](https://en.wikipedia.org/wiki/Dependency_injection" \l ":~:text=In%20software%20engineering%2C%20dependency%20injection,object%20is%20called%20a%20service.)

Dependency Injection-A Practical Introduction.pdf

依赖注入（Dependency Injection，简称DI）是一种在软件设计和开发中广泛使用的技术，它允许一个对象（称为客户端）在运行时接收其依赖项（服务、资源或其他对象），而不是在编译时由客户端自己创建或查找这些依赖项。这种技术有助于实现松耦合的代码，提高代码的可测试性和可维护性。

**（1）依赖注入的基本概念**

依赖：在面向对象编程中，一个对象依赖于另一个对象来完成其职责。这种依赖关系可能会导致代码紧密耦合，使代码难以测试和重用。

依赖注入：通过将依赖项从客户端的代码中分离出来，并在运行时将它们注入到客户端中，可以实现松耦合的代码。这样，客户端就不再需要知道如何创建或查找其依赖项，只需专注于完成其自己的职责。

**（2）依赖注入的类型**

① 构造器注入：在创建客户端对象时，通过构造函数传递依赖项。这种方式可以确保在客户端对象被使用之前，其依赖项已经被正确初始化。

② 属性注入：通过客户端对象的公共属性或字段来设置依赖项。这种方式比较灵活，但可能会暴露客户端的内部状态，降低封装性。

③ 方法注入：通过客户端对象的方法传递依赖项。这种方式通常用于在客户端对象的生命周期中的特定时间点注入依赖项。

**（3）依赖注入容器**

依赖注入容器（也称为IoC容器）是一个负责创建、配置和管理对象及其依赖项的工具。它可以根据配置或约定自动创建对象，并将依赖项注入到这些对象中。使用依赖注入容器可以大大简化依赖注入的过程，并减少代码中的样板代码。

**（4）依赖注入的好处**

① 松耦合：通过将依赖项从客户端代码中分离出来，可以实现松耦合的代码，提高代码的可重用性和可维护性。

② 可测试性：通过注入模拟或存根（stub）对象来替换实际依赖项，可以更容易地编写单元测试，提高代码的可测试性。

③ 灵活性：由于依赖项是在运行时注入的，因此可以轻松更改客户端对象的行为，而无需修改其代码。

④代码重用：通过减少代码中的样板代码和重复代码，可以提高代码的重用性。

**（5）依赖注入的实践**

在实际项目中，可以使用各种依赖注入框架（如Spring、Guice、Unity等）来实现依赖注入。这些框架通常提供了丰富的功能和灵活的配置选项，可以根据项目的需求进行定制。此外，也可以手动实现依赖注入，但需要注意保持代码的清晰和可维护性。

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。