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。

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。

依赖注入（Dependency Injection，DI）是一种软件设计模式，它可以帮助我们实现依赖倒置原则和开闭原则，从而实现更好的解耦和代码重用。以下是一些关于依赖注入的关键概念：

依赖： 在面向对象编程中，一个类的实例（对象）依赖于另一个对象，如果它使用了另一个对象的方法或属性。例如，如果一个类A的方法需要一个类B的对象才能工作，我们就说类A依赖于类B。

注入： 注入是将依赖对象传递（注入）给依赖它的对象的过程。这通常通过构造函数、属性或者方法来实现。最常见的注入方式是构造器注入和setter注入。

依赖注入容器： 容器是用来创建、配置和管理类及其依赖关系的工具。它负责创建对象，并自动处理对象之间的依赖关系。例如，Spring框架的IoC（Inversion of Control）容器就是一个依赖注入容器。

服务： 在依赖注入中，服务通常指的是可以被其他对象使用的对象。一个服务可能会依赖其他的服务。

客户： 在依赖注入中，客户是依赖一个或多个服务的对象。客户不应直接负责查找或创建它所依赖的服务，而是应该有容器或其他机制将这些服务注入到客户中。

使用依赖注入的好处包括：代码更加解耦，更容易测试，更容易重用和重构，更符合单一职责原则和开闭原则等。

3.

里氏代换原则（Liskov Substitution Principle）：程序应该能够在不改变程序正确性的前提下，将基类对象替换为其子类对象。例如，在网络博客中，可能有一个基类叫Post，然后有多个子类如TextPost，ImagePost，VideoPost等，这些子类都应该能够在需要Post的地方被替换使用。

单一职责原则（Single Responsibility Principle）：一个类应该只有一个引起变化的原因。例如，在博客系统中，你可能会有一个PostService类来处理与文章相关的逻辑，如创建、删除、更新文章等，但不应该让这个类处理用户认证或评论管理等不相关的功能。

开闭原则（Open-Closed Principle）：软件实体（类、模块、函数等）应该对扩展开放，对修改关闭。例如，如果你想在博客系统中添加一个新的特性（例如，支持 Markdown 格式），你应该能够添加这个特性，而不需要修改已经存在的代码。

德米特法则（Law of Demeter）：最少知识原则，一个对象应该对其他对象有最少的了解。例如，CommentService类应该只能通过Post类的接口来获取所需信息，而不应该直接访问到Post的内部或者Post的其他依赖对象。

依赖倒转原则（Dependency Inversion Principle）：高层模块不应该依赖低层模块，两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。例如，在博客系统中，PostService类（高层模块）不应该直接依赖数据库操作类（低层模块），而应该依赖一个抽象的接口，如IPostRepository。

合成复用原则（Composition Over Inheritance）：应该尽量使用组合/聚合的方式，而不是使用继承。例如，在博客系统中，你可能有一个Post类和一个Comment类，而每个Post都有一组Comment。在这种情况下，我们可以说Post聚合了Comment，而不是Post继承了Comment。