**会议记录表**

|  |  |  |  |
| --- | --- | --- | --- |
| 时间 | 2023年5月12日 | 星期 | 第十三周 |
| 队伍 | 啊对对队 | 会议地点 | 线上会议 |
| 进行时长 | 1h 56min |
| 参会人员 | 孟小凡、尹国泰、黄泰达、吴百川 | | |
| 会议主题 | 设计原则的模块评估、依赖注入技术的学习、原则实践应用 | | |
| 会议内容 | 议题1：从设计原则的几个方面对设计模块进行评估  通过对教材6.2中相关内容的学习，分别从模块化、接口、信息隐蔽、增量式开发、抽象、通用性几个方面对于本组设计的项目系统进行模块评估，组内成员编写评估报告，完成整体评价并找到系统中出现的问题并提出对应的解决方案。  议题2：依赖注入技术的学习  依赖注入（Dependency Injection，简称DI）是一种软件设计模式和技术，用于实现模块之间的解耦和管理对象之间的依赖关系。在依赖注入中，对象的依赖关系由外部实体（通常是容器或框架）在创建对象时注入，而不是对象自己创建或查找依赖。  特点：  解耦性：依赖注入可以将对象之间的依赖关系解耦，使得每个对象只关注自身的功能，而不需要关心依赖对象的创建和管理。  可测试性：通过依赖注入，可以将依赖的对象替换为模拟或测试对象，从而方便进行单元测试和模块测试，提高代码的可测试性。  可扩展性：依赖注入使得系统更易于扩展和修改。通过注入不同的依赖实现，可以动态改变系统的行为，而无需修改依赖对象本身。  可维护性：依赖注入使代码结构更清晰，依赖关系更明确，易于理解和维护。同时，由于依赖关系集中在一个地方进行管理，也方便进行修改和调整。  可重用性：通过依赖注入，可以将可复用的依赖对象独立管理，并在不同的模块或系统中共享使用，提高代码的重用性和模块化程度。  松耦合：依赖注入降低了对象之间的耦合度，使得对象更加独立和可替换。这有助于提高系统的灵活性和可维护性，同时也支持更好的代码重构和优化。  配置灵活性：依赖注入可以通过配置来管理依赖关系，使得系统的行为和组成可以在不修改代码的情况下进行调整和配置，提高系统的灵活性和可配置性。  议题3：原则理论的实践应用  组内学习讨论利斯科夫替换原则（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、合成复用原则的相关内容，每人举例一个原则与本组项目中对应模块的结合，综合评判原则的实践应用。 | | |
| 备注 | 无 | | |