**实验十三设计模块（二）**

**一、实验目的**

**面向对象设计原则**

**二、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. **利斯科夫替换原则（里氏代换原则）**

里氏替换原则（LSP）由麻省理工学院计算机科学实验室的里斯科夫女士在1987年的“面向对象技术的高峰会议”（OOPSLA）上发表的一篇文章《数据抽象和层次》里提出来的，她提出：继承必须确保超类所拥有的性质在子类中仍然成立。

里氏替换原则主要阐述了有关继承的一些原则，也就是什么时候应该使用继承，什么时候不应该使用继承，以及其中蕴含的原理。里氏替换原是继承复用的基础，它反映了基类与子类之间的关系，是对开闭原则的补充，是对实现抽象化的具体步骤的规范。里氏替换原则通俗来讲就是：子类可以扩展父类的功能，但不能改变父类原有的功能。也就是说，子类继承父类时，除添加新的方法完成新增功能外，尽量不要重写父类的方法。如果通过重写父类的方法来完成新的功能，这样写起来虽然简单，但是整个继承体系的可复用性会比较差，特别是运用多态比较频繁时，程序运行出错的概率会非常大。如果程序违背了里氏替换原则，则继承类的对象在基类出现的地方会出现运行错误。这时其修正方法是：取消原来的继承关系，重新设计它们之间的关系。

**应用**：在人才招聘系统中，假设有一个基类Job（职位），它有一个方法apply()（申请）。如果有一个子类ITJob（IT职位），它继承了Job类，那么ITJob类应该能够无缝替换Job类，而不需要改变apply()方法的预期行为。如果ITJob需要特殊处理，比如技术面试，它应该添加新的方法而不是重写apply()。

1. **单一职责原则**

单一职责原则（Single Responsibility Principle，SRP）又称单一功能原则，由罗伯特·C.马丁（Robert C.Martin）于《敏捷软件开发：原则、模式和实践》一书中提出的。这里的职责是指类变化的原因，单一职责原则规定一个类应该有且仅有一个引起它变化的原因，否则类应该被拆分。

该原则提出对象不应该承担太多职责，如果一个对象承担了太多的职责，至少存在以下两个缺点：

* 一个职责的变化可能会削弱或者抑制这个类实现其他职责的能力；
* 当客户端需要该对象的某一个职责时，不得不将其他不需要的职责全都包含进来，从而造成冗余代码或代码的浪费。

**应用：**在人才招聘系统中，如果有一个类ResumeParser（简历解析器），它负责解析简历和验证简历格式。如果解析和验证逻辑都放在一个类中，这个类就承担了两个职责。根据SRP，应该将这两个功能分离到两个不同的类中，比如ResumeParser和ResumeValidator。

1. **开闭原则**

开闭原则（Open Closed Principle，OCP）由勃兰特·梅耶（Bertrand Meyer）提出，他在1988年的著作《面向对象软件构造》（Object Oriented Software Construction）中提出：软件实体应当对扩展开放，对修改关闭，这里的软件实体包括项目中划分出的模块、类与接口、方法，也就是说在设计一个模块的时候，应当使这个模块可以在不被修改的前提下被扩展，即实现在不修改源代码的情况下改变这个模块的行为。

开闭原则是面向对象程序设计的终极目标，它使软件实体拥有一定的适应性和灵活性的同时具备稳定性和延续性。如果软件遵守开闭原则，软件测试时只需要对扩展的代码进行测试，因为原有的测试代码仍然能够正常运行。开闭原则可以提高代码的可复用性，提高软件的可维护性。

抽象化是开闭原则的关键。绝大部分的设计模式都符合开闭原则，在对每一个模式进行优缺点评价时都会以开闭原则作为一个重要的评价依据，以判断基于该模式设计的系统是否具备良好的灵活性和可扩展性。开闭原则可以通过“抽象约束、封装变化”来实现，即通过接口或者抽象类为软件实体定义一个相对稳定的抽象层，而将相同的可变因素封装在相同的具体实现类中。

**应用：**人才招聘系统可能需要在未来支持更多的职位类型或招聘流程。设计时，可以通过定义抽象类或接口来允许新的职位类型或流程轻松添加，而无需修改现有代码。例如，可以定义一个JobPosition接口，不同的职位类型实现这个接口。

1. **德（迪）米特法则**

迪米特法则（Law of Demeter，LoD）又叫作最少知识原则（Least Knowledge Principle，LKP)，产生于1987年美国东北大学的一个名为迪米特（Demeter）的研究项目，由伊恩·荷兰（Ian Holland）提出，被UML创始者之一的布奇（Booch）普及，后来又因为在经典著作《程序员修炼之道》提及而广为人知。

迪米特法则的定义是：只与你的直接朋友交谈，不跟“陌生人”说话。其含义是：如果两个软件实体无须直接通信，那么就不应当发生直接的相互调用，可以通过第三方转发该调用。其目的是降低类之间的耦合度，提高模块的相对独立性。迪米特法则中的“朋友”是指：当前对象本身、当前对象的成员对象、当前对象所创建的对象、当前对象的方法参数等，这些对象同当前对象存在关联、聚合或组合关系，可以直接访问这些对象的方法。

迪米特法则的主要用途在于控制信息的过载：在类的划分上，应当尽量创建松耦合的类，类之间的耦合度越低，就越有利于复用，一个处在松耦合中的类一旦被修改，不会对关联的类造成太大波及；在类的结构设计上，每一个类都应当尽量降低其成员变量和成员函数的访问权限；在对其他类的引用上，一个对象对其他对象的引用应当降到最低。

**应用：**在人才招聘系统中，一个InterviewScheduler（面试调度器）类不应该直接与Database（数据库）类交互来获取面试时间，因为这会使得InterviewScheduler与数据库实现紧密耦合，它应该通过InterviewRepository（面试仓库）这样的中介与数据库通信，降低耦合度。

1. **依赖倒转原则**

依赖倒转原则是Robert C.Martin在1996年为“C++Reporter”所写的专栏EngineeringNotebook的第三篇，后来加入到他在2002年出版的经典著作“Agile Software Development, Principles, Patterns, and Practices”一书中。依赖倒转原则定义如下：抽象不应该依赖于细节，细节应当依赖于抽象。换言之，要针对接口编程，而不是针对实现编程。简单来说，依赖倒转原则就是指：代码要依赖于抽象的类，而不要依赖于具体的类；要针对接口或抽象类编程，而不是针对具体类编程。实现开闭原则的关键是抽象化，并且从抽象化导出具体化实现，如果说开闭原则是面向对象设计的目标的话，那么依赖倒转原则就是面向对象设计的主要手段。

依赖倒转原则要求我们在程序代码中传递参数时或在关联关系中，尽量引用层次高的抽象层类，即使用接口和抽象类进行变量类型声明、参数类型声明、方法返回类型声明，以及数据类型的转换等，而不要用具体类来做这些事情。为了确保该原则的应用，一个具体类应当只实现接口或抽象类中声明过的方法，而不要给出多余的方法，否则将无法调用到在子类中增加的新方法。

在引入抽象层后，系统将具有很好的灵活性，在程序中尽量使用抽象层进行编程，而将具体类写在配置文件中，这样一来，如果系统行为发生变化，只需要对抽象层进行扩展，并修改配置文件，而无须修改原有系统的源代码，在不修改的情况下来扩展系统的功能，满足开闭原则的要求。

在实现依赖倒转原则时，我们需要针对抽象层编程，而将具体类的对象通过依赖注入（Dependency Injection, DI）的方式注入到其他对象中，依赖注入是指当一个对象要与其他对象发生依赖关系时，通过抽象来注入所依赖的对象。常用的注入方式有三种，分别是：构造注入，设值注入（Setter注入）和接口注入。构造注入是指通过构造函数来传入具体类的对象，设值注入是指通过Setter方法来传入具体类的对象，而接口注入是指通过在接口中声明的业务方法来传入具体类的对象。这些方法在定义时使用的是抽象类型，在运行时再传入具体类型的对象，由子类对象来覆盖父类对象。

**应用：**在人才招聘系统中，业务逻辑层不应该直接依赖于数据访问层的具体实现，它应该依赖于定义良好的接口或抽象类。例如，CandidateService（候选人服务）应该依赖于ICandidateRepository接口，而不是具体的数据库访问类。

1. **合成复用原则**

合成复用原则又称为组合/聚合复用原则，其定义如下：合成复用原则（CRP）：尽量使用对象组合，而不是继承来达到复用的目的。合成复用原则就是在一个新的对象里通过关联关系（包括组合关系和聚合关系）来使用一些已有的对象，使之成为新对象的一部分；新对象通过委派调用已有对象的方法达到复用功能的目的。简言之：复用时要尽量使用组合/聚合关系（关联关系），少用继承。

在面向对象设计中，可以通过两种方法在不同的环境中复用已有的设计和实现，即通过组合/聚合关系或通过继承，但首先应该考虑使用组合/聚合，组合/聚合可以使系统更加灵活，降低类与类之间的耦合度，一个类的变化对其他类造成的影响相对较少；其次才考虑继承，在使用继承时，需要严格遵循里氏代换原则，有效使用继承会有助于对问题的理解，降低复杂度，而滥用继承反而会增加系统构建和维护的难度以及系统的复杂度，因此需要慎重使用继承复用。

**应用：**在人才招聘系统中，如果需要实现一个具有特定行为的SpecializedJob（特定职位），而不是通过继承Job类，应该考虑使用组合。SpecializedJob可以包含一个Job对象，并添加特定的行为或属性，而不是修改Job类。

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。