**实验十三 设计模块（二）**

**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. 利斯科夫替换原则（Liskov Substitution Principle, LSP）

定义： 子类应当能够替换其父类并保持程序的正确性。换句话说，使用基类的地方应能无差别地使用子类对象。

在系统中的应用：

例子： 设计一个基类Candidate表示应聘者，有子类ExperiencedCandidate（有工作经验的应聘者）和FreshGraduate（应届毕业生）。在处理应聘者信息时，无论是统计应聘人数、安排面试还是评估技能，系统都应能透明地处理这些子类对象，而无需关心它们的具体类型。

应用： 确保所有应聘者的公共操作（如基本信息录入、面试安排）在基类中定义，子类仅添加特定行为（如经验候选人的工作历史审查，应届毕业生的毕业院校验证），保证了多态性的正确使用。

2. 单一职责原则（Single Responsibility Principle, SRP）

定义： 一个类应该只有一个引起它变化的原因，即一个类只负责一项职责。

在系统中的应用：

例子： 设计JobPosting类来管理职位发布信息，包括职位描述、要求等，而不应包含筛选简历的逻辑。筛选简历的逻辑应放在单独的ResumeFilter类中。

应用： 这样做可以确保当职位发布的需求改变时（比如增加职位类别），不会影响到简历筛选的逻辑，反之亦然，降低了模块间的耦合度。

3. 开闭原则（Open-Closed Principle, OCP）

定义： 对扩展开放，对修改关闭。软件实体（类、模块、函数等）应该可以扩展功能，但不应该通过修改源代码的方式进行。

在系统中的应用：

例子： 系统需要支持多种面试方式（线上、线下、电话面试等）。设计一个InterviewStrategy接口，定义一个执行面试的方法，然后为每种面试方式创建实现类（如OnlineInterview, OfflineInterview）。

应用： 当需要添加新的面试方式时，只需实现InterviewStrategy接口并注册到系统中，无需修改现有面试策略的代码。

4. 迪米特法则（Law of Demeter, LoD）

定义： 一个对象应当对其他对象有最少的了解，即尽量减少对象之间的交互，提高模块的独立性。

在系统中的应用：

例子： 在处理应聘者面试流程时，InterviewScheduler类仅需知道Candidate的预约面试方法，而不需要知道Candidate内部如何管理其联系方式或简历详情。

应用： 通过引入中介者或服务类来协调不同对象之间的交互，减少直接依赖，例如使用CandidateService来封装与候选人相关的数据访问和业务逻辑。

5. 依赖倒转原则（Dependency Inversion Principle, DIP）

定义： 高层模块不应依赖于低层模块，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。

在系统中的应用：

例子： 系统需要持久化职位信息到数据库中。不直接让业务逻辑依赖具体的数据库操作类，而是定义一个IJobPostingRepository接口，业务逻辑依赖此接口。

应用： 实现时，可以有SQLJobPostingRepository或NoSQLJobPostingRepository等具体存储策略，根据需要动态切换，而不影响上层业务逻辑。

6. 合成复用原则（Composition over Inheritance）

定义： 尽量使用对象组合而非继承来达到复用的目的。

在系统中的应用：

例子： 考虑到应聘者的技能列表，如果使用继承来为每一种技能创建一个类（如JavaSkill, PythonSkill），将很快变得难以管理。更好的方式是，让Candidate类包含一个List<Skill>，其中Skill是一个抽象基类，不同技能作为其子类。

应用： 这样做不仅简化了类结构，还允许灵活地添加、删除或修改应聘者的技能集，而无需创建大量继承类。