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）。

面向对象设计原则（SOLID）

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

定义：一个类应该只有一个职责，并且该职责应该被封装在一个类中。当一个类负责多项任务时，它可能需要因不同原因进行更改，这违反了单一职责原则。

在人才招聘系统中，简历解析器类只负责解析简历文件，用户资料管理类只负责管理用户资料。这样，每个类都只有一个职责，当需求变化时，只需修改一个类。

1. 开闭原则（Open-Closed Principle, OCP）

定义：软件实体应该对扩展开放，对修改关闭。这意味着设计时应该允许在不修改现有代码的基础上进行扩展。

系统设计时使用抽象层，比如职位发布接口，可以有多个实现类，如全职职位、兼职职位、实习职位等。当需要添加新的职位类型时，只需添加新的实现类，而不需修改现有代码。

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

定义：子类型必须能够替换掉它们的父类型，而不影响程序的行为。这要求子类能够完全继承父类的行为。

如果系统中有一个基础职位类和多个特定类型的职位类，这些特定类型的职位类可以替换基础职位类出现在系统中的任何位置，而不破坏系统功能。

1. 接口隔离原则（Interface Segregation Principle, ISP）

定义：客户端不应该依赖于它不使用的方法。一个类不应该被迫实现它不需要的接口。

在系统中，不同的服务如简历上传、职位搜索、候选人筛选等，各自实现它们自己的接口，而不是依赖于一个大而全的接口。

1. 依赖倒置原则（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。抽象不应依赖于细节，细节应依赖于抽象。

系统的业务逻辑不直接依赖于具体的数据访问实现，而是依赖于数据访问接口。这样，更换数据库或数据访问技术时，只需更改实现这些接口的类。

1. 迪米特法则（Law of Demeter, LoD）

定义：一个对象应该对它的友元对象保持最少的了解，只与直接的朋友通信，不与“朋友的朋友”通信。

系统中的职位发布模块不需要直接访问候选人的私人信息，而是通过候选人模块提供的接口来请求这些信息，减少模块间的直接依赖。

1. 合成复用原则（Composite Reuse Principle, CRP）

定义：尽量通过对象的组合/聚合来实现代码复用，而不是通过继承。

系统可以设计为由多个组件组成，如简历分析组件、面试安排组件等，这些组件可以被不同的招聘流程复用和组合，而不是通过继承来扩展功能。