**实验十三 设计模块（二）**

**夏思晓 202100031020**

**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，Liskov Substitution Principle, LSP）**

定义：所有引用基类（父类）的地方必须能透明地使用其子类的对象。即子类可以扩展父类的功能，但不能改变父类原有的功能。

重要性：确保软件系统的稳定性，在增加新功能时，能够尽量不影响原有的功能。

项目应用：在“职等你来”人才招聘系统中，假设有一个User类作为基类，表示所有用户。我们可以创建Employer（雇主）和JobSeeker（求职者）作为User的子类。里氏代换原则要求，在任何使用User对象的地方，我们都可以用Employer或JobSeeker的对象替换，且系统的行为保持一致。例如，在发送通知的功能中，无论是给雇主还是求职者发送通知，我们都可以通过User接口进行，无需关心具体是哪种类型的用户。

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

定义：一个类应该只有一个引起变化的原因。即一个类只负责一项职责。

重要性：降低类的复杂性，提高系统的可维护性。

项目应用：在“职等你来”人才招聘系统中，我们有一个处理用户登录的类。按照单一职责原则，这个类应该只负责处理登录相关的逻辑，而不应该包含其他如用户注册、密码找回等功能。这样，当登录逻辑发生变化时，我们只需要修改这一个类，而不是其他相关的类。

**（3）开闭原则（Open-Closed Principle, OCP）**

定义：软件实体（类、模块、函数等）应该对扩展开放，对修改关闭。即在不修改现有代码的情况下，通过扩展实现新的功能。

重要性：提高软件的可维护性和可扩展性。

项目应用：假设在招聘系统中，我们想要增加一种新的招聘方式，如视频面试。开闭原则要求我们在不修改现有代码的基础上，通过扩展系统来实现这一功能。我们可以创建一个新的VideoInterview类，并实现与招聘相关的接口或继承自相应的基类，而不是修改现有的招聘或面试类。

**（4）德（迪）米特法则（Law of Demeter, LoD）**

定义：一个对象应该对其他对象保持最少的了解。即一个类应该尽可能少地与其他类发生相互作用。

重要性：降低类之间的耦合度，提高系统的可维护性和可重用性。

项目应用：在“职等你来”人才招聘系统中，当用户提交简历时，我们可能需要与多个系统或服务进行交互，如简历存储系统、邮件通知系统等。迪米特法则要求我们在处理用户简历提交时，只与必要的系统或服务进行交互，并尽量减少对其他系统或服务的依赖和了解。这样可以提高系统的松耦合性，降低模块间的耦合度。

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

定义：高层模块不应该依赖于低层模块，二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。即要面向接口编程，不要面向实现编程。

重要性：降低模块之间的耦合度，提高系统的可维护性和可扩展性。

项目应用：在“职等你来”人才招聘系统中，我们有一个处理简历的模块，它依赖于具体的简历解析库或数据库。依赖倒转原则要求我们不直接依赖这些具体的实现，而是依赖于它们的抽象接口或基类。这样，当需要更换简历解析库或数据库时，我们只需要修改相应的抽象接口或基类的实现，而不需要修改依赖这些接口或基类的模块。

**（6）合成复用原则（Composite/Aggregate Reuse Principle, CARP）**

定义：尽量使用对象组合（has-a）/聚合（contains-a），而不是继承（is-a）来达到复用的目的。

重要性：降低系统的复杂性，提高系统的可维护性和可扩展性。通过对象组合和聚合，可以更加灵活地构建软件系统。

项目应用：在招聘系统中，我们有多个模块需要访问用户信息。按照合成复用原则，我们不应该通过继承User类来获取用户信息，而是应该将User对象作为参数传递给这些模块，或者在这些模块中创建一个User对象的引用。这样，我们可以更灵活地控制用户信息的访问和使用，降低类与类之间的耦合度。

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。