**实验十四 设计模块（三）**

**夏思晓 202100031020**

**一、实验目的：**

学习设计模式，能在项目设计中运用设计模式进行面向对象设计

**二、实验内容：**

1. 阅读下面设计模式资料（或查阅其它相关资料），结合项目的进程和开发历程，分析项目采用了哪些设计模式。

Design Patterns-Elements of Reusable Object-Oriented Software.pdf

The GoF Design Patterns Reference.pdf

[Design Patterns - Wikipedia](https://en.wikipedia.org/wiki/Design_Patterns)

**（1）MVC（Model-View-Controller）设计模式：**

MVC是一种用于构建用户界面的设计模式，它将应用程序的数据（Model）、用户界面（View）以及处理用户输入和数据更新的逻辑（Controller）分离开来。

在人才招聘系统中，MVC模式可以确保用户界面（如求职者注册、职位搜索等）与后台数据（如求职者信息、职位信息）和业务逻辑（如验证、搜索算法）的清晰分离。具体来讲，求职者信息、职位信息等可以作为数据模型（Model），求职者界面、企业界面等可以作为视图（View），而搜索、注册、登录等功能则可以通过控制器（Controller）来实现。

**（2）单例模式（Singleton Pattern）：**

单例模式确保一个类只有一个实例，并提供一个全局访问点来访问该实例。

在“职等你来”人才招聘系统项目中，存在一些需要全局访问的资源或服务，如数据库连接、配置管理、日志记录等。这些资源或服务可以通过单例模式来管理，确保在整个应用程序中只有一个实例，并方便全局访问。

**（3）工厂模式（Factory Pattern）：**

工厂模式用于封装对象的创建过程，使得在创建对象时无需直接指定其具体的类。

在“职等你来”人才招聘系统项目中，工厂模式被用于创建不同类型的用户对象（如求职者、招聘者、管理员等）、职位对象或简历对象等。通过工厂模式，可以方便地管理对象的创建过程，降低代码的复杂性和耦合度。

**（4）观察者模式（Observer Pattern）：**

观察者模式定义了一种一对多的依赖关系，让多个观察者对象同时监听某一个主题对象。当主题对象状态发生变化时，它的所有依赖者（观察者）都会自动收到通知并更新。

在“职等你来”人才招聘系统项目中，观察者模式被用于实现实时消息通知功能，如当有新职位发布时，通知所有关注的求职者；或者当简历被查看时，通知求职者简历已被查看等。

**（5）策略模式（Strategy Pattern）：**

策略模式定义了一系列的算法，并将每一个算法封装起来，使它们可以互相替换。策略模式使得算法可以独立于使用它的客户端变化。

在“职等你来”人才招聘系统项目中，策略模式被用于实现不同的简历筛选策略、职位匹配算法等。通过定义不同的策略类，可以方便地切换不同的算法，以满足不同的业务需求。

**（6）模板方法模式（Template Method Pattern）：**

模板方法模式定义了一个操作中的算法的框架，而将一些步骤延迟到子类中。模板方法使得子类可以不改变一个算法的结构即可重新定义该算法的某些特定步骤。

在“职等你来”人才招聘系统项目中，模板方法模式被用于处理一些具有固定流程但部分步骤可以定制的业务场景，如简历审核流程、职位发布流程等。通过定义模板方法，可以确保流程的一致性和可控性，同时允许在子类中自定义特定步骤的实现。

2. 给出4种设计模式的例子（语言不限，以组为单位），并总结其特点 （保存到每个小组选定的协作开发平台上）

**（1）单例模式（Singleton Pattern）**

在数据库连接管理中，通常希望只有一个数据库连接实例，以避免过多的连接和资源浪费。此时，可以使用单例模式来确保在整个应用程序中只有一个数据库连接实例。

特点：

① 全局唯一：确保一个类只有一个实例，并提供一个全局访问点。它适用于那些只需要一个实例的场景，如配置管理、线程池、缓存等。

② 延迟加载：在需要时才创建实例，避免不必要的资源浪费。

③ 线程安全：在多线程环境下，需要确保单例模式的线程安全性。

**（2）工厂模式（Factory Pattern）**

假设我们有一个用于创建不同形状（如圆形、矩形、三角形）的绘图应用程序。我们可以使用工厂模式来根据用户请求创建相应的形状对象。

特点：定义一个用于创建对象的接口，让子类决定实例化哪一个类。工厂方法使一个类的实例化延迟到其子类。其特点可概括为：

① 解耦：将对象的创建与使用分离，使得代码更加灵活和可扩展，降低了类之间的耦合度。

② 灵活性：通过子类扩展增加对象种类，无需修改已有代码。

③ 封装性：隐藏了对象的创建细节，使得代码更加易于维护。

**（3）观察者模式（Observer Pattern）**

在新闻订阅系统中，当用户订阅了某个新闻类别后，他们希望在新闻发布时立即收到通知。此时，可以使用观察者模式，让新闻发布系统（被观察者）在发布新闻时通知所有订阅了该类别新闻的用户（观察者）。

特点：定义了一种一对多的依赖关系，让多个观察者对象同时监听某一个主题对象。当主题对象在状态上发生变化时，它的所有依赖者（观察者）都会自动收到通知并更新。这使得对象之间的耦合度降低，提高了系统的可维护性和可扩展性。总结其特点为：

① 松耦合：对象之间通过事件进行通信，降低了耦合度。

② 灵活性：可以动态地添加或删除观察者，无需修改已有代码。

③ 广播机制：当被观察对象状态发生变化时，可以通知所有观察者。

**（4）策略模式（Strategy Pattern）**

在电商系统中，需要为不同的用户群体提供不同的支付策略（如信用卡支付、货到付款、第三方支付等）。我们可以使用策略模式来定义这些支付策略，并在运行时根据用户选择或系统条件来切换策略。

特点：定义了一系列的算法，并将每一个算法封装起来，使它们可以互相替换。策略模式使得算法可以独立于使用它的客户端变化。这提高了代码的复用性和可维护性，使得系统更加灵活和可扩展。同时，它也降低了客户端与算法之间的耦合度。总结其特点为：

① 封装性：策略模式将每个具体的算法都封装起来，并且使它们之间可以互相替换。这使得算法可以独立于使用它的客户端变化。

② 可维护性：由于策略模式把算法的使用者和算法的实现分离，所以算法可以独立于使用它的客户端而演变。这大大简化了算法的维护过程。如果需要修改或增加算法，只需要修改或增加相应的策略类，而不需要修改客户端代码。

③ 可扩展性：策略模式使得系统可以方便地扩展新的算法。只需要定义一个新的策略类，并在上下文中进行配置即可。这使得系统具有很高的可扩展性。

④ 灵活性：客户端可以根据运行时的情况动态地选择算法。这种灵活性使得系统能够根据不同的需求或环境选择最合适的算法。

⑤ 简化单元测试：由于策略模式将算法封装在独立的类中，这使得对算法的测试变得简单。可以对每个策略类进行独立的单元测试，而不需要在客户端代码中测试所有的算法。

⑥ 符合开闭原则：开闭原则要求软件实体（类、模块、函数等）应该对扩展开放，对修改封闭。策略模式完全符合这一原则，因为可以在不修改客户端代码的情况下添加新的策略。

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。