实验十四 设计模块（三 ）

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

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，分析项目采用了那些设计模式

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)

学习设计模式可以帮助你在项目设计和开发过程中更好地应对各种需求和问题，提高代码的质量和可维护性。设计模式是针对特定问题和场景的解决方案，通过将问题和解决方案进行抽象和总结，形成了一系列被广泛接受的设计模式。这些设计模式通常可以分为以下几类：

1）创建型模式 (Creational Patterns):

* 用于处理对象的创建机制，包括对象实例化的方式、对象的创建和销毁等。常见的创建型模式包括：
* 工厂模式 (Factory Pattern)
* 抽象工厂模式 (Abstract Factory Pattern)
* 单例模式 (Singleton Pattern)
* 原型模式 (Prototype Pattern)
* 建造者模式 (Builder Pattern)

2）结构型模式 (Structural Patterns):

* 用于处理类或对象之间的组合，以形成更大的结构。常见的结构型模式包括：
* 适配器模式 (Adapter Pattern)
* 桥接模式 (Bridge Pattern)
* 装饰器模式 (Decorator Pattern)
* 组合模式 (Composite Pattern)
* 享元模式 (Flyweight Pattern)
* 代理模式 (Proxy Pattern)

3）行为型模式 (Behavioral Patterns):

* 用于处理对象之间的通信、职责分配等行为。常见的行为型模式包括：
* 观察者模式 (Observer Pattern)
* 状态模式 (State Pattern)
* 策略模式 (Strategy Pattern)
* 模板方法模式 (Template Method Pattern)
* 命令模式 (Command Pattern)
* 访问者模式 (Visitor Pattern)
* 责任链模式 (Chain of Responsibility Pattern)
* 中介者模式 (Mediator Pattern)
* 备忘录模式 (Memento Pattern)
* 解释器模式 (Interpreter Pattern)

4）并发型模式 (Concurrency Patterns):

* 用于处理多线程、并发编程中的问题。常见的并发型模式包括：
* 保护性暂挂模式 (Guarded Suspension Pattern)
* 生产者-消费者模式 (Producer-Consumer Pattern)
* 读写锁模式 (Read-Write Lock Pattern)
* 信号量模式 (Semaphore Pattern)
* 资源池模式 (Resource Pool Pattern)
* 并行计算模式 (Parallel Computing Pattern)

我们设计的人才招聘系统结合多种设计模式来实现不同的功能和解决不同的问题。

**1.工厂模式 (Factory Pattern):**

人才招聘系统使用工厂模式来创建不同类型的对象，如用户、职位、简历等。通过工厂模式，系统可以根据需要动态创建相应的对象实例，而不需要直接依赖于具体的类实现，从而降低了耦合度。

**2.观察者模式 (Observer Pattern):**

人才招聘系统用观察者模式来实现消息通知和事件处理机制。当有新的职位发布或者用户提交简历时，系统可以通知相关的观察者，如求职者或招聘方，从而及时更新页面内容或发送通知。

**3.策略模式 (Strategy Pattern):**

人才招聘系统使用策略模式来实现不同的推荐算法或搜索算法。根据用户的搜索条件或行为，系统可以动态选择合适的算法来为用户提供最佳的搜索结果或职位推荐，从而提升用户体验。

**4.适配器模式 (Adapter Pattern):**

人才招聘系统需要与各种外部服务或数据源进行集成，如第三方支付服务、地图服务等。适配器模式可以帮助系统与这些外部服务进行交互，将外部服务的接口适配为系统内部统一的接口，从而实现数据的统一访问和交互。

这些设计模式可以帮助人才招聘系统实现各种功能，并提高系统的灵活性、可维护性和可扩展性。通过合理地运用这些设计模式，可以使系统更易于开发、测试和维护，同时也更容易满足不断变化的需求和业务规则。

1. 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):

例子: 在一个电子商务网站的购物车系统中，购物车对象是唯一的，因此可以使用单例模式确保系统中只有一个购物车实例。

特点: 单例模式确保一个类只有一个实例，并提供全局访问点。这可以节省系统资源，避免了频繁创建对象和管理实例的复杂性。但需要注意线程安全问题和懒加载的实现方式。

2)观察者模式 (Observer Pattern):

例子: 在一个社交网络应用中，用户可以关注其他用户的动态。当被关注用户发布新动态时，关注者会收到通知。

特点: 观察者模式定义了对象之间的一对多依赖关系，当一个对象状态发生变化时，其所有依赖对象都会收到通知并自动更新。这种松耦合的设计使得系统更易于维护和扩展。

3)工厂模式 (Factory Pattern):

例子: 在一个餐厅点餐系统中，根据用户点餐的种类（如主菜、饮料、甜点等），可以使用工厂模式创建对应的菜品对象。

特点: 工厂模式用于创建对象，将对象的实例化过程封装在工厂类中，客户端无需直接依赖具体的类实现，而是通过工厂来创建所需的对象。这样可以降低代码耦合度，同时也方便了对象的管理和扩展。

4)装饰器模式 (Decorator Pattern):

例子: 在一个图片编辑软件中，可以通过装饰器模式给图片添加滤镜、边框或水印等效果，而无需修改原始图片类。

特点: 装饰器模式允许动态地给对象添加新的功能，通过一系列装饰器的组合，可以实现不同的组合效果，而且对客户端来说是透明的。这样可以避免子类的爆炸性增长，提高系统的灵活性和可扩展性。

这些设计模式都是在特定情景下提出的解决方案，可以帮助开发者更好地设计和组织代码，提高系统的可维护性、可扩展性和重用性。

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。