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

在设计人事管理系统时，前端和后端通常采用了不同的设计模式和架构风格。以下是采用的设计模式：
一、前端设计模式
模型-视图-控制器（MVC）模式：
视图：前端部分使用 JavaScript 框架（如 React、Angular 或 Vue.js）开发，负责用户界面和用户交互。
控制器：在前端，控制器负责处理用户输入，并将这些输入转发给后端 API。
模型：数据模型通常由后端提供，通过 API 获取和提交数据。
二、后端设计模式
1.分层架构：
表现层：处理 HTTP 请求和响应，通常是 RESTful API。
业务逻辑层：包含系统的核心业务逻辑。
持久层：负责与数据库交互，进行数据的持久化。
2.外观模式：后端使用外观模式来提供一个简化的接口，以便前端能够方便地调用复杂的子系统。
3.代理模式：在前后端分离的架构中，代理模式用于管理客户端和服务器之间的通信，处理请求、缓存以及访问远程服务。
4.单例模式：
后端使用单例模式来确保某些类（如配置管理器、日志记录器或连接池）在应用程序生命周期内只有一个实例。
三、整体架构
综合来看，该人事管理系统采用了前后端分离的架构，通过 API 进行通信：
前后端分离架构：
前端负责呈现和用户交互，后端负责业务逻辑和数据处理。通过 RESTful API 或 GraphQL 进行通信。

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
（1）工厂模式（Factory Pattern）
例子：
在电商平台中，我们可能需要创建多种类型的产品，如手机、电脑等。使用工厂模式，我们可以定义一个抽象的产品接口（如 Product）和不同的具体产品类（如 MobilePhone、Computer），然后通过一个工厂类（如 ProductFactory）来创建不同类型的产品对象。这样，在需要创建具体产品对象的地方，我们只需要调用工厂类的方法即可。

特点：
解耦：将对象的创建过程封装在一个工厂类中，实现了对象的解耦。
可扩展性：当需要增加新的产品类型时，只需要增加相应的产品类和工厂方法，无需修改已有代码。
（2）单例模式（Singleton Pattern）
例子：
在数据库连接管理或线程池管理等场景中，我们需要确保某个类只能有一个实例。例如，通过单例模式，我们可以确保数据库连接对象只有一个，避免了资源浪费和线程安全问题。
特点：
全局唯一：保证一个类只有一个实例，并提供一个全局访问点。
线程安全：在多线程环境中，单例模式通常需要确保线程安全，避免多个线程同时创建实例。
（3）适配器模式（Adapter Pattern）
例子：
假设我们有一个旧的类库（如 LegacyLibrary），其接口与新的系统不兼容。通过适配器模式，我们可以创建一个适配器类（如 Adapter），将旧类库的接口转换成新系统期望的接口，从而使得新旧系统能够协同工作。
特点：
接口转换：将一个类的接口转换成客户端希望的另一个接口，使得原本不兼容的类能够协同工作。
复用性：在不修改已有类的情况下，通过适配器模式可以使旧代码与新系统兼容。
（4）观察者模式（Observer Pattern）
例子：
在 GUI（图形用户界面）或消息通知系统中，当一个对象（如用户界面元素或消息源）的状态发生改变时，它可能会通知多个依赖它的对象（如监听器或观察者）。例如，在 Android 中的广播机制，当广播事件发生时，注册的接收器都会收到通知并执行相应的操作。
特点：
一对多依赖：建立了一个一对多的依赖关系，使得当被观察者状态发生改变时，能够自动通知所有观察者。
解耦：观察者和被观察者之间通过抽象接口进行通信，降低了它们之间的耦合度。