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)

****Design Patterns: Elements of Reusable Object-Oriented Software（GoF设计模式）****

GoF设计模式，即《设计模式：可复用的面向对象软件的基础》一书中介绍的设计模式，由Erich Gamma、Richard Helm、Ralph Johnson和John Vlissides四位作者合著，他们被称为“四人帮”（Gang of Four，简称GOF）。这本书提出了23种设计模式，这些模式是在长期面向对象软件开发中积累的经验的总结，为开发人员提供了在软件开发过程中遇到的一般问题的最佳解决方案。

GoF设计模式主要分为以下三类：

****创建型模式（Creational Patterns）****：这类模式提供了一种在隐藏创建逻辑的同时创建对象的方法，而不是使用new运算符直接实例化对象。这包括Abstract Factory（抽象工厂模式）、Builder（建造者模式）、Factory Method（工厂方法模式）、Prototype（原型模式）和Singleton（单例模式）。这些模式的主要目的是解耦对象的创建过程，使得代码更加灵活和易于扩展。

****结构型模式（Structural Patterns）****：这类模式涉及类和对象组成。它们采用继承的概念来组合接口并定义组合对象以获得新功能的方法。包括Adapter（适配器模式）、Bridge（桥接模式）、Composite（组合模式）、Decorator（装饰器模式）、Facade（外观模式）、Flyweight（享元模式）和Proxy（代理模式）。这些模式主要用于处理类或对象之间的组合关系，以便实现更灵活的软件设计。

****行为型模式（Behavioral Patterns）****：这类模式特别关注对象之间的通信。包括Chain of Responsibility（责任链模式）、Command（命令模式）、Interpreter（解释器模式）、Iterator（迭代器模式）、Mediator（中介者模式）、Memento（备忘录模式）、Observer（观察者模式）、State（状态模式）、Strategy（策略模式）、Template Method（模板方法模式）和Visitor（访问者模式）。这些模式主要用于描述对象之间的交互和职责分配，以便实现更加灵活和可维护的软件系统。

在实际项目中，开发人员可以根据具体需求选择合适的设计模式来解决问题。通过学习和应用这些设计模式，开发人员可以提高代码的可读性、可维护性和可扩展性，从而提高软件的质量和效率。

在分析博客网站项目可能采用的设计模式时，我们参考《设计模式：可复用的面向对象软件的元素》（通常称为“GoF设计模式”）一书中的设计模式，并结合博客网站的开发历程和特性来进行分析。以下是一些在设计博客网站项目时被应用的设计模式：

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

****模型（Model）****：表示数据和业务逻辑，如博客文章、评论、用户等的数据模型以及相关的业务处理逻辑。

****视图（View）****：负责数据的展示，如博客文章的列表、详情页、用户界面等。

****控制器（Controller）****：处理用户输入，如点击按钮、提交表单等，并调用相应的业务逻辑来处理请求，然后选择合适的视图进行渲染。

1. ****工厂模式（Factory Pattern）****：

在博客网站中，可能会使用工厂模式来创建不同类型的对象，如根据用户请求创建不同类型的文章对象或评论对象。工厂模式可以隐藏对象的创建细节，使得代码更加灵活和易于维护。

1. ****单例模式（Singleton Pattern）****：

博客网站中可能有一些只需要一个实例的类，如数据库连接池、配置管理类等。使用单例模式可以确保这些类只有一个实例，并提供一个全局访问点。

1. ****观察者模式（Observer Pattern）****：

在博客网站中，观察者模式可能用于实现某些事件的通知机制，如当有新文章发布时，通知所有订阅了该博客的用户。发布者（如博客系统）会维护一个观察者列表，并在事件发生时通知所有观察者。

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

在博客网站中，可能存在一些需要按照特定步骤执行的操作，如用户注册、文章发布等。这些操作通常包含一些公共的步骤和一些可能因情况而异的步骤。使用模板方法模式可以定义一个操作中的算法框架，而将一些步骤延迟到子类中。这样可以在不改变算法结构的情况下重新定义某些步骤的具体实现。

1. ****策略模式（Strategy Pattern）****：

在博客网站中，策略模式可能用于实现不同的推荐算法或排序算法等。这些算法可以根据不同的需求进行替换，而不需要修改使用这些算法的代码。

1. ****组合模式（Composite Pattern）****：

在博客网站中，文章和评论通常具有层次结构（如文章包含评论，评论还可以包含子评论）。使用组合模式可以将这些对象组织成树形结构，并使得客户端能够以一致的方式处理单个对象和组合对象。

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）

以下是四种设计模式的例子及其特点：

****1. 工厂方法模式（Factory Method）****

****例子****：  
假设有一个博客网站，需要支持多种不同类型的文章（如技术文章、新闻文章、评论等）。每种类型的文章都有其特定的创建和初始化过程。为了解耦文章类型的创建逻辑，可以使用工厂方法模式。定义一个文章工厂接口，然后为每个文章类型实现一个具体的工厂类。

****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。

核心工厂类不再负责所有产品的创建，而是将具体创建的工作交给子类去做。

****2. 单例模式（Singleton）****

****例子****：  
在博客网站中，有一个全局唯一的配置管理类，用于管理整个网站的配置信息。这个类应该只被实例化一次，并且可以在任何地方被访问到。此时可以使用单例模式来实现。

****特点****：

确保一个类仅有一个实例，并提供一个全局访问点。

构造函数被声明为私有，防止外部类实例化该类的对象。

提供了一个静态方法来获取该类的实例。

****3. 建造者模式（Builder）****

****例子****：  
在博客网站中，创建复杂的用户对象可能涉及到多个属性的设置，如用户名、密码、邮箱、地址等。为了避免使用大量的setter方法来设置这些属性，可以使用建造者模式。定义一个用户建造者接口，然后提供一个具体的建造者类来构建用户对象。

****特点****：

将一个复杂对象的构建与它的表示分离，使得同样的构建过程可以创建不同的表示。

允许用户只通过指定复杂对象的类型和内容就可以构建它们。

建造者模式隐藏了复杂对象的创建过程，并允许按照步骤构造对象。

****4. 适配器模式（Adapter）****

****例子****：  
假设博客网站需要与一个旧的评论系统进行集成，但新旧系统的接口不兼容。为了解决这个问题，可以创建一个适配器类，它实现了新系统的接口，但内部使用旧系统的接口来调用相关功能。

****特点****：

将一个类的接口转换成客户希望的另一个接口，使得原本由于接口不兼容而无法一起工作的类能够一起工作。

适配器模式分为类结构型模式和对象结构型模式两种。

适配器模式允许由于接口不兼容而不能交互的类协同工作。

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。

