1. 单例模式（Singleton）

目的：

保证一个类在系统中只有一个实例，并提供一个全局访问点。

优点：

确保唯一性：避免资源冲突（如数据库连接、线程池）；

全局访问点：可在程序中任何地方访问该实例；

可延迟加载：懒加载提升启动性能；

缺点：

违背单一职责原则：类既管理实例又包含业务逻辑；

难以扩展：不易继承和测试（不利于依赖注入）；

多线程实现需注意同步，否则可能创建多个实例；

隐藏依赖：单例类似全局变量，可能导致代码耦合、难以追踪。

适用场景：

程序中只需一个实例控制资源的场景，例如：

配置管理器（Configuration）

日志记录器（Logger）

线程池、数据库连接池

操作系统资源管理器（打印机管理等）

典型应用：

操作系统中的任务管理器（全局唯一）

浏览器中的缓存管理器

游戏中的全局控制器（如声音管理、资源加载）

2.工厂模式（Factory）

目的：

定义一个用于创建对象的接口，让子类决定实例化哪个类，将对象的创建和使用解耦。

优点：

解耦：将对象的创建和使用解耦，使用者无需知道具体的创建逻辑。

易于扩展：增加新产品时，无需修改现有代码，只需增加新的子类和对应的工厂方法。

灵活性：可以根据不同的输入参数动态创建不同的对象实例。

缺点：

增加系统复杂度：引入工厂类或方法，可能导致代码复杂性增加，尤其是当产品类型较多时。

违反开闭原则：添加新产品时，通常需要修改工厂类代码，违反了“对修改关闭”的原则。

紧密耦合：工厂类需要知道所有具体产品类的细节，导致工厂与产品类耦合度较高。

测试困难：工厂类包含复杂逻辑，且依赖于具体产品类，增加了单元测试的难度。

适用场景：

产品创建：例如在图形界面库中，根据不同的需求动态创建按钮、窗口等控件。

数据库访问：根据不同的数据库类型（如 MySQL、PostgreSQL），动态创建相应的数据库访问对象。

日志框架：根据不同的日志记录方式（如文件、控制台、网络），动态创建相应的日志记录对象。

用户角色管理：在博客系统中，根据用户角色（如普通用户、管理员）动态创建不同的用户对象。

典型应用：

用户管理：根据用户角色动态创建普通用户或管理员对象。

资源分配：根据不同的资源需求动态创建资源对象。

插件系统：根据不同的插件类型动态加载和创建插件实例。

3.观察者模式(Observer)

目的：

建立对象之间一对多的依赖关系，当一个对象状态变化时，其所有依赖对象都会自动收到通知并更新。

在个人博客系统中，适用于例如用户订阅博主、文章发布通知、评论通知等功能 —— 一旦事件发生，所有关注者或观察者可以自动获知变化。

优点：

低耦合：被观察者（Subject）无需知道观察者（Observer）具体实现，只需维护接口引用；易于扩展：动态添加或删除观察者，不影响其他逻辑；

增强模块独立性：事件源与监听者解耦，便于系统维护与测试；

支持广播通信：一个事件通知多个观察者，无需显式调用每一个观察者对象；

缺点：

性能问题：观察者过多时，通知广播可能影响性能；

通知顺序难以控制：多个观察者响应顺序可能不一致，可能导致状态不一致；

内存管理复杂：观察者未及时注销可能导致内存泄漏（尤其在 GUI、WebSocket 等长连接场景）；

适用场景：

文章发布通知系统（作者发布文章时，通知所有订阅该作者的用户更新文章流或推送提醒）

评论消息提醒（某篇文章收到新评论后，通知作者或其他评论者）

后台事件记录器（页面访问、登录、点赞等触发日志观察器进行持久化记录）

消息推送系统（用户关注动态或被 @ 时触发推送模块发送站内信或邮件）

典型应用：

事件系统/发布订阅系统

模型-视图-控制器（MVC）框架

数据同步与缓存刷新

消息队列和日志收集

4.装饰模式（Decorator）

目的：

在不修改原始类的前提下，动态地为对象添加额外功能（遵循开闭原则）。

结构：

由抽象组件、具体组件、抽象装饰器、具体装饰器组成，装饰器持有组件的引用，实现同一接口。

实现机制：

使用 组合（而非继承） 来扩展功能；

装饰器可以 递归嵌套，形成功能叠加；

所有装饰器和原始对象对外 接口一致，可互换。

优点：

功能可灵活组合、任意叠加；

保持原类封闭，无需修改原始代码；

每个装饰器关注一个功能，符合 单一职责原则。

缺点：

多层装饰会导致系统结构复杂、调试困难；

对象数量增多，不易理解装饰链的实际行为。

适用场景：

需要按需、运行时添加功能；

不希望使用继承或无法修改已有类时；

有多个功能组合需求，但不希望为每种组合写一个子类。

典型应用：

Java I/O 流（如 BufferedReader 装饰 InputStreamReader）；

图形界面控件（为按钮添加边框、阴影等）；

日志框架、权限校验、中间件链式处理等。