**一、设计模式（Design Patterns）简介：**

设计模式**代表了最佳实践**，通常被有经验的面向对象的开发人员使用。

设计模式是程序猿在软件开发过程中面临的**一般问题的解决方案**。

设计模式是一套**反复使用**的、**多数人知晓**的、经过**分类编目**的、**代码设计经验**的总结。

*（注：分类编目的意思是设计模式有很多分类，每一种设计模式有其不同作用和使用场景，本人大胆猜测…）*

**二、设计模式对编程的影响：**

使用设计模式可以增强代码的**重用性**，保证代码的**可靠性**，使代码**更加的优雅容易被后续人员理解**。

设计模式对于维护人员和系统都是多赢的，它**使代码编制真正的工程化**，设计模式**是代码设计的基石**，如同大厦的一块块砖头一样。

项目中合理的使用设计模式可以很完美的解决很多问题，**每一种模式都有其应用场景**，且都描述了一个反复发生的问题以及该问题的解决方案。

**三、那么什么是GoF呢？设计模式跟这个又有何牵扯？**

GoF是Gang of Four的简称，中文翻译：四人帮，23种设计模式最先就是他们提出来的，也可以称之为设计模式的四个霸霸。

在 1994 年，由 Eric Gamma、Richard Helm、Ralph Johnson 和 John Vlissides 四位博士合著出版了一本名为 Design Patterns - Elements of Reusable Object-Oriented Software（中文译名：设计模式 - 可复用的面向对象软件元素） 的书，该书首次提到了软件开发中设计模式的概念。

四位作者合称 GoF。他们所提出的设计模式主要是基于以下的面向对象设计原则：

**1）对接口编程而不是对实现编程。**

**2）优先使用对象组合而不是继承。**

**四、GoF设计模式的分类：**

GoF提出的设计模式有23种，且大致可以分成三类：

**创建型模式（Creational Patterns）**

**结构型模式（Structural Patterns）**

**行为型模式（Behavioral Patterns）**

*（注：下面粗体标记的是我接触过比较多的设计模式，你可以重点研究一下，也可以选择忽略… 通透理解设计模式能帮助我们很好的阅读框架或第三方依赖源码）*

1）创建型模式

这些设计模式提供了一种在创建对象的同时隐藏创建逻辑的方式，而不是使用 new 运算符直接实例化对象。这使得程序在判断针对某个给定实例需要创建哪些对象时更加灵活。

**工厂模式（Factory Pattern）**

**抽象工厂模式（Abstract Factory Pattern）**

**单例模式（Singleton Pattern）**

**建造者模式（Builder Pattern）**

**原型模式（Prototype Pattern）**

2）结构型模式

这些设计模式关注类和对象的组合。继承的概念被用来组合接口和定义组合对象获得新功能的方式。

**适配器模式（Adapter Pattern）**

桥接模式（Bridge Pattern）

**过滤器模式（Filter、Criteria Pattern）**

组合模式（Composite Pattern）

**装饰器模式（Decorator Pattern）**

外观模式（Facade Pattern）

享元模式（Flyweight Pattern）

**代理模式（Proxy Pattern）**

3）行为型模式

这些设计模式特别关注对象之间的通信。

责任链模式（Chain of Responsibility Pattern）

命令模式（Command Pattern）

解释器模式（Interpreter Pattern）

**迭代器模式（Iterator Pattern）**

中介者模式（Mediator Pattern）

备忘录模式（Memento Pattern）

**观察者模式（Observer Pattern）**

状态模式（State Pattern）

空对象模式（Null Object Pattern）

策略模式（Strategy Pattern）

**模板模式（Template Pattern）**

访问者模式（Visitor Pattern）

**\* J2EE 模式**

上面这23种只是GoF提出来的，如果要延伸的话肯定还有其他设计模式，比如Java里面的J2EE设计模式，Spring里面就有用到很多J2EE设计模式。

（*注：建议直接去研究Spring框架，推荐《Spring in Action》这本书最新版，中文名是《Spring实战》*）

这些设计模式特别关注表示层。这些模式是由 Sun Java Center 鉴定的。

MVC 模式（MVC Pattern）

业务代表模式（Business Delegate Pattern）

组合实体模式（Composite Entity Pattern）

数据访问对象模式（Data Access Object Pattern）

前端控制器模式（Front Controller Pattern）

拦截过滤器模式（Intercepting Filter Pattern）

服务定位器模式（Service Locator Pattern）

传输对象模式（Transfer Object Pattern）

**五、日本IT大佬结城浩在《图解设计模式》一书中对设计模式做的分类：**

1）适应设计模式：

这里面都是比较简单容易理解的设计模式，帮助你理解概念的，入门级的。

迭代器模式（Iterator Pattern）

适配器模式（Adapter Pattern）

2）交给子类：

继承相关的

模板模式（Template Pattern）

工厂模式（Factory Pattern）

3）生成实例

抽象工厂模式（Abstract Factory Pattern）

单例模式（Singleton Pattern）

建造者模式（Builder Pattern）

原型模式（Prototype Pattern）

4）分开考虑

分开考虑易变得杂乱无章的处理

桥接模式（Bridge Pattern）

策略模式（Strategy Pattern）

5）一致性

能够让两个看上去不同的对象的操作变得统一

装饰器模式（Decorator Pattern）

组合模式（Composite Pattern）

6）访问数据结构

访问者模式（Visitor Pattern）

责任链模式（Chain of Responsibility Pattern）

7）简单化

让类关系变得更简单

外观模式（Facade Pattern）

中介者模式（Mediator Pattern）

8）管理状态

与对象的状态相关

观察者模式（Observer Pattern）

备忘录模式（Memento Pattern）

状态模式（State Pattern）

9）避免浪费 提高处理效率

享元模式（Flyweight Pattern）

代理模式（Proxy Pattern）

10）用类来表现特殊的东西

解释器模式（Interpreter Pattern）

解释器模式（Interpreter Pattern）

**六、设计模式的六大原则：**

**1、开闭原则（Open Close Principle）**

开闭原则的意思是：**对扩展开放，对修改关闭**。在程序需要进行拓展的时候，不能去修改原有的代码，实现一个热插拔的效果。简言之，是为了使程序的扩展性好，易于维护和升级。想要达到这样的效果，我们需要使用接口和抽象类，后面的具体设计中我们会提到这点。

**2、里氏代换原则（Liskov Substitution Principle）**

里氏代换原则是面向对象设计的基本原则之一。 里氏代换原则中说，**任何基类可以出现的地方，子类一定可以出现**。LSP 是继承复用的基石，只有当派生类可以替换掉基类，且软件单位的功能不受到影响时，基类才能真正被复用，而**派生类也能够在基类的基础上增加新的行为**。里氏代换原则是对开闭原则的补充。实现开闭原则的关键步骤就是抽象化，而基类与子类的继承关系就是抽象化的具体实现，所以里氏代换原则是对实现抽象化的具体步骤的规范。

**3、依赖倒转原则（Dependence Inversion Principle）**

这个原则是开闭原则的基础，具体内容：**针对接口编程，依赖于抽象而不依赖于具体**。

**4、接口隔离原则（Interface Segregation Principle）**

这个原则的意思是：**使用多个隔离的接口，比使用单个接口要好**。它还有另外一个意思是：**降低类之间的耦合度**。由此可见，其实设计模式就是从大型软件架构出发、便于升级和维护的软件设计思想，它强调降低依赖，降低耦合。

**5、迪米特法则，又称最少知道原则（Demeter Principle）**

最少知道原则是指：**一个实体应当尽量少地与其他实体之间发生相互作用，使得系统功能模块相对独立**。

**6、合成复用原则（Composite Reuse Principle）**

合成复用原则是指：**尽量使用合成/聚合的方式，而不是使用继承**。