面向对象思想设计原则

在实际的开发中，我们要想更深入的了解面向对象思想，就必须熟悉前人总结过的面向对象的思想的设计原则

单一职责原则

开闭原则

里氏替换原则

依赖注入原则

接口分离原则

迪米特原则

单一职责原则

其实就是开发人员经常说的”高内聚，低耦合”

也就是说，每个类应该只有一个职责，对外只能提供一种功能，而引起类变化的原因应该只有一个。在设计模式中，所有的设计模式都遵循这一原则。

开闭原则

核心思想是：一个对象对扩展开放，对修改关闭。

其实开闭原则的意思就是：对类的改动是通过增加代码进行的，而不是修改现有代码。也就是说软件开发人员一旦写出了可以运行的代码，就不应该去改动它，而是要保证它能一直运行下去，如何能够做到这一点呢?这就需要借助于抽象和多态，即把可能变化的内容抽象出来，从而使抽象的部分是相对稳定的，而具体的实现则是可以改变和扩展的

里氏替换原则

核心思想：在任何父类出现的地方都可以用它的子类来替代。

其实就是说：同一个继承体系中的对象应该有共同的行为特征。

依赖注入原则

核心思想：要依赖于抽象，不要依赖于具体实现。

其实就是说：在应用程序中，所有的类如果使用或依赖于其他的类，则应该依赖这些其他类的抽象类，而不是这些其他类的具体类。为了实现这一原则，就要求我们在编程的时候针对抽象类或者接口编程，而不是针对具体实现编程。

接口分离原则

核心思想：不应该强迫程序依赖它们不需要使用的方法。

其实就是说：一个接口不需要提供太多的行为，一个接口应该只提供一种对外的功能，不应该把所有的操作都封装到一个接口中。

迪米特原则

核心思想：一个对象应当对其他对象尽可能少的了解

其实就是说：降低各个对象之间的耦合，提高系统的可维护性。在模块之间应该只通过接口编程，而不理会模块的内部工作原理，它可以使各个模块耦合度降到最低，促进软件的复用

设计模式概述

设计模式（Design pattern）是一套被反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结。使用设计模式是为了可重用代码、让代码更容易被他人理解、保证代码可靠性。

设计模式不是一种方法和技术，而是一种思想

设计模式和具体的语言无关，学习设计模式就是要建立面向对象的思想，尽可能的面向接口编程，低耦合，高内聚，使设计的程序可复用

学习设计模式能够促进对面向对象思想的理解，反之亦然。它们相辅相成

设计模式的几个要素

**A:**名字 必须有一个简单，有意义的名字

B:问题 描述在何时使用模式

C:解决方案 描述设计的组成部分以及如何解决问题

D:效果 描述模式的效果以及优缺点

设计模式：经验的总结

A:创建型 创建对象

B:结构型 对象的组成

C:行为型 对象的功能

设计模式的分类

创建型模式 对象的创建

创建型模式：简单工厂模式，工厂方法模式，抽象工厂模式，建造者模式，原型模式，单例模式。(6个)

结构型模式 对象的组成(结构)

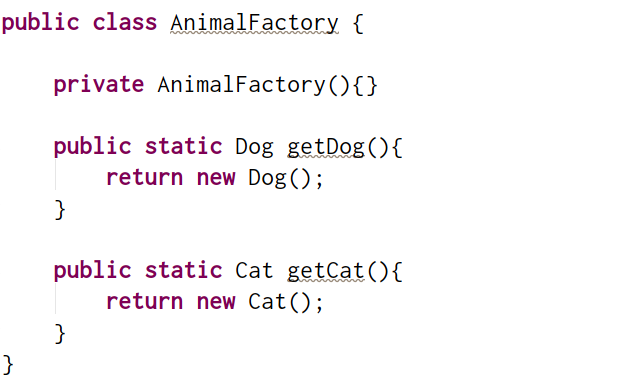
结构型模式：外观模式、适配器模式、代理模式、装饰模式、桥接模式、组合模式、享元模式。(7个)

行为型模式 对象的行为

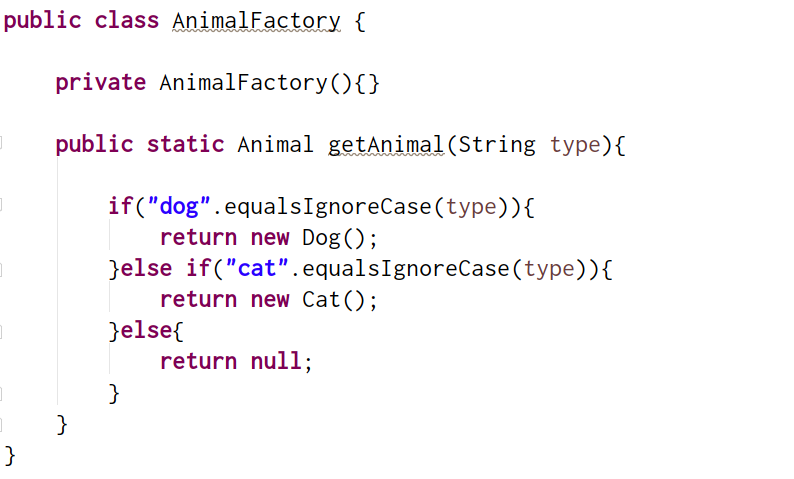
行为型模式：模版方法模式、观察者模式、状态模式、职责链模式、命令模式、访问者模式、策略模式、备忘录模式、迭代器模式、解释器模式。(10个)

简单工厂模式：

我有一个动物类，又有猫类和狗类继承动物类，当我想创建猫和狗的对象时，我需要自己new一个，这样的方式不好，很麻烦。那么我就可以提供一个动物工厂类，这个类的构造方法是私有化的，但是专门提供了公有的静态方法，可以让用户获取猫和狗，这样非常方便具体代码如下：



但是这样并不好，因为动物千千万，我不可能在动物工厂类中写出所有的get各种动物的方法，所以需要改进，我能不能只提供一个get方法，你想要什么，我就造什么呢？



这种办法，只提供了一个获取方法，非常好的解决了上述问题

简单工厂模式概述

又叫静态工厂方法模式，它定义一个具体的工厂类负责创建一些类的实例

优点

客户端不需要在负责对象的创建，从而明确了各个类的职责

缺点

这个静态工厂类负责所有对象的创建，如果有新的对象增加，或者某些对象的创建方式不同，就需要不断的修改工厂类，不利于后期的维护

工厂方法模式(优化简单工厂模式)

工厂方法模式概述

工厂方法模式中抽象工厂类负责定义创建对象的接口，具体对象的创建工作由继承抽象工厂的具体类实现。

优点

客户端不需要在负责对象的创建，从而明确了各个类的职责，如果有新的对象增加，只需要增加一个具体的类和具体的工厂类即可，不影响已有的代码，后期维护容易，增强了系统的扩展性

缺点

需要额外的编写代码，增加了工作量

