单例模式

设计模式

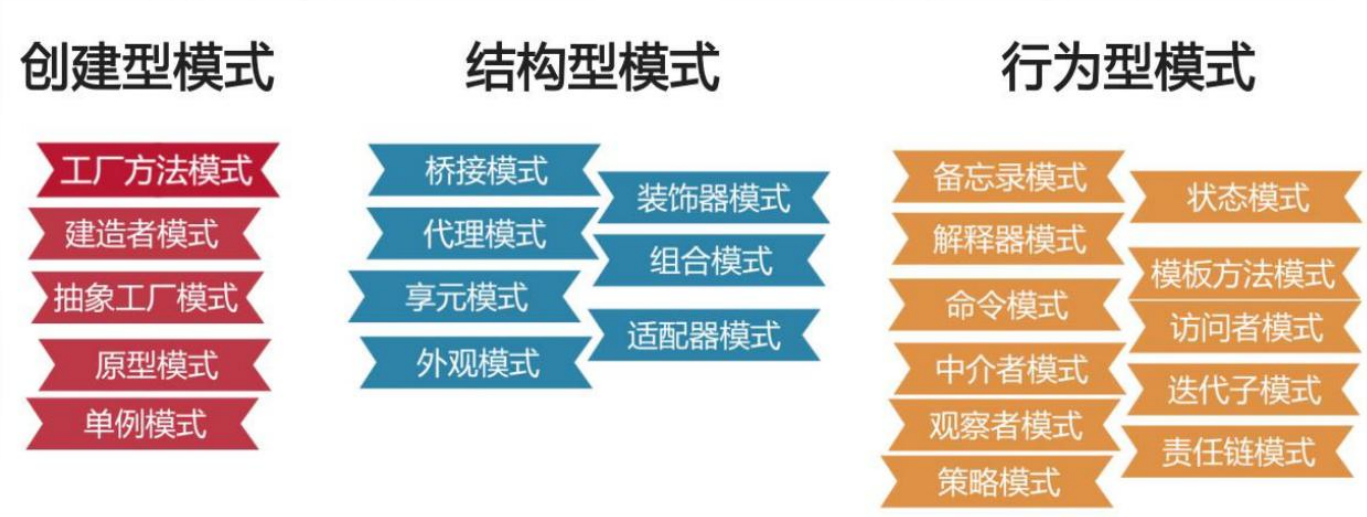
- 设计模式（Design pattern）是一套被反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结。
- 设计模式是软件开发人员在软件开发过程中面临的一般问题的解决方案。

设计模式

- 项目中合理的运用设计模式可以完美的解决很多问题
- 每种模式在现在中都有相应的原理来与之对应
- 每一个模式描述了一个在我们周围不断重复发生的问题，以及该问题的核心解决方案

设计模式

- 面向对象的设计模式很多，但大家认为这23个模式是其它模式的基础



单例模式（Singleton）

- 滚滚历史，朝代更迭，永恒定律：
  - 一个朝代皇帝只有一个
  - 军队的最高司令官只有一个
  - 一山不容二虎
- 计算机系统：
  - 驱动程序
  - 打印机后台处理程序（Printer Spooler）
  - 线程池
  - 缓存
  - 日志

单例模式（Singleton）

目的：

使得类的一个对象成为该类系统中的唯一实例

定义：

一个类有且仅有一个实例，并且自行实例化向整个系统提供

单例模式

要点：

1、某个类只能有一个实例

2、必须自行创建实例

3、必须自行向整个系统提供这个实例

实现：

1、只提供私有的构造方法

2、含有一个该类的静态私有对象

3、提供一个静态的公有方法用于创建、获取静态私有对象

单例模式

代码实现方案：

1、饿汉式

2、懒汉式

饿汉式 PK 懒汉式

1、饿汉式在类加载时就创建实例， 第一次加载速度快；
懒汉式第一次使用时才进行实例化，第一次加载速度慢
饿汉式：空间换时间
懒汉式：时间换空间

2、饿汉式线程安全；懒汉式存在线程风险
解决方案：

1、同步锁　　3、静态内部类
2、双重校验锁　　4、枚举

单例模式

优点：

1、在内存中只有一个对象，节省内存空间
2、避免频繁的创建销毁对象，提高性能
3、避免对共享资源的多重占用

缺点：

1、扩展比较困难
2、如果实例化后的对象长期不利用，系统将默认为垃圾进行回收，造成对象状态丢失

单例模式

适用场景：

1、创建对象时占用资源过多，但同时又需要用到该类对象

2、对系统内资源要求统一读写，如读写配置信息

3、当多个实例存在可能引起程序逻辑错误，如号码生成器