## 单例模式

单例模式的要素：私有的构造方法；指向自己实例的私有静态引用；以自己实例为返回值的静态的公有的方法。

### 饿汉式单例

public class Singleton {

//在类被加载时就实例化一个对象

private static Singleton singleton = new Singleton();

private Singleton(){}

public static Singleton getInstance(){

return singleton;

}

}

### 懒汉式单例

public class Singleton {

private static Singleton singleton;

private Singleton(){}

public static synchronized Singleton getInstance(){

if(singleton==null){

singleton = new Singleton();

}

return singleton;

}

}

## prototype原型模式

使用原型模式复制对象不会调用类的构造方法

因为对象的复制是通过调用Object类的clone方法来完成的，它直接在内存中复制数据，因此不会调用到类的构造方法。不但构造方法中的代码不会执行，甚至连访问权限都对原型模式无效

使用场景：

创建新对象成本较大（如初始化需要占用较长的时间，占用太多的CPU资源或网络资源），新的对象可以通过原型模式对已有对象进行复制来获得，如果是相似对象，则可以对其成员变量稍作修改。

如果系统要保存对象的状态，而对象的状态变化很小，或者对象本身占用内存较少时，可以使用原型模式配合备忘录模式来实现。

原型模式的主要思想是基于现有的对象克隆一个新的对象出来，一般是有对象的内部提供克隆的方法，通过该方法返回一个对象的副本，这种创建对象的方式，相比我们之前说的几类创建型模式还是有区别的，之前的讲述的工厂模式与抽象工厂都是通过工厂封装具体的new操作的过程，返回一个新的对象，有的时候我们通过这样的创建工厂创建对象不值得，特别是以下的几个场景的时候，可能使用原型模式更简单也效率更高。

当一个系统应该独立于它的产品创建、构成和表示时，要使用 Prototype模式

当要实例化的类是在运行时刻指定时，例如，通过动态装载；

为了避免创建一个与产品类层次平行的工厂类层次时

当一个类的实例只能有几个不同状态组合中的一种时。建立相应数目的原型并克隆它们可能比每次用合适的状态手工实例化该类更方便一些。（也就是当我们在处理一些对象比较简单，并且对象之间的区别很小，可能只是很固定的几个属性不同的时候，可能我们使用原型模式更合适）。