《EffectiveJava》学习笔记

**1.考虑静态工厂方法代替构造器：**

优势：

1）.有名称；

2）.不必在每次调用它们的时候都创建一个新对象；

3）.可以返回原返回类型的任何子类型的对象；

4）.在创建参数化类型实例的时候，它们使代码变得更加简洁。

缺点：

1）.类如果不含公有的或者受保护的构造器，就不能被子类化；

2）.它们与其他静态方法实际上没有区别。















**2.遇到多个构造器参数时要考虑用构建器：**

1）、重叠构造器模式可行，但是当有许多参数的时候，客户端代码会很难编写，并且难以阅读；

2）、JavaBean模式也可行，但是有个很严重的缺点，JavaBean模式阻止了把类做成不可变的可能，因此线程不安全；

3）、因此第三种方式，既可以保证像重叠构造器模式那样的安全性，也能保证像JavaBean模式那么好的可读性，也就是Builder模式。

Builder模式：

不直接生成想要的对象，而是让客户端利用所有必要的参数调用构造器，得到一个builder对象，然后客户端在builder对象上调用类似setter的方法，来设置每个相关的可选参数，最后客户端调用无参的builder方法生成不可变的对象。







**3.用私有构造器或者枚举类型强化Singleton属性：**

Singleton指仅仅被实例化一次的类。在jdk1.5前有多中方式实现单例模式，例如静态工厂方法、静态内部类等，但是它们各有弊端，比如线程问题或者反射问题，当然，抵御反射攻击时，可以修改构造器，让它在被要求创建第二个实例的时候抛出异常。



jdk1.5之后，单元素的枚举类型已经成为实现Singleton的最佳方法，既线程安全，又可避免反射攻击，同时代码也很优雅。

**4.通过私有构造器强化不可实例化的能力**

一些只包含静态方法或者静态域的类，比如工具类等，我们并不希望实例化该类，因为实例化并没有意义，此时就可以显示的私有化构造器来禁止类被实例化。但是也有副作用就是不能被子类继承了。