第1条：考虑用静态工厂方法代替构造器

优点：

1.它们有名称。

1.1有适当名称的静态工厂方法会更容易阅读，更容易使用。

2.1使用名称来区分多个带有相同签名的构造器。

2.不必在每次调用它们的时候都创建一个新对象。

2.1缓存预先构建好的对象，重复利用。

2.2编写实例受控的类，严格控制在哪个时刻哪些对象应该存在。

3.可以返回原类型的类型的任何子类型对象。

3.1API可以选择返回对象，同时又不会使对象的类变成公有的，以这种方式隐藏公有类会使API变的更加灵活，简洁。

4.在创建参数化类型的时候，它们使代码变得更简洁。

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。弥补方案使用组合代替继承。

2.它们与静态方法实际上没有任何区别。弥补方案时在类或者接口注释中关注静态工作，并遵守标准的命名习惯。

例如：valueOf、of、getInstance、newInstance、getType、newType

第2条：遇到多个构造器参数时要考虑用构建器

当遇到多个构造器参数时，可以考虑使用重叠构造器模式、JavaBeans模式

1.重叠构造器模式缺点：

当有许多参数的时候，客户端代码会很难编写，并且较难阅读。

2.JavaBeans模式缺点：

构造过程中JavaBeans可能会处于不一致的状态。

JavaBeans模式阻止把类做成不可变的可能，这需要付出额外的努力来确保它的现场安全性。

构建器模式优点：

既能保证重叠构造器模式那样的安全性，也可以保证像JavaBeans模式那么好的可读性。

缺点：存在一定的性能开销，构建器模式比重叠构造器模式更加冗长

第3条：用私有构造器或者枚举类型强化Singleton属性

1.公有域方法的好处：组成类的成员的声明很清楚地表明这个类时一个Singleton，公有的静态域是final的，所以该域将总是包含相同的对象引用。
2.工厂方法的优势：在不改变其API的前提下，我们可以改变该类是否应该是Singleton的想法。
3.单元素的枚举类型：在功能上与公有域方法相近，但是它更加简洁。面对复杂的序列化或者反射攻击可以防止多次实例化。

第4条：通过私有构造器强化不可实例化的能力

第5条：避免创建不必要的对象

1.如果对象是不可变的，它就始终可以被重用。
2.对于同时提供了静态工厂方法和构造器的不可变类，通常可以使用静态工厂方法而不是构造器，以避免创建不必要的对象。
3.重用那些已知不会被修改的可变对象。
4.要优先使用基本类型而不是装箱基本类型，要当心无意识的自动装箱。

对象池使用缺点缺点：
代码很乱，不容易维护，同时增加内存占用。
优点：
如果池中的对象是非常重量级的，创建对象需要付出非常昂贵的代价，使用对象池是很有意思的。

第6条：消除过期的对象引用

1.没有必要对于每一个对象引用，一旦程序不再用到它，就把它清空。这样会把代码弄得很乱。清空对象引用应该是一种例外，而不是一种规范行为。
2.只要类是自己管理内存，就应该警惕内存泄漏问题。

第7条：避免使用终结方法

存在的问题：
1.终结方法存在行为不稳定、可移植性问题。

Java不仅不能保证终结方法会被及时地执行，而且根本就不能保证它们会被执行。
2.使用终结方法有一个非常严重的性能损失。

使用场景：
1.终结方法可以充当最后的安全网。
2.终止非关键的本地资源。

替代方案
1.显示的终止方法通常与try-finally结构结合起来使用，以确保及时终止。
2.公有非final方法如果需要使用终结方法，优先考虑终结方法守卫者。