链滴

《高质量 java 代码》总结

作者：valarchie
原文链接：https://ld246.com/article/1577722762853
来源网站：链滴

<blockquote>
整理自《编写高质量代码:改善 Java 程序的 151 条建议》
</blockquote>
<h3 id="一-Java开发中通用的方法和准则">一、Java 开发中通用的方法和准则</h3>

不要在常量和变量中出现易混淆的字母;
莫让常量蜕变成变量;
三元操作符的类型务必一致;
避免带有变长参数的方法重载;
别让 null 值和空值威胁到变长方法;
覆写变长方法也要循规蹈矩;
警惕自增的陷阱;
不要让旧语法困扰你;
少用静态导入;
不要在本类中覆盖静态导入的变量和方法;
养成良好习惯，显式声明 UID;
避免用序列化类在构造函数中为不变量赋值
避免为 final 变量复杂赋值;
使用序列化类的私有方法巧妙解决部分属性持久化问题;
break 万万不可忘;
易变业务使用脚本语言编写;
慎用动态编译;
避免 instantceof 非预期结果;
断言对决不是鸡肋;
不要只替换一个类;

<h3 id="二-基本类型">二、基本类型</h3>

使用偶判断，不用奇判断;
用整数类型处理货币;
不要让类型默默转换;
边界，边界，还是边界;
不要让四舍五入亏了一方;
提防包装类型的 null 值;
谨慎包装类型的大小比较;
优先使用整型池;
优先选择基本类型;
不要随便设置随机种子;

<h3 id="三-类-对象及方法">三、类、对象及方法</h3>

在接口中不要存在实现代码;
静态变量一定要先声明后赋值;
不要覆写静态方法;
构造函数尽量简化;
避免在构造函数中初始化其他类;
使用构造代码块精炼程序;
使用静态内部类提供封装性;
使用匿名类的构造函数;
匿名类的构造函数很特殊;
让多重继承成为现实;
让工具类不可实例化;
避免对象的浅拷贝;
推荐使用序列化实现对象的拷贝;

覆写 equals 方法时不要识别不出自己;
equals 应该考虑 null 值情景;
在 equals 中使用 getClass 进行类型判断;
覆写 equals 方法必须覆写 hashCode 方法;
推荐覆写 toString 方法;
使用 package-info 类为包服务;
不要主动进行垃圾回收;

<h3 id="四-字符串">四、字符串</h3>

推荐使用 String 直接量赋值;
注意方法中传递的参数要求;
正确使用 String、StringBuffer、StringBuilder;
注意字符串的位置;
自由选择字符串拼接方法;
推荐在复杂字符串操作中使用正则表达式;
强烈建议使用 UTF 编码;
对字符串排序持一种宽容的心态;

<h3 id="五-数组和集合">五、数组和集合</h3>

性能考虑，数组是首选;
若有必要，使用变长数组;
警惕数组的浅拷贝;
在明确的场景下，为集合指定初始容量;
多种最值方法，适时选择;
避开基本类型数组转换列表陷阱;
asList 方法产生的 List 对象不可更改;
不同的列表选择不同的遍历方法;
频繁插入和删除时使用 LinkedList;
列表相等只需关心元素数据;
推荐使用 subList 处理局部列表;
生成子表后不要再操作原列表;
使用 Comparator 进行排序;
不推荐使用 binarySearch 对列表进行检索;
集合中的元素必须做到 compareTo 和 equals 同步;
集合运算时使用更优雅的方式;
使用 shuffle 大乱列表;
减少 HashMap 中元素的数量;
集合中的哈希码不要重复;
多线程使用 Vector 或者 HashTable;
非稳定排序推荐使用 List;

<h3 id="六-枚举和注解">六、枚举和注解</h3>

推荐使用枚举定义常量;
使用构造函数协助描述枚举项;
小心 switch 带来的空值异常;
在 switch 的 default 代码块中增加 AssertionError 错误;
使用 valueOf 前必须进行校验;
用枚举实现工厂方法模式更简洁;
枚举项的数量限制在 64 个以内;
小心注解继承;
枚举和注解结合使用威力更大;

原文链接：《高质量 java 代码》总结

注意 @Override 不同版本的区别;

<h3 id="七-枚举和注解">七、枚举和注解</h3>

Java 的泛型是类型擦除的;
不能初始化泛型参数和数组;
强制声明泛型的实际类型;
不同的场景使用不同的泛型通配符;
警惕泛型是不能协变和逆变的;
建议采用的顺序是 List、List<?>、List<object>;
严格限定泛型类型采用多重界限;
数组的真实类型必须是泛型类型的子类型;
注意 Class 类的特殊性;
适时选择 getDeclaredXXX 和 getXXX;
反射访问属性或方法时将 Accessible 设置为 true;
使用 forName 动态加载类文件;
动态加载不合适数组;
动态代理可以使代理模式更加灵活;
反射让模板方法模式更强大;
不需要太多关注反射效率;

<h3 id="八-异常">八、异常</h3>

提倡异常封装;
采用异常链传递异常;
受检异常尽可能转化为非受检异常;
不要在 finally 块中处理返回值;
使用 Throwable 获取栈信息;
异常只为异常服务;
多使用异常，把性能问题放一边;

<h3 id="九-多线程和并发">九、多线程和并发</h3>

不推荐覆写 start 方法;
启动线程前 stop 方法是不可靠的;
不使用 stop 方法停止线程;
线程优先级只使用三个等级;
使用线程异常处理器提升系统可靠性;
volatile 不能保证数据同步;
异步运算多考虑使用 Callable 接口;
优先选择线程池;
适时选择不同的线程池来实现;
Lock 与 synchronized 是不一样的;
预防线程死锁;
适当设置阻塞队列长度;
使用 CountDownLatch 协调子线程;
CyclicBarrier 让多线程齐步走;

</object>