**华为 Java 编程军规，每季度代码验收标准**

**1、引言：**

这个标准是衡量代码本身的缺陷，也是衡量一个研发人员本身的价值。华为作为一家全球化的 IT 公司，十几万员工，无论是人事管理，还是代码管理，都是一件不容易的事情，没有规范的约束，想想都是件可怕的事情。下面挑选了一些网上流传的编程规范，一起来学习下，以下内容不涉及基础的语法规范（请见 Refer），更侧重于一些编程习惯，如何提高程序的健壮性、可维护性等。（PS：以下内容未经官方考证，如阅读者出现不适，请选择立即关闭本页 -\_-||| ）

**2、军规简介：**

**军规一：**【避免在程序中使用魔鬼数字，必须用有意义的常量来标识。】

**军规二：**【明确方法的功能，一个方法仅完成一个功能。】

**军规三：**【方法参数不能超过5个】

**军规四：**【方法调用尽量不要返回null，取而代之以抛出异常，或是返回特例对象（SPECIAL CASE object，SPECIAL CASE PATTERN）；对于以集合或数组类型作为返回值的方法，取而代之以空集合或0长度数组。】

**军规五：**【在进行数据库操作或IO操作时，必须确保资源在使用完毕后得到释放，并且必须确保释放操作在finally中进行。】

**军规六：**【异常捕获不要直接catch (Exception ex) ，应该把异常细分处理。】

**军规七：**【对于if „ else if „(后续可能有多个else if …)这种类型的条件判断，最后必须包含一个else分支，避免出现分支遗漏造成错误；每个switch-case语句都必须保证有default，避免出现分支遗漏，造成错误。】

**军规八：**【覆写对象的equals()方法时必须同时覆写hashCode()方法。】

**军规九：**【禁止循环中创建新线程，尽量使用线程池。】

**军规十：**【在进行精确计算时(例如:货币计算)避免使用float和double，浮点数计算都是不精确的，必须使用BigDecimal或将浮点数运算转换为整型运算。】

**3、军规说明**

**军规一：【避免在程序中使用魔鬼数字，必须用有意义的常量来标识。】**

说明：是否是魔鬼数字要基于容易阅读和便于全局替换的原则。0、1作为某种专业领域物理量枚举数值时必须定义常量，严禁出现类似NUMBER\_ZERO的“魔鬼常量”。

**军规二：【明确方法的功能，一个方法仅完成一个功能。】**

说明：方法功能太多，会增加方法的复杂度和依赖关系，不利于程序阅读和将来的持续维护，无论是方法还是类设计都应符合单一职责原则。

**军规三：【方法参数不能超过5个】**

说明：参数太多影响代码阅读和使用，为减少参数，首先要考虑这些参数的合理性，保持方法功能单一、优化方法设计，如果参数确实无法减少，可以将多个参数封装成一个类（对象），同时考虑在新的类（对象）中增加相应的行为，以期更符合OOP。

**军规四：【方法调用尽量不要返回null，取而代之以抛出异常，或是返回特例对象（SPECIAL CASE object，SPECIAL CASE PATTERN）；对于以集合或数组类型作为返回值的方法，取而代之以空集合或0长度数组。】**

说明：返回null会增加不必要的空指针判断，遗漏判断也会导致严重的NullPointerException错误。

**军规五：【在进行数据库操作或IO操作时，必须确保资源在使用完毕后得到释放，并且必须确保释放操作在finally中进行。】**

说明：数据库操作、IO操作等需要关闭对象必须在try -catch-finally 的finally中close()，如果有多个IO对象需要关闭，需要分别对每个对象的close()方法进行try-catch,防止一个IO对象关闭失败其他IO对象都未关闭。推荐做法如下：

Connection jdbcConnection = null;

Statement stmt = null;

try

{

........

}

catch (SQLException e)

{

........

}

finally

{

if (stmt != null)

{

try

{

stmt.close();

}

catch (SQLException e)

{

logger.log(Level.WARNING, "异常说明", e);

}

}

if (jdbcConnection != null)

{

try

{

jdbcConnection.close();

}

catch (SQLException e)

{

logger.log(Level.WARNING, "异常说明", e);

}

}

}

**军规六：【异常捕获不要直接 catch(Exception ex) ，应该把异常细分处理。】**

说明：catch (Exception ex)的结果会把RuntimeException异常捕获，RuntimeException是运行期异常，是程序本身考虑不周而抛出的异常，是程序的BUG，如无效参数、数组越界、被零除等，程序必须确保不能抛出RuntimeException异常，不允许显示捕获RuntimeException异常就是为了方便测试中容易发现程序问题。

**军规七：【对于if „ else if „(后续可能有多个elseif …)这种类型的条件判断，最后必须包含一个else分支，避免出现分支遗漏造成错误；每个switch-case语句都必须保证有default，避免出现分支遗漏，造成错误。】**

**军规八：【覆写对象的equals()方法时必须同时覆写hashCode()方法。】**

说明：equals和hashCode方法是对象在hash容器内高效工作的基础，正确的覆写这两个方法才能保证在hash容器内查找对象的正确性，同时一个好的hashCode方法能大幅提升hash容器效率。

**军规九：【禁止循环中创建新线程，尽量使用线程池。】**

**军规十：【在进行精确计算时(例如:货币计算)避免使用float和double，浮点数计算都是不精确的，必须使用BigDecimal或将浮点数运算转换为整型运算。】**

说明：浮点运算在一个范围很广的值域上提供了很好的近似，但是它不能产生精确的结果。二进制浮点对于精度计算是非常不适合的，因为它不可能将0.1——或者10的其它任何次负幂精确表示为一个长度有限的二进制小数。

具体案例请参考：浮点数加法引发的问题：浮点数的二进制表示

<http://my.oschina.net/leejun2005/blog/156793>

**4、有关开发效率和协作的几点建议与心得体会**

今天看到某同学写给团队成员的一封邮件，发现比较通用，分享出来吧:

**1. 小提交:**

把大的任务拆分成多个独立小任务，每完成小任务确保无 Bug 后就可以提交合并到主分支甚至发布；频繁提交有利于自己把控项目进度、降低风险、同其他人协作和代码 Review ; 每天可以提交合并多次。每个小任务是 1-2 个小时可以完成的粒度，最大的一天完成。并行做多个任务的时候，优先做最短时间能够实现的任务。

**2. 命名规范:**

尽量避免无意义的字符做变量 比如 a, b, t 。可以逐步改善，可以参考:

<http://google-styleguide.googlecode.com/svn/trunk/javaguide.html>

**3. 避免过度设计:**

能够用简单方式实现的功能，不引入复杂的类，对象，避免不必要的 new 对象，避免引入不必要的泛型、线程。开发初期冗余大于抽象和依赖。避免自己重新实现比较通用的组件和函数。调研多种实现方式的时候，选用做简单的实现方式。尽量少写代码。

**4. Web 工程尽量避免在应用内部保存“状态”，这样可以适应频繁发布、重启无影响。**

**5. 善于用打日志的方式调试，在程序关键点打日志。尽量少用断点方式，日志方式可以批量调试一批功能，效率相对高。**

**6. 避免一屏显示不下的超大函数。**

**7. 添加必要、简洁的注释：**

循环中的 continue, break 尽量加上单行注释；尽量避免非函数结尾的 return，必要的时候加注释。类自动生成 toString() 方法，方便调试和打日志。

**8. 不把自己局限到做某个功能，每个人都是整个项目的 Owner ，尽量交叉 Review ，交叉开发。**

**9. 遇到问题及时和其他人沟通，避免浪费时间。**

**10. 从最终产品的目标审视自己细小的设计，熟悉自己负责部分的上下游代码。时刻关注最终产品(Web 界面和日志)，发现 Bug 和可以改善的地方。**

Refer：<http://blog.eood.cn/team_performance>

**5、推荐阅读：**

**（1）华为JAVA编程规范**

<http://wenku.baidu.com/view/096477d8ce2f0066f5332263.html>

**（2）Google Java Style**

<http://google-styleguide.googlecode.com/svn/trunk/javaguide.html>

<http://google-styleguide.googlecode.com/svn/trunk/>

**（3）Oracle/Sun Java语言编码规范(Java Code Conventions)**

<http://www.oracle.com/technetwork/java/codeconvtoc-136057.html>

<http://huihoo.org/code/java_code_conventions.html#71>

（4） java 八种不规范的代码写法

<http://blog.csdn.net/tk1905345/article/details/8015604>

原文地址：<http://www.cnblogs.com/Javame/p/3513670.html>