java代码规范

1. 程序内部的文档

（1）标识符规约

【规范】类名使用UpperCamelCase 风格，必须遵从驼峰形式

【规范】方法名、参数名、成员变量、局部变量都统一使用lowerCamelCase 风格，必须遵从驼峰形式。

【规范】抽象类命名使用 Abstract 或 Base 开头 ； 异常类命名使用 Exception 结尾 ； 测试类命名以它要测试的类的名称开始，以 Test 结尾。

【规范】不允许任何魔法值（ 即未经定义的常量 ） 直接出现在代码中。

【规范】各层命名规约：

Service / DAO 层方法命名规约

1 ） 获取单个对象的方法用 get 做前缀。

2 ） 获取多个对象的方法用 list 做前缀（习惯：getXXXList）。

3 ） 获取统计值的方法用 count 做前缀。

4 ） 插入的方法用 save（ 推荐 ） 或 insert 做前缀。

5 ） 删除的方法用 remove（ 推荐 ） 或 delete 做前缀。

6 ） 修改的方法用 update 做前缀(或modify)。

（2）注释规约

【规范】类、类属性、类方法的注释必须使用 Javadoc 规范，使用/\*\*内容\*/格式，不得使用 //xxx 方式。

【规范】所有的抽象方法(包括接口中的方法)必须要用 Javadoc 注释、除了返回值、参数、 异常说明外，还必须指出该方法做什么事情，实现什么功能。

说明:对子类的实现要求，或者调用注意事项，请一并说明。

【风格】方法内部单行注释，在被注释语句上方另起一行，使用//注释。方法内部多行注释使用/\* \*/注释，注意与代码对齐。

【规范】所有的枚举类型字段必须要有注释，说明每个数据项的用途。

【规范】代码修改的同时，注释也要进行相应的修改，尤其是参数、返回值、异常、核心逻辑 等的修改。

【规范】注释掉的代码尽量要配合说明，而不是简单的注释掉。

（3）程序的视觉组织

【规范】缩进只使用TAB键，不能使用空格键；方法体的开始、类的定义数据说明代码都要采用缩进方式

1. 数据说明

【规范】多个变量名在一个语句说明时，按字母顺序排列 这些变量。必要时，对于较复杂的数据结构进行注释说明其实现方法和特点。

1. 语句构造

【规范】当一个类有多个构造方法，或者多个同名方法，这些方法应该按顺序放置在一起，便于阅读。

【风格】类内方法定义顺序依次是：公有方法或保护方法 > 私有方法 > getter / setter方法。

【规范】不要为了节省空间而把多个语句写在同一行

【规范】在 if/else/for/while/do 语句中必须使用大括号，即使只有一行代码

【规范】除常用方法(如 getXxx/isXxx)等外，不要在条件判断中执行其它复杂的语句，将复杂逻辑判断的结果赋值给一个有意义的布尔变量名，以提高可读性。

1. 输入输出

【规范】保持输入格式简单

【规范】对以下场景中的方法进行参数校验:

1) 调用频次低的方法。

2) 执行时间开销很大的方法，参数校验时间几乎可以忽略不计，但如果因为参数错误导致中间执行回退，或者错误，那得不偿失。

3) 需要极高稳定性和可用性的方法。

4) 对外提供的开放接口，不管是RPC/API/HTTP接口。

5) 敏感权限入口。

【规范】明确提示交互式输入的请求，详细说明可用的选择或边界数值。（输入数据规范）

1. 效率
2. 程序运行时间

·写程序之前先简化算术的和逻辑的表达式。

·仔细研究嵌套的循环,以确定是否有语句可以从内层往外移。

·尽量避免使用多维数组。

·尽量避免使用指针和复杂的表。

·使用执行时间短的算术运算。不要混合使用不同的数据类型。

·尽量使用整数运算和布尔表达式。

1. 存储器效率

使用能保持功能域的结构化控制结构。

提高执行效率的技术。

提高存储器效率的关键是“简单”。

1. 输入输出的效率

·所有输入输出都应该有缓冲,以减少用于通信的额外开销。

·对二级存储器(如磁盘)应选用最简单的访问方法。

·二级存储器的输入输出应该以信息组为单位进行。