Java 代码开发规范

格式规范：

1、TAB 空格的数量。编辑器上的 TAB 空格数量统一取值为 4

2、换行， 每行 120 字符

3、if 语句的嵌套层数 3 层以内

4、匿名内部类 20 行以内 ，太长的匿名内部类影响代码可读性，建议重构为命名的

（普通）内部类。

5、文件长度 2000 行以内

6、方法长度 150 行以内

7、逻辑上相关序代码与其前后之程序代码间应以空白行加以分隔；在注释段与程序

段、以及不同程序段插入空行。提高可读性

8、方法（构造器）参数在 5 个以内 ， 太多的方法（构造器）参数影响代码可读性。

考虑用值对象代替这些参数或重新设计。

9、CC 度量值不大于 10

解释：CC(CyclomaticComplexity)圈复杂度指一个方法的独立路径的数量，可以

用一个方法内 if,while,do,for,catch,switch,case,?:语句与&&,||操作符的

总个数来度量。

10、NPath 度量值不大于 200

解释：NPath 度量值表示一个方法内可能的执行路径的条数。

11、布尔表达式中的布尔运算符(&&,||)的个数不超过 3 个

命名规范：

（开发人员如果遇到以下未列举的类型，请书面通知相关管理人员，由管理人员集中更新列

表内容，不得擅自启用未经确定的新变量前缀）

包名 必须全部用小写。

命名方式：业务领域名.子系统名.层名 如 com.iteach.dao.weibo

类名 以英文单词取名，首字母必须大写，多个英文单词以大写字母间隔，避免使用单词的缩写，除非它的缩写已经广为人知，如 HTTP。类名中不允许'_'、 '-'等符号。[A4]

属性 在类定义的开始，按照 public,protected,package,private 顺序放置。定义 local 变量尽量在那段代码的开始处，如方法的开始处。

如果是 if，for,while 段，尽量在左大括号"{"的下一行处定义要使用的 local 变量。

尽量用相同含义英文单词表示，不允许'_'、 '-'等符号，如：cust Name。第一个字母小写，中间单词的第一个字母大写。不要用_或&等符号作为第一个字母。 单字符的变量名一般只用于生命期非常短暂的变量。如：i,j,k,m,n 一般用于 int。如果变量是集合，则变量名应用复数，即以小写 s 结尾 。例如：

序号变量名称注　　释

1strFileName"文件名"字符串类型

2intFileCount"文件总数"整型

3strFames 多个"文件名"的集合

4gMemory 全局变量

常量名 均全部大写，单词间以'_'隔开。例如：

序号	常量名称	注　释
1	MAX_NUM	最大数
2	public static final String FUNCTION_LIST = ″function_list″;	…

方法 命名采用"动作+属性" 的方法。并且，动作以小写字母开始，属性以大写字母开始。常用的动作有：is、get、set、add、 update、del 等。

例如：getName、setName、isSysManager、saveXXX、mdfXXX、delXXX 等。

规则名称规则说明

新增数据 addXXX

修改数据 updateXXX

删除数据 deleteXXX

查询数据

findXXX

getXXX

findUserByName() 获取单个

getUserByName() 获取所有

备注：

遇到缩写如 XML 时，仅首字母大写，即 loadXmlDocument() 而不是

loadXMLDocument()

为了基于接口编程，不采用首字母为 I 或加上 IF 后缀的命名方式，如

IBookDao,BookDaoIF。

页面部件名建议命名为：btnOK、lblName 或 okBtn、nameLbl

其中 btn、lbl 缩写代表按钮(Button)、标签(Label)。

注释规范：

（在类、方法开始之前需要添加中文注释，类和方法的注释采用 Java 自动生成的注释格式。）

1、类注释：

```
/**
 * 类功能说明
 * 类创建者 创建日期
*/
```

2、函数注释

```
/**
 * 函数功能说明
 * 创建者名字 创建日期
 * 修改者名字 修改日期
 * 修改内容
 * @param 参数名称 参数类型 参数说明
 * @return 返回值类型 返回值说明
 */
```

3、程序段注释

如果做过修改需加上修改者和日期 //修改者 修改日期 说明

或者

/**

*修改者 修改日期

*说明

*/

4、变量或属性注释

//说明

5、失效代码注释

由/*...*/界定，标准的 C-Style 的注释，专用于注释已失效的代码

注：没有意义的注释语句删掉，不留空的注释语句

备注建议的注释：（非下划线标注的规范建议使用，不强制）

循环语句和判断语句前必须注释。

特殊变量声明时需要注释。

如果方法允许 Null 作为参数，或者允许返回值为 Null，必须在 JavaDoc 中说明。

注释中的第一个句子要以（英文）句号、问号或者感叹号结束。Javadoc 生成

工具会将注释中的第一个句子放在方法汇总表和索引中。

为了在 JavaDoc 和 IDE 中能快速链接跳转到相关联的类与方法，尽量多的使用

@see xxx.MyClass，@see xx.MyClass#find(String)。

如果注释中有超过一个段落，用<p>分隔。

示例代码以<pre></pre>包裹。

标识(java keyword, class/method/field/argument 名，Constants) 以<code></code>

包裹。

标识在第一次出现时以{@linkxxx.Myclass}注解以便 Java Doc 与 IDE 中可以链

接。

如果该注释是废话，连同标签删掉它，而不是自动生成一堆空的标签，如空的

@param name，空的@return。

推荐的注释内容：

对于 API 函数如果存在契约，必须写明它的前置条件(precondition)，后置条件

(postcondition)，及不变式(invariant)。

对于调用复杂的 API 尽量提供代码示例。

对于已知的 Bug 需要声明。

在本函数中抛出的 unchecked exception 尽量用@throws 说明。

代码质量不好但能正常运行，或者还没有实现的代码用//TODO: 或 //XXX:

声明存在错误隐患的代码用//FIXME:声明

异常处理：

重新抛出的异常必须保留原来的异常，即 throw new NewException("message", e);

而不能写成 throw new NewException("message")。

在所有异常被捕获且没有重新抛出的地方必须写日志。

如果属于正常异常的空异常处理块必须注释说明原因，否则不允许空的 catch 块。

框架尽量捕获低级异常并封装成高级异常重新抛出，隐藏低级异常的细节，方便

系统能够更好的跟踪运行情况。

如果一个层要抛出多个异常，那么所有自定义异常必须统一继承一个父类异常。

这样上层可以通过父类异常捕获。

编写细节建议规范：

1、 为了提高可读性，一般情况下，字符串的连接使用"+"，而不是 StringBuffer

中的方法。在考虑速度性能的时候才考使用 StringBuffer。

2、（不强制）没有特殊原因，不要定义 synchronized 的方法。而是在方法内实

际需要同步的代码段加入 synchronized 限定，如：

```
public void sharedMethod() {
    String display = null;
    synchronized( this ) {
```

```
            display = mySharedObject.getHelloWorld();
        }
        System.out.println( display );
    }
```

3、 捕捉例外的标准书写规则如下：

```
try{
        // some stuff
} catch ( FileNotFoundException fnfe ) {
        // some stuff
} finally {
        // some stuff
}
```

例外的变量名统一规定为例外类名中大写字母的组合。

4、 （不强制）对于一个方法或实例化类调用是否成功，不采用返回 boolean 值来判断，而采用捕捉例外的方法，如：

```
Order m_order = new Order();
try {
        m_order.init();
```

```
} catch ( OrderNotFoundException onfe ) {
      // some stuff
}
```