在软件开发的过程中总是强调注释的规范，但是没有一个具体的标准进行说明，通常都是在代码编写规范中简单的描述几句，不能作为一个代码注释检查的标准和依据，做什么都要有一个依据吗:),现在我特整理了一个《Java的注释规范》，内容来自网络、书籍和自己的实际积累。   
JAVA注释规范   
  
版本/状态 作者 版本日期   
1.0 ghc 2008-07-02   
  
一、背景   
1、当我们第一次接触某段代码，但又被要求在极短的时间内有效地分析这段代码，我们需要什么样的注释信息？   
2、怎么样避免我们的注释冗长而且凌乱不堪呢？   
3、在多人协同开发、维护的今天，我们需要怎么样的注释来保证高质、高交的进行开发和维护工作呢？   
二、意义   
程序中的注释是程序设计者与程序阅读者之间通信的重要手段。应用注释规范对于软件本身和软件开发人员而言尤为重要。并且在流行的敏捷开发思想中已 经提出了将注释转为代码的概念。好的注释规范可以尽可能的减少一个软件的维护成本 , 并且几乎没有任何一个软件，在其整个生命周期中，均由最初的开发人员来维护。好的注释规范可以改善软件的可读性，可以让开发人员尽快而彻底地理解新的代 码。好的注释规范可以最大限度的提高团队开发的合作效率。长期的规范性编码还可以让开发人员养成良好的编码习惯，甚至锻炼出更加严谨的思维能力。   
三、注释的原则   
1、 注释形式统一   
在整个应用程序中，使用具有一致的标点和结构的样式来构造注释。如果在其他项目组发现他们的注释规范与这份文档不同，按照他们的规范写代码，不要试图在既成的规范系统中引入新的规范。   
2、 注释的简洁   
内容要简单、明了、含义准确，防止注释的多义性，错误的注释不但无益反而有害。   
3、 注释的一致性   
在写代码之前或者边写代码边写注释，因为以后很可能没有时间来这样做。另外，如果有机会复查已编写的代码，在今天看来很明显的东西六周以后或许就 不明显了。通常描述性注释先于代码创建，解释性注释在开发过程中创建，提示性注释在代码完成之后创建。修改代码的同时修改相应的注释，以保证代码与注释的 同步。   
4、 注释的位置   
保证注释与其描述的代码相邻，即注释的就近原则。对代码的注释应放在其上方相邻或右方的位置，不可放在下方。避免在代码行的末尾添加注释；行尾注释使代码更难阅读。不过在批注变量声明时，行尾注释是合适的；在这种情况下，将所有行尾注释要对齐。   
5、 注释的数量   
注释必不可少，但也不应过多，在实际的代码规范中，要求注释占程序代码的比例达到20%左右。注释是对代码的“提示”，而不是文档，程序中的注释不可喧宾夺主，注释太多了会让人眼花缭乱，注释的花样要少。不要被动的为写注释而写注释。   
6、删除无用注释   
在代码交付或部署发布之前，必须删掉临时的或无关的注释，以避免在日后的维护工作中产生混乱。   
7、 复杂的注释   
如果需要用注释来解释复杂的代码，请检查此代码以确定是否应该重写它。尽一切可能不注释难以理解的代码，而应该重写它。尽管一般不应该为了使代码更简单便于使用而牺牲性能，但必须保持性能和可维护性之间的平衡。   
8、 多余的注释   
描述程序功能和程序各组成部分相互关系的高级注释是最有用的，而逐行解释程序如何工作的低级注释则不利于读、写和修改，是不必要的，也是难以维护的。避免每行代码都使用注释。如果代码本来就是清楚、一目了然的则不加注释，避免多余的或不适当的注释出现。   
9、必加的注释   
典型算法必须有注释。在代码不明晰或不可移植处必须有注释。在代码修改处加上修改标识的注释。在循环和逻辑分支组成的代码中添加注释。为了防止问题反复出现，对错误修复和解决方法的代码使用注释，尤其是在团队环境中。   
10、注释在编译代码时会被忽略，不编译到最后的可执行文件中，所以注释不   
会增加可执行文件的大小。   
四、JAVA注释技巧   
1、空行和空白字符也是一种特殊注释。利用缩进和空行，使代码与注释容易区   
别，并协调美观。   
2、当代码比较长，特别是有多重嵌套时，为了使层次清晰，应当在一些段落的   
结束处加注释（在闭合的右花括号后注释该闭合所对应的起点），注释不能   
写得很长，只要能表示是哪个控制语句控制范围的结束即可，这样便于阅读。   
3、将注释与注释分隔符用一个空格分开，在没有颜色提示的情况下查看注释时，   
这样做会使注释很明显且容易被找到。   
4、不允许给块注释的周围加上外框。这样看起来可能很漂亮，但是难于维护。   
5、每行注释（连同代码）不要超过120个字(1024×768)，最好不要超过80   
字(800×600) 。   
6、Java编辑器（IDE）注释快捷方式。Ctrl+/ 注释当前行,再按则取消注释。   
7、对于多行代码的注释，尽量不采用“/\*......\*/”，而采用多行“//”注释，   
这样虽然麻烦，但是在做屏蔽调试时不用查找配对的“/\*......\*/”。   
8、注释作为代码切换开关，用于临时测试屏蔽某些代码。   
例一：   
//\*/   
   codeSegement1;   
//\*/   
改动第一行就成了：   
/\*/   
   codeSegement1;   
//\*/   
例二：   
//----------------------第一段有效，第二段被注释   
//\*/   
   codeSegement1;   
/\*/   
   codeSegement2;   
//\*/   
只需删除第一行的/就可以变成：   
//----------------------第一段被注释，第二段有效   
/\*/   
   codeSegement1;   
/\*/   
   codeSegement2;   
//\*/   
五、JAVA注释方法及格式   
1、单行(single-line)--短注释：//……      
单独行注释：在代码中单起一行注释， 注释前最好有一行空行，并与其后的代码具有一样的缩进层级。如果单行无法完成，则应采用块注释。   
注释格式：/\* 注释内容 \*/   
  
行头注释：在代码行的开头进行注释。主要为了使该行代码失去意义。   
注释格式：// 注释内容   
     
行尾注释：尾端(trailing)--极短的注释，在代码行的行尾进行注释。一般与代码行后空8（至少4）个格，所有注释必须对齐。   
注释格式：代码 + 8（至少4）个空格 + // 注释内容   
2、块(block)--块注释：/\*……\*/   
注释若干行，通常用于提供文件、方法、数据结构等的意义与用途的说明，或者算法的描述。一般位于一个文件或者一个方法的前面，起到引导的作用，也可以根据需要放在合适的位置。这种域注释不会出现在HTML报告中。注释格式通常写成：   
/\*   
  \* 注释内容   
  \*/   
3、文档注释：/\*\*……\*/   
注释若干行，并写入javadoc文档。每个文档注释都会被置于注释定界符   
/\*\*......\*/之中，注释文档将用来生成HTML格式的代码报告，所以注释文   
档必须书写在类、域、构造函数、方法，以及字段(field)定义之前。注释文档由两部分组成——描述、块标记。注释文档的格式如下：   
/\*\*   
\* The doGet method of the servlet.   
\* This method is called when a form has its tag value method   
   \* equals to get.   
\* @param request   
\*  the request send by the client to the server   
\* @param response   
\*  the response send by the server to the client   
\* @throws ServletException   
\*  if an error occurred   
\* @throws IOException   
\*  if an error occurred   
\*/   
public void doGet (HttpServletRequest request, HttpServletResponse response)   
throws ServletException, IOException {   
    doPost(request, response);   
}   
前两行为描述，描述完毕后，由@符号起头为块标记注释。更多有关文档注   
释和javadoc的详细资料，参见javadoc的主页： http://java.sun.com/javadoc/index.html   
4、javadoc注释标签语法   
@author    对类的说明 标明开发该类模块的作者   
@version   对类的说明 标明该类模块的版本   
@see      对类、属性、方法的说明 参考转向，也就是相关主题   
@param    对方法的说明 对方法中某参数的说明   
@return    对方法的说明 对方法返回值的说明   
@exception  对方法的说明 对方法可能抛出的异常进行说明   
六、JAVA注释具体实现   
1、源文件注释   
源文件注释采用 /\*\* …… \*/，在每个源文件的头部要有必要的注释信息，包括：文件名；文件编号；版本号；作者；创建时间；文件描述包括本文件历史修改记录等。中文注释模版：   
/\*\*   
\* 文 件 名 :   
    \* CopyRright (c) 2008-xxxx:   
\* 文件编号：   
\* 创 建 人：   
\* 日    期：   
\* 修 改 人：   
\* 日   期：   
\* 描   述：   
\* 版 本 号：   
\*/   
  
2、类（模块）注释：   
类（模块）注释采用 /\*\* …… \*/，在每个类（模块）的头部要有必要的注释信息，包括：工程名；类（模块）编号；命名空间；类可以运行的JDK版本；版本号；作者；创建时间；类（模 块）功能描述（如功能、主要算法、内部各部分之间的关系、该类与其类的关系等，必要时还要有一些如特别的软硬件要求等说明）；主要函数或过程清单及本类 （模块）历史修改记录等。   
英文注释模版：   
/\*\*   
\* CopyRright (c)2008-xxxx:   <展望软件Forsoft >                            
    \* Project:                     <项目工程名 >                                            
\* Module ID:   <(模块)类编号，可以引用系统设计中的类编号>      
    \* Comments:  <对此类的描述，可以引用系统设计中的描述>                                             
\* JDK version used:      <JDK1.6>                                
\* Namespace:           <命名空间>                                
\* Author：        <作者中文名或拼音缩写>                   
\* Create Date：  <创建日期，格式:YYYY-MM-DD>   
\* Modified By：   <修改人中文名或拼音缩写>                                           
\* Modified Date:  <修改日期，格式:YYYY-MM-DD>                                      
    \* Why & What is modified  <修改原因描述>      
\* Version:                  <版本号>                         
\*/   
如果模块只进行部分少量代码的修改时，则每次修改须添加以下注释：   
//Rewriter   
//Rewrite Date：<修改日期:格式YYYY-MM-DD> Start1：   
/\* 原代码内容\*/   
//End1：   
将原代码内容注释掉，然后添加新代码使用以下注释：   
//Added by   
//Add date：<添加日期，格式：YYYY-MM-DD> Start2：   
//End2：   
如果模块输入输出参数或功能结构有较大修改，则每次修改必须添加以下   
注释：   
//Log ID：<Log编号,从1开始一次增加>   
//Depiction：<对此修改的描述>   
//Writer：修改者中文名   
//Rewrite Date：<模块修改日期，格式：YYYY-MM-DD>   
  
2、接口注释：   
接口注释采用 /\*\* …… \*/，在满足类注释的基础之上，接口注释应该包含描述接口的目的、它应如何被使用以及如何不被使用，块标记部分必须注明作者和版本。在接口注释清楚的前提下对应的实现类可以不加注释。   
  
3、构造函数注释：   
构造函数注释采用 /\*\* …… \*/，描述部分注明构造函数的作用，不一定有块标记部分。   
注释模版一：   
/\*\*   
\* 默认构造函数   
\*/   
注释模版二：   
/\*\*   
\* Description :       带参数构造函数,   
\*                       初始化模式名,名称和数据源类型   
\* @param schema：   模式名   
\* @param name：   名称   
\* @param type： 数据源类型   
\*/   
  
4、函数注释：   
函数注释采用 /\*\* ……\*/，在每个函数或者过程的前面要有必要的注释信息，包括：函数或过程名称；功能描述；输入、输出及返回值说明；调用关系及被调用关系说明等。函数注释里面可以不出现版本号（@version）。   
注释模版一：   
/\*\*   
  \* 函 数 名 :   
  \* 功能描述：   
\* 输入参数:     <按照参数定义顺序>   
\*             <@param后面空格后跟着参数的变量名字   
\*            （不是类型），空格后跟着对该参数的描述。>   
\*   
\* 返 回 值:  - 类型 <说明>   
\*            <返回为空（void）的构造函数或者函数，   
\*            @return可以省略; 如果返回值就是输入参数，必须 \*            用与输入参数的@param相同的描述信息; 必要的时\*            候注明特殊条件写的返回值。>   
\* 异    常：<按照异常名字的字母顺序>   
\* 创 建 人:   
\* 日    期:   
\* 修 改 人:   
\* 日    期:   
\*/   
注释模版二：   
/\*\*   
\* FunName:           getFirstSpell   
  \* Description :      获取汉字拼音首字母的字符串，   
\*                   被生成百家姓函数调用   
  \* @param：         str the String是包含汉字的字符串   
  \* @return String：汉字返回拼音首字母字符串；   
\*                  英文字母返回对应的大写字母；   
\*                 其他非简体汉字返回 '0'；   
\* @Author:       ghc   
\* @Create Date: 2008-07-02   
\*/   
  
5、方法注释：   
方法注释采用 /\*\* …… \*/，对于设置 (Set 方法 ) 与获取 (Get 方法 ) 成员的方法，在成员变量已有说明的情况下，可以不加注释；普通成员方法要求说明完成什么功能，参数含义是什么且返回值什么；另外方法的创建时间必须注释清 楚，为将来的维护和阅读提供宝贵线索。   
  
6、方法内部注释：   
控制结构，代码做了些什么以及为什么这样做，处理顺序等，特别是复杂的逻辑处理部分，要尽可能的给出详细的注释。   
      
7、 全局变量注释：   
要有较详细的注释，包括对其功能、取值范围、哪些函数或者过程存取以及存取时注意事项等的说明。   
  
8、局部（中间）变量注释：   
主要变量必须有注释，无特别意义的情况下可以不加注释。   
  
9、实参/参数注释：   
参数含义、及其它任何约束或前提条件。   
  
10、字段/属性注释： 字段描述，属性说明。   
  
11、常量：常量通常具有一定的实际意义，要定义相应说明。