**Coding Style Guidelines**

1. 编码风格
2. **格式规范**,

。缩进被括起来的代码。 //关于匿名内置类  
。打破长句子  
。使用空格和空行  
。不要用tab键

1. **命名规范**

。用有语义的、大家熟悉的名字

。避免过长的名字

。尽量保留名字中的元音字母

。对于像XML这样全部大写的缩写，在名字中不必全大写

。不要只通过大小写使得两个名字不同

。包命名格式：小写；以域名开头

。用有意义的名字

。不同版本要用新的包名//这样的话新版本包的命名是个麻烦事

1. **注释规范**  
   。注释和代码同步  
   。用准确的语言  
   。在写代码前写注释  
   。为所有的成员写文档注释  
   。为包写注释  
   。为一个应用或者一组包写注释  
   。用一种统一的格式做文档注释  
   。叙述一个类的功能时，尽量不用主语，而谓语采用第三人称单数  
   。写简述，简述放在第一句  
   。指代当前类是用this而不用the  
   。为每个类、接口、成员变量、方法写注释  
   。为方法做完整的描述  
   。在必要时包含例子  
   。为方法的前提（例如参数值的限制）、结果（例如类的状态改变）写注释

。为错误和缺陷做注释  
。为同步语义做注释  
。描述代码为什么这样做，而不是做什么。代码本身能说明自己在做什么

。尽量避免把行末注释放在行末，因为修改这行代码时可能导致一行很长而把注释推到不容易看到的地方，把行末注释放在要解释的代码之前，而且另起一行  
。对变量的注释放在行末  
。确定一组关键字来说明没有解决的事情，尤其用在代码还没有完成的时候，一般还包括一个时间，例如：UNRESOLEVED: EBW, 2005-7-20  
。对多层嵌套的控制结构，对每个右大括弧做注释，例如 //end if 之类的，如果采用每个大括弧单独成行而且相应括号对齐的话，这种注释就可以取消了  
。在switch语句中，如果一个case之后没有break语句，则一定要加上一句注释://fall though   
。如果一个控制结构之后的语句块是空的，则请注明 //Empty!。例如  
for (int i=0; i<n; i++);  
//Empty!

**二、编码准则**

**1， 坚持原有代码的规范**  
当修改现有代码时，修改应该仍然遵守原来代码的规范。不要试图只修改代码规范来重写原有代码，否则很容易产生错误。

**2， 坚持最少astonishment原则**  
应该避免程序产生让用户感到奇怪的行为。应该遵循下面的准则：  
简单：写简单的类和简单的方法。决定做到何种程度就可以满足用户的需求。  
明确：保证每个类、接口、方法、变量和对象都有明确的目的。  
完整：创建完整的文档；对所有的特征和工作做文档。  
一致：风格要一致。相似的东西应该看起来相似，行为相似。尽量创建和应用标准。  
鲁棒性：提供对错误和异常的可预见的处理。

**3， 立即应用这些规则**  
把这些规则应用到你要写的任何代码中。包括原型开发的代码，等等。

**4， 对任何偏离规范的变化做文档**  
没有标准是完美的。有时候不得不打破它。首先要理解你要打破的规则的存在理由和后果，其次要为所作的变化做文档。

**5， 先实现后优化**

**6， 记住“分而治之”的原则，将大问题分解为若干小问题逐个击破**

**7， 记住代码可重用，不要写重复代码**

**8， 避免使用多继承**

**9， 避免使用全局变量**

**10， 更喜欢编译时错误，而不是运行时错误**