**编码规范文档**

**一、命名风格**

使用匈牙利表示法，命名时应始终采用完整的英文描述符，一般应采用小写字母。编码规范文档 

1）类的命名

类的名字必须由大写字母开头而其他字母都小写的单词组成，其中包含的所有单词都应紧靠在一起，而且大写中间单词的首字母。 如public class HighFrequencyWords extends JFrame {}

2）类的变量命名

变量的名字必须由一个小写字母开头，后面的单词用大写字母开头。如inputButton

3）静态变量的命名

变量的名字应该都大写，并且指出完整定义。

如private static JButton closeButton;

4）参数的命名

参数的名字必须和变量的命名规范一致。

**二、代码格式**

**1、排版**

1）较长的语句、表达式要等分成多行书写。

2）长表达式要在低优先级操作符处划分新行，操作符放在新行之首。

3）循环、判断等语句中若有较长的表达式或语句，则要进行适当的划分。

4）若函数中的参数较长，则要进行适当的划分。

5）不允许把多个短语句写在一行中，即一行只写一句语句。

**2、缩进和间隔**

缩进应该是每行4个空格。不要在源文件中保存Tab字符，在使用不同的源代码管理工具时Tab字符将因为用户设置的不同而扩展为不同的宽度。

1）关键词和操作符之间加适当的空格。

2）相对独立的程序块与块之间加空行。

3）函数或过程的开始、结构的定义及循环、判断等语句中的代码都要采用缩进风格。‘{'和‘}'应各独占一行并位于同一列，同时与引用它们的语句左对齐。

4）划分出的新行要进行适应的缩进，使排版整齐，语句可读。

如：

 public HighFrequencyWords (final Map<String, Integer> maps)

{

Font font =new Font("黑体", Font.PLAIN, 20);

statistics=new JFrame("输出词频前N个单词");

statistics.setSize(500, 400);

input=new JLabel();

}

**三、注释规约**

1）编写代码边注释，修改代码同时修改相应的注释，以保证注释和代码的一致性。

2）在必要的地方注释，注释量要适中。注释的内容要清楚、明了、含义准

确，防止注释二义性。保持注释与其描述的代码相邻，即注释的就近原则。

3）对代码的注释应放在其上方相邻位置，不可放在下面。对数据结构的注释应放在其上方相邻位置，不可放在下面；对结构中的每个域的注释应放在此域的右方；统一结构中不同域的注释要对齐。

如：

/\*\*按字典顺序输出到文件\*/

public class PrintToFile

{

Map<String,Integer> Map = new LinkedHashMap<String, Integer>();

/\*\*按字典顺序排序\*/

void Sort(Map<String, Integer> map)

{

}

}

4）变量、常量的注释应该放在其上方相邻位置或右方。

如：ArrayList<String> lists = new ArrayList<String>(); // 用来存储过滤后的单词列表

5）全局变量要有较详细的注释，包括对其功能、功能范围、那些函数或过程存取它以及存取时注意事项等的说明。

6）在源文件的头部要有必要的注释信息，包括：作者，生成日期，模块功能描述等。

如：

/\*\*

\* @author Lin Zhou,YanXiao Zhao

\* @data 2019.3.31

\*/

**四、编码时应注意**

1）代码编译

  编写代码时要注意随时保存，并定期备份，防止由于断电、磁盘损坏等

原因造成代码丢失。

2）在进行文件的读取操作时，除了可以正常的捕捉异常外，必须在文件读取完毕后，关闭整个流，特别是在系统发生异常时也要保证相关流操作能够正常关闭。

 3）在代码编写的过程中，尽量减少方法中临时变量以及类中全局静态变量

的使用，这些变量都将增加系统的额外开销。

4）对类设计：

不同的功能用不同的类来分别实现，使类尽可能短小精悍，以便于之后的修改。

5）对于接口和抽象类

* 如果要创建不带任何方法定义和成员变量的基类，那么就应该选择接口而不是抽象类。
* 如果知道某个类应该是基类，那么第一个选择的应该是让它成为一个接口，只有在必须要有方法定义和成员变量的时候，才应该选择抽象类。因为抽象类中允许存在一个或多个被具体实现的方法，只要方法没有被全部实现该类就仍是抽象类。