**Slarker小组代码规范**

1. 说明
   1. 为什么要有代码规范

代码规范对于整个程序日后的维护扩展都非常重要，因为：

* 一个软件的生命周期中，需要大量的维护
* 软件的发展过程中，难免会有人员的更替，维护人员不一定是当初的开发人员
* 可以增加代码的可读性，易维护性
* 在项目范围内统一代码风格

1. 文件名
   1. 文件后缀

Java源文件的文件后缀为.java，Java字节码文件的文件后缀为.class

1. 文件组织

尽量避免超过2000行的文件，因为超过2000行的文件难以阅读。

* 1. java源文件
* 每个源文件包含一个单一的公共类或者接口
* 开头注释

注释应包含：类名，版本，日期，功能，版权声明

* 包和引入语句

package java.awt;

import java.awt.peer;

* 类和接口声明

1. 缩进排版
   1. 缩进排版单位

4个空格作为缩进排版的一个单位。

* 1. 行长度

行长度应避免超过80个字符。

* 1. 换行

当一个表达式无法容纳在一行内时：

* 在一个逗号后面断开
* 在一个操作符前面断开
* 新的一行应与上一行同一级表达式的开头处对齐

1. 注释
   1. 实现注释

实现注释用来注释代码或者实现细节，注释为了帮助程序员理解，但应该避免提供代码已经清晰表达的信息，同时应该避免频繁注释。因为多次的更新会使更新注释比更新代码更加困难。

* 块注释

第一行/\*开头，后面每一行\*开头，最后一行\*/结束

/\*

\* 这是块注释

\*

\*

\*/

* 单行注释

短注释可以显示一行内，并和其后代码具有意义的缩进。如果一个注释不能一行写完，则使用块注释。单行注释前有一个空行。

If（condition）{

/\* 这是一个行注释 \*/

……

}

1. 声明
   1. 每行声明变量

每一行进行一个声明，这样便于写注释。

Int level; // level

Int size; // size of table

* 1. 初始化

尽量在声明局部变量的同时进行初始化。

* 1. 布局
* 只在代码块的开始处声明变量，不要在首次使用该变量之前才声明，这样会妨碍代码在该作用域内的可移植性。
* 避免声明的局部变量覆盖上一级的声明变量，不要在内部代码块中声明相同的变量名。

1. 语句

7.1简单语句

每一行最多包含一条语句

7.2 复合语句

If-else，for等后面应该使用大括号，避免由此带来的bug

7.3 返回语句

一个带返回值的return语句不使用小括号，除非它们以某种方式使得返回值更加显而易见。

7.4 if-else语句应该具有如下格式：

if (condition) {

statements;

}

if (condition) {

statements;

} else {

statements;

}

if (condition) {

statements;

} else if (condition) {

statements;

} else if (condition) {

statements;

}

7.5 while语句

while (condition) {

statements;

}

7.6 do-while语句

do {

statements;

} while (condition);

7.7 try-catch语句

try {

statements;

} catch (ExceptionClass e) {

statements;

} finally {

statements;

}

1. 空白
   1. 空行

两个方法之间总是使用一个空行

* 1. 空格
* 空白应该位于参数列表中逗号的后面
* 所有二元运算符，除了”.”，应该使用空格将之与操作数分开，一元操作符和操作数之间不应该加空格。
* for语句中的表达式应该被空格分开

for（expr1; expr2; expr3）

* 强制转换类型后面应该跟一个空格

1. 命名规范

命名规范使程序更易读，从而更易于理解。它们也可以提供一些有关标识符功能的信息，以助于理解代码。

* 1. 包

一个唯一包名的前缀总是全部小写的ASCⅡ字母并且是一个顶级域名，通常是com，gov，edu，mil，net，org。包名的后续部分根据不同机构各自内部的命名规范而不尽相同。这类命名规范可能以特定目录名的组成来区分部门(department)，项目(project)，机器(machine)，或注册名(login names)。

例如： com.apple.quicktime.v2

* 1. 类和接口

类名和接口名是一个名词，采用大小写混合方式，每个单词的首字母大写。尽量使你的类名简洁而富于描述。使用完整单词，避免缩写词(除非该缩写词被广泛的使用，如URL，HTML)。

* 1. 方法

方法名是一个动词，采用大小写混合方式，第一个单词的首字母小写，其后单词的首字母大写。

* 1. 变量

除了变量名外，所有实例，包括类，类常量，均采用大小写混合的方式，第一个单词的首字母小写，其后单词的首字母大写。变量名不应以下划线或美元符号开头，尽管这在语法上是允许的。

* 1. 实例变量

实例变量名应简短且富于描述。变量名的选用应该易于记忆，即，能够指出其用途。

* 1. 临时变量

临时变量通常被取名为i，j，k，m和n，它们一般用于整形；c，d，e，它们一般用于字符型。

* 1. 常量

大小写规则和变量名相似，除了前面需要一个下划线隔开。尽量避免ANSI常量，容易引起错误。

1. 编程惯例
   1. 常量

位于for循环中作为计数器值的数字常量，除了-1，0和1之外，不应被直接写入代码。

* 1. 圆括号

一般而言，在含有多种运算符的表达式中使用括号来避免运算符优先级问题，是个好方法。即便运算符的优先级对你而言可能很清楚，但对其他人未必如此。你不能假设别的程序员和你一样清楚运算符的优先级。