# Programming Style 备忘

阅读了关于程序风格的经典书籍《The elements of programming style》之后觉得有必要记录一下阅读的时候有意义的一些重点。整本书

## 写的清楚一点，不要显得太聪明。

程序员们总是喜欢使用一些比较巧妙的方法来实现功能，但是实际情况中不能指望别的程序员都能够明白如此聪明的做法。为了程序的可读性，我们有时候需要把程序写的“傻瓜”一点。作者把这条规则写在了第一章里，我相信这是一条对于绝大多数的程序员都非常重要的准则。

* **Example：**

**public** **void** init(**int**[][] array) {

**for** (**int** i = 0; i < array.length; ++i)

**for** (**int** j = 0; j < array[i].length; ++j)

array[i][j] = (i / j) \* (j / i);

}

在这个例子里面，程序员巧妙的利用只有当i==j的时候，(i / j) \* (j / i)才为1的特性，创建了一个Indentity Matrix(单位矩阵)，但是这样的程序可读性不高。修改后的程序可以这样写：

**public** **void** init(**int**[][] array) {

**for** (**int** i = 0; i < array.length; ++i) {

**for** (**int** j = 0; j < array[i].length; ++j)

array[i][j] = 0;

array[i][i] = 1;//这里好像需要加检查

}

}

## 简单直接的把逻辑写出来

* **Example:**

**public** **int** min(**int** x, **int** y, **int** z) {

**if** (y < z)

**if** (y < x) **return** y;

**else** **return** x;

**if** (x < z) **return** x;

**return** z;

}

这个例子就是求三个数的最小数，使用了不少的判断，看起来就比较啰嗦，但是逻辑不清楚。修改之后可以这样写。

**public** **int** min(**int** x, **int** y, **int** z) {

**int** small = x;

**if** (y < small) small = y;

**if** (z < small) small = z;

**return** small;

}

## 尽量使用类库函数

Example:

**if** (a < 0) a = -a;

在这个例子里面就是将a取绝对值，本来这个例子用fortran来写好像更加合适一点，这里用java写可能不是特别明显，主要意思就是可以用库函数解决的问题就用库函数解决，不要自己写，需要养成习惯。修改之后：

a = Math.*abs*(a);

## 减少临时变量的使用

这个规则好像比较显而易见，很多时候不可避免的需要使用临时变量，怎么减少临时变量的使用才是我们需要动脑子的地方。

## 不要为了提高一点点效率，降低程序可读性

这里我稍稍修改了书中的规则，添加了一点点作为形容词。是因为我觉得在使用某些高性能算法的时候，我们不可避免地需要写更复杂的程序，降低可读性变得不可避免，但是如果只是一点点的效率提高（没有提高算法复杂度），就没有必要写一些降低可读性的程序。

## 让机器来做“脏活”

这里的脏活是指我们很多时候不需要做一些应该机器完成的事情，例如不少位操作其实应该交给机器来处理，例如没有必要不要把数字表示成2进制或16进制。