Java编码规范

**源文件结构**

一个源文件包含(按顺序地)：

1. 许可证或版权信息(如有需要)
2. package语句
3. import语句
4. 一个顶级类(**只有一个**)

以上每个部分之间用一个空行隔开。

**许可证或版权信息**

如果一个文件包含许可证或版权信息，那么它应当被放在文件最前面。

**package语句**

package语句不换行，列限制(4.4节)并不适用于package语句。(即package语句写在一行里)

**import语句**

**import不要使用通配符**

即，不要出现类似这样的import语句：import java.util.\*;

**不要换行**

import语句不换行，列限制(4.4节)并不适用于import语句。(每个import语句独立成行)

**顺序和间距**

import语句可分为以下几组，按照这个顺序，每组由一个空行分隔：

所有的静态导入独立成组

com.google imports(仅当这个源文件是在com.google包下)

第三方的包。每个顶级包为一组，字典序。例如：android, com, junit, org, sun

java imports

javax imports

组内不空行，按字典序排列。

### 类声明

#### 只有一个顶级类声明

每个顶级类都在一个与它同名的源文件中(当然，还包含.java后缀)。

例外：package-info.java，该文件中可没有package-info类。

#### 类成员顺序

类的成员顺序对易学性有很大的影响，但这也不存在唯一的通用法则。不同的类对成员的排序可能是不同的。 最重要的一点，每个类应该以某种逻辑去排序它的成员，维护者应该要能解释这种排序逻辑。比如， 新的方法不能总是习惯性地添加到类的结尾，因为这样就是按时间顺序而非某种逻辑来排序的。

##### 重载：永不分离

当一个类有多个构造函数，或是多个同名方法，这些函数/方法应该按顺序出现在一起，中间不要放进其它函数/方法。

**命名约定**

**所有标识符都通用的规则**

标识符只能使用ASCII字母和数字，因此每个有效的标识符名称都能匹配正则表达式\w+。

在Google其它编程语言风格中使用的特殊前缀或后缀，如name\_, mName, s\_name和kName，在Java编程风格中都不再使用。

**标识符类型的规则**

**包名**

包名全部小写，连续的单词只是简单地连接起来，不使用下划线。

**类名**

类名都以UpperCamelCase风格编写。

类名通常是名词或名词短语，接口名称有时可能是形容词或形容词短语。现在还没有特定的规则或行之有效的约定来命名注解类型。

测试类的命名以它要测试的类的名称开始，以Test结束。例如，HashTest或HashIntegrationTest。

**方法名**

方法名都以lowerCamelCase风格编写。

方法名通常是动词或动词短语。

下划线可能出现在JUnit测试方法名称中用以分隔名称的逻辑组件。一个典型的模式是：test<MethodUnderTest>\_<state>，例如testPop\_emptyStack。 并不存在唯一正确的方式来命名测试方法。

**常量名**

常量名命名模式为CONSTANT\_CASE，全部字母大写，用下划线分隔单词。那，到底什么算是一个常量？

每个常量都是一个静态final字段，但不是所有静态final字段都是常量。在决定一个字段是否是一个常量时， 考虑它是否真的感觉像是一个常量。例如，如果任何一个该实例的观测状态是可变的，则它几乎肯定不会是一个常量。 只是永远不打算改变对象一般是不够的，它要真的一直不变才能将它示为常量。

// Constants

static final int NUMBER = 5;

static final ImmutableList<String> NAMES = ImmutableList.of("Ed", "Ann");

static final Joiner COMMA\_JOINER = Joiner.on(','); // because Joiner is immutable

static final SomeMutableType[] EMPTY\_ARRAY = {};

enum SomeEnum { ENUM\_CONSTANT }

// Not constants

static String nonFinal = "non-final";

final String nonStatic = "non-static";

static final Set<String> mutableCollection = new HashSet<String>();

static final ImmutableSet<SomeMutableType> mutableElements = ImmutableSet.of(mutable);

static final Logger logger = Logger.getLogger(MyClass.getName());

static final String[] nonEmptyArray = {"these", "can", "change"};

这些名字通常是名词或名词短语。

**非常量字段名**

非常量字段名以lowerCamelCase风格编写。

这些名字通常是名词或名词短语。

**参数名**

参数名以lowerCamelCase风格编写。

参数应该避免用单个字符命名。

**局部变量名**

局部变量名以lowerCamelCase风格编写，比起其它类型的名称，局部变量名可以有更为宽松的缩写。

虽然缩写更宽松，但还是要避免用单字符进行命名，除了临时变量和循环变量。

即使局部变量是final和不可改变的，也不应该把它示为常量，自然也不能用常量的规则去命名它。

**5.2.8 类型变量名**

类型变量可用以下两种风格之一进行命名：

* 单个的大写字母，后面可以跟一个数字(如：E, T, X, T2)。
* 以类命名方式(5.2.2节)，后面加个大写的T(如：RequestT, FooBarT)。

**编程实践**

### @Override：能用则用

只要是合法的，就把@Override注解给用上。

### 捕获的异常：不能忽视

除了下面的例子，对捕获的异常不做响应是极少正确的。(典型的响应方式是打印日志，或者如果它被认为是不可能的，则把它当作一个AssertionError重新抛出。)

### 静态成员：使用类进行调用

使用类名调用静态的类成员，而不是具体某个对象或表达式

### Finalizers: 禁用

极少会去重载Object.finalize。