Java 编码规范

1. 介绍/说明

1.1 声明

本文档内容描述 JAVA 编码规范。

1.2 为什么要有编码规范

编码规范对于开发人员来说是非常重要的，有以下几个原因：

- 一个软件的生命周期中，80%的花费在于维护
- 几乎没有任何一个软件，在其整个生命周期中，均由最初的开发人员来维护
- 编码规范可以改善软件的可读性，可以让程序员尽快而彻底地理解新的代码
- 如果你将源码作为产品发布，就需要确任它是否被很好的打包并且清晰无误，一如你已构建的其它任何产品

2. 目标

- 为来自不同的项目组或个人提供标准的代码格式。
- 增加易读性。

3. 命名规定

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意图的信息，有助于开发人员理解代码。

3.1 包的命名

包的命名应该都是小写字母，单词之间用“.”分开。所有的 JAVA 文件必须建立在 cn.com.biceng 包下。例如：

Package cn.com.biceng.dao;

package cn.com.biceng.web.struts.action;

3.2 类的命名

类的命名应该都是名词，第一个字母都要大写，其他每个单词的第一个字母都要大写。要用完整的单词，除非是被公认的单词缩写。例如：

class Container
class ShippingLine

3.3 接口的命名

接口的命名应该都是名词或形容词，以大写的“I”开头，第一个字母都要大写，其他每个单词第一个字母都要大写。要用完整的单词，除非是被公认的单词缩写。例如：

interface ICommonService
interface IFlowService

3.4 方法的命名

方法的命名应该都用动词或是惯用短语描述，第一个字母都要小写，其他每个单词第一个字母都要大写。例如：

run()
loginEx ()
getContainerId()

3.5 变量的命名

所有非静态变量名的第一个字母都要小写，其他每个单词的第一个字母都要大写。命名应尽量简单并且要有意义。变量名的选用应该易于记忆，能够指出其用途。尽量避免单个字符的变量名，除非是一次性的临时变量。临时变量通常被取名为 i，j，k，m 和 n，它们一般用于整型；c，d，e，它们一般用于字符型。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。下面是一些正确的变量命名例子：

int numOfContainers
String containerId;
Date today;

3.6 常量命名

对于静态的 final 变量，在命名的时候每一个单词都要大写，单词之间用“_”分开。例如：

final static MIN_WIDTH = 4;
final static DEFAULT_CONTAINER_SIZE = 20;

3.7 文件的命名

java 源程序文件以.java 结尾，编译后的文件以.class 结尾。例如：

Container.java
Container.class

3.8 推荐的命名

3.8.1 类名推荐

当要区别接口和实现类的时候，可以在类的后面加上“Impl”。例如：
interface ICommonService
class CommonServiceImpl
class ICtainerImpl
class ContainerImpl

3.8.2 Exception 类名推荐

Exception 类最好能用“Exception”做为类命名的结尾。例如：
InvalidArgumentException
RemoteCallException

3.8.3 抽象类名推荐

抽象类最好能用“Abstract”做为类命名的开头。例如：
AbstractBeanDefinition
AbstractBeanFactory

3.8.4 Test 类名推荐

Test 类最好能用“Test”做为类命名的结尾。例如：
ContainerTest
而测试方法最好以“test”开头：
testLogin()

3.8.5 工厂类方法推荐

工厂方法最好能把该方法做要创建的对象类型描述出来。例如：
public Container createContainer();
public Location newLocation();

4. Java 文件组织

一个文件由被空行分割而成的段落以及标识每个段落的可选注释共同组成。超过 2000 行的程序难以阅读，所以一个 java 程序文件中的代码行数不能超过 2000 行，除非有特殊原因。

每个 Java 源文件都包含一个单一的公共类或接口。若私有类和接口与一个公共类相关联，可以将它们和公共类放入同一个源文件。公共类必须是这个文件中的第一个类或接口。

Java 源文件还遵循以下规则，这个规则规定了 java 程序段落的顺序：

- 开头注释
- 包和引入语句
- 类和接口声明

以下摘自林锐《高质量程序设计指南--c++/c 语言》一书第七章内容，供参考。

比较著名的命名规则当推 Microsoft 公司的“匈牙利”法，该命名规则的主要思想是“在变量和函数名中加入前缀以增进人们对程序的理解”。例如所有的字符变量均以 ch 为前缀，若是指针变量则追加前缀 p。如果一个变量由 ppch 开头，则表明它是指向字符指针的指针。

“匈牙利”法最大的缺点是烦琐，例如

```
int    i,  j,  k;
float  x,  y,  z;
```

倘若采用“匈牙利”命名规则，则应当写成

```
int    iI,  iJ,  ik;  // 前缀 i 表示 int 类型
float  fX,  fY,  fZ;  // 前缀 f 表示 float 类型
```

如此烦琐的程序会让绝大多数程序员无法忍受。

据考察，没有一种命名规则可以让所有的程序员赞同，程序设计教科书一般都不指定命名规则。命名规则对软件产品而言并不是“成败悠关”的事，我们不要化太多精力试图发明世界上最好的命名规则，而应当制定一种令大多数项目成员满意的命名规则，并在项目中贯彻实施。

本节论述的共性规则是被大多数程序员采纳的，我们应当在遵循这些共性规则的前提下，再扩充特定的规则（见 7.2 节）。

- **【规则 7-1-1】**标识符应当直观且可以拼读，可望文知意，不必进行“解码”。

标识符最好采用英文单词或其组合，便于记忆和阅读。切忌使用汉语拼音来命名。程序中的英文单词一般不会太复杂，用词应当准确。例如不要把 CurrentValue 写成 NowValue。

- **【规则 7-1-2】**标识符的长度应当符合“min-length && max-information”原则。

几十年前老 ANSI C 规定名字不准超过 6 个字符，现今的 C++/C 不再有此限制。一般来说，长名字能更好地表达含义，所以函数名、变量名、类名长达十几个字符不足为怪。那么

名字是否越长约好？不见得！例如变量名 maxval 就比 maxValueUntilOverflow 好用。单字符的名字也是有用的，常见的如 i, j, k, m, n, x, y, z 等，它们通常可用作函数内的局部变量。

- **【规则 7-1-3】**命名规则尽量与所采用的操作系统或开发工具的风格保持一致。

  例如 Windows 应用程序的标识符通常采用“大小写”混排的方式，如 AddChild。而 Unix 应用程序的标识符通常采用“小写加下划线”的方式，如 add_child。别把这两类风格混在一起用。

- **【规则 7-1-4】**程序中不要出现仅靠大小写区分的相似的标识符。

  例如：

```
int   x,   X;      // 变量 x 与 X 容易混淆
void foo(int x);// 函数 foo 与 F00 容易混淆
void F00(float x);
```

- **【规则 7-1-5】**程序中不要出现标识符完全相同的局部变量和全局变量，尽管两者的作用域不同而不会发生语法错误，但会使人误解。

- **【规则 7-1-6】**变量的名字应当使用“名词”或者“形容词＋名词”。

  例如：

```
float   value;
float   oldValue;
float   newValue;
```

- **【规则 7-1-7】**全局函数的名字应当使用“动词”或者“动词＋名词”（动宾词组）。类的成员函数应当只使用“动词”，被省略掉的名词就是对象本身。

  例如：

```
DrawBox();          // 全局函数
box->Draw();        // 类的成员函数
```

- **【规则 7-1-8】**用正确的反义词组命名具有互斥意义的变量或相反动作的函数等。

  例如：

```
int   minValue;
int   maxValue;

int   SetValue(…);
int   GetValue(…);
```

✧ **【建议 7-1-1】**尽量避免名字中出现数字编号，如 Value1,Value2 等，除非逻辑上的确需要编号。这是为了防止程序员偷懒，不肯为命名动脑筋而导致产生无意义的名字（因为用数字编号最省事）。我长这么大，从来都没有见到过有人把子女的名字叫做张三或李四。

✧

✧ **简单的 Windows 应用程序命名规则**

作者对“匈牙利”命名规则做了合理的简化，下述的命名规则简单易用，比较适合于

Windows 应用软件的开发。

- **【规则 7-2-1】**类名和函数名用大写字母开头的单词组合而成。

  例如：

```
class Node;                  // 类名
class LeafNode;              // 类名
void  Draw(void);            // 函数名
void  SetValue(int value);   // 函数名
```

- **【规则 7-2-2】**变量和参数用小写字母开头的单词组合而成。

  例如：

```
BOOL flag;
int  drawMode;
```

- **【规则 7-2-3】**常量全用大写的字母，用下划线分割单词。

  例如：

```
const int MAX = 100;
const int MAX_LENGTH = 100;
```

- **【规则 7-2-4】**静态变量加前缀 s_（表示 static）。

  例如：

```
void Init(…)
{
    static int s_initValue;     // 静态变量
    …
}
```

- **【规则 7-2-5】**如果不得已需要全局变量，则使全局变量加前缀 g_（表示 global）。

  例如：

```
int g_howManyPeople; // 全局变量
int g_howMuchMoney; // 全局变量
```

- **【规则 7-2-6】**类的数据成员加前缀 m_（表示 member），这样可以避免数据成员与成员函数的参数同名。

  例如：

```
void Object::SetValue(int width, int height)
{
    m_width = width;
    m_height = height;
}
```

- **【规则 7-2-7】**为了防止某一软件库中的一些标识符和其它软件库中的冲突，可以统一为各种标识符加上能反映软件性质的前缀。例如三维图形标准 OpenGL 的所有库函数均以 gl 开头，所有常量（或宏定义）均以 GL 开头。

以上摘自《高质量程序设计指南》一书。

5. JAVA 文件声明顺序

类或接口应该按以下顺序声明：

- 包的定义
- impot 类（输入包的顺序、避免使用*）
  - 输入包应该按照 java.*.*，javax.*.*，org.*.* ，com.*.*的顺序 import
  - 在 import 的时候不应该使用* (例如: java.util.*)
- 类或接口的定义
- 静态变量定义，按 public，protected，private 顺序
- 实例变量定义，按 public，protected，private 顺序
- 构造方法
- 方法定义顺序按照 public 方法(类自己的方法)，实现接口的方法，重载的 public 方法，受保护方法，包作用域方法和私有方法。

建议：类中每个方法的代码行数不要超过 100 行。

- 内部类的定义

6. JAVA 文件格式缩进定义

6.1 缩进尺寸

用 4 个空格做为缩进尺寸。不建议用 TAB 代替，因为不同的 Text Edit 工具对 TAB 的设置是不同的。

6.2 行的尺寸

每行不要超过 80 个字符。

6.3 行的格式定义

当一行表达式不能在一行内显示，请按下列顺序要求拆行：

- 在“(”或“=”符号后拆行
- 在“,”拆行
- 在一个操作符后拆行
- 把并发的拆行放到同一级别上的缩进
- 如果在拆行中再次拆分的时候遇到“(”，应该新拆出来的行放在更远的一个缩进级别上

例如：methodWithLongName(
expression1, expression2, expression3,
expression4, expression5);

```
var =
    method1(
        expression1, expression2,
        method2(
            expression3, expression4));
```

7. 注释

Java 有两种注释方法。“/* This is a comment */”或 “// This is a comment”
第一种应该被用到写 JavaDoc 上，并且都用“/**”开头。
第二种适合于在做部分代码的注释，但只适合做非常短内容的注释。

8. 声明

8.1 变量声明

推荐每行声明一个变量，并加注释。例如：

```
int count;                  // number of containers
int size;                   // size of table
int count, size;            // AVOID THIS!
```

数组声明应该采用前缀方式。例如：

```
int[] table;
String[] args;
```

8.2 类或接口声明

- “{”和声明语句在同一行。
- 如果不能在同一行显示，就将“extends”或“implements”进行拆行，并放在两个缩进级别后。
- “}”符号应该独自占一行。

例如：

```
public class Manager extends Employee {
    ...
}
public class ChiefExecutiveOfficer
        extends Manager
        implements Person {
```

```
    ...
}
```

8.3 方法声明

- “{”和声明语句在同一行。
- “}”符号应该独自占一行。

例如：

```
public int myMethod(int i, int j) {
    ...
}
```

9. 语句格式

9.1 return 语句

return 后面的 value 在比较明显的时候不要用“()”。例如：

```
return;
return myDisk.size();
return (size ? size : defaultSize);
```

9.2 if, if-else, if-else-if-else 语句

例如：

```
if (condition) {
    statements;
}

if (condition) {
    statements;
} else {
    statements;
}

if (condition) {
    statements;
} else if (condition) {
    statements;
} else if (condition) {
    statements;
```

```
}
```

9.3 for 语句

例如：

```
for (initialization; condition; update) {
    statements;
}
```

9.4 while 语句

例如：

```
while (condition) {
    statements;
}
```

9.5 do-while 语句

例如：

```
do {
    statements;
}
while (condition);
```

9.6 switch 语句

例如：

```
switch (condition) {
case ABC:
    statements;
case DEF:
    statements;
    break;
case XYZ:
    statements;
    break;
default:
    statements;
    break;
}
```

9.7 try-catch 语句

例如：

```
try {
    statements;
} catch (ExceptionClass e) {
    statements;
} finally {
    statements;
}
```

10. JavaDoc 的格式定义

10.1 文件头

应该包括 Copyright，文件版本等信息。例如：

```
/*
 * Copyright (C) 2010  百润百成科技有限公司.
 *
 *  本系统是商用软件,未经授权擅自复制或传播本程序的部分或全部将是非法的.
 *
 * $$Id:$$
 * Date          Author          Description
 */
```

10.2 类说明信息

定义文件描述，作者，版本。例如：

```
/**
 * Class description goes here.
 *
 *
 * @author <a href="machunlin@biceng.com.cn">马春林</a>
 * @version CVS $$Revision$$ $$Date$$
 */
```

10.3 变量定义

定义变量描述。例如：

```
/**
```

```
 * Comment for <code>${field}</code>
 */
```

10.4 方法定义

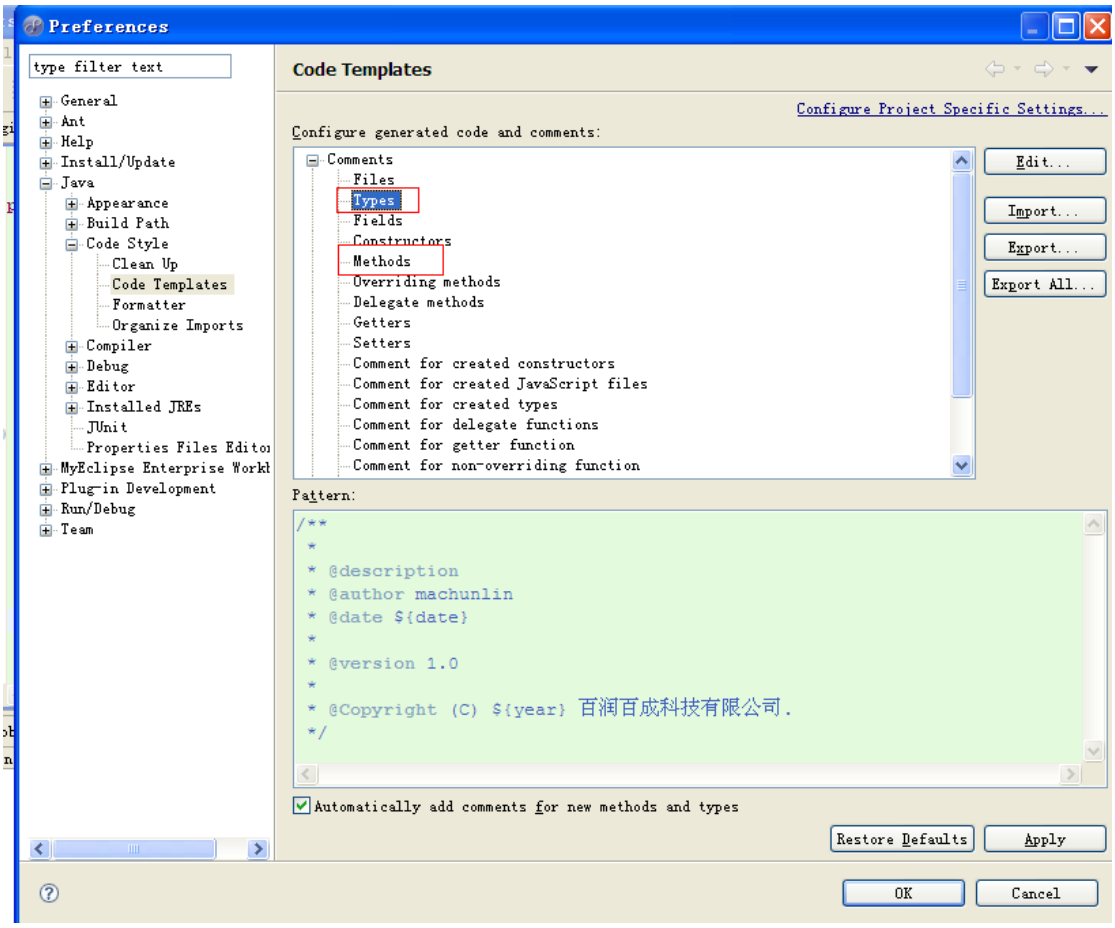
定义方法的描述，参数，返回值，参照文档，Exception。例如：

```
/**
   * description
   *
   * @param
   * @return
   * @exception
   * @see
   * @since
   */
```

11. 关于文档注释

MyEclispe 设置注释模板的入口： Window->Preference->Java->Code Style->Code Template 然后展开 Comments 节点就是所有需设置注释的元素啦。

图例：



现就每一个元素逐一介绍：

文件(Files)注释标签：

```
/**
* @Title: ${file_name}
* @Package ${package_name}
* @Description: ${todo}(用一句话描述该文件做什么)
* @author machunlin machunlin@biceng.com.cn
* @date ${date} ${time}
* @version V1.0
*/
```

类型(Types)注释标签（类的注释）：

```
/**
* @ClassName: ${type_name}
```

```
* @Description: ${todo}(这里用一句话描述这个类的作用)
* @author machunlin machunlin@biceng.com.cn
* @date ${date} ${time}
*
* ${tags}
*/
```

字段(Fields)注释标签：

```
/**
* @Fields ${field} : ${todo}(用一句话描述这个变量表示什么)
*/
```

构造函数标签：

```
/**
* <p>Title: </p>
* <p>Description: </p>
* ${tags}
*/
```

方法(Constructor & Methods)标签：

```
/**
* @Title: ${enclosing_method}
* @Description: ${todo}(这里用一句话描述这个方法的作用)
* @param ${tags}    设定文件
* @return ${return_type}    返回类型
* @throws
*/
```

覆盖方法(Overriding Methods)标签：

```
/* (非 Javadoc)
* <p>Title: ${enclosing_method}</p>
* <p>Description: </p>
* ${tags}
* ${see_to_overridden}
*/
```

参考文档：

寰信通公司 JAVA 编码规范，

中创软件编码规范，

《高质量程序设计指南--c++/c 语言》

Ps:其他

1.测试类，新建 test 源文件，类以 test 结尾，方法以 test 开头:

```
public class UserManagerTest {

    public static BeanFactory factory = new
ClassPathXmlApplicationContext(
            "applicationContext.xml");
   public static IUserManagerService userManager = (IUserManagerService)
factory
            .getBean("userManagerService");

    @Test
    public void testUserSave() {
        userManager.save(userMap);
    }
```

2.修改别人的代码要有注释 modify_by ,modify_date :

```
//==========后台添加用户需要数字证书 modify_by machunlin 2011-11-11 start
        if(userMap.get("certificates")!=null
&& !userMap.get("certificates").equals("")){

    user.setCertificates(userMap.get("certificates").toString().trim(
));
        }
        if(userMap.get("status")!=null
&& !userMap.get("status").equals("")){
```

```
            user.setStatus(userMap.get("status").toString().trim());
        }

//==============后台添加用户需要数字证书 end
```

3.Action 中的异常要记录到 log4j 中，代码通过测试之后，所有 catch 语句中不要有 e.printStatckTrace():

```
public class UserManagerAction extends ActionSupport {
    private static final long serialVersionUID = 1L;
    Logger log = Logger.getLogger(UserManagerAction.class);
     /**
     *
     * @description 执行用户修改
     * @param
     * @return String
     */
@SuppressWarnings("unchecked")
public String doUpdate() {

    try {
            userManagerService.save(userMap);
        } catch (RemoteCallException e) {
            log.error(e.getMessage());
            System.out.println(e.getMessage());
            setReturnMsg(request, e.getMessage());
            return "addFail";
        }

        return "addSuccess";

 }

    /**
     *
     * @description 将系统处理异常结果返回页面
     * @param returnMsg
     *            页面提示
     * @return void
     */
    public void setReturnMsg(HttpServletRequest request, String
returnMsg) {
        request.setAttribute("message", returnMsg);
```

```
    }
}
```

4.编码格式统一 utf-8，web 服务器 uriencode=”utf-8”。