Java 编码规范

术语

Pascal case －所有单词第一个字母大写，其它字母小写。

例：Person, OrderDetail, OilTank。

Camel case －除了第一个单词， 所有单词第一个字母大写， 其他字母小写。

例：oilLevel, customerName

在后面的描述中我们使用 Pascal 代表第一种表示方式， Camel 代表第二

种表示方式。

约束

没有一个规范可以到处适用，也不可能无所不包。

1．本规范并非完全强制性规范，对于任何违背本规范但能提高代码可读性

的措施，都可以采用。

2．本规范第 3，4，5 章为强制性规范， 其它章节为建议规范。 如违背本条，

请参考第一条。

一般规则

1．所有包，类，接口，方法，属性，变量，参数均使用英文单词进行命名，

具体细节请参见命名规范一章。

2．命名包，类，接口，方法以及变量时，尽量使用贴近问题域的表意丰富

的名称。

3．修改源代码时，应尽量保持与所修改系统的编码风格保持一致。

4．所有包名使用必须使用 com.[company]前缀，所有项目使用 com.

[company].projects.[project name], company 是公司简称， project name 是项

目的缩写。

格式规范

1．包的导入应该按照相关性进行分组。

2

import java.io.IOException;

import java.net.URL;

import java.rmi.RmiServer;

import java.rmi.server.Server;

import javax.swing.JPanel;

import javax.swing.event.ActionEvent;

import org.linux.apache.server.SoapServer;

2．只倒入明确需要的类，这样只要看导入列表，就可以知道该类依赖于哪

些类和接口，保证可读性。

import java.util.List; // 避免: import java.util.\*

import java.util.Arraylist;

import java.util.HashSet;

3．类和接口中元素的布局顺序。

（1）类和接口的文档描述

（2）类和接口的声明

（3）类的静态变量，按照 public，protected，package，private 的顺序。

（4）实例变量，按照 public，protected，package，private 的顺序。

（5）类的方法，无固定顺序。

4．方法修饰关键字定义顺序。

<public, protected, private > static abstract synchronizedunuaual final native

methodName注意访问标示符一定要在最前面。

public static double square(double a);

//避免: static public double square(double a);

5．变量声明，采用 Camel 表示法不要在一行声明多个变量。

//推荐

int level;

int size;

//避免

int level, size;

6．数组指示符紧跟类型变量

int[] a = new int[20]; // 避免: int a[] = new int[20]

3

一个变量要代表独立的意思，不要在其生命周期赋予它不同的概念。

int tempValue;

tempValue = maxValue;

…

tempValue = minValue;

…

tempValue = anotherValue;

tempValue 在生命周期内表示了各种各样的意图，增加理解代码的难度。

应该为每个独立概念定义单独的变量：

int tempMaxValue;

int tempMinValue;

int tempAnotherValue;

7．循环变量应靠近循环体初始化

isDone = false

while(!isDone){

…

}

//避免

isDone = false;

…

…

while(!isDone){

…

}

8．避免长的布尔表达式，应换成多个更容易理解的表达式。

bool isFinished = (elementNo < 0) || (elementNo > maxElement);

bool isRepeatedEntry = elementNo == lastElement;

if (isFinished || isRepeatedEntry) {

…

}

// 避免

if ((elementNo < 0) || (elementNo > maxElement)|| elementNo ==

lastElement) {

…

4

}

9．不要在条件语句中执行方法，以提高可读性

InputStream stream = File.open(fileName, "w");

if (stream != null) {

…

}

//避免

if (File.open(fileName, "w") != null)) {

…

}

10．条件语句的主要形式，即使单条语句，也要使用括号括起来。

if (condition) {

statements;

}

if (condition) {

statements;

} else {

statements;

}

if (condition) {

statements;

} else if (condition) {

statements;

} else {

statements;

}

11．空循环体也要使用完整的 {} 块

for (initialization; condition; update) {

;

}

12．switch 语句的使用格式

switch (condition) {

case ABC :

statements;

5

//穿透，一定要做出注释

case DEF :

statements;

break;

case XYZ :

statements;

break;

default :

statements;

break;

}

13．空格的使用

（1）运算符两边应该各有一个空格。

（2）Java 保留字后面应跟随一个空格。

（3）逗号后面应跟随一个空格。

（4）冒号两个应各有一个空格。

（5）分号后面应跟随一个空格。

18．空行的使用

（1）文件头部注释、 package 语句和 import 语句之间。

（2）class 之间

（3）方法之间

（4）方法中，变量的申明和具体代码之间。

（5）逻辑上相关的语句段之间

（6）块注释和行注释的前面

▽ --- 代表空行

/\*\*

\*

\*/

▽

package XXX.XXX;

▽

import XXX.XXX.XXX.XXX;

▽

/\*\*

6

\*

\*/

public class UserFileAccess {

▽

//

private int myObjId;

▽

/\*\*

\*

\*/

public UserFileAccess() {

???

}

▽

/\*\*

\*

\*/

public void getCtlInfo() {

int count;

String msg;

▽

count = 100;

???

▽

//实现代码注释前空行

msg = “MESSAGE”;

▽

count = dataCount;

if (count == 0) {

???

}

}

▽

/\*\*

7

\*

\*/

private class UserFileRead {

14．逻辑上紧密相关的代码块应该用一个空行分开。

// Create a new identity matrix

Matrix4x4 matrix = new Matrix4x4();

// Precompute angles for efficiency

double cosAngle = Math.cos(angle);

double sinAngle = Math.sin(angle);

// Specify matrix as a rotation transformation

matrix.setElement(1, 1, cosAngle);

matrix.setElement(1, 2, sinAngle);

matrix.setElement(2, 1, -sinAngle);

matrix.setElement(2, 2, cosAngle);

// Apply rotation

transformation.multiply(matrix);

15．三元条件运算符

可以使用如下三种表达方式，条件要用括号括起来。

alpha = (aLongBooleanExpression) ? beta : gamma;

alpha = (aLongBooleanExpression) ? beta

: gamma;

alpha = (aLongBooleanExpression)

? beta

: gamma

8

命名规范

一般命名规范

1．包名应该用小写字母，不要出现下划线等符号，名词用有意义的缩写或

者英文单词。

//推荐

com.esse.business

java.lang.util

//避免

com.Esse-tech.buSiness

2．所有类命名使用 Pascal 表示方式，使用名词组合。

UserManager, ClassLoader, HttpHeaderResult

3．使用名词组合或形容词去命名一个接口，接口声明了一个对象能提供的

服务，也描述了一个对象的能力。一般以 “able和”“ible作为后缀，代表了一种能 ”

力。

public interface Runnable{

public void run();

}

public interface Accessible{

public Context getContext();

}

4．变量名和参数名使用 Camel 表示方式。

userName, objectFactory, entrys, list

5．对于常量名，使用大写字母，并使用下划线做间隔。

MAX\_TIMES, DEFAULT\_NAME

程序中应该使用常量代替 “25，

”“100等实际的数字。 ”

6．方法名应该使用动词开头， 使用 Camel 表示方式 ,一般由动词 +名词组成。

getName, initialize, addParameter, deleteUser

7．缩写字母也应该保持首字母大写

exportHtmlSource(); // 避免: exportHTMLSource();

openDvdPlayer(); // 避免: openDVDPlayer();

8．变量的名字应该和类型名称一致

void setTopic(Topic topic) // 避免: void setTopic(Topic value)

9

// 避免: void setTopic(Topic aTopic)

// 避免: void setTopic(Topic t)

void connect(Database database)

// 避免: void connect(Database db)

// 避免: void connect(Database oracleDB)

当同时定义多个属于同一个类的变量时，把类型作为实例的后缀，如：

Point startPoint;

Point centerPoint;

这样做是为了从实例名就可以推断它的类型名称。

特殊命名规范

1．使用 get/set 对类属性进行访问，这是 Java 社区的核心编码规范。

2．使用 is 前缀表示一个布尔变量和方法。

isUsed, isEmpty,isVisible,isFinished

有时也可以使用 has，can，should：

boolean hasLicense();

boolean canEvaluate();

boolean shouldAbort = false;

3．在查询方法中应使用 find 作为前缀

vertex.findNearestVertex();

matrix.findSmallestElement();

node.findShortestPath(Node destinationNode);

4．使用 initialize 做为对象初始化的方法前缀，也可以简写为 init

initializeFiles();

init();

initFontSet();

5．对于对象集合，变量名称应使用复数。

Collection<Point> points;

int[] values;

6．对于抽象类，应该使用 Abstract 前缀。

AbstractReportBuilder，AbstractBeanFactory

7．对于表示编号的变量，应加 No 后缀。

tableNo, userNo,employeeNo

8．常在一起使用的对称词汇，这些词汇一起使用，方法的表达意图自然可

以互相推测和演绎。

10

get/set, add/remove, create/destroy, start/stop,

insert/delete, increment/decrement, begin/end, first/last,

up/down, min/max, next/previous, old/new, open/close,

show/hide, suspend/resume

9．避免使用否定布尔变量

bool isError; // 避免: isNoError

bool isFound; // 避免: isNotFound

10．异常类应该使用 Exception 做为后缀。

AccessException, RuntimeException

11．对于单例类（ Singleton），应该使用 getInstance 方法得到单例。

class UnitManager {

private final static UnitManager instance = new UnitManager();

private UnitManager() {

...

}

public static UnitManager getInstance(){

return instance\_;

}

}

12．对于工厂类，进行创建对象的方法，应该使用 new 前缀

class PointFactory {

public Point newPoint(...) {

...

}

}

11

注释规范

概述

代码中为什么要包含注释？

1．别人要调用你的程序中的公共接口，对这部分进行文档描述，使别人能

够正确而有效的使用它。

2．除了自己，别人要阅读和维护你的代码。为了使代码更容易维护，首先

要使代码更易于理解， 才能在理解的基础上进行维护。 对这些代码进行文档描述，

将使这个过程变得更加容易。

对代码进行注释， 是在代码可读性的基础上， 使用自然语言对代码所表达的

意思进行阐述。并不是说代码可以写的很烂， 注释写的很详细，这不是好的方式。

如果代码可读性很好， 命名表意丰富，清晰，一般不需要特别多的注释。 对于类，

主要着重要描述它的职责， 即它能干什么， 对于复杂的算法实现， 应该使用内部

实现注释， 说明算法的主要思路， 对于长方法， 要让阅读代码的人比较容易的明

白方法实现的主要流程。反之，对于一看就懂的方法，则不需要进行注释，比如

get/set 方法。

一般原则

1．代码应该和注释保持同步，如果代码和注释不同步，则阅读代码的人会

想，“到底是代码准确，还是注释准确啊 ”，换谁都会糊涂。

2．注释尽量简洁，尺度没有准确的定义，大部分人能明白即可，可以将自

己的代码给同事看看。太简单的方法就不要注释了，比如上面提到的 get/set 方

法。

①代码的版权信息。

②类描述信息，描述类的主要职责和用处。

③方法描述信息，描述方法是做什么的， 如何调用，最好给出调用代码示例。

④ JavaDoc tags ,用来生成 Html 形式的 API 文档

⑤内部实现注释， 用于描述复杂的算法， 长方法，从为什么要这么做角度去

描述