JAVA 编程规范

目录

[**1.Java编程命名规范** 1](#_Toc10210427)

[**1.1包的命名标准** 1](#_Toc10210428)

[**1.1.1命名包** 2](#_Toc10210429)

[**1.2类、接口的命名标准** 2](#_Toc10210430)

[**1.2.1命名类** 2](#_Toc10210431)

[**1.2.2命名接口** 3](#_Toc10210432)

[**1.2.3命名编译单元** 3](#_Toc10210433)

[**1.3成员函数命名标准** 3](#_Toc10210434)

[**1.3.1命名成员函数** 3](#_Toc10210435)

[**1.3.2获取函数** 4](#_Toc10210436)

[**1.3.3设置函数** 4](#_Toc10210437)

[**1.3.4命名构造函数** 4](#_Toc10210438)

[**1.3.5成员函数的可见性** 4](#_Toc10210439)

[**1.4字段/属性 命名标准** 5](#_Toc10210440)

[**1.4.1命名字段** 5](#_Toc10210441)

[**1.4.2命名组件（部件）** 5](#_Toc10210442)

[**1.5局部变量命名标准** 6](#_Toc10210443)

[**1.5.1命名流** 6](#_Toc10210444)

[**1.5.2命名循环计数器** 6](#_Toc10210445)

[**1.5.3命名异常对象** 6](#_Toc10210446)

[**1.6成员函数参数命名标准** 7](#_Toc10210447)

[**2  Java编程注释规范** 8](#_Toc10210448)

[**2.1注释约定：** 8](#_Toc10210449)

[**2.1.1Java 注释语句类型** 8](#_Toc10210450)

[**2.1.2javadoc标志** 9](#_Toc10210451)

[**2.2包的注释标准** 10](#_Toc10210452)

[**2.3类、接口的注释标准** 11](#_Toc10210453)

[**2.3.1注释类** 11](#_Toc10210454)

[**2.3.2注释接口** 12](#_Toc10210455)

[**3  成员函数注释标准** 13](#_Toc10210456)

[**3.1成员函数的函数头** 13](#_Toc10210457)

[**3.1.1内部注释** 15](#_Toc10210458)

[**3.2字段/属性 注释标准** 16](#_Toc10210459)

[**3.2.1注释一个字段** 16](#_Toc10210460)

[**3.3局部变量注释标准** 17](#_Toc10210461)

[**3.3.1声明和注释局部变量** 17](#_Toc10210462)

[**3.4成员函数参数注释标准** 17](#_Toc10210463)

[**4  Java编程格式规范** 18](#_Toc10210464)

[**4.1if-else-elseif** 18](#_Toc10210465)

[**4.2switch** 18](#_Toc10210466)

[**4.3 while** 18](#_Toc10210467)

[**4.4do … while** 19](#_Toc10210468)

[**4.5for** 19](#_Toc10210469)

[**4.6try-catch** 19](#_Toc10210470)

[**5    Java错误和异常处理规范** 20](#_Toc10210471)

[**5.1Java程序错误和异常处理原则** 20](#_Toc10210472)

[**5.2在底层的类中** 20](#_Toc10210473)

[**5.3在高层的类中** 20](#_Toc10210474)

[**6  Java编程一般约定：** 20](#_Toc10210475)

[**6.1首先定义公共接口** 20](#_Toc10210476)

[**6.2编程一般约定** 21](#_Toc10210477)

[**7  其他** 22](#_Toc10210478)

[**Ø  让代码分段/缩进** 22](#_Toc10210479)

**1.Java编程命名规范**

**一般约定：**

命名时应始终采用完整的英文描述符。此外，一般应采用小写字母，但类名、接口名以及任何非初始单词的第一个字母要大写。

1. **使用可以准确说明变量/字段/类的完整的英文描述符**。例如， **firstName**，**grandTotal** 或**CorporateCustomer** 而非**x1**，**y1** 或 **fn** ，使代码以理解、维护和改进。
2. **采用该领域的术语**。如果用户称他们的“客户” (clients) 为“顾客”Customers)，那么就采用术语 **Customer** 来命名这个类，而不用 **Client**。
3. **采用大小写混合，提高名字的可读性**。一般应该采用小写字母，但是类和接口的名字的首字母，以及任何中间单词的首字母应该大写。
4. **尽量少用缩写，但如果需要使用，可以选择单词的头4个字母作为缩写**。如：DeptInfoValue，MapActi, MapActiInst 等等。
5. **避免使用长名字（最好不超过 15 个字母）**。名字太长时，应该给它起个短一点的别名。
6. **避免使用相似或者仅在大小写上有区别的名字**。
7. **避免使用下划线作为名字的首末字母**。以下划线为首末字母的名字通常为系统保留。

**1.1包的命名标准**

**1.1.1命名包**

关于包的命名有几条规则，这些规则是：

1、 包的命名都是使用小写的英文字母或汉字组成，每个包名称之间用点号分隔开来。如java.lang;javax.servlet.http 等。

2、 全局包的名字用所在机构的 Internet保留域名开头。如：com.sun   com.oracle. 等等

3、 我们的命名约定：

com.excellence.common excellence下的通用文件，可能被以后不同项目通用的类文件。

**1.2类、接口的命名标准**

**1.2.1命名类**

标准Java类约定是使用完全的英文描述符，所有单词的第一个字母要大写，并且单词中大小写混合。

类名应是单数形式。

示例：

Customer  Employee        Order   OrderItem       FileStream        String

    相关的类如果使用了某些已经约定的开发模式，应该使用相关约定的命名：如：LoggerFactory , UserDAO 等。

    在同一个模块中的相关类，可以采用同样开头的名称以区分他们的相关性。如：UserValue, UserDAO 等。

**1.2.2命名接口**

采用完整的英文描述符说明接口封装，所有单词的第一个字母大写。习惯上，名字后面加上后缀 able,. ible 或者 er。

示例：

Runnable, Contactable,

Prompter, Singleton

**1.2.3命名编译单元**

编译单元，在这个情况下是一个源码文件。一个源码文件中最多只能有一个public类，且源码文件名必须与该public类同名。

示例：

Customer.java

Singleton.java

DeptInfo.java

**1.3成员函数命名标准**

**1.3.1命名成员函数**

成员函数的命名应采用完整的英文描述符，大小写混合使用：所有中间单词的第一个字母大写。**成员函数名称的第一个单词常常采用一个有强烈动作色彩的动词。**

**示例：**

openAccount()  printMailingLabel() createUser()   delete()

**1.3.2获取函数**

获取函数作为一个成员函数，返回一个字段的值。除了布尔字段之外，应采用get 作为字段的前缀；布尔字段采用 is 作为前缀。

**示例：**

       getUserName()

**1.3.3设置函数**

设置函数，也叫变值函数，是可以修改一个字段值的成员函数。无论何种字段类型，都要在字段名的前面加上 set 前缀。

**示例：**

setUserName(StringuserName)

**1.3.4命名构造函数**

构造函数与它所属类的名字总是相同的。例如，类**Customer** 的构造函数是 **Customer()**。注意大小写一致。

这个命名约定由 Sun 公司设定，必须严格遵守。

**1.3.5成员函数的可见性**

良好的程序设计应该尽可能减小类与类之间耦合，所遵循的经验法则是：尽量限制成员函数的可见性。如果成员函数没必要公有 (public)，就定义为保护 (protected)；没必要保护 (protected)，就定义为私有 (private)。

**1.4字段/属性 命名标准**

字段/属性是说明一个对象或者一个类的一段数据。字段可以是像字符串或者浮点数这样的基本数据类型，也可以是一个对象，例如一个消费者或者一个银行帐户。

**1.4.1命名字段**

应采用完整的英文描述符来命名字段。象数组或者矢量这样是集合的字段，命名时应使用复数来表示它们代表多值。

示例：

firstName  zipCode unitPrice  discountRate    orderItems（复数）

**1.4.2命名组件（部件）**

应采用完整的英文描述符命名组件（接口部件），名字的后缀是组件类型名。

示例：

okButton       customerList       fileMenu        newFileMenuItem

**1.5局部变量命名标准**

一般说来，命名局部变量遵循与命名字段一样的约定，即使用完整的英文描述符，任何非开头的单词的第一个字母要大写。

但是为方便起见，对于如下几个特殊的局部变量类型，这个约定可以放宽：

* 流
* 循环计数器
* 异常

**1.5.1命名流**

当有一个单输入和/或单输出流在一个成员函数中被打开、使用和关闭时，通常的约定是对这些流分别采用 in 和 out 。对于既用于输入又用于输出的流，采用 inOut 来命名。

一个常用的取代这种约定的方法是分别采用 inputStream，outputStream 和 ioStream 这样的名字，而不是 in，out 和 inOut，虽然这与 Sun 公司的建议相抵触。

**1.5.2命名循环计数器**

因为局部变量常用作循环计数器，并且它为 C/C++ 所接受，所以在 Java 编程中，可以采用 i， j 或 k 作为循环计数器。

**1.5.3命名异常对象**

因为在 Java 代码中异常处理也非常普遍，所以字母 e 作为**一般的异常**符被广泛地接受。

如果有多个异常同时出现，应该使用该类型异常全称的缩写表示。

catch(SQLExceptionsqlexception)

**1.6成员函数参数命名标准**

命名参数

参数命名遵循与局部变量命名完全一样的约定。对于局部变量，名字隐藏是一个问题。

一个可行的取代方法，是采用局部变量的命名约定，但在名字之前加入“a”或“an”。加上“a”或“an”有助于让参数与局部变量和字段区分开来，避免名字隐藏的问题。这种方法较好。

示例：

aCustomer  anInventoryItem     aPhotonTorpedo anInputStream       anException

但是，如果是设置/获取函数，由于采用了jbuilder 下的bean/properties工具生成了代码，可以不采用这种命名方法。而采用，this关键字将局部变量和函数参数区分开。

如：public void setUserName(String userName) {

   this.userName = userName;

  }

**2  Java编程注释规范**

**2.1注释约定：**

**2.1.1Java 注释语句类型**

Java 有三种注释语句风格：

以 /\*\* 开始， \*/ 结束的文档注释，

以 /\* 开始，以 \*/ 结束的C语言风格注释，

以及以 // 开始，代码行末尾结束的单行注释。

|  |  |  |
| --- | --- | --- |
| **注释语句类型** | **用法** | **示例** |
| **文档注释** | 在接口、类、成员函数和字段声明之前紧靠它们的位置用文档注释进行说明。文档注释由 *javadoc* 处理，为一个类生成外部注释文档，如下所示。 | /\*\*  Customer （顾客）.顾客是指作为我们的服务及产品的销售对象的任何个人或组织。  @author S.W. Ambler  \*/ |
| *C***语言风格注释** | 采用 C 语言风格的注释语句将无用的代码注释掉。保留这些代码是因为用户可能改变想法，或者只是想在调试中暂时不执行这些代码。 | /\*  这部分代码已被它前面的代码替代，所以于 1999 年6 月 4 日被 B. Gustafsson 注释掉。如果两年之后仍未用这些代码，将其删除。  . . . （源代码）  \*/ |
| **单行注释** | 在成员函数内部采用单行注释语句对业务逻辑、代码片段和临时变量声明进行说明。 | // 因为让利活动  // 从 1995 年 2 月开始， |

**2.1.2javadoc标志**

Sun 公司的 JavaDevelopment Kit (JDK) 中有一个名为*javadoc* 的程序。它可以处理Java 的源代码文件，并且为 Java 程序产生 HTML 文件形式的外部注释文档。Javadoc 支持一定数目的标记，标识注释文档中各段起始位置的保留字。详情请参考 JDK *javadoc* 文档。

|  |  |  |
| --- | --- | --- |
| **标记** | **用于** | **目的** |
| @author name | 类、  接口 | 说明特定某一段程序代码的作者。每一个作者各有一个标记。 |
| @deprecated | 类、  成员函数 | 说明该类的应用程序编程接口 (API) 已被废弃，因此应不再使用。 |
| @exception name description | 成员函数 | 说明由成员函数发出的异常。一个异常采用一个标记，并要给出异常的完整类名。 |
| @param name description | 成员函数 | 用来说明传递给一个成员函数的参数，其中包括参数的类型/类和用法。每个参数各有一个标记。 |
| @return description | 成员函数 | 若成员函数有返回值，对该返回值进行说明。应说明返回值的类型/类和可能的用途。 |
| @since | 类、成员函数 | 说明自从有 JDK 1.1 以来，该项已存在了多长时间。 |
| @see ClassName | 类、接口、成员函数、字段 | 在文档中生成指向特定类的超文本链接。可以并且应该采用完全合法的类名。 |
| @see ClassName#member functionName | 类、接口、成员函数、字段 | 在文档中生成指向特定成员函数的超文本链接。可以并且应该采用完全合法的类名。 |
| @version text | 类、接口 | 说明特定一段代码的版本信息。 |

**2.2包的注释标准**

**注释包**

应保有一个或者多个外部文档以说明你的机构所创建的包的用途。对于每个包，应说明：

1. 包的基本原理。其他开发者需要了解一个包是用来做什么的，这样他们才能判断是否可以用它，如果是一个共享包，他们可以判断是否需要改进或是扩展它。
2. 包中的类。在包中要包含一个类和接口的列表，每个类或接口用简短的一行文字来说明，以便让其他的开发者了解这个包中含有什么。

技巧：生成一个以包名命名的 HTML 文件，将它放到包的适当的目录中去。这个文件应具有后缀*.htm*。

**2.3类、接口的注释标准**

**2.3.1注释类**

以下的信息应写在文档注释中紧靠类的定义的前面：

1. 类的目的和作用。开发者需要了解一个类的一般目的，以判断这个类是否满足他们的需求。养成一个注释与类有关的任何好东西的习惯。
2. 已知的问题。如果一个类有任何突出的问题，应说明出来，让其他的开发者了解这个类的缺点/难点。此外，还应注明为什么不解决问题的原因。
3. 类的开发/维护历史。通常要包含一个历史记录表，列出日期、类的作者和修改概要。这样做的目的是让进行维护的程序员了解过去曾对一个类所做的修改，是谁做了什么样的修改。
4. 版权信息。

对于以上几点：其中类的目的和作用这项是必须要填写的。采用Javadoc形式标志的一个示例如下：

/\*\*

 \* <p>类说明:LoggerFactory,创建Logger的工厂类</p>

 \*<p>Copyright 2003: Excellence Network Co.,LTD</p>

 \* @author fjy

 \* @version 1.0

 \* @since2003-06-10

 \* @modified fjy2003-06-13  修改了XXXX

\* @modified fjy2003-06-23  修改了XXXX

 \*/

**2.3.2注释接口**

以下的信息应写在文档注释中紧靠接口定义的前面：

1. 目的。在其他开发者应用一个接口之前，他们需要理解接口封装的概念。换句话说，他们需要了解接口的目的。一个好的检测是否有必要定义一个接口的方法是你是否可以很容易地说明它的目的。 如果说明起来有困难，很可能从一开始起就不需要这个接口。在 Java 中接口的概念较新，所以人们对如何正确使用它们还不是很有经验，常常滥用它们。
2. 它应如何被使用以及如何不被使用。开发者需要了解接口应如何使用以及如何不被使用。

因为成员函数的标识在接口中定义，所以对于每个成员函数的标识应遵循第二章中所讨论的注释约定。

示例如下：

/\*\*

 \* <p>接口说明:Runnable，作为线程的接口。实现run()方法，可以实现线程的效果。</p>

 \*<p>Copyright 2003: Excellence Network Co.,LTD</p>

 \* @author fjy

 \* @version 1.0

 \* @since2003-06-10

 \* @modified fjy2003-06-13  修改了XXXX

\*/

**3  成员函数注释标准**

注释一个成员函数是为了函数更加可被理解，进而可维护和可扩展。

**3.1成员函数的函数头**

每一个 Java 成员函数都应包含某种称之为“成员函数文档”的函数头。这些函数头在源代码的前面，用来记录所有重要的有助于理解函数的信息。这些信息包含但不仅仅局限于以下内容：

1. **成员函数做什么以及它为什么做这个。**通过给一个成员函数加注释，让别人更加容易判断他们是否可以复用代码。注释出函数为什么做这个可以让其他人更容易将你的代码放到程序的上下文中去。也使其他人更容易判断是否应该对你的某一段代码加以修改（有可能他要做的修改与你最初为什么要写这一段代码是相互冲突的）。
2. **哪些参数必须传递给一个成员函数。**还必须说明，如果带参数，那么什么样的参数必须传给成员函数，以及成员函数将怎样使用它们。这个信息使其他程序员了解应将怎样的信息传递给一个成员函数。*javadoc @param* 标识便用于该目的。
3. **成员函数返回什么。**如果成员函数有返回值，则应注释出来，这样可以使其他程序员正确地使用返回值/对象。*javadoc @return* 标识便用于此目的。
4. **已知的问题。**成员函数中的任何突出的问题都应说明，以便让其他程序开发者了解该成员函数的弱点和难点。如果在一个类的多个成员函数中都存在着同样的问题，那么这个问题应该写在类的说明里。
5. **任何由某个成员函数抛出的异常。**应说明成员函数抛出的所有异常，以便使其他程序员明白他们的代码应该捕获些什么。*javadoc @exception* 标识便用于此目的。
6. 成员函数的开发/维护历史。通常要包含一个历史记录表，列出日期、修改的作者和修改概要。这样做的目的是让进行维护的程序员了解过去曾对一个成员函数所做的修改，是谁做了什么样的修改。

示例：

/\*\*

   \* 执行SELECT函数，可以分页。按照输入的页码、每页的行数，返回当前页的结果集

   \* 又多行记录：结果放入VECTOR，VECTOR里面的每一项代表一行记录。

  \*          VECTOR里面包含的String[]就是真正的结果

   \* @parampageNo 返回SELECT结果中的第几页数据

   \* @parampageCount 每页的最大行数

   \* @throwsjava.lang.Exception

   \* modified by fjy 2003-07-03 增加了pageCount参数的检查

   \*/

public voidExecuteSelect(int pageNo,int pageCount)  throws Exception  {

}

**3.1.1内部注释**

内部注释应采用两种方式：C 语言风格的注释 (/\* 和 \*/) 和单行注释 (//)。**对业务逻辑采用单行注释，**因为它可用于整行注释和行末注释。**采用 C 语言风格的注释语句去掉无用的代码，**因为这样仅用一个语句就可以容易地去掉几行代码。所以，应尽量减少使用它们。

在函数内，需要注释说明的地方：

1. **控制结构**。说明每个控制结构，例如比较语句和循环。你无须读完整个控制结构内的代码才判断它的功能，而仅需看看紧靠它之前的一到两行注释即可。
2. **代码做了些什么以及为什么这样做**。通常你常能看懂一段代码做了什么，但对于那些不明显的代码，你很少能判断出它为什么要那样做。
3. **局部变量**。虽然我们在第 4 章将仔细讨论这一点，在一个成员函数内定义的每一个局部变量都应在它代码的所在行声明，并且应采用一个行内注释说明它的用法。
4. **难或复杂的代码**。若发现不能或者没有时间重写代码，那么应将成员函数中的复杂代码详细地注释出来。一般性的经验法则是，如果代码并非显而易见的，则应说明。
5. **处理顺序**。如果代码中有的语句必须在一个特定的顺序下执行。

**在循环或者控制结构的闭括号后加上注释。**类似 **//endif**，**//end for**，**//end switch**，&这样的注释加在闭括号所在行的行后，可以使代码更易理解。

**3.2字段/属性 注释标准**

**3.2.1注释一个字段**

字段都应很好地加以注释，以便其他开发者理解它。要想有效地注释，以下的部分需要说明：

1. 字段的说明。需说明一个字段，才能使人了解如何使用它。
2. 注释出所有采用的不变量。字段中的不变量是指永远为“真”的条件。例如，字段 dayOfMonth 的不变量可能是它的值只能在 1 到 31 之间（显然，可以用基于某一年里的某个月份来限制这个字段值，使其变的更加复杂）。通过说明字段值的限制条件，有助于定义重要的业务规则，使代码更易理解。

  示例：

  //需要连接数据源的名称

 private   String     dsName;

**3.3局部变量注释标准**

**3.3.1声明和注释局部变量**

在 Java 中声明和注释局部变量有几种约定。这些约定是：

1. 一行代码只声明一个局部变量。这与一行代码应只有一个语句相一致，并使得对每个变量采用一个行内注释成为可能。
2. **用一个行内注释语句说明局部变量。行内注释是一种紧接在同一行的命令代码后**，用符号 // 标注出来的单行注释风格（它也叫“行末注释”）。应注释出一个局部变量用于做什么、在哪里适用、为什么要用等等，使代码易读。

如：

int count = 1;  //记录循环次数

**3.4成员函数参数注释标准**

**注释参数**

成员函数的参数在采用*javadoc @param* 标识的头文件中注释。应说明：

1. 参数用来做什么。需要注释出参数用来做什么，以便其他开发者了解使用参数的上下文。
2. 任何约束或 前提条件。 如果一个参数的值域不能被成员函数接收，则应让调用者知道。可能一个成员函数只接收正数，或者字符数小于五的字符串。
3. 示例。如果应传递什么样的参数不明显，那么应该在注释中给出一个或多个例子。

**4  Java编程格式规范**

        定义大括号“{”一般放在一行语句的结尾。

**4.1if-else-elseif**

如果是if…elseif…else的情况，超过一行的执行体语句要分行，如果其中有任何一个的执行体使用了花括号，就应该都使用花括号。

**4.2switch**

Switch语句应该都带有一个default，当出现了意料之外的情况时可以在default中抛出异常。

**4.3 while**

while语句也是循环性质，所以也要使用花括号

**4.4do … while**

do语句因为使用的情况较少，最好是尽量使用while来代替do

**4.5for**

for语句因为具有循环的性质，应该使用花括号明确的标出语句块的范围

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

   System.out.println(aa[i]);

}

**4.6try-catch**

try-catch是异常捕捉的语句，做为语言本身的规定，就必须使用花括号。

try {

}catch (Exception ex) {

         ex.printStackTrace();

}catch(ClassNotFoundException classnotfoundexception) {

         classnotfoundexception.printStackTrace();

}finally {

   Logger.log("ddd");

}

**5    Java错误和异常处理规范**

**5.1Java程序错误和异常处理原则**

为了系统的稳定性，应该保证代码的健壮性。时时刻刻检查是否有异常产生，并按以下的原则处理。在底层的类中，如果没有显示的界面，应该在相关的log日志文件中记录错误；如果在有显示界面的层中，应该在记录log日志文件的同时，显示相关的出错信息。

在成员函数中应该检查参数的合法性，运行状态中参数的合法性，执行结果是否为null等。

不同模块可以定义自己的异常类,按照实际运行过程可能出现的

**5.2在底层的类中**

在底层的类中出现的异常可以在log的同时，再往外抛出异常。如：

**5.3在高层的类中**

在高层的类中，在页面中出现的异常，则应该引导到出错页面，并显示相关的错误信息。

**6  Java编程一般约定：**

**6.1首先定义公共接口**

首先定义公共接口。大多数有经验的开发者在开始编写类的代码之前就先定义类的公共接口。第一，如果你不知道一个类要完成怎样的服务/行为，你仍有一些设计工作要做。第二，这样做使这个类很快地初具雏形，以便其他有赖于该类的开发者在“真正的”类被开发出来以前至少可以用这个雏形开始工作。第三，这种方法给你提供了一个初始框架，围绕着这个框架你构造类。

**6.2编程一般约定**

|  |  |
| --- | --- |
| 约定目标 | 约定 |
| 存取成员函数 | 考虑对数据库中的字段使用滞后初始化  使用存取函数获得和修改所有字段  对常量采用存取函数  对于集合，加入成员函数来插入和删除项  一旦可能，将存取函数置为被保护类型，不是公共类型 |
| 字段 | 字段永远为私有类型  不要直接访问字段，应使用存取成员函数  不要使用静态常量字段（常量），应使用存取成员函数  不要隐藏名字  一定要初始化静态字段 |
| 类 | 最小化公共和保护接口  在开始写代码之前定义一个类的公共接口  按以下顺序声明一个类的字段和成员函数：   * 构造函数 * 公共成员函数 * 被保护成员函数 * 私有成员函数 * 私有字段 |
| 局部变量 | 不要隐藏名字  一行代码只声明一个局部变量  用一个行内注释说明局部变量  在使用局部变量之前声明它  仅将局部变量用于一件事 |
| 成员函数 | 给代码加上注释  给代码分段  使用空白，控制结构之前用一个空行，成员函数之前用两个空行  一个成员函数应能在 30 秒内让人理解  写短小单独的命令行  尽量限制成员函数的可见性  说明操作的顺序 |

**7  其他**

**Ø  让代码分段/缩进**

**Ø  在代码中使用空白**

建议采用一个空行来分隔代码的逻辑组，例如控制结构，采用两个空行来分隔成员函数定义。

**Ø  采用字段存取函数**