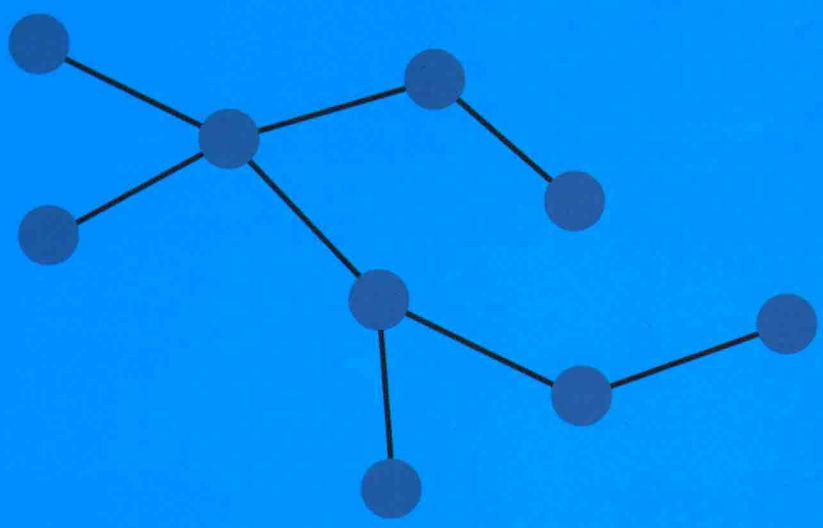
高等院校计算机任务驱动教改教材

Java高级编程

魏 勇 编 著



清华大学出版社

高等院校计算机任务驱动教改教材

Java 高级编程

魏　勇　编著

清华大学出版社
北　京

内 容 简 介

本书是一本针对 Java 解决方案的书籍。随着开发项目的增大，以及开发团队人员的增加，项目管理显得越来越重要。本书将介绍注释文档自动生成、Java 应用程序转换为操作系统平台直接运行的程序、实时监控程序的 JMX 技术、利用 SVN 版本控制等具有 Java 项目管理特征的技术作为第 1 章的开头。接下来的主要内容是在具有 Java 基础知识的前提下，学习如何利用 Java 类库实现数据结构的主要算法、Java 网络编程、MINA 框架、Java 安全技术、远程对象调用、动态模块等内容。

本书适合软件技术相关专业高年级学生学习，也是 Java 工程师重要的参考资料。

图书在版编目(CIP)数据

Java 高级编程/魏勇编著. —北京：清华大学出版社，2017
(高等院校计算机任务驱动教改教材)
ISBN 978-7-302-45094-8

Ⅰ. ①J…　Ⅱ. ①魏…　Ⅲ. ①JAVA 语言－程序设计－高等学校－教材　Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字(2016)第 227217 号

责任编辑：张龙卿
封面设计：徐日强
责任校对：李　梅
责任印制：刘海龙

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62770175-4278
印 装 者：北京泽宇印刷有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：21.25　　**字　　数**：514 千字
版　　次：2017 年 1 月第 1 版　　**印　　次**：2017 年 1 月第1次印刷
印　　数：1～1500
定　　价：45.00 元

产品编号：050894-01

前　言

Java 自从诞生以来，一直是编程语言中的“万金油”，其使用范围广，市场占有率高。随着全球云计算和移动互联网的发展，Java 进一步表现出其明显的优势和广阔的发展前景。因而 Java 是现在大多数企业在从事电子商务开发、企业信息化建设、Web 应用开发时的首选技术。

实际开发过程中，Java 程序员很少碰到只涉及语言本身的问题。因为 Java 在诸多方面都提供了解决方案。譬如在利用 Java 进行项目开发过程中如何进行有效的项目管理；如何直接利用 Java 类库实现数据结构中的算法；如何依靠典型的通信框架实现稳定的系统及建立安全的通信机制；如何实现远程对象的调用；如何实现动态模块等。随着本书学习的深入，读者会越来越感觉到 Java 不仅是一门编程语言，更重要的是 Java 提供了多种解决方案。

本书每一部分的内容都从提出一个具体的实际工作任务开始，分别通过详细设计、编码实现、源代码、测试与运行、技术分析、问题与思考几个步骤来完成。每个步骤各自需要达到的目的如下。

(1) 详细设计。提出实现本任务的基本程序框架和主要算法等。

(2) 编码实现。用 Java 语句实现详细设计，并对重点语句进行分析和说明。

(3) 源代码。给出实现程序的完整源程序。读者可以逐步尝试并练习如何在前两个步骤的基础上写出自己的源程序，从而达到最终完成设计和编写源程序的目的。

(4) 测试与运行。对以上编写的程序进行测试。有时用几组数据直接运行程序进行测试；有时需要编写测试程序，并对结果进行基本的分析。

(5) 技术分析。该步骤是围绕提出的一个工作任务而进行的，对引出的知识需要系统地整理。如果按学科体系组织教学内容，这个步骤应放在最前面，然后再通过一些例子验证。本书基于工作过程，每个具体内容都先让读者知道如何做，再去梳理设计过程中所涉及的知识。

(6) 问题与思考。这个步骤对学习过程中有疑问的一些问题进行讨论，既可以为以后的知识做一些铺垫，又可以对所学内容起到举一反三的作用。

各章内容如下。

第 1 章主要介绍注释文档自动生成、Java 应用程序转换为操作系统平

台直接运行的程序、实时监控程序的 JMX 技术、利用 SVN 版本控制等具有 Java 项目管理特征的技术。

第 2 章主要介绍标准 Java 库提供的最基本的数据结构，讲述如何利用 Java 编程语言实现各种传统的数据结构。

第 3 章从服务器端和客户端两个角度重点介绍利用 Socket 实现网络通信的示例。Java 中网络程序有 TCP 和 UDP 两种协议，TCP 通过握手协议进行可靠的连接，UDP 则是不可靠的连接。

第 4 章介绍如何利用 MINA 框架开发通信软件。MINA 封装了 TCP/IP、线程等内容，由于其安全、稳定，以及开发人员无须考虑通信细节等特点，广泛应用在 Client/Server 模式的环境中。成功的案例包括 Openfire 和 Spark 搭建的及时通信环境。

第 5 章在介绍加密/解密基本知识的前提下，向读者展示如何用 Java 的类库实现私钥加密/解密、公钥加密/解密、数字签名等技术。

第 6 章介绍 RMI 框架及 EJB 框架，让读者能够实现 RMI 和 CORBA 编程，能够建立基本的 EJB 和发布技术。

第 7 章让读者了解 OSGi 动态模块——Bundle 的基本结构，Bundle 之间如何调用以及如何实现 OSGi 的 Web 应用等。

书中实例程序都已调试通过，因而读者在上机实践时，不会出现不必要的困惑。

本书在编写过程中得到了清华大学出版社的大力支持，在此表示衷心的感谢！由于时间紧迫，本书难免有不妥之处，欢迎各界专家和读者朋友批评指正，也欢迎读者交流。本人的联系方式是 email-weiuser@hotmail.com。

编　者

2016 年 8 月

目 录

第 1 章　Java 开发环境及工具

随着项目规模的扩大、开发队伍人数的增加，项目的管理显得更为重要。例如在开发过程中项目成员如何实时地进行版本控制，如何更快地生成注册文档，如何将 Java 的运行包转换为操作系统下可直接运行的文件格式，如何有效地监控程序运行过程等。

本章将学习项目管理中一些带有 Java 开发特征的管理技术。

在 Java 的编写过程中需要对一些程序进行注释，除了自己阅读方便，也便于别人更好地理解自己的程序，所以需要增加一些注释，可以介绍编程思路或者程序的作用，以方便程序员更好地阅读。

用 JDK 提供的 javadoc 工具可自动生成注册文档，当程序修改时可及时更新生成的注释文档。本章将介绍如何使用 javadoc 读取源文件中的文档注释，并按照一定的规则与 Java 源程序一起进行编译，并最终生成文档。

作为跨平台的语言，用户更希望软件以操作系统直接运行的文件形式呈现，本章将介绍一些工具，能把 Java 提供的.jar 运行包转换为操作系统下直接运行的文件格式。

JMX(Java Management Extensions)是一个为应用程序、设备、系统等植入管理功能的框架。JMX 可以跨越一系列异构操作系统平台、系统体系结构和网络传输协议，灵活地开发无缝集成的系统、网络和服务管理应用。JMX 让程序有被管理的功能，如收到了多少数据，有多少人登录等。通过“配置”这个软件，在访问人数比较多时，可以把数据连接池设置得大一些。

SVN(Subversion)是一个开放源代码的版本控制系统，相对于 RCS、CVS，它采用了分支管理系统。SVN 取代了 CVS，互联网上很多版本控制服务已从 CVS 迁移到 SVN。

1.1　注释文档的生成

利用 JDK 的 javadoc 命令可以为类生成高质量的类似 API 的注释文档。

【实例】 代码如下：

```
package weiyong.demo.javadoc;
public class JavadocDemo {
   public final String message="This is a demo for java doc.";
   public static void main(String[] args) {
      JavadocDemo demo=new JavadocDemo();

      System.out.println(demo.message);
      System.out.println(demo.upcaseMessage());
```

```
        System.out.println(demo.getChars(2, 6));
    }

    public String upcaseMessage(){
        return message.toUpperCase();
    }

    public String getChars(int beginIndex, int endIndex){
        return message.substring(beginIndex, endIndex);
    }
}
```

操作要求：按照 javadoc 命令的语法要求，为 JavadocDemo 类加上注释，并生成注释文档。

1. 详细设计

在生成 javadoc 文档之前首先要对类进行注释。

注释有如下三种方式：

(1) //注释内容

(2) /*注释内容*/(选中要注释的内容后按“Shift+Ctrl+/”快捷键，可以添加注释；取消注释可以按“Shift+Ctrl+\”快捷键)

(3)

```
/**
*注释内容
*/
```

提示

只有第(3)种方式在使用 javadoc 命令生成文档时才有用。因为只有 public 类型才能被其他类访问，所以只需要对 public 方法和变量进行注释。下面举例说明。

① 在类前面添加类的说明和创作者，如：

```
/**
* 复制文件。将字符串输入到文件中，将文件内容输出到控制台中
*
* @author xiaoxu
*
*/
```

② 在 public 方法前面用注释来说明方法的作用，如对 public static String readFile(File file)方法的注释如下：

```
/**
* 读取一个文件中的内容
*
* @param file
* 要读的文件
* @return 返回读取文件的字符串
*/
```

③ 对 public 变量进行注释，假设有一个表示窗口宽度的变量 WIDTH，则注释如下：

```
/**
 * 主窗口的宽度
 */
public static final int WIDTH=1000;
```

2. 输出文字编码的实现

(1) 对类的注释

语句如下：

```
/**
 * 类的说明
 *
 * <p>
 * JavadocDemo 类演示如何生成类的 API 文档<br>
 *
 * @author weiyong
 * @version 1.0  weiyong 2013.11.12<br>
 * 1.1 water 2013.11.12 增加了说明
 */
```

分析：@anthor 声明作者，@version 声明版本。

(2) 方法的注释

语句如下：

```
/**
 * 从 message 中获取指定的子串
 *
 * @param beginIndex  子串开始的下标
 * @param endIndex  子串结束的下标
 * @return  从 beginIndex 到 endIndex 之间的子串
 */
```

分析：@param 说明方法的参数；@return 说明方法返回的值。

3. 源代码

```
package weiyong.demo.javadoc;
/**
 * 类的说明
 *
 * <p>
 * JavadocDemo 类演示如何生成类的 API 文档<br>
 *
 * @author weiyong
 * @version 1.0  weiyong 2013.11.12<br>
 * 1.1 water 2013.11.12 增加了说明
 */
public class JavadocDemo {
   /* 对于类公有的属性，也要写上相关的注释 */
   /** 用于显示一个提示信息 */
```

```
    public final String message="This is a demo for java doc.";
    /**
     *
     * 这个方法是程序的入口，虚拟机载入这个类的时候，
     * 将从这个方法开始运行程序
     *
     * @param args  命令行参数<br>
     */
    public static void main(String[] args) {
        JavadocDemo demo=new JavadocDemo();

        System.out.println(demo.message);
        System.out.println(demo.upcaseMessage());
        System.out.println(demo.getChars(2, 6));
    }

    /**
     * 将 message 转换成一个大写的字符串
     *
     * @return  转换成大写字串后的 message
     */
    public String upcaseMessage(){
        return message.toUpperCase();
    }

    /**
     * 从 message 中获取指定的子串
     *
     * @param beginIndex  子串开始的下标
     * @param endIndex  子串结束的下标
     * @return  从 beginIndex 到 endIndex 之间的子串
     */
    public String getChars(int beginIndex, int endIndex){
        return message.substring(beginIndex, endIndex);
    }
}
```

4. 测试与运行

因为 JavadocDemo 类属于 weiyong. demo. javadoc 包，所以先把 JavadocDemo. java 保存在工作目录的 weiyong\demo\javadoc 子目录下，如图 1-1 所示。

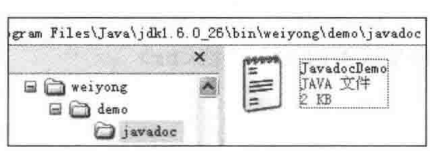


图 1-1　类文件所在目录

接着用 javadoc weiyong. demo. javadoc JavadocDemo. java 命令生成注释文档，如图 1-2 所示。

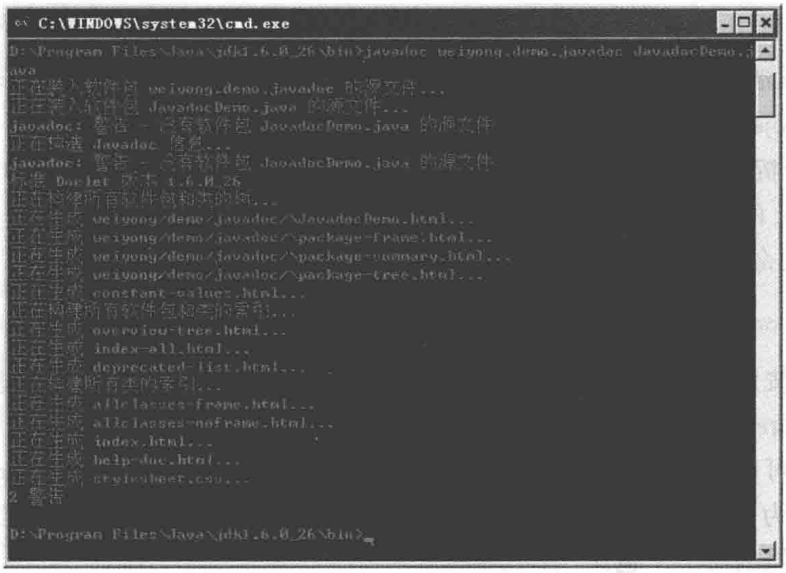


图 1-2　生成注释文档的过程

再看目录 weiyong\demo\javadoc 下的内容，发现多了 4 个文档，表明注释文档生成成功，如图 1-3 所示。

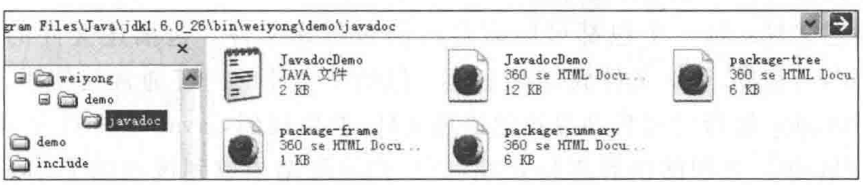


图 1-3　生成的注释文档

用浏览器打开 JavadocDemo. html 文件，效果如图 1-4 所示。

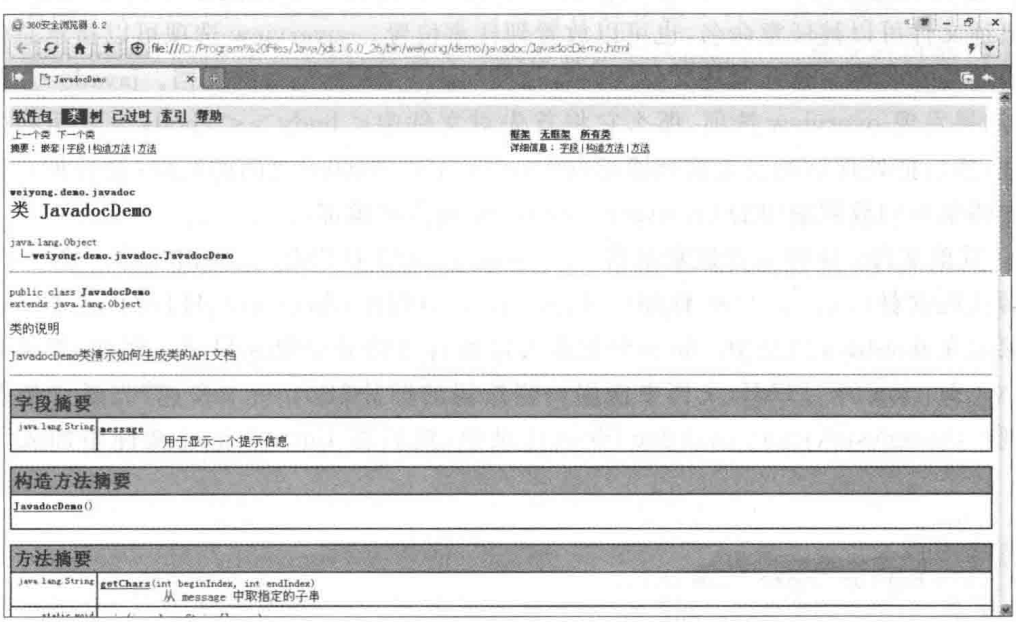


图 1-4　在浏览器中查看注释的文档

注意

用 Eclipse 很容易生成 javadoc 文档，方法如下：选中要生成 javadoc 文档的项目，再选择菜单中的 project→Generate Javadoc...→next→finish 命令。

5. 技术分析

1) Javadoc 的命令行语法

Javadoc 的命令行语法如下：

```
javadoc [ options ] [ packagenames ] [ sourcefiles ] [ @files ]
```

参数可以按照任意顺序排列。下面分别对这些参数和相关的一些内容进行说明。

(1) packagenames(包列表)

这个选项可以是一系列的包名(用空格隔开)，例如“java.lang　java.lang.reflect java.awt”。不过因为 javadoc 不递归作用于子包，不允许对包名使用通配符，所以必须显式地列出希望建立文档的每一个包。

(2) Sourcefiles(源文件列表)

这个选项可以是一系列的源文件名(用空格隔开)，可以使用通配符。javadoc 允许使用四种源文件：类源代码文件、包描述文件、总体概述文件、其他文件。

① 类源代码文件：类或者接口的源代码文件。

② 包描述文件：每一个包都可以有自己的包描述文件。包描述文件的名称必须是 package.html，与包的.java 文件放置在一起。包描述文件的内容通常是使用 HTML 标记写的文档。javadoc 运行时将自动寻找包描述文件，如果找到，javadoc 将首先对描述文件中＜body＞＜/body＞之间的内容进行处理，然后把处理结果放到该包的 Package Summary 页面中，最后把包描述文件的第一句(紧靠＜body＞)放到输出的 Overview summary 页面中，并在语句前面加上该包的包名。

③ 总体概述文件：javadoc 可以创建一个总体概述文件来描述整个应用或者所有包。总体概述文件可以被任意命名，也可以放置到任意位置。-overview 选项可以指示总体概述文件的路径和名称。总体概述文件的内容是使用 HTML 标记写的文档。javadoc 在执行的时候，如果发现-overview 选项，那么它将首先对文件中＜body＞＜/body＞之间的内容进行处理；然后把处理后的结果放到输出的 Overview summary 页面的底部；最后把总体概述文件中的第一句放到输出的 Overview summary 页面的顶部。

④ 其他文件：这些文件通常是指与 javadoc 输出的 HTML 文件相关的一些图片文件、Java 源代码文件(.java)、Java 程序(.class)、Java 小程序(Applets)、HTML 文件。这些文件必须放在 doc-files 目录中。每一个包都可以有自己的 doc-files 目录。例如，如果希望在 java.awt.Button 的 HTML 文档中使用一幅按钮的图片(Button.gif)，首先必须把图片文件放到 C:\user\src\java\awt\doc-files\目录中，然后在 Button.java 文件中加入下面的注释。

```
/**
* This button looks like this:
* <img src="doc-files/Button.gif">
*/
```

(3) @files(包含文件)

为了简化 javadoc 命令,可以把需要建立文档的文件名和包名放在一个或多个文本文件中。例如,为了简化下面的命令:

```
javadoc -d apidoc com.mypackage1 com.mypackage2 com.mypackage3
```

可以建立一个名称为 mypackage.txt 的文件,其内容如下:

```
com.mypackage1
com.mypackage2
com.mypackage3
```

然后执行下面的命令即可:

```
javadoc -d apidoc @mypackage.txt
```

(4) options(命令行选项)

javadoc 使用 doclets(doclets 是指用 doclet API 编写的程序)来确定输出的内容和格式。命令行选项中一部分是可用于所有 doclet 的通用选项,还有一部分是由默认的标准 doclet 提供的专用选项。下面对一些常用的选项进行介绍。

① 通用选项。

- -1.1　生成与 javadoc 1.1 版本生成的外观和功能一样的文档。不是所有的参数都可以用于-1.1 选项,具体可以使用 javadoc -1.1 -help 命令查看。
- -help　显示联机帮助。
- -bootclasspath classpathlist　指定"根类"(通常是 Java 平台自带的一些类。例如 java.awt.* 等)的路径。
- -sourcepath sourcepathlist　指定包的源文件搜索路径。但是必须注意,只有在 javadoc 命令中指定了包名的时候才可以使用-sourcepath 选项。如果指定了包名而省略了-sourcepath,那么 javadoc 使用类路径查找源文件。举例说明:假定打算为 com.mypackage 建立文档,其源文件的位置是 C:\user\src,那么可以使用下面的命令:

  ```
  javadoc -sourcepath c:\user\src com.mypackage
  ```

- -classpath classpathlist　指定用 javadoc 查找"引用类"的路径。引用类是指带文档的类加上它们引用的任何类。javadoc 将搜索指定路径的所有子目录。Classpathlist 可以包含多个路径(用";"隔开)。如果省略-classpath 选项,则 javadoc 使用-sourcepath 选项查找源文件和类文件。例如,假定打算为 com.mypackage 建立文档,其源文件的位置是 C:\user\src,包依赖于 C:\user\lib 中的库,那么可以使用下面的命令:

  ```
  javadoc -classpath c:\user\lib -sourcepath c:\user\src com.mypackage
  ```

- -overview path\filename　告诉 javadoc 从 path\filename 所指定的文件中获取概述文档,并且把它放到输出的概述页面(overview-summary.html)中。其中 path\filename 是相对于-sourcepath 的路径。

- -public　只显示公共类以及成员。
- -protected　只显示受保护的公共类以及成员。这也是默认选项。
- -package　只显示包、受保护的和公共的类以及成员。
- -private　显示所有类和成员。
- -doclet class　指定 javadoc 产生输出内容的自定义 doclet 类。如果忽略这个选项，javadoc 将使用默认的 doclet 产生一系列的 HTML 文档。
- -docletpath classpathlist　与-doclet 选项相关，指定自定义的 doclet 类文件的路径。classpathlist 可以包含多条路径(用分号隔开)。
- -verbose　在 javadoc 运行时提供更详细的信息。

② 标准 doclet 专用选项。

- -author　在生成的文档中包含“作者”项。
- -d directory　指定 javadoc 生成的 HTML 文件的保存目录。省略该选项，将把文件保存在当前目录。directory 可以是绝对目录，也可以是相对当前目录的目录。
- -version　在生成的文档中包含“版本”项。
- -use　为类和包生成 use(用法)页面。这些页面描述了该类和包在 javadoc 命令涉及的文件中被使用的情况。例如，对于给定的类 C，在 C 的用法页面中将包含 C 的子类，类型为 C 的域，返回变量类型为 C 的方法以及在参数中有变量类型为 C 的方法和构造器。
- -splitindex　把索引文件按照字母顺序分为多个文件，每一个文件对应一个字母。
- -windowtitle title　指定输出的 HTML 文档的标题。
- -header header　指定输出的 HTML 文档的页眉文本。
- -footer footer　指定输出的 HTML 文档的脚注文本。
- -bottom text　指定输出的 HTML 文档底部的文本。
- -group groupheading packagepatten;packagepatten;...在总体概述页面中按照命令的指定方式分隔各个包。例如执行下面的命令：

```
javadoc -group "Core Packages" "java.lang* :java.util"
-group "Extension Packages" "javax.* "
java.lang java.lang.reflect java.util javax.servlet java.new
```

在页面中将有如下结果：

```
Core Packages
java.lang
java.lang.reflect
java.util
Extension Packages
javax.servlet
Other Packages
java.new
```

- -noindex　不输出索引文件。
- -help　在文件的导航条中忽略 help 链接。
- -helpfile path\filename　指定导航条中的 help 链接所指向的帮助文件。忽略该选

项，javadoc 将生成默认的帮助文件。

- -stylesheetfile path\filename　指定 javadoc 的 HTML 样式表文件的路径。忽略该选项，javadoc 将自动产生一个样式表文件 stylesheet. css。

通过上面的介绍，大家基本了解了 javadoc 的命令行语法。下面介绍 javadoc 文档的注释方法。

2）javadoc 注释

javadoc 注释以“/**”开始，以“*/”结束，里面可以包含普通文本、HTML 标记和 javadoc 标记。javadoc 只处理源文件中在类/接口定义、方法、域、构造器之前的注释，忽略位于其他地方的注释。举例如下：

```
/**
* 第一个程序--<b>Helloworld</b>
* @author 张军
* @version 1.0 2014/10/15
*/
public class myHelloworld{
/**
* 在 main()方法中使用的用于显示字符串的方法
* @see #main(java.lang.String[])
*/
static String SDisp
```

使用下面的命令即可以生成关于 myHelloworld. java 的 API 文档：

```
javadoc -private -d doc -author -version myHelloworld.java
```

上面例子中以@开头的标记就是 javadoc 标记。在 Java 程序中正确使用 javadoc 标记是一个良好的注释习惯，将非常有助于 javadoc 自动从源代码文件生成完整的格式化 API 文档。下面就对各种标记进行详细说明。

（1）@author name-text　指定生成文档中的“作者”项。从 JDK/SDK 1. 0 开始引入。name-text 可以指定多个名字（使用“,”隔开）。文档注释可以包含多个类。

（2）{@docroot}　代表产生文档的根路径。从 JDK/SDK 1. 3 开始引入。用法举例如下：

```
/**
* see the <a href={@docroot}/copyright.html>copyright</a>
*/
```

假定生成文档的根目录是 doc，上面注释所在的文件最后生成的文件是 doc\utility\utl. html，那么 copyright 的链接会指向“..\copyright. html”。

（3）@deprecated deprecated-text　添加注释，表明不推荐使用该 API。

（4）@exception class-name description　@throw 的同义标记。从 JDK/SDK 1. 0 开始引入。

（5）{@link package. class # member label}　插入指向 package. class # member 的内嵌链接。从 JDK/SDK 1. 2 开始引入。例如，假定注释中有如下文档：

```
/** Use the {@link #getComponentAt(int, int) getComponentAt} method. */
```

那么 javadoc 最终生成的 HTML 页面中将有如下内容：

```
Use the <a href="Component.html#getComponentAt(int,int)"
>getComponentAt </a>method.
```

(6) @param parameter-name description　描述参数。从 JDK/SDK 1.0 开始引入。

(7) @return description　描述返回值。从 JDK/SDK 1.0 开始引入。

(8) @see reference　添加“引用”标题，其中有指向 reference 的链接或者文本项，从 JDK/SDK 1.0 开始引入。@see 标记有三种形式，下面分别说明。

① @see "string"　为 string 添加文本项，不产生链接。

② @see Label　使用 HTML 标记产生链接。

③ @see package　添加包标题。

3) 创建自己的类库

方法一：假设现在要创建一个项目 Test，里面要用到自定义类库 tools.jar 中的 S 类。新建一个类 Test.java，里面的 main 方法如下：

```
public static void main(String[] args) {
  S.pl("java");  //S 类中的 S.pl()封装了 System.out.println(obj)
}
```

这时 Eclipse 会报错，会提示 S 这个类不存在，问用户是否要创建这个类。这是因为默认的 JRE 系统库中并不存在 tools.jar 包，如图 1-5 所示。

如何将这个包安装到默认的 JRE 系统库中呢？

选择 Window→Preferences 命令，会打开一个对话框，选择 Java→Installed JREs，如图 1-6 所示。

图 1-5　默认包

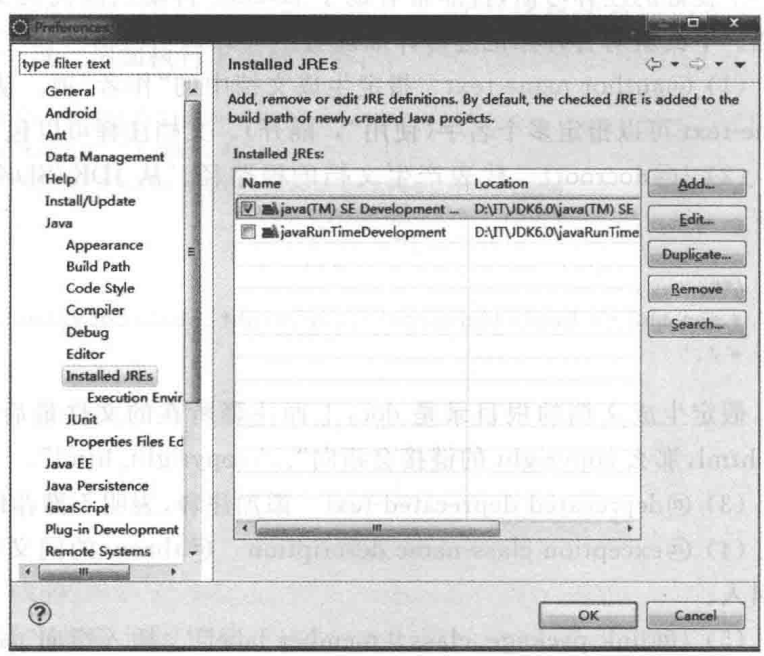


图 1-6　Preferences 对话框