JDK 1.5类库大全

陈烨 张蓓 等编著

- Java语言及JDK简介
- Java虚拟机和java.lang包
- Collection框架
- Number及其子类
- 字符及字符串处理
- 输入输出流
- 数学运算工具类
- 正则表达式
- ZIP压缩工具和Java归档工具
- 时间日期工具和日志工具
- 属性配置工具
- Java反射机制
- XML

清华大学出版社

JDK 1.5 类库大全

陈 烨 张 蓓 等编著

清华大学出版社

北 京

内 容 简 介

本书从实用的角度出发，系统地介绍了 JDK 1.5 中各种实用类，尤其是新增类的结构和使用方法。全书由 19 章组成，主要内容包括 Java 语言及 JDK 简介、Java 虚拟机、Java.lang 包、Collection 框架、Number 及其子类、字符及字符串处理、输入输出流、数学运算工具类、正则表达式、ZIP 压缩工具、Java 归档工具、时间日期工具、日志工具、属性文件工具、Java 反射机制、网络、XML 和 JDK 1.5 编程实践等。

本书内容丰富，从各个方面介绍了 JDK 1.5 中主要包和类的使用方法。在对类的 API 进行讲解时，结合了大量的实例，使读者能够快速掌握各个类的使用方法。

本书既可以作为 Java 初学者学习 Java 语言的教材，也可以作为专业程序员进行程序开发的参考书。

版权所有，翻印必究。举报电话：010-62782989　13501256678　13801310933
本书封面贴有清华大学出版社防伪标签，无标签者不得销售。
本书防伪标签采用特殊防伪技术，用户可通过在图案表面涂抹清水，图案消失，水干后图案复现；或将表面膜揭下，放在白纸上用彩笔涂抹，图案在白纸上再现的方法识别真伪。

图书在版编目(CIP)数据

JDK 1.5 类库大全/陈烨，张蓓等编著. —北京：清华大学出版社，2005.5

ISBN 7-302-10085-3

Ⅰ.J…　Ⅱ.①陈…②张…　Ⅲ.JAVA 语言—程序设计　Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2004)第 130006 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　　**客户服务**：010-62776969
组稿编辑：孟毅新　　**文稿编辑**：鲍　芳
封面设计：久久度文化　　**版式设计**：康　博
印 装 者：北京鑫霸印务有限公司
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：36.75　**字数**：940 千字
版　　次：2005 年 5 月第 1 版　2005 年 5 月第 1 次印刷
书　　号：ISBN 7-302-10085-3/TP・6905
印　　数：1～3000
定　　价：58.00 元

前　言

Java 语言是 Sun 公司开发的新一代面向对象的编程语言，具有简单性、面向对象、分布性、健壮性、安全性、体系结构中立、可移植性、解释执行、高性能、多线程和动态性等特点，并提供了并发机制，具有很高的性能。其体系结构中立和可移植性的重要性在于 Java 解释器生成与体系结构无关的字节码指令，只要安装了 Java 运行时系统，Java 程序就可在任意处理器上运行。这些字节码指令对应于 Java 虚拟机中的表示，Java 解释器得到字节码后，对它进行转换，使之能够在不同的平台运行。

Sun 公司在推出 Java 语言的同时，也推出了 Java 的一系列开发工具，如 JDK（Java Developer's Kit）。JDK 是可以从网上免费下载的 Java 开发工具集。

2004 年 Sun 公司发布了 Java 2 平台标准版(J2SE)5.0 版，这是一个快速开发和配置跨平台的企业级应用与服务的综合性平台。Java 2 平台标准版的这一最新版本提升了 Java 计算的性能与缩放能力，代表了 Java 技术的大跨步进展。有了 J2SE5.0 版，企业就可以通过更简易的步骤、花费更少的时间，采用 Java 技术开发与配置要求更高的应用。在 2004 年，Sun 计划推出 JDK 的最新版本 1.5 版，代号为 Tiger。过去的 J2SE 版本主要关注新类和性能，而 Tiger 的目标则是通过使 Java 编程更易于理解、对开发人员更为友好、更安全来增强 Java 语言本身，同时最大限度地降低与现有程序的不兼容性。

本书系统地介绍了使用 JDK 1.5 进行程序开发的方法，着重讲解了 JDK1.5 中各个实用类库。在讲解的同时是还结合了大量 API 使用示例和综合实例，使读者能够快速掌握各个类库的使用方法。

全书由 19 章组成，主要内容包括：Java 语言及 JDK 简介、Java 虚拟机、Java.lang 包、Collection 框架、Number 及其子类、字符及字符串处理、输入输出流、数学运算工具类、规则表达式、ZIP 压缩工具、Java 归档工具、时间日期工具、日志工具、属性文件工具、Java 反射机制、网络、XML 和 JDK 1.5 编程实践。

通过本书的学习，读者不仅可以轻松掌握 Java 语言，而且能够熟练使用 JDK1.5 中提供的类库进行 Java 程序开发，从而极大提高开发效率。

本书既可以作为 Java 初学者学习 Java 语言的教材，也可以作为专业程序员进行程序开发的参考书。

本书由陈烨和张蓓共同执笔编写。此外，蓝荣香、王昊亮、喻波、马天一、魏勇、郝荣福、孙明、李大宇、武思宇、牟博超、李彬、付鹏程、高翔、朱丽云、崔凌、张巧玲、李辉、李欣、柏宇、郭强、金春范、程梅、黄霆、钟华、高海峰、王建胜、张浩、刘湘和邵蕴秋等同志在整理材料方面给予了编者很大的帮助，在此，编者对他们表示衷心的感谢。

由于时间仓促，再加上编者水平有限，书中不足之处希望广大读者不吝赐教并提出宝贵意见。

编　者

目　录

试读结束：需要全本请在线购买：　www.ertongbook.com

第1章 JDK 1.5概述

本章主要介绍 Java 语言的特点、JDK 的发展历史和 JDK 最新版本 1.5 版的新增功能，使读者在使用本书前对 JDK 有一个大致的了解。另外，为方便读者的使用，本章还简单介绍了 JDK 1.5 的安装方法。

1.1 Java 语言简介

Java 是一种广泛使用的网络编程语言，它是一种新的计算概念。

首先，作为一种程序设计语言，它简单、面向对象、不依赖于机器的体系结构、具有可移植性、鲁棒性(健壮性)、安全性等，并且提供了并发的机制、具有很高的性能。其次，它最大限度地利用了网络，Java 的小应用程序(applet)可在网络上运行而不受 CPU 和环境的限制。另外，Java 还提供了丰富的类库，使程序设计者可以很方便地建立自己的系统。

(1) 简单性

Java 语言是一种面向对象的语言，它通过提供最基本的方法来完成指定的任务，只需理解一些基本的概念，就可以用它编写出适合于各种情况的应用程序。Java 略去了运算符重载、多重继承等模糊的概念，并且通过实现自动垃圾回收，极大简化了程序设计者的内存管理工作。另外，Java 也适合于在小型机上运行，它的基本解释器及类的支持只有 40KB 左右，加上标准类库和线程的支持也只有 215KB 左右。

(2) 面向对象

Java 语言的设计集中于对象及其接口，它提供了简单的类机制及动态的接口模型。对象中封装了它的状态变量及相应的方法，实现了模块化和信息隐藏；而类则提供了一类对象的原型，并且通过继承机制，子类可以使用父类所提供的方法，实现代码的复用。

(3) 分布性

Java 是面向网络的语言。通过它提供的类库可以处理 TCP/IP 协议，用户可以通过 URL 地址在网络上很方便地访问其他对象。

(4) 鲁棒性(健壮性)

Java 在编译和运行程序时，都要对可能出现的问题进行检查，以消除错误的产生。它提供自动垃圾收集来进行内存管理，以防止程序员在管理内存时容易产生的错误。通过集成的面向对象的异常处理机制，在编译时，Java 揭示出可能出现但未被处理的异常，帮助程序员正确地进行选择以防止系统崩溃。另外，Java 在编译时还可捕获类型声明中的许多常见错误，防止出现动态运行时不匹配的问题。

(5) 安全性

用于网络、分布环境下的 Java 必须要防止病毒的入侵。Java 不支持指针，一切对内存的访问都必须通过对象的实例变量来实现，这样就防止程序员使用“特洛伊”木马等欺骗手段访问对象的私有成员，同时也避免了指针操作中容易产生的错误。

(6) 体系结构中立

Java 解释器生成与体系结构无关的字节码指令，只要安装了 Java 运行时系统，Java 程序就可在任意的处理器上运行。这些字节码指令对应于 Java 虚拟机中的表示，Java 解释器得到字节码后，对它进行转换，使之能够在不同的平台运行。

(7) 可移植性

与平台无关的特性使 Java 程序可以方便地被移植到网络上的不同机器。同时，Java 的类库中也实现了与不同平台的接口，使这些类库可以移植。另外，Java 编译器是由 Java 语言实现的，Java 运行时系统由标准 C 实现，这使得 Java 系统本身也具有可移植性。

(8) 解释执行

Java 解释器直接对 Java 字节码进行解释执行。字节码本身携带了许多编译时信息，使得连接过程更加简单。

(9) 高性能

和其他解释执行的语言如 BASIC、TCL 不同，Java 字节码的设计使之能很容易地直接转换成对应于特定 CPU 的机器码，从而得到较高的性能。

(10) 多线程

多线程机制使应用程序能够并行执行，而且同步机制保证了对共享数据的正确操作。通过使用多线程，程序设计者可以分别用不同的线程完成特定的行为，而不需要采用全局的事件循环机制，这样就很容易地实现网络上的实时交互行为。

(11) 动态性

Java 的设计使它适合于一个不断发展的环境。在类库中可以自由地加入新的方法和实例变量而不会影响用户程序的执行。并且 Java 通过接口来支持多重继承，使之比严格的类继承具有更灵活的方式和扩展性。

1.2 Java 开发环境——JDK 介绍

Sun 公司在推出 Java 语言的同时，也推出了 Java 的一系列开发工具，如 JDK(Java Developer's Kit)。JDK 是可以从网上免费下载的 Java 开发工具集。通常以 JDK 的版本来定义 Java 的版本。JDK 1.0 版于 1996 年初发布，JDK 1.1 版于 1997 年初发布，JDK 1.2 版于 1998 年底发布。Sun 在 JDK 1.2 版发布后将 Java 改名为 Java 2，将 JDK 改名为 Java 2 Software Development Kit(为了方便起见，本书仍将其简称为 JDK)。JDK 1.3 于 2000 年 4 月发布。

2004 年 Sun 公司发布了 Java 2 平台标准版(J2SE)5.0 版，这是一个快速开发和配置跨平台的企业级应用与服务的综合性平台。Java 2 平台标准版的这一最新版本提升了 Java 计算的性能与缩放能力，代表了 Java 技术的大跨步进展。有了 J2SE 5.0 版，企业就可以通过更简易的步骤、花费更少的时间，采用 Java 技术开发与配置要求更高的应用。

在 2004 年，Sun 推出了 JDK 的最新版本 1.5 版，代号为 Tiger。过去的 J2SE 版本主要关注新类和性能，而 Tiger 的目标则是通过使 Java 编程更易于理解、对开发人员更为友好、更安全来增强 Java 语言本身，同时最大限度地降低与现有程序的不兼容性。

1.3 JDK 1.5 新增特性概述

为使编程更加方便，JDK 1.5 中增加了如下特性。

(1) 通过泛型(generics)来改进类型检查

generics 使用户能够指定一个集合中使用的对象的实际类型，而不是像过去那样只是使用 Object。generics 也称为“参数化类型”，因为在 generics 中，一个类的类型接受影响其行为的类型变量。

(2) 自动装箱/拆箱

Java 有一个带有原始类型和对象(引用)类型的分割类型系统。原始类型被认为是更轻便的，因为它们没有对象开销。新的 autoboxing 特性使编译器能够根据需要隐式地从 int 转换为 Integer，从 char 转换为 Character 等；auto-unboxing 则进行相反的操作。

(3) 增强的 for 循环

试读结束：需要全本请在线购买：www.ertongbook.com

JDK 1.5 中为 for 循环引入了新的语法，使得代码更加简便、易读。

(4) 类型安全的枚举类型

枚举类型(enums)是定义具有某些命名的常量值的类型的一种方式。熟习 C 和 C++ 的读者对于枚举类型不会陌生，但在 JDK 1.5 之前的版本中并没有引入该类型。在 JDK 1.5 中，将会添加枚举类型，方便开发人员的使用。

(5) 静态导入

静态导入使得用户可以将一套静态方法和域放入作用域(scope)。它是关于调用的一种缩写，可以忽略有效的类名。

1.4　JDK 1.5 的安装

JDK 1.5 可以在 SUN 公司的网站上免费下载，下载地址为：

http://java.sun.com

下面以 Windows 平台的安装为例介绍 JDK 1.5 的安装过程，其他平台的安装过程大致相同。

双击安装文件，文件开始自解压，解压完成后出现安装的首界面，如图 1-1 所示。

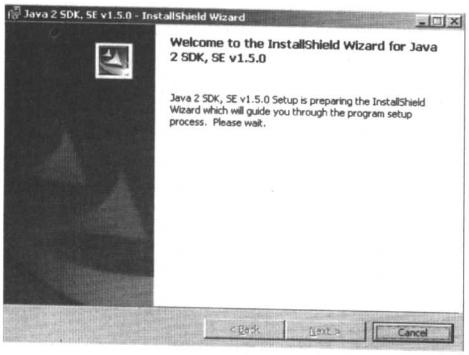

图 1-1　安装首界面

单击 Next 按钮可进入下一步，如图 1-2 所示。该界面中描述了 JDK 1.5 的权限许可信息，接受相关的授权后，可以单击 Next 按钮进入下一步，如图 1-3 所示。

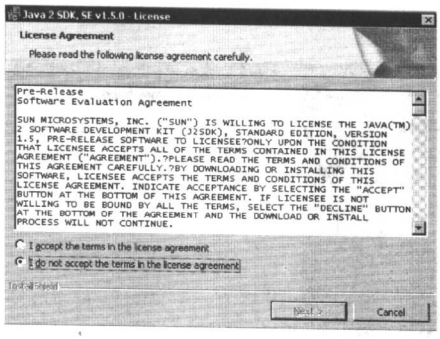

图 1-2　权限许可

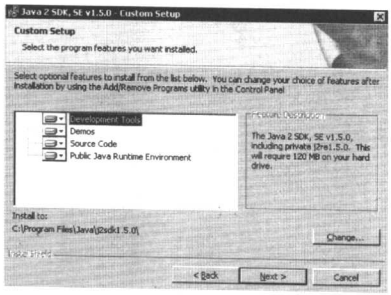

图 1-3　选择安装路径

在该界面中可以选择 JDK 1.5 的安装路径，单击 Change 按钮可以选择其他的安装路径。

在该界面中用户还可以选择需要的安装组件，包括程序文件(必需)、本地接口头文件、示例、Java 源码和 Java 2 运行时环境。选择好相应组件后，单击 Next 按钮进入下一步，如图 1-4 所示。

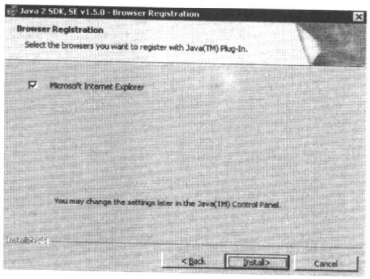

图 1-4　选择浏览器

在该界面中，可以选择在什么浏览器中安装 Java 运行时环境。选择好浏览器后，单击 Install 按钮可进入 JDK 1.5 的安装，完毕后，单击 Finish 按钮可完成 JDK 1.5 的安装，如图

1-5 所示。

图 1-5　完成安装

第2章 Java虚拟机

Java 虚拟机(JVM)是 Java 平台无关的基础，在 JVM 上，有一个 Java 解释器用来解释 Java 编译器编译后的程序。Java 编程人员在编写完软件后，通过 Java 编译器将 Java 源程序编译为 JVM 的字节代码。本章主要介绍了 Java 2 SDK 中包含的 Java 虚拟机。

2.1 Java 2 SDK 中的 JVM

在 Java 2 SDK 标准版中，包含了两类 Java 虚拟机(VM)的实现。

(1) Java HotSpot 客户端虚拟机

Java HotSpot 客户端虚拟机是 Java 2 SDK 和 Java 2 运行时环境的默认虚拟机。从该虚拟机的名称就可以看出，当虚拟机在客户端环境下，可以通过减少应用程序的启动时间和内存使用量，使得运行的应用程序获得最佳的性能。

(2) Java HotSpot 服务器虚拟机

Java HotSpot 服务器虚拟机是为运行在服务器环境的应用程序设计的，用来获取应用程序的最快运行速度。Java HotSpot 服务器虚拟机可以在启动一个应用程序时在命令行通过 -server 命令调用，如：

```
java -server MyApp
```

通过使用 Java HotSpot 技术，在两个虚拟机中均实现了如下的特性。

- 自适应编译器

在这两个虚拟机中，应用程序由标准的解释器来启动，但代码在运行期间会被分析，用以检测出性能的瓶颈。Java HotSpot 虚拟机会编译这些性能瓶颈点来推进性能的提升，同时，虚拟机也会避免编译那些很少使用到的代码。在空闲时，Java HotSpot 虚拟机还会利用自适应编译器，使用 in-lining 等技术来决定如何对代码进行编译以获取最佳的性能。自适应编译器的运行时分析功能可以估算哪一种优化方法能够达到最好的优化效果。

- 快速内存分配和垃圾收集

Java HotSpot 技术为对象提供了一个快速的内存分配机制，另外还提供了一种快速、高效的垃圾收集器。

- 线程同步

Java 程序语言允许用户进行程序的多线程、并发执行。Java HotSpot 技术提供了一种线程同步机制，该机制可以使用户在大规模、分享内存的多处理器服务器上十分方便地实现线程同步。

2.2 命令行选项

通过 java 命令行工具可以使用一些 Java HotSpot 虚拟机提供的新功能，下面介绍这些功能在 Windows 环境下的使用方式，其他平台的使用方式大致相同。

2.2.1 基本用法

java 命令行工具的语法如下：

```
java [ options ] class [ argument ... ]
java [ options ] -jar file.jar [ argument ... ]
javaw [ options ] class [ argument ... ]
javaw [ options ] -jar file.jar [ argument ... ]
```

java 命令行工具用来启动一个 Java 应用程序，该工具的工作流程是启动 Java 运行时环境、加载指定的类、最后调用相关类的 main 方法。main 方法必须由如下方式定义：

```
public static void main(String args[])
```

该方法必须声明为 public 和 static，不能有返回值，必须接受一个 String 类型的数组作为参数。默认条件下，第一个必需的参数是调用的类的名称。如果指明了-jar 选项，则第一个必需的参数为包含源文件和类的 JAR 包的名称。

javaw 命令与 java 命令的使用方法相同。它们之间的区别在于使用 javaw 不会弹出一个命令行窗口。如果程序启动时出现错误，使用 javaw 也会弹出一个对话框用于显示相应的错误信息。

在启动时可以设定一系列标准的选项(options)，这些选项是当前运行时环境支持的，并且在今后的版本中也会支持。另外，Java HotSpot 虚拟机还提供了一些非标准的选项，这些选项在今后的版本中可能发生变化。

2.2.2 标准选项

标准的选项(options) 包括如下几种。

- -client

选择 Java HotSpot 客户端虚拟机，默认使用的就是该虚拟机。

- -server

使用 Java HotSpot 服务器端虚拟机。

- -cp classpath

指明一系列的目录、JAR 归档文件或 ZIP 归档文件为类的查找路径。各文件间用“;”间隔。注意使用-cp 会忽略原先设置的 CLASSPATH 环境变量。

- -Dproperty=value

设置系统属性的值。如果属性值中包含空格，则必须用引号括起来，例如：

```
java -Dfoo="some string" SomeClass
```

- -ea[:<package name>"..." | :<class name>]

可以使用断言(默认为不能使用)，例如：

```
java -ea:com.wombat.fruitbat... <Main Class>
```

- da[:<package name>"..." | :<class name>]

不使用断言(默认)。如果程序需要在包 com.wombat.fruitbat 中使用断言，但在类 com.wombat.fruitbat.Birickbat 中不使用断言，则可在命令行中输入：

```
java -ea:com.wombat.fruitbat...
-da:com.wombat.fruitbat.Brickbat lt;Main Class>
```

- -esa

在所有的系统类中使用断言。

- -dsa

在所有的系统类中屏蔽断言。

- -verbose:class

显示每一个类的加载信息。

- -verbose:gc

显示每一个垃圾收集事件的信息。

- -verbose:jni

显示使用本地方法和 JNI 行为的信息。

- -version

显示版本信息并退出。

- -showversion

显示版本信息并继续运行。

- -X

显示非标准的选项并退出。

2.2.3 非标准选项

非标准的选项包括如下几种。

- -Xint

仅以解释器模式运行，此时不能编译本地代码，所有的字节码将由解释器执行。

- -Xbatch

不以后台方式编译。默认情况下虚拟机会以后台任务方式进行编译，使用该命令可以强制系统使用前台任务方式进行编译，直至编译

结束。

- -Xdebug

使用调试器开始。

- -Xnoclassgc

不使用类垃圾收集。

- -Xincgc

使用增强的垃圾收集。

- -Xloggc:file

报告每次垃圾收集事件，并将信息记录在 file 指定的文件中。

- -Xmsn

指定初始化内存池的大小(默认值为 2MB)，以字节为单位，其值必须是大于 1MB 的 1024 的整数倍。例如：

```
-Xms6291456
-Xms6144k
-Xms6m
```

- -Xmxn

以字节为单位指明内存池的最大值(默认值为 64MB)。例如：

```
-Xmx83886080
-Xmx81920k
-Xmx80m
```

- -Xssn

设置线程堆栈的大小。

- -Xprof

剖析运行的程序，并将剖析数据发送到标准输出设备。

第3章　java.lang包

java.lang 包是 JDK 提供的最基本的开发包，其中包含了许多 Java 程序中需要使用到的基本类和接口。如果要进行 Java 程序的开发，就不可避免地要使用到 java.lang 包。

3.1　简介

在 Java 基本类中，最主要的类是 Object，该类是其他所有类的父类，处于 Java 类层次结构的顶层。

在 Java 语言中有一些通用的接口，如 Cloneable、Comparable 等。其他的类通过实现这些接口可以实现一些额外的功能。

在 java.lang 包中还提供了其他一些类：ClassLoader 类、Runtime 类、System 类提供了一些系统级操作，例如动态类加载、获取系统信息等。

3.2　基本接口

java.lang 包中包含了一些基本的接口，Java 类通过继承这些接口可以获取额外的功能。

3.2.1　java.lang.Cloneable

java.lang.Cloneable 接口是一个空接口，该接口用来指明一个对象是否可以进行克隆。实现了该接口的对象可以调用 clone()方法来进行对象的浅克隆。如果对象没有实现该接口，则调用 clone()方法时会抛出 CloneNotSupportedException 异常。

通常情况下，当一个类需要进行克隆操作时，应该覆盖 Object.clone()方法。

Cloneable 接口的类图如图 3-1 所示。

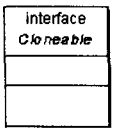

图 3-1　java.lang.Cloneable 接口类图

下面是一个实现 Java 浅克隆的例子：

```
import java.util.Vector;
public class Example   implements Cloneable{
   private Vector vector=new Vector();
   private String name;
   public void setName(String name){
      this.name=name;
   }
   public String getName(){
      return name;
   }
   public void add(String s){
      vector.add(s);
   }
   public int getSiza(){
      return vector.size();
   }
   public Object clone(){
      Example temp = null;
      try{
                temp = (Example) super.clone();
                return temp;
      }
      catch(CloneNotSupportedException e){
```

试读结束：需要全本请在线购买：　www.ertongbook.com