特色规划

免费赠送
电子课件

高等学校应用型特色规划教材

Java
网络编程技术

刘永华　于春花　李晓利　主　编

清华大学出版社

高等学校应用型特色规划教材

Java 网络编程技术

刘永华　于春花　李晓利　主　编

清华大学出版社
北　京

内 容 简 介

本书以 Java 语言为基础，比较深入地介绍了 Java 网络编程技术。内容包括 Java 与数据库的连接、JSP 技术、查找 Internet 地址、用 URL 检索数据、Socket 编程、收发 E-mail、Servlet 编程等。在内容选取上以基础、实用、够用为原则，并注重培养读者的编程能力。

本书可作为应用型本科计算机及相关专业“Java 网络编程技术”、“Java 网站开发技术”等课程的教材。也可供有一定 Java 语言程序设计基础，需进一步学习 Java 网络编程技术的人员自学，或供相关领域的工程技术人员作为参考书使用。

图书在版编目(CIP)数据

Java 网络编程技术/刘永华，于春花，李晓利主编. —北京：清华大学出版社，2008.5
(高等学校应用型特色规划教材)
ISBN 978-7-302-17385-4

Ⅰ. JAVA…　Ⅱ. ①刘…　②于…　③李…　Ⅲ. JAVA 语言—程序设计—高等学校—教材　Ⅳ. TP312

中国版本图书馆 CIP 数据核字(2008)第 050651 号

责任编辑：刘建龙　宋延清
封面设计：杨玉兰
版式设计：北京东方人华科技有限公司
责任印制：何　芊
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**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
印 装 者：北京宏伟双华印刷有限公司
经　　销：全国新华书店
开　　本：185×260　**印　张**：14.75　**字　数**：351 千字
版　　次：2008 年 5 月第 1 版　　**印　　次**：2008 年 5 月第1 次印刷
印　　数：1～4000
定　　价：23.00 元

前　言

Java 是由美国 Sun 公司开发的面向对象的、具有可移植性的、功能强大的多线程计算机编程语言。用 Java 编写的程序可以在各种类型的计算机和操作系统上运行。因此 Java 非常适合于企业网络和 Internet 网络环境，已经成为 Internet 中最受欢迎、最有影响力的编程语言之一。

本书以深入提高 Java 网络编程技术为目的，旨在介绍和解释网络技术的基本概念，并讨论 Java 网络编程实践技巧。通过本书的阅读与学习，读者可以提高 Java 网络编程的能力。如果读者已经拥有另一种语言的某些编写网络程序的经验，那么将发现那些编程经验也可以应用在 Java 中。本书内容广泛且通俗易懂，全书循序渐进、由浅入深地引导读者逐步掌握 Java 网络编程技术。

本书共分 10 章，各章的主要内容说明如下。

- 第 1 章：Java 基础。介绍 Java 技术和 Java 环境的设置。
- 第 2 章：Java 进阶。介绍 Java 流处理、线程和异常处理的相关知识。
- 第 3 章：Java 与数据库的连接。介绍如何使用 JDBC 实现与数据库的连接。
- 第 4 章：JSP 技术。介绍 JSP 的基本概念，并通过示例帮助读者较快地理解。
- 第 5 章：查找 Internet 地址。介绍 InternetAddress 类及其应用。
- 第 6 章：用 URL 检索数据。介绍如何使用 URL/URLConnection 类检索数据。
- 第 7 章：基于 TCP/IP 协议的 Socket 编程。介绍 TCP/IP 协议、套接字、Socket 类和 ServerSocket 类及其应用。
- 第 8 章：基于 UDP 协议的 Socket 编程。介绍 UDP 协议、DatagramPacket 类和 DatagramSocket 类及其应用。
- 第 9 章：用 Java 收发 E-mail。介绍常用邮件协议、JavaMail 基础和收发 E-mail 的实现。
- 第 10 章：Servlet 编程。包括 Servlet 概述和 Servlet 编程。

本书是一本全面介绍 Java 网络编程技术的实用书籍。本书的特色是以示例为载体来介绍 Java 网络编程技术，每节内容均辅以示例和说明，将方法和知识融汇到示例之中，使读者进一步理解和掌握理论知识。

全书由刘永华、于春花、李晓利主编并统稿。其中刘永华完成了第 2、9、10 章，于春花完成了第 5、6、7、8 章，李晓利完成了第 1、3、4 章。解圣庆、张淑玉、周金玲、周建梁参与了部分章节的编写与讨论。

本书可作为普通高等学校应用型本科计算机科学与技术、网络工程、通信工程及相关专业“Java 网络编程技术”、“Java 网站开发技术”等课程的教材。也可供有一定 Java 语言程序设计基础、需进一步学习 Java 网络编程技术的人员自学或作为相关工程技术人员的科技参考书使用。

由于作者水平有限，错误和不足之处在所难免，敬请广大读者批评指正。

目　录

第 1 章　Java 基础

Java 语言是目前推广速度最快的程序设计语言，它采用面向对象的编程技术，功能强大而又简单易学。Java 伴随着 Internet 的发展而成熟，内置了多线程和网络支持能力，可以说是网络世界的通用语言。本章将介绍 Java 语言的基本特点和开发的一般过程，使读者对 Java 程序的开发有一个概括性的了解。

1.1　Java 技术

Java 是一种简单易用、完全面向对象、具有平台无关性、安全可靠、主要面向 Internet 的程序开发工具。自从 Java 正式问世以来，它的快速发展已经让整个 Web 世界发生了翻天覆地的变化。

随着 Java Servlet 的推出，Java 在电子商务方面开始崭露头角，而 Java Server Page(JSP) 技术的推出，更是让 Java 成为基于 Web 的应用程序的首选开发工具。

1.1.1　Java 语言的特点

Java 以跨平台、面向对象、多线程、兼具编译型语言和解释型语言优点、稳定性好并具有良好的安全性等众多特点，尤其是与 Internet 的完美结合，而获得了巨大的成功。

- Java 语言最突出的特点是跨平台性，也叫与平台无关性。也就是说，用 Java 语言编写的程序可以在任何时候、在任何一台计算机上运行，这是因为 Java 语言中并没有任何与具体的工作平台绑定的功能。
- Java 语言的第二个重要特点是面向对象。这是指把程序实现的每一个具体功能作为类，然后由类来构成对象，这种方法我们会在以后的章节中详细地介绍。
- Java 语言的第三个特点是多线程。线程是指正在运行的程序，但线程有别于进程，即多个线程共用一个内存区域，也共享同一组系统资源，对每个线程来讲，只有堆栈和寄存器数据是独立的，所以在线程之间进行通信和切换时，系统开销要比进程机制小得多。
- Java 语言的第四个特点是具有编译型语言和解释型语言的优点。编译是指一次性把源程序翻译成可以运行的目标程序，以后翻译好的目标程序可以作为一个独立的文件无数次地运行。编译过程所需要的存储空间大，同时，编译所需要的时间较长，但程序执行速度快，C、Fortran、Pascal 等都属于编译型语言。解释是指对源程序翻译一句执行一句，翻译和执行交叉进行，每运行一次，都必须重新翻译、执行，翻译完即执行完，这类语言包括 Perl、Python、Rebol、Ruby 等，也常被称作 Script(脚本)语言，由于边解释边执行，所以解释型语言的速度远远低于编译型语言。Java 语言是半编译半解释的语言，一个 Java 语言源程序要运行，必

须先由 Java 编译器编译成字节码，但这个编译过程是不彻底的，因为字节码不是最终的执行程序，不能在具体平台上运行，必须由运行于系统上的解释器将其翻译成机器语言，Java 解释器采取边解释边执行的方式，但速度仍相当快。

- Java 语言还有一个非常重要的特点，就是 Applet 功能以及与此相关的图形功能。Applet 是 Java 特有一种小应用程序，Java 系统提供特殊的技术，可以使这些小应用程序的功能被方便地加入到 Internet 网页上去，从而为 Internet 网页增加各种图形效果和多媒体功能。
- 除了上述特点外，Java 还具有稳定性好、安全性高和编程简单等优点。

1.1.2 Java 与 C/C++的差别

熟悉 C 语言和 C++语言的读者一定想搞清这个问题，实际上，Java 确实从 C/C++语言继承了许多优秀的成分，比如 Java 在变量声明、操作符形式、参数传递、流控制等方面与 C/C++语言相同。但是，Java 与 C/C++语言又有许多差别，主要反映在以下几个方面：

- Java 对内存的分配是动态的，它采用面向对象的机制，通过 new 运算符为每个对象分配内存空间，而且实际内存还会随程序的运行情况而改变，同时，Java 能自动回收不再使用的内存，具有自动垃圾搜集功能。
- Java 不使用 goto 语句，而用 try-catch-finally 异常处理语句来代替 goto 语句来处理出错的功能。
- Java 不在所有类之外定义全局变量，而是在某个类中定义一种公用静态的变量来完成全局变量的功能。
- Java 不支持头文件。
- Java 不支持宏定义，而是用关键字 final 来定义常量。
- Java 为每种数据类型都分配固定长度，例如在 Java 中，int 类型总是 32 位的，而 C/C++中对于不同的平台同一个数据类型分配不同的字节数，例如同是 int，在有的计算机中为 16 位，而在另一些计算机中为 32 位，从而造成 C 语言移植性差，而 Java 则具有跨平台性。
- Java 不使用指针，保证了系统的安全性。

1.1.3 两类 Java 程序

1. Java 应用程序

Java 应用程序的程序框架如下：

```
public class 类名
{
    public static void main(String args[])  {
        ...//程序代码
    }
}
```

上面的程序框架在类中定义了一个 main()方法，其中 public 表示访问权限，指明所有的类都可以使用这一方法；static 指明该方法是一个静态方法，它可以通过类名直接调用；void 则指明 main()方法不返回任何值。对于一个应用程序来说，main()方法是必需的，而且必须按照如上的格式来定义。Java 应用程序(Java Application)是一种能在支持 Java 的平台上独立运行的程序，它是通过 Java 虚拟机(Java Virtual Machine，JVM)解释执行的。Java 解释器在没有生成任何实例的情况下，以 main()作为入口来执行程序。Java 程序中可以定义多个类，每个类中可以定义多个方法，但是最多只能有一个公共类(public class)。在 main()方法定义中，括号“()”中的 String args[ ]是传递给 main()方法的参数，参数名为 args，它是类 String 类型的数组。

2. Java 小程序

Java 小程序的程序框架如下：

```
import java.applet.*;
import java.awt.*;
public class Applet1 extends Applet
{
    //变量或属性定义;
    public void init()  //初始化方法
    {
        //...;
    }
    public void paint(Graphics g)  //小程序显示结果的方法
    {
        //...;
    }
    //其他方法
}
```

Java 小程序(Java Applet)是一种通过<applet></applet>标记将字节码文件内嵌在 HTML 文档中并由支持 Java 的浏览器来运行的程序。Java 小程序没有自己的主进程。

小程序不能独立运行，它是通过支持 Java 的浏览器来运行的。因此，必须将小程序编译后形成的字节码(.class)文件利用标记<applet></applet>嵌入 HTML 文档中：

```
<html>
<head><title>...</title></head>
<body>
<applet code = 类名.class  width = 300  height = 150>
</applet>
</body>
</html>
```

import 语句的作用是：装入 java.awt 包和 java.applet 包下所有的类，使得程序可能使用这些包中所定义的类。小程序中没有实现 main()方法，由于 Applet 不是一个能单独运行的程序，需要支持 Java 的浏览器来解释运行，因此应编写 HTML 文件，通过<applet>标记把该 Applet 嵌入其中，然后用 Appletviewer 命令来运行嵌入了 Applet 的.htm 文件，或在支持

Java 的浏览器上运行.htm 文件。其中用<applet>标记来启动该 Applet，code 指明字节码文件的名称，width 和 height 指明 Applet 所占的宽和高。在不支持 Java Applet 的浏览器中，将不能运行 Java Applet。

1.1.4 Java 程序的编辑、编译和运行

开发一个 Java 程序的过程包括下列步骤。

(1) 建立 Java 源程序。Java 源程序包含 Java 命令语句，可用任何文本编辑器建立。注意，使用一些带格式的文本编辑器(如 Word 等)在保存源程序文件时，应选择以 MS-DOS 文本格式保存。

(2) 编译源程序。在命令行状态下执行 javac.exe，将源程序编译成字节码文件，字节码文件的内容是 Java 虚拟机(JVM)可执行的指令，编译时如果出现错误，则终止编译，应修改程序中的错误，直至最终通过编译为止。

(3) 运行 Java 程序。Java 虚拟机由 Java 解释器实现，在命令行状态下执行 java.exe，可将 Application 字节码文件解释为本地计算机能够执行的指令并予以执行，运行结果在 MS-DOS 窗口中显示，如果是图形方式的 Application，将会显示图形界面。

如果程序是 Java Applet，应建立一个 HTML 文件，在适当位置加入 Applet 字节文件名，并用 Applet 查看器或直接用浏览器打开 HTML 文件，Applet 的运行结果会在查看器或浏览器窗口中显示出来。

下面通过一个简单的 Java 应用程序，来看一下其整个编译和运行过程。

【例 1.1】第一个 Java 程序——HelloWorld.java。

在 Windows 环境下打开“记事本”工具，输入如下几行语句：

```
// HelloWorld.java
// A first program in Java
public class HelloWorld {
    public static void main(String args[ ])
    {
        System.out.println("Hello World!");
    }
}
```

输入代码后，以 HelloWorld.java 作为文件名保存。

下一步是对源程序 HelloWorld.java 进行编译，在 MS-DOS 命令提示符下执行如下编译命令：

```
C:\jdk1.6.0_02\bin>javac HelloWorld.java
```

编译之后生成一个名为 HelloWorld.class 的字节码文件。

最后一步就是用 Java 解释器边解释边运行字节码文件：

```
C:\jdk1.6.0_02\bin>java HelloWorld
```

运行结果如图 1.1 所示。

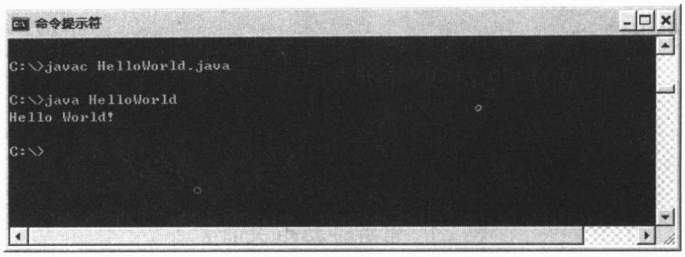

图 1.1　HelloWorld.java 程序的运行结果

1.1.5　Java 语言的注释和分隔符

1. Java 语言的注释

注释是为源程序增加必要的解释和说明内容，以提高程序的可读性，使用注释是一种良好的编程习惯。Java 语言的注释分为单行注释和多行注释两种，说明如下。

- 多行注释(块注释)。通常以“/*”开始，以“*/”结束，可跨越多行，格式为：

```
/*  注释部分 */
/** 注释部分 */
```

- 单行注释。以“//”开始的行的后面部分为注释内容，用于单行的注释，格式为：

```
//  注释部分
```

2. 几个重要的分隔符

程序中经常使用的几个重要的分隔符是：

- 大括号({})——用来定义类体、方法体、复合语句和数组的初始化。
- 分号(;)——是语句的结束标志。
- 逗号(,)——区分方法的各个参数，区分变量声明的各个变量。
- 冒号(:)——用于语句标号。

1.1.6　Java 语言的标识符和关键字

1. Java 语言的标识符

标识符用来给自定义的变量、方法和类及类对象命名。

Java 语言的标识符必须由字母、下划线(_)或美元符号($)开始，其余字符可以是上述三种符号或数字(0~9)。由于 Java 语言使用 Unicode 字符集(16 位)，所以字母包括下面几种：

- A~Z 之间的 26 个大写字符。
- a~z 之间的 26 个小写字符。
- Unicode 字符集中序号大于 0xC0 的所有符号。如中文字符、日文字符、阿拉伯字符等都可用作 Java 语言的标识符。

标识符不能以数字开头，不能用关键字给变量或对象命名。

2. Java 语言的关键字

关键字(又称保留字)，是 Java 中具有特定含义的标识符。用户只能按系统规定的方式使用，不能用于用户自己的目的。关键字一律用小写字母来表示。表 1.1 给出了一些常用的 Java 关键字。

表 1.1 常用的 Java 关键字

abstract	Default	goto	null	switch
boolean	Do	if	package	synchronized
break	Double	implements	private	this
byte	Else	import	protected	throw/throws
case	Extends	instanceof	public	transient
catch	False	int	return	true
char	Final	interface	short	try
class	Finally	long	static	void
const	Float	native	strictfp	volatile
continue	For	new	super	while

注意： 关键字使用大写会引起语法错误。

3. Java 语言的命名规则

Java 语言的命名规则如下：

- 类名的第一个字母大写，如 System、Applet 等。
- 方法名第一个字母小写，如 main()、print()、println()等。
- Java 语言中，类名和方法名都比较长，这样便于阅读程序。如果类名或方法名由多个单词构成，单词的第一个字母大写。
- Java 语言的关键字小写。
- 自定义的变量名或一个类的对象名一般小写。

1.1.7 Java 语言的数据类型

数据类型说明了常量、变量或表达式的性质，在 Java 语言中定义了 8 种基本的数据类型。这些基本类型又称作基本元素，可划分为下述 3 类。

- 数值型：既可以是整数，也可以是浮点数。
- 字符型(char)：存储一个字符，如‘a’、‘d’等。
- 布尔型(boolean)：也称为逻辑型，值为 true 或 false。

在 Java 中，所有基本类型的类型名都是关键字，因此不能挪作它用，下面让我们仔细来看一下各种基本数据类型。

1. 整型数据类型

可以用来存储整型数值的数据类型有 4 种，它们都是有符号数，即可以存储正的或负的数值，这 4 种整数类型所存储的数值范围的差别如下。

- byte 型：这种类型的变量可以表示-128~127 之间的整数，在内存中占 1 个字节。
- short 型：这种类型的变量可以表示-32768~32767 之间的整数数值，在内存中占 2 个字节。
- int 型：这种类型的变量可以表示-2147483648~2147483647 之间的整数数值，在内存中占 4 个字节。
- long 型：这种类型的变量可以表示-9223372036854775808~9223372036854775807 之间的整数数值，在内存中占 8 个字节。

让我们看一下上述类型的变量声明的例子：

```
byte smallValue;
short pageCount;
int wordCount;
long bigValue;
```

上述每条语句都声明了一个特定类型的变量，对于 Java 语言，无论你使用何种机型，每种整数类型所存储的数值范围均如上所列，范围都是相同的。Java 中的任何直接使用数值都被称为直接量。任何整型直接量的默认类型都是整型(int)，因此，1、-999、1234567 都是 int 型直接量。如果要定义一个长整型(long)的整形变量，并且赋给该变量一个大于 int 型的数值，就需要在该数值后面追加一个 L。也可以使用小写字母 l，但它与数字 1 太容易混淆，所以最好不要使用。例如 2L、-239L、7654321L 都属于 long 型的。

整型直接量也可以被指定为基数 16，也就是十六进制数值。在 Java 中，十六进制数值的直接量要以 0x 为前缀，0~9 和 A~F(或 a~f)跟随其后，它们分别表示 0~15 之间的数值。同样，1~7 数字前加 0 可以表示八进制数。十六进制数值转化为十进制数的方法如下：

0x100　$1\times16^2+0\times16^1+0\times16^0=256$

0xCAB　$12\times16^2+10\times16^1+11\times16^0=3243$

让我们重新回过头来看一下变量的声明，例如我们可以用如下语句声明一个 long 型的变量：

```
long bigOne;
```

这条语句是变量 bigOne 的一个声明，它指出变量 bigOne 将存储一个 long 型的数值。当这条语句被编译时，系统就分配给它一块 8 个字节的内存空间。Java 并不会自动地对一个变量进行初始化，如果你希望变量有初始值，就必须在声明中为它指定值，例如：

```
long bigOne = 999L;
```

注意：*在声明变量时对它进行初始化是个好习惯。*

如果在计算中使用了一个未赋值的变量，程序将无法通过编译。

可以在程序的任何地方声明变量，但是，必须在使用该变量进行运算之前声明它，因此，变量声明的位置将影响一个给定变量在程序的某个位置上是否可以访问。

2. 浮点数据类型

非整型数值被称为浮点型数值，浮点型数值有固定的精度，但它的取值范围非常大，虽然数字位数是固定的，但由于小数点可以“浮动”，所以可以获得一个非常大的取值范围。

在 Java 中有两种基本的浮点类型——float 和 double。这两种类型使程序员能够对表示的数据精确度以及取值范围进行选择，具体说明如下。

- float：该类型变量可表示-3.4×10^{38}~$+3.4\times10^{38}$之间的数值，占用 4 个字节的内存空间，表示的数值精确度大约为 7 位。
- double：该类型的变量可表示-1.7×10^{308}~$+1.7\times10^{308}$之间的数值，占用 8 个字节的内存空间，表示的数值精确度大约为 17 位。

浮点型直接量的默认类型为 double，比如 1.0 和 345.67 都是 double 型数值，若要声明一个 float 类型数值，就需要在其后追加字母 f 或 F，比如 1.0f 和 345.67F。

对于非常大或非常小的浮点数值，通常使用指数形式进行书写，即一个十进制小数乘以 10 的幂次的形式。在 Java 中，程序员可以用十进制小数后跟 E 或 e，之后再跟 10 的幂次来表示浮点数值，例如 1.496E8 和 3.56E-9 等。

声明浮点型变量与声明整型变量的方法相同，例如可以用下列语句来声明和初始化浮点型变量：

```
double sunDistance = 1.496E8;
float electronMass = 9E-28;
```

1.1.8 Java 中的常量

常量是在程序运行期间值不改变的量，常量不能被程序修改。

在 Java 语言中，定义常量通过 final 关键字来实现。Java 的常量包括整型、实型、字符型及布尔型常量，还有字符串常量。

1. 整型常量

整型常量默认为 int 类型，用 4 个字节的存储单元存放。

要表示一个数为长整型，需在这个数后面加上一个字母 L(或 l)。

八进制整数只能包含 0 到 7 这八个数字及正、负号，而且必须以数字 0 为前导。

十六进制整数只能包含数字 0 到 9、字母 A 到 F 或 a 到 f、正/负号。

2. 实型常量

实型数据分单精度(float)和双精度(double)两种类型。单精度的实数占 4 个字节内存，双精度实数占 8 个字节内存。

实型常量的小数形式由一个整数加上小数点，然后加上小数部分组成。

在 Java 中，无类型后缀的实型常量默认为双精度类型(double)，也可加后缀 D 或 d。

指定单精度浮点类型常量时，必须在常量后面显式地加上后缀 F(或 f)。

实型常量也可表示为指数形式：如双精度数 2.1E8、5.3e-9D；单精度数 9e-2f 等。其中的 8、-9、-2 分别表示指数。

3. 字符常量

字符常量是无符号的常量，占两个字节内存，其范围是 0~65535。

字符常量有 4 种表达方式：

- 用单引号括起来的字符。例如 'a'、'+'、'-'。
- 用单引号括起来的转义序列，以 "\" 开始表示转义。例如制表符'\t'、换行符'\n'、反斜杠'\\'等。
- 用单引号括起来的八进制转义序列，形式为'\ddd'，此处的 ddd 表示 3 位八进制数。例如'\141'是字母 a。
- 用单引号括起来的 Unicode 转义字符形式为'\U××××'，此处的××××为 4 位十六进制数。例如'\U0061'是字母 a。

表 1.2 列出了常用的转义字符。

表 1.2　常用的转义字符

转义字符	功　能
'\b'	退格
'\r'	回车
'\n'	换行
'\t'	水平制表符
'\f'	进纸
'\''	单引号
'\"'	双引号
'\\'	反斜杠

4. 布尔常量

布尔常量仅有两个值：true 和 false，分别代表布尔逻辑中的“真”和“假”。

在 Java 语言中，布尔常量不能转换成任何数据类型，true 常量不等于 1，而 false 常量也不等于 0。

5. 字符串常量

字符串(String)不是简单数据类型，而是复合数据类型(类类型)。

一个字符串常量是用双引号括起来的 0 个或多个字符组成的序列。例如：

```
""  //空串
"Hello World\n"  //一行字符，最后一个是回车换行符
```

两个字符串可以用连接符“+”来连接(此时允许分行写)。例如：“This is a string ” + “That is an other string”等价于“This is a string That is an other string”。

1.1.9 变量

变量是 Java 程序中的基本存储单元，它具有名称、类型、值和作用域等特性。在使用任何变量之前必须先定义。如果是基本数据类型的变量，在声明变量类型时，就为其分配了内存单元，因为它们的存储长度是固定的。如果是复合数据类型的变量，声明完其数据类型之后，还要用关键字 new 为其分配内存单元。

1. 变量定义

每个变量必须定义为某一种数据类型，而且只能声明为唯一的数据类型，不允许重复定义。

变量定义是用标识符为变量起名、确定其数据类型，还可以为它赋初值(称作变量初始化)。

变量必须是先定义后使用。

变量的定义格式为：

```
数据类型 标识符 1[ = 初值], [标识符 2[ = 初值], ... ;
```

标识符用于为变量命名；赋初值使变量一开始有确定的值，为可选项；可用同一类型定义多个变量，各变量之间用逗号(,)作为分隔符。

2. 变量赋值与类型转换

在进行赋值之前，首先检查赋值运算符左右两端数据的类型是否一致。当出现类型不匹配的赋值操作时，表示数的范围较小的数据类型可自动向表示数的范围较大的数据类型转换；否则，必须强制类型转换，不然将导致编译错误。

所有整型数据类型都能向 float 或 double 自动转换，反之必须强制转换。

字符类型可向 int、long、float 或 double 自动转换，反之必须强制转换。

字符类型转换为字节类型必须强制转换。

布尔型不能与任何其他数据类型转换。

强制类型转换一般会丢失部分信息。而自动类型转换不会造成信息丢失(见表 1.3)。

表 1.3 不会丢失信息的类型转换

原始类型	目标类型
byte	short、char、int、long、float、double
short	int、long、float、double
char	int、long、float、double
int	long、float、double
long	float、double
float	Double