第 2 章　Java 语言基础

本章学习目标

本章主要讲解 Java 语言的基础知识，为后面章节的学习打下基础。通过对本章的学习，读者应该掌握以下内容:

- 标识符、保留字和分隔符
- 数据类型
- 运算符和表达式
- Java 程序的简单输入输出

2.1　标识符、保留字和分隔符

2.1.1　Java 标识符

在现实生活中任何事物都有自己的名字，在程序中也是如此。编程人员要对程序中的变量、类、方法、标号、数组、字符串和对象等元素进行命名，这种命名记号称为标识符（Identifier）。在 Java 中，标识符的定义规则为：以字母、下划线（_）、美元符（$）开始，其后面可是任意个字母、数字（0~9）、下划线、美元符的字符序列。Java 标识符区分大小写，对长度没有限制。用户定义的标识符不可以是 Java 关键字，但关键字可以作为用户定义标识符的一部分，例如，my_class 是一个合法的用户标识符，其中关键字 class 作为它的一部分。

Java 程序是由统一字符编码标准字符集（Unicode Character Set）写成的，其字母包括：'A'~'Z'，'a'~'b'和字符集中序号大于 0xC0（192）的所有符号。在这种字符集中字母和汉字以及其他语言文字的长度是一样的。统一字符编码标准是一种十六位的字符编码标准，而 ASCII 则是七位编码，只适用于英文。Unicode 共有 65536 个编码，其中有近 39000 种已被定义完成，而中国字就占了 21000 种 ！由于只有较少数的文本编辑器支持 Unicode，因此，大多数的 Java 程序是用 ASCII 码编写的。

例如，合法的标识符：

Class　program　_system　$value　a2　my_int

非法的标识符：

class　2x　hello!　Build#2　mailbox-2

其中，class 是保留字，不可以作为用户自定义标识符。2x 以数字开头，hello!、Build#2、

mailbox-2 包含非法字符，所以都不是合法的标识符。

2.1.2 保留字

保留字又称为关键字，是 Java 语言中具有特殊意义和用途的标识符，这些标识符由系统使用，不能作为一般用户定义的标识符使用。因此，这些标识符称为保留字（Reserved Word）。Java 语言中的保留字如表 2-1 所示。

表 2-1 Java 语言中的保留字

abstract	break	byte	boolean	catch
case	char	class	continue	default
do	double	else	extends	false
final	float	for	finally	if
import	implements	int	interface	Instanceof
long	length	native	new	null
package	private	protected	public	return
switch	synchronized	short	static	super
try	true	this	throw	throws
threadsafe	transient	void	volatile	while

2.1.3 分隔符（Seprators）

在程序中为了便于区分任意两个相邻的标识符、操作数、保留字或两个语句，在它们之间必须插入一个分隔的符号，这样的符号称为分隔符。常见的分隔符有：空格、圆括号、花括号、分号等。

每条语句无论一行还是多行都以分号（Semicolons）结束。块（Block）是包含在{}里面的多条语句，块可以嵌套。空白（White space）插在代码的元素中间：由一个或多个空格（Spaces）组成，也可以由一个或多个 tab 空格（Tabs）组成多个空行（Newlines）。

2.1.4 注释（Comments）

Java 中的注释有 3 种形式。

单行：//…

多行：/*

…

*/

文档注释：

/**

…

*/

第三种形式是第二种形式的变形，可用 javadoc.exe 提取程序文件中的文档注释来制作 HTML 帮助文档。

2.2　数据类型概述

2.2.1　数据类型的划分

Java 语言中的数据类型分为基本数据类型、空类型和复合数据类型。具体划分如图 2-1 所示。基本数据类型是 Java 中定义的数据类型，是不可再分的原始类型。复合数据类型是用户根据需要用基本数据类型经过组合而形成的类型。

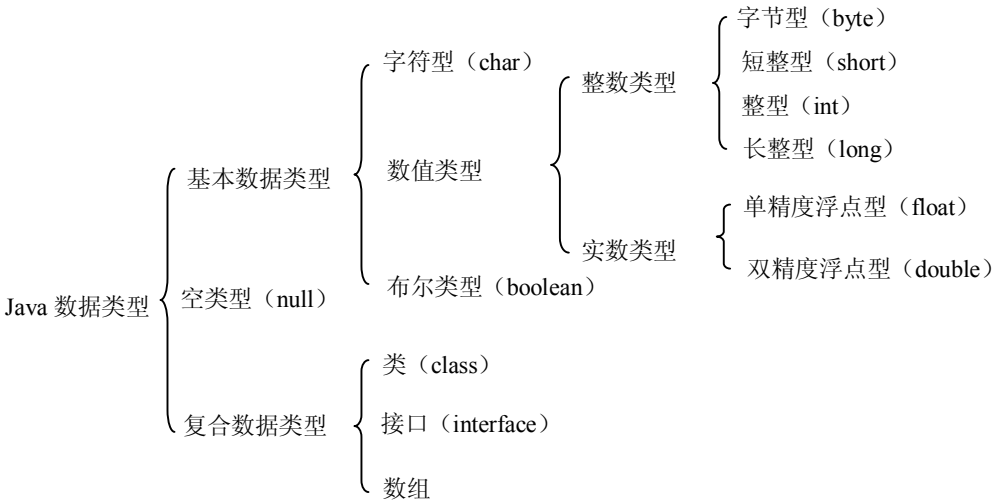

图 2-1　Java 数据类型

2.2.2　常量和变量

在程序中使用各种数据类型时，其表现形式有两种：变量和常量。

1. 常量

常量有字面（Literal）常量和符号常量两种形式。

字面常量是指其数值意义如同字面所表示的一样，例如 69，就表示值和含义均为 69。常量区分不同的数据类型，如整型常量 789，实型常量 2.34，字符常量'A'，布尔类型常量 true 和 false，字符串常量"I like use Java to program. "。

符号常量是用 Java 标识符表示的一个常量，可以使用保留字 final 定义符号常量，符号常量定义的一般格式如下：

<final><数据类型><符号常量标识符>=<常量值>;

例如：
```
final double PI=3.141593;
final int COUNT=1000;
```

2. 变量

变量是 Java 程序中用于标识数据的存储单元。定义变量包括变量类型、变量标识符及作用域部分。变量定义的一般格式如下：

<数据类型><变量标识符>=<值>，<变量标识符>=<值>，…;

例如：
```
double x=1.2345;
int  j=10;
char  ch1,ch2;
String str1,str2;
```

变量的作用域是指能够访问该变量的一段程序代码。在声明一个变量的同时也就指明了变量的作用域。变量按作用域划分，可以有以下几种类型：局部变量、成员变量、方法参数、异常处理参数。

局部变量在方法或方法的一个块内声明，它的作用域为它所在的代码块。

成员变量在类中声明，而不是在类的某个方法中声明，它的作用域是整个类。

方法参数传递给方法，它的作用域就是这个方法。

异常处理参数传递给异常处理程序，它的作用域就是异常处理代码部分。

在一个确定的域中，变量名是惟一的。通常情况，一个域用大括号{}来划定。有关类变量、参数传递及异常处理在后续章节中讲述。

3. 变量默认值

若不给变量赋初值，则变量默认值如表 2-2 所示。

表 2-2 变量默认值

数据类型	默认值（初始值）
boolean	flase
char	'\000'（空字符）
byte	0（byte）
short	0（short）
int	0
long	0L
float	0.0F
double	0.0

2.3 基本数据类型

Java 基本数据类型在内存中占的位数及范围如表 2-3 所示。

表 2-3　Java 基本数据类型在内存中占的位数及范围

数据类型	关键字	bit	取值范围
布尔型	boolean	8	true，false
字节型	byte	8	−128~127
字符型	char	16	0~65535
短整型	short	16	−32768~32767
整型	int	32	-2^{31}~$2^{31}-1$
长整型	long	64	-2^{63}~$2^{63}-1$
单精度型	float	32	3.4028E−38~3.4028E+38

2.3.1　整型数据

1. 整型常量

（1）整型多为十进制数形式，也可为八进制或十六进制形式：无任何前缀的是十进制数，以 0 开头为八进制数，以 0x 或 0X 开头的为十六进制数。

十进制数的 10 个基数为 0~9。八进制数的 8 个基数为 0~7，不包括 8 和 9；十六进制数的 16 个基数为 0~9，a~f（或 A~F）。

例如：十进制数：1234，−5678；

八进制数：0234 表示十进制数 156，

−0123 表示十进制数−83；

十六进制数：0x64 表示十进制数 100，

−0xff 表示十进制数−255。

（2）Java 整型数都为带符号数。

（3）整型默认为 int 型，若为长整型需在数据后加字母 l 或 L。强烈推荐使用 L，以免与数字 1 混淆。

2. 整型变量

整型变量的类型包括：byte、short、int、long。

int 类型是最常用的整数类型，它表示的数据范围相当大，基本满足现实生活的需要，而且无论是 32 位还是 64 位的处理器，int 类型在内存中的存储长度都是 4 个字节。如果遇到更大的整数，int 类型不能表示，要使用 long 类型。

byte 类型表示的数据范围很小，容易造成溢出，使用时要注意。

short 类型很少使用，它限制数据的存储为先高字节，后低字节，在某些机器中会出错。

整型变量的定义如下所示：

```
byte   bval;            // 定义变量 bval 为 byte 型
short   sval;           // 定义变量 sval 为 short 型
int   ival;             // 定义变量 ival 为 int 型
long  lval ;            // 定义变量 lval 为 long 型
```

2.3.2 实型数据

1. 实型常量

（1）用十进制数形式表示，由数字和小数点组成，必须包含小数点，如 0.1234，12.345、12345.0。

（2）用科学计数法形式表示，格式如下：

尾数 E（或 e）指数

E（或 e）之前必需有数字，指数必须为整数，如 1.2345E+3、12345E−3。

（3）数后加 f 或 F 为 float，加 d 或 D 为 double，没有后缀修饰的则默认为 double 类型。

2. 实型变量

实型变量，即为浮点型变量，包括 float 和 double 类型。

双精度浮点型 double 比单精度浮点型 float 的精度更高，表示数据的范围更大。

实型变量的定义如下所示：

```
float fval;          // 定义变量 fval 为 float 型
double dval;         // 定义变量 dval 为 double 型
```

2.3.3 字符型数据

1. 字符常量

字符常量是用单引号括起来的一个字符，如：'J'、'*'。另外，Java 中有以反斜杠（\）开头的字符，反斜杠将其后面的字符转变为另外的含义，称为转义字符。Java 中的转义字符如表 2-4 所示。

表 2-4 Java 中的转义字符

转义字符	Unicode	含义
\b	\u0008	退格（backspace）
\f	\u000C	换页（form feed）
\n	\u000A	换行（line feed）
\r	\u000D	回车（carriage return）
\t	\u0009	水平跳格（Tab）
\'	\u0027	单引号（single quote）
\ "	\u0022	双引号（ double quote）
\\	\u005c	倒斜线（backslash）
\ddd		八进位转义序列（d 介于 0~7）
\uxxxx		十六进制转义序列

2. 字符变量

Java 中的字符型数据是 16 位的 Unicode，汉字与英文字母占的内存空间相同。

字符变量的定义如下所示：

```
char  ch='i';            //定义 ch 为字符型变量，且赋初值为'i'
```

2.3.4　字符串数据

1. 字符串常量

字符串常量是使用双引号括起来的字符序列（注意：最后字符不是'\0'）。例如："Let's learn Java! "。Java 编译器自动为每一个字符串常量生成一个 String 类的对象，因此可以使用字符串常量初始化一个 String 类的对象。

在字符串中允许出现转义字符，如："天苍苍，\t 野茫茫…\n 风吹草低现牛羊"。输出上面的字符串结果为：

```
天苍苍，        野茫茫…
风吹草低现牛羊
```

2. 字符串变量

在 Java 中，字符串变量是作为对象处理的。通过使用 String 类或 StringBuffer 类的构造函数来生成字符串对象。在后面的字符串处理一章中会详细介绍。

2.3.5　布尔型数据

1. 布尔型常量

布尔类型的常量值只有两个：true 和 false。

2. 布尔型变量

布尔类型变量为 boolean 类型。布尔类型变量只有两个取值 true 和 false，且它们不对应任何数值，与 C++不同。布尔类型变量的定义如下所示：

```
boolean  bval=false;        //定义 bval 为 boolean 类型变量，且赋初值为 false
```

2.3.6　类型转换

1. 自动类型转换

Java 允许不同类型的数据进行混合运算，如果在 Java 表达式中出现了数据类型不一致的情形，那么 Java 运行时系统先自动将低优先级的数据转换成高优先级类型的数据，然后才进行表达式值的计算。Java 数据类型的优先级关系如下所示：

低 ⟶ 高

byte、short、char ⟶ int ⟶ long ⟶ float ⟶ double

2. 强制类型转换

强制类型转换：优先级高的类型转换成优先级低的类型，使用方法如下：

（数据类型）表达式

例如：

```
float x=5.5;          // x 为 float 类型
int y;                // y 为 int 类型
y=(int)x+100;         // 先把 x 转换为 int 型，放到临时变量中，
```

```
// 然后与 100 相加，结果赋给 y，x 类型不变，仍为 int 型。
```

2.4 运算符

Java 语言中对数据的处理过程称为运算，用于表示运算的符号称为运算符，它由一至三个字符结合而成，虽然运算符是由数个字符组合而成，但 Java 将其视为一个符号。参加运算的数据称为操作数。按操作数的数目来划分运算符的类型有：一元运算符（如：++）、二元运算符（如：*）和三元运算符（如：？：）；按功能划分运算符的类型有：算术运算符、关系运算符、布尔运算符、位运算符、赋值运算符、条件运算符和其他运算符。

2.4.1 算术运算符

算术运算符主要完成算术运算。常见的算术运算符如表 2-5 所示。

表 2-5 Java 算术运算符

运算符	运算	例子	结果
+	正号	+8	8
-	负号	a=8;b=-a;	a=8,b=−8
+	加	a=6+6;	a=12
-	减	a=16-9;	a=7
*	乘	a=16*2;	a=32
/	除	a=16/3;	a=5
%	模除（求余）	a=16%3;	a=1
++	前缀增	a=10;b=++a;	a=11,b=11
++	后缀增	a=10;b=a++;	a=11,b=10
--	前缀减	a=10;b=--a;	a=9,b=9
--	后缀减	a=10;b=a--;	a=9,b=10

Java 对加运算符进行了扩展，使它能够进行字符串的连接，如："Java"+" Applet" 得到字符串"Java Applet"。

另外，Java 模除运算%对浮点型操作数也可以进行计算，这与 C/C++不同。

【例 2-1】 Java 运算符++、--的使用。

```
public class ProfixApp{
    public static void main(String[] args){
      int a=10;
      System.out.println("a="+a);
      int b=++a;
      System.out.println("a="+a);
      System.out.println("b="+b);
      int c=a++;
```

```
            System.out.println("a="+a);
            System.out.println("c="+c);
            int d=--a;
            System.out.println("a="+a);
            System.out.println("d="+d);
            int e=a--;
            System.out.println("a="+a);
            System.out.println("e="+e);
        }
    }
```

程序输出结果：

```
a=10
a=11
b=11
a=12
c=11
a=11
d=11
a=10
e=11
```

算术运算符中，单目运算符+（正号）、-（负号）、++、--的优先级最高，二元运算符*、/、%其次，二元运算符+、-优先级最低。算术运算符的执行顺序自左至右。

2.4.2　关系运算符

关系运算符主要完成操作数的比较运算，结果为布尔值。Java 的关系运算符如表 2-6 所示。

表 2-6　Java 关系运算符

运算符	运算	例子	结果
==	等于	4==2	false
!=	不等于	1!=2	true
<	小于	18<18	false
<=	小于等于	18<=18	true
>	大于	2>1	true
>=	大于等于	2>=1	true
instanceof	检查是否为类实例	"Java" instanceof　String	true

关系运算符的优先级别低于算术运算符，关系运算符的执行顺序自左至右。

2.4.3　布尔逻辑运算符

布尔逻辑运算符主要完成操作数的布尔逻辑运算，结果为布尔值。Java 的布尔逻辑运

算符如表 2-7 所示。

表 2-7 Java 布尔逻辑运算符

运算符	运算	例子	结果
&	与	5>2&2>3	false
|	或	5>2|2>3	true
!	非	! true	false
^	异或	5>2^8>3	false
&&	简洁与	5>12&&22>3	false
||	简洁或	5>2||2>3	true

简洁与、或和非简洁与、或对整个表达式的计算结果是相同的，但有时操作数的计算结果不同。简洁与、或运算时，若运算符左端表达式的值能够确定整个表达式的值，则运算符右端表达式将不会被计算。而非简洁与、或运算时，运算符两端的表达式都要计算，最后计算整个表达式的值。

例如：

```
int a=6,b=8,c=12,d=15;
boolean x=++a>b++&&c++>d--;
```

则结果为：

```
a=7,b=9,c=12,d=15,x=false;
```

而

```
int a=6,b=8,c=12,d=15;
boolean x=++a>b++&c++>d--;
```

结果为：

```
a=7,b=9,c=13,d=14,x=false;
```

在布尔逻辑运算符中，单目运算符“!”的优先级最高，高于算术运算符和关系运算符，运算符“&”、“|”等低于关系运算符。布尔逻辑运算符的执行顺序自左至右。

2.4.4 位运算符

位运算符是对二进制位进行操作，Java 提供的位运算符如表 2-8 所示。

表 2-8 Java 位运算符

运算符	运算	例子	结果
~	按位取反	~00011001	11100110
&	按位与	00110011&10101010	00100010
|	按位或	00110011|10101010	10111011
^	按位异或	10100001^00010001	101100000

续表

运算符	运算	例子	结果
<<	左移位	a=00010101; a<<2;	01010100
>>	右移位	a= 10101000; a>>2;	11101010
>>>	无符号右移	a= 10101000; a>>>2;	00101010

在计算机中，Java 使用补码来表示二进制数，最高位为符号位，正数的符号位为 0，负数的符号位为 1。对正数而言，补码就是正数的二进制形式，对于负数，首先把该数绝对值的补码取反，然后再加 1，即得该数的补码。例如：123 的补码为 01111011，−123 的补码为 10000101。

（1）按位取反运算符 ~。

~是一元运算符，对数据的每个二进制位进行取反，即把 0 变为 1，把 1 变为 0。

例如：~00011001=11100110

（2）按位与运算符 &。

对应运算的两个位都为 1，则该位的结果为 1，否则为 0。即：

0&0=0　　0&1=0　　1&0=0　　1&1=1

例如：10110011&10101010=10100010

（3）按位或运算符 |。

对应运算的两个位都为 0，则该位的结果为 0，否则为 1。即：

0|0=0　　0|1=1　　1|0=1　　1|1=1

例如：10110011|10101010=10111011

（4）按位异或运算符 ^。

对应运算的两个位相同，都为 1 或 0，则该位的结果为 0，否则为 1。即：

0^0=0　　0^1=1　　1^0=1　　1^1=0

例如：10100001^00010001=101100000

（5）左移位运算符 <<。

用来将一个数的各二进制位全部左移若干位，右端补 0。在不溢出的情况下，每左移一位，相当于乘 2。

例如：a=00010101; a<<2=01010100

（6）右移位运算符 >>。

用来将一个数的各二进制位全部右移若干位，前补符号值。每右移一位，相当于除以 2。

例如：a= 10101000; a>>2= 11101010

（7）无符号右移运算符 >>>。

用来将一个数的各二进制位全部右移若干位，移出的位被舍弃，前面空出的位补 0。每右移一位，相当于除以 2。

例如：a= 10101000; a>>>2= 00101010

【例 2-2】 有关位运算的实例。

```
// BitOperate.java
public class BitOperate{
       public static void main(String []args){
           int i=123;
           int j=45;
           OutBitInt("i   ",i);
           OutBitInt("~i  ",~i);
           OutBitInt("-i  ",-i);
           OutBitInt("j   ",j);
           OutBitInt("i&j ", i&j);
           OutBitInt("i|j ", i|j);
           OutBitInt("i^j ", i^j);
           OutBitInt("i<<2", i<<2);
           OutBitInt("i>>2", i>>2);
}
static void OutBitInt(String str,int i)
{
        System.out.print(str+",int: "+i+" ,binary:");
        System.out.print("    ");
        for(int j=31;j>=0;j--)
                if(((1<<j)&i)!=0) System.out.print("1");
                else System.out.print("0");
        System.out.println();
       }
}
```

程序执行结果为：

```
i   ,int: 123 ,binary:  00000000000000000000000001111011
~i  ,int: -124 ,binary: 11111111111111111111111110000100
-i  ,int: -123 ,binary: 11111111111111111111111110000101
j   ,int: 45 ,binary:   00000000000000000000000000101101
i&j ,int: 41 ,binary:   00000000000000000000000000101001
i|j ,int: 127 ,binary:  00000000000000000000000001111111
i^j ,int: 86 ,binary:   00000000000000000000000001010110
i<<2,int: 492 ,binary:  00000000000000000000000111101100
i>>2,int: 30 ,binary:   00000000000000000000000000011110
```

2.4.5 赋值运算符

赋值运算符“=”，用来把一个表达式的值赋给一个变量。如果赋值运算符两边的类型不一致，当赋值运算符右侧表达式的数据类型比左侧的数据类型优先级别低时，则右侧的数据自动被转化为与左侧相同的高级数据类型，然后将值赋给左侧的变量。当右侧数据类型比左侧数据类型高时，则需进行强制类型转变，否则出错。

例如：

```
int a=100;
```

```
long x=a;            //自动类型转换
int a=100;
byte x=(byte)a;      //强制类型转换
```

在赋值运算符“=”的前面加上其他运算符，构成复合赋值运算符。如：i+=8 等价于 i= i+8。Java 赋值运算符如表 2-9 所示。

表 2-9　Java 赋值运算符

运算符	运算	例子	结果
=	赋值	a=8;b=3;	a=8,b=3
+=	加等于	a+=b;	a=11,b=3
-=	减等于	a-=b;	a=5,b=3
*=	乘等于	a*=b;	a=24,b=3
/=	除等于	a/=b;	a=2,b=3
%=	模除等于	a%=b;	a=2,b=3

2.4.6　条件运算符

条件运算符为 （？:），它的一般形式为：

表达式 1？表达式 2: 表达式 3

其中表达式 1 的值为布尔值，如果为 true，则执行表达式 2，表达式 2 的结果作为整个表达式的值，否则执行表达式 3，表达式 3 的结果作为整个表达式的值。

例如：
```
int max,a=20,b=19;
max=a>b?a:b;
```

执行结果 max=20。

条件运算符的优先级要低于赋值运算符。

2.4.7　运算符优先级

对表达式进行运算时，要按照运算符的优先顺序从高到低进行，同级的运算符则按从左到右的顺序进行。表 2-10 列出了 Java 中运算符的优先顺序。

表 2-10　Java 运算符优先顺序

优先顺序	运算符	结合性
1	. [] ()	从左到右
2	+（正号） -（负号） ++ -- ！ ~ instanceof	从右到左
3	new （type）	从右到左
4	* / %	从左到右
5	+（加） -（减）	从左到右
6	>> >>> <<	从左到右

续表

优先顺序	运算符	结合性
7	> < >= <=	从左到右
8	== !=	从左到右
9	&	从左到右
10	^	从左到右
11	|	从左到右
12	&&	从左到右
13	||	从左到右
14	?:	从右到左
15	= += -= *= /= %= ^= &= |= <<= >>= >>>==	从右到左

2.5 表达式

表达式是由操作数和运算符按一定语法形式组成的用来表达某种运算或含义的符号序列，例如，以下是合法的表达式。

```
a+b   (a+b)*(a-b)   "name="+"黄荣盛"   (a>b)&&(c!=d)
```

每个表达式经过运算后都会产生一个确定的值，称为表达式的值。表达式值的数据类型称为表达式的类型。一个常量或一个变量是最简单的表达式。表达式可以作为一个整体也可以看成一个操作数参与到其他运算中，形成复杂的表达式。根据表达式中所使用的运算符和运算结果的不同，可将表达式分为：算术表达式、关系表达式、逻辑表达式、赋值表达式、条件表达式等。

例如：

a++*b、12+c、a%b-23*d 为算术表达式；

x>y、c!=d 为关系表达式；

m&&n、(m>=60)&&(n<=100)、(a+b>c)&&(b+c>a)&&(a+c>b)为逻辑表达式；

i=12*a/5、x=78.98 为赋值表达式；

x>y? x:(z>100:60:100) 为条件表达式。

2.6 简单的输入输出

输入和输出是程序的重要组成部分，是实现人机交互的手段。输入是指把需要加工处理的数据放到计算机内存中，而输出则把处理的结果呈现给用户。在 Java 中，通过使用 System.in 和 System.out 对象分别与键盘和显示器发生联系而完成程序的输入和输出。

2.6.1　输出

System.out 对象包含着多个向显示器输出数据的方法。System.out 对象中包含的最常用的方法是：

- println()方法：向标准输出设备（显示器）输出一行文本并换行。
- print()方法：向标准输出设备（显示器）输出一行文本但不换行。

例如：

```
System.out.println("Example ! ");
System.out.println("Input 10 data: ");
```

执行该代码将显示下述输出结果：

```
Example !
Input 10 data:
```

print()方法与 println()方法非常相似，两者的惟一区别在于 println()方法完成输出后开始一个新行，而 print()方法输出后不换行。下面的代码说明了它们的不同之处：

```
System.out.print("Example ! ");
System.out.print("Input 10 data: ");
```

执行该代码将显示下列输出结果：

```
Example !  Input 10 data:
```

【例 2-3】　输出数据。

```
public  class Output {
    public static void main(String args[]) {
        char  ch='a';
        int i=1;
        double  d=1234.56789;
        String  str="China";
        System.out.println("ch="+ch);
        System.out.println("i="+i);
        System.out.println("d="+d);
        System.out.println("str="+str);
    }
}
```

程序执行结果为：

```
ch=a
i=1
d=1234.56789
str=China
```

2.6.2　输入

1. 使用 System.in 对象

System.in 对象用于在程序运行时从键盘输入数据。在 Java 中输入数据时，为了处理在输入数据的过程中可能出现的错误，需要使用异常处理机制，使得程序具有“健壮性”（异常处理在第 7 章详细介绍）。使用异常处理命令行输入数据有两种格式：

- 使用 try～catch 语句与 read 方法或 readLine 方法相结合。
- 使用 throws IOException 与 read 方法或 readLine 方法相结合。

下面是从键盘读入一个字符，一个字符串或一个整数的程序示例。当程序中需要实现键盘输入功能时可以参考这些例子。

【例 2-4】 从键盘读一个字符。

```
import java.io.*;
/* 引入 java.io 中的类（输入输出类），因为程序中要用到输入类的 read()方法，* 表示引入该包中的所有类 */
public  class ReadChar {
    public static void main(String args[]) {
        try {                                          // 异常处理中的 try 语句
            // 调用 read 方法，读一个字符存入 ch 中
            char ch=(char)System.in.read();
            System.out.print(ch);
        } catch (IOException e) { }    // catch 语句，IOException 为异常
                                       //也可使用 Exception 异常类
    }
}
```

或程序为：

```
import java.io.*;
public  class ReadChar {
    public static void main(String args[]) throws IOException{
        char ch=(char)System.in.read();
        System.out.print(ch);
    }
}
```

【例 2-5】 从键盘读入一个数字串或一个整数并输出。

```
import  java.io.*;
class ReadStringOrInt {          public static void main(String args[])
{
        byte buf[]=new byte[50];         //50 个字节的数组用于存字符串
        String str;
        double anDouble=0.0;             //局部变量要初始化
        try {                            //try 语句
               System.in.read(buf);      //从键盘读一个数字串保存于 buf 中
               //buf 转换成 String 对象 str(ASCII 字符串转换成 Unicode 码串)
               str=new String(buf,0);
               //数字串转换成整数
               anDouble=Double.parseDouble(str.trim());
        } catch (Exception e) { }        //catch 语句，Exception 为异常类
        System.out.println(anDouble);
    }
}
```

程序执行时输入：12345.6789

程序执行结果为：

```
12345.6789
```

Java 程序中将数字字符串转换为数值类型的常用方法如表 2-11 所示。

表 2-11　Java 中字符串转换成数值类型的方法

数据类型	转换方法
long	Long.parseLong(数字字符串)
int	Integer.parseInt(数字字符串)
short	Short.parseShort(数字字符串)
byte	Byte.parseByte(数字字符串)
double	Double.parseDouble(数字字符串)
float	Float.parseFloat(数字字符串)

2. 使用命令行参数

在程序执行时，通过在命令行中输入参数，来获得数据，可通过 main()方法的 args[]参数来实现。main()方法的参数是一个字符串类型的数组，程序从 main()方法开始执行，Java 虚拟机会自动创建一个字符串数组，并将程序执行时输入的命令行参数放在数组中。最后将数组的地址赋给 main()方法的参数。

【例 2-6】　从键盘读入一个数字字符串和一个整数并输出。

```
class ReadFromCommandLine {
   public static void main(String args[ ]) {
      int anInt=0;                                    //局部变量初始化
      System.out.println(args[0]);
      anInt=Integer.parseInt(args[1].trim());         //数字串转换成整数
      System.out.println(anInt);
   }
}
```

在命令行输入下面一行代码：

```
Java ReadFromCommandLine  Java  1234
```

程序执行时输入的参数“Java”和“1234”放入字符串数组的第一个和第二个元素，而 main()中的参数 args 将指向该数组。在程序中把“1234”数字串转换为数值类型，进行输出，程序的输出结果为：

```
Java
1234
```

本章小结

本章主要介绍了 Java 语言的基本要素：标识符、保留字和分隔符；基本数据类型：char、byte、short、int、long、float、double、boolean；基本运算符：算术运算符、关系运算符、逻辑运算符、条件运算符、位运算符、赋值运算符和复合赋值运算符。最后，介绍 Java 程序的简单输入输出。

习题二

2-1 Java 中怎样进行注释?

2-2 Java 中标识符定义的规则有哪些?

2-3 下面字符串中哪些是关键字?

CLASS sizeof abstract NULL INTEGER LONG native import

2-4 下面哪些是 Java 中的标识符?

&Moon9 $_1234computer MyVariance My%INTEGER INT $_$_You 86xyz new class You&Me

2-5 Java 中包含哪些基本数据类型?

2-6 判断下面常量的数据类型:

true 123 3f 8.23e-2 4.6789 345L 'a' "Hello world!"

2-7 Java 运算符有哪些?并给出运算实例。

2-8 在下列符号中不属于字符常量的有哪些?说明理由。

'x' '\101' '\r' '\\' '\%' '\u0030' '+' M s '\='

2-9 已知 int i=10,j=0;计算下面表达式的值:

(1) j=5+++i

(2) j=5+i++

(3) j=8+3*9/7-6

(4) j=i+3*9%i-4

2-10 已知 int i=10,j=20,k=30;计算下面表达式的值:

(1) i<10&&j>10&&k!=10

(2) i<10||j>10||k!=10

(3) !(i+j>k)&&!(k-j>i)

(4) !(i==j)&&!(j==k)&&!(i==k)

2-11 已知 int i=6,j=8;求下面表达式计算后 j 的值:

(1) j+=++i

(2) j-=5+i++

(3) j*=j+3*i*j--;

(4) j+=j-=j*=j

2-12 编写程序,从键盘输入圆的半径,求圆的周长和面积并输出。

2-13 编写程序,从键盘输入平行四边形的底和高,求面积并输出。

2-14 编写程序,实现摄氏温度和华氏温度的转换,要求输入摄氏温度,输出华氏温度,输入华氏温度,输出摄氏温度。摄氏温度和华氏温度的转换公式为:

华氏温度=9×摄氏温度÷5+32

2-15 编写程序,实现英里到公里的转换,其转换公式为:

1 英里=1.6 公里