第 3 章　Java 语言基础

随着甲骨文收购 Sun，Java 语言将迎来新的起点，将会有越来越多的软件开发者选择 Java 语言作为编程的首选语言。Java 语言继承和融合了 C/C++ 的优点，并且沿用了其基本语法格式，这样既符合人们的编程习惯，又适应当今网络编程的需要，因而它成为主流的程序设计语言。本章主要讲解 Java 语言的基本语法、语句类型、控制语句以及数组和字符串。

本章主要内容：

- Java 语言的基本语法
- Java 语言的控制语句
- 数组和字符串

3.1　Java 语言的基本语法

Java 语言使用国际字符格式标准(Unicode)和浮点数(IEEE 754)。Unicode 字符集采用 16 位编码，其前 256 个字符与 ASCII 字符集完全一致。除了数字 0～9、英文字母 A～Z、a～z、下划线(_)、美元符号($)以及＋、－、*、/等 ASCII 字符之外，Unicode 字符集还提供了其他语言文字字符，如汉字、希腊文、韩文等。

在开发全球使用的软件产品时(国际化)，如果使用了不一致的字符编码(即与字符关联的不一致的数字值)，将会产生非常严重的问题，因为计算机是使用数字来处理信息的。例如，字符“a”被转换为数字值，计算机对这个数字进行处理。许多国家和公司开发了各自的编码系统，但互不兼容。例如，值 0XC0 在微软的 Windows 操作系统中表示“带重音记号的 A”，而这个值在 Apple Macintosh 操作系统中表示的是“颠倒的问号”。这会导致数据的错误解释，甚至可能造成数据毁坏。

如果没有统一的字符编码标准，要开发全球通用的软件，开发人员就必须在软件发布之前进行大量的本地化工作，其中包括语言翻译和内容调整。

Unicode 标准(Unicode Standard)正是为了解决这一问题而创建的，它是一个编码标准，有助于软件的开发和发布。Unicode 标准规定了世界字符和符号的一致编码规则。采用 Unicode 标准编码的软件产品仍需要进行本地化，但是其过程更简单有效，因为不需要转换数字值，并且字符编码是统一的。

3.1.1　标识符

Java 标识符(identifier)是以字母开头的字母数字序列。标识符是用户定义的单词，用于标识变量、常量、类、方法、对象和文件等。只要编写程序，就不可避免地使用标识符。

标识符的命名规则如下。

(1) “字母”和“数字”具有宽泛含义。字母通常指大小写英文字母、下划线(_)、美元符

(\$)等,也可以是 Unicode 字符集中的字符,如汉字等。数字通常指 0～9。

(2) 标识符可以是字母、数字等字符的任意组合,除此之外,不能包含其他字符(如+、-及空格等)。

(3) 标识符区分字母大小写,或者说标识符是大小写敏感的。

(4) 标识符不能使用 Java 中的关键字。

(5) 标识符长度不受限制。

符合上述规则的标识符都是正确的 Java 标识符。正确的标识符举例如下:

```
String name;
int age;
double salary;
public class Salary{}
String 性别,籍贯,爱好;
```

而以下则是错误的标识符:

```
int 2x;              //不能以数字开头
double my salary;    //标识符中不能有空格
String x+y;          //+既不是 Java 中的字母,也不是 Java 中的数字,不能使用
String test1-2-3;    //-既不是 Java 中的字母,也不是 Java 中的数字,不能使用
String class;        //标识符不能用关键字
String Java&JSP;     //"&"既不是 Java 中的字母,也不是 Java 中的数字,不能使用
```

在 Java 语言规范中,有些标识符虽然正确,但是不提倡使用,应该规范化书写。因为不规范的命名习惯会大大降低代码的可读性。

在 Java 规范中约定:关键字、变量名、对象名、方法名和包名通常将全部字母小写;如果由多个单词构成标识符,则第一个单词的首字母小写,其后的单词首字母大写,如 toString;类名首字母大写,如 FirstJava;常量名全部字母均大写,如 BOOK。

3.1.2 关键字

关键字(Keyword)是 Java 语言保留的具有特定含义的英文单词。每一个关键字都有一种特定含义,不能被赋予别的含义,也不能把关键字作为标识符来使用。Java 中的关键字以及其含义如表 3-1 所示。

表 3-1 Java 关键字及其含义

关键字	含　义
abstract	用于声明类或者方法的抽象属性
boolean	数据类型关键字,布尔类型
break	控制转移关键字,提前跳出一个块
byte	数据类型关键字,字节类型
case	用在选择/条件语句 switch 中,表明其中的一个分支
catch	用在异常处理中的关键字,用来处理异常

续表

关键字	含　义
char	数据类型关键字，字符类型
class	用于声明类的关键字
continue	控制转移关键字，用于回到一个块的开始处
default	默认值，例如，用在 switch 语句中，表明一个默认的分支
do	循环语句关键字，用在 do-while 循环结构中
double	数据类型关键字，双精度浮点数类型
else	用在选择/条件语句 if 中，表明当条件不成立时的分支
enum	枚举类型关键字
extends	声明一个类继承另外一个类，该类是另外一个类的子类
final	用来声明最终属性，表明一个类不能派生出子类，或者方法不能被覆盖，或者变量的值不能被更改
finally	用于处理异常情况，用来声明一个基本肯定会被执行到的语句块
float	数据类型关键字，单精度浮点数类型
for	一种循环结构的关键字
if	选择/条件语句的关键字
implements	声明一个类用于实现接口
import	导入需要使用的指定类或包
instanceof	用来测试一个对象是否是指定类型的实例对象
int	数据类型关键字，整数类型
interface	声明一个接口
long	数据类型关键字，长整数类型
new	用来创建实例对象
package	用来创建一个包
private	私有访问权限
protected	受保护的访问权限
public	公有访问权限
return	从成员方法中返回数据
short	数据类型关键字，短整数类型
static	声明静态属性
super	调用当前对象父类的内容
switch	分支结构语句的关键字
synchronized	表明代码需要同步执行

续表

关键字	含　义
this	指向当前实例对象
throw	抛出异常
throws	在当前定义的成员方法中所需要抛出的异常
try	可能抛出异常的程序块
void	无返回值
while	循环控制语句

读者在学完本书后，就能够掌握上述关键字的用途。

3.1.3　分隔符

为便于阅读，程序也需要如同自然语言一样恰当地使用分隔符。这些分隔符不能互相代用，即该用空格的地方只能用空格，该用逗号的地方只能用逗号。圆括号、大括号、中括号、空格、逗号和分号等称为分隔符，Java 规定任意两个相邻标识符、数字、关键字或两个语句之间必须至少有一个分隔符，以便编译程序时能识别。

常用的分隔符如下：

(1) 圆括号()：在方法声明和调用时可以包括一组参数；在控制语句或者强制类型转换中用于数据的执行和数据类型的转换。

(2) 大括号{}：在类、方法体、语句块以及初始化数组中的值声明时使用。

(3) 中括号[]：在声明数组以及在访问数组元素中使用。

(4) 空格“ ”：在源代码中用空格符改善源代码的书写形式，可以分割相邻的两个语法符号，使程序易读，空格符号可以是空格、Tab 制表符、回车符和换行符等。

(5) 逗号“,”：在声明多个变量或在方法中的参数之间等可以使用。

(6) 分号“;”：在语句结束以及 for 控制语句中等可以使用。

(7) 句号“.”：在调用方法和变量中可以使用。

【例 3-1】 分隔符的使用(Separator.java)

```
/*
    功能简介：本程序演示分隔符的使用。首先声明一个数组，然后通过 for 语句把数组的值取出
    并相加，最后输出计算结果。
*/
public class Separator
{
    public static void main(String args[]){
        //声明整型变量 i 和 sum 并初始化
        int i,sum=0;
        //声明整型数组
        int a[]={1,2,3,4,5};
        //控制语句和数组遍历
        for(i=0;i<a.length;i++)
```

```
        {
            //遍历数组并计算结果
            sum+=a[i];
        }//for 结束
        System.out.println(sum);
    }//main()方法结束
}//类 Separator 结束
```

3.1.4 数据类型

程序在执行的过程中需要调用数据进行运算，同时也需要存储数据。获取的数据可能是从键盘输入，也可能是从文件中取出，甚至是从网络上得到。在程序的执行过程中，数据存储在内存中，便于程序的使用。

数据是描述客观事物的数字、字符以及所有能输入到计算机中并能被计算机接收的各种符号集合。数据是计算机程序的处理对象的状态。

类型是具有相同逻辑意义的一组值的集合。数据类型是指一个类型和定义在这个类型上的操作集合。数据类型定义了数据的性质、取值范围以及对数据所能进行的运算和操作。

程序中的每一个数据都属于一种数据类型。决定了数据的类型也就相应决定了数据的性质以及对数据进行的操作，同时数据也受到类型的保护，确保对数据不进行非法操作。

Java 语言中的数据类型分为两大类：基本数据类型和引用数据类型，如图 3-1 所示。

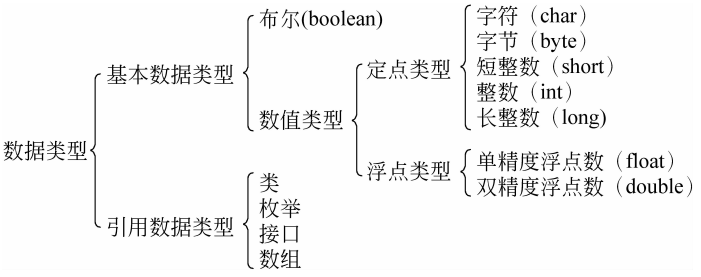

图 3-1　Java 数据类型

1. 布尔类型

布尔类型(boolean)，又称逻辑类型，只有两个取值 true(真)和 false(假)，它们全是小写，计算机内部用 8 位来表示。Java 语言不允许数值类型和布尔类型之间进行转换。而在 C/C++ 中，允许用数值表示，如用 0 表示 false，非 0 表示 true。Java 语言中不允许这样做。

所有关系运算(如 a<b)的结果值都是布尔类型的。布尔类型也用于流程控制语句中的条件表达式，如 if、for 和 while 等语句中。

2. 整数类型

整数类型是具有固定上、下界的整数，包括正整数、零和负整数，与数学中的整数概念并不完全一样。Java 语言根据数据在内存中占用的位数不同提供了 4 种整数类型，分别是 byte、short、int 和 long，它们的位数递增，表示数的范围也越来越大。Java 的整数类型如表 3-2 所示。

表 3-2　Java 整数类型

整数类型	长度/位	字节数/B	取 值 范 围
byte	8	1	−128～127
short	16	2	−32 768～32 767
int	32	4	−2 147 483 648～2 147 483 647
long	64	8	−9 223 372 036 854 775 808～9 223 372 036 854 775 807

整数型常量有 3 种表示形式：

（1）十进制整数。如 56、−24、0。

（2）八进制整数。以 0 开头的数是八进制整数。如 017。

（3）十六进制整数。以 0x 开头的数是十六进制整数。如 0x17、0x0、0xf、0xD。十六进制整数可以包含数字 0～9、字母 a～f 或 A～F。

3. 浮点类型

浮点类型是包含有小数部分的数值。Java 中的浮点类型按其取值范围的不同可区分为 float(浮点类型)和 double(双精度型)两种，如表 3-3 所示。

表 3-3　Java 浮点类型

浮点类型	长度/位	字节数/B	取 值 范 围
float	32	4	−3.402 823 47E+38F～3.402 823 47E+38F
double	64	8	−1.797 693 134 862 315 7E+308～1.797 693 134 862 315 7E+308

Java 用浮点数类型表示数学中的实数，一个浮点数值包括整数部分和小数部分。浮点数有两种表示方式：

（1）标准记数法。由整数部分、小数点和小数部分构成。如 1.0、123.45 等。

（2）科学记数法。由十进制整数、小数点、小数和指数部分构成，指数部分由字母 E 或 e 跟上带正负号的整数表示。

在浮点型常量后加上 f 或 F 表示单精度。如 2.112.3e3F、2.6f、2.6F。在浮点型常量后不加任何字符或加 d 或 D 表示双精度，如 2.11、2.3e3、2.3e3d、2.3e3D、2.6、2.6d、2.6D。

4. 字符类型

字符类型(char)数据是由一对单引号('')括起来的单个字符，该字符是 16 位的 Unicode 码。

字符常量有两种表示法：

（1）用单引号将 ASCII 字符括起来的值，如'A'、'a'、'V'等。

（2）用 Unicode 值表示。

另外，用“\”开头，后面跟一个字母表示某个特定的控制符，这就是 ASCII 控制符，常称为转义字符。Java 中的转义字符如表 3-4 所示。

表 3-4　转义字符

转义字符	含义	转义字符	含义
\'	单引号	\b	退格
\"	双引号	\t	制表符

续表

转义字符	含义	转义字符	含义
\r	回车符	\\	反斜杠
\n	换行符		

char 类型在 Java 语言中不常用，一般使用字符串类型。字符串类型是用一对双引号括起来的字符序列，字符串由 String 类实现。

【例 3-2】 转义字符的使用(转义字符.java)

```
/*
    功能简介：表 3-4 中转义字符的使用。
*/
public class 转义字符
{
    public static void main(String args[]){
        System.out.println( "\"中国\n 欢迎\n 你!\t\"" );
    }
}
```

5. 类型转换

Java 程序中的类型转换可分为显式类型转换和隐式类型转换两种形式。

1) 隐式类型转换

对于由二元运算中的算术运算符组成的表达式，一般要求运算符两边的两个操作数的类型一致，如果两者的类型不一致，则系统会自动转换为较高(即取值范围较大)的类型，这便是隐式数据类型转换。有关运算符和表达式请参考 3.1.6 节。

2) 显式类型转换

隐式类型转换只能由较低类型向较高类型转换，但是在实际工作中，有时也可能需要由较高类型向较低类型转换。例如，在计算数值时为了保证其精度，为某些变量取了较高的数据类型(如 double 型)，但在输出时，往往只需要保留两三位小数或者只输出整数，这时只能进行显式类型转换。显式类型转换需要人为地在表达式前面指明所需要转换的类型，系统将按这一要求把某种类型强制性地转换为指定的类型。格式如下：

(<类型名>)<表达式>

基本数据类型之间的相互转换规则如表 3-5 所示。

表 3-5 基本数据类型的转换

	char ch;	byte b;	short s;	int i;	long k;	float f;	double d;
char ch;		ch=(char)b;	ch=(char)s;	ch=(char)i;	ch=(char)k;	ch=(char)f;	ch=(char)d;
byte b;	b=(byte)ch;		b=(byte)s;	b=(byte)i;	b=(byte)k;	b=(byte)f;	b=(byte)d
short s;	s=(short)ch;	s=b;		s=(short)i;	s=(short)k;	s=(short)f;	s=(short)d;
int i;	i=ch;	i=b;	i=s;		i=(int)k;	i=(int)f;	i=(int)d;
long k;	k=ch;	k=b;	k=s;	k=i;		k=(long)f;	k=(long)d;
float f;	f=ch;	f=b;	f=s;	f=i;	f=k;		f=(float)d;
double d;	d=ch;	d=b;	d=s;	d=i;	d=k;	d=f;	

【例 3-3】 类型转换的使用(类型转换.java)

```
/*
    功能简介:使用类型转换将数据类型进行转换。
*/
public class 类型转换
{
    public static void main(String args[]){
        int x=100;
        //隐式类型转换
        long y=x;
        System.out.println("类型转换:整型"+x+"转换为长整型"+y);
        double d=1212;
        //显式类型转换,强制类型转换
        int a=(int)d;
        System.out.println("类型转换:double 类型"+d+"转换为 int 类型"+a);
    }
}
```

3.1.5 常量和变量

任何一种程序设计语言都要使用和处理数据,而数据又可以区分为不同的类型。Java 语言中提供常量和变量来存储数据。

常量是指在程序的整个运行过程中其值始终保持不变的量。在 Java 语言中,常量有两种形式:一种是以数字形式直接给出值的常量;另一种则是以关键字 final 定义的标识符常量。不论哪种形式的常量,一旦声明,在程序的整个运行过程中其值始终不会改变。

变量是在程序的运行过程中其值可以被改变的量。变量除了区分为不同的数据类型外,更重要的是每个变量还具有变量名和变量值两重含义。变量名是用户自己定义的标识符。通过指明变量所属的数据类型,将相关的操作封装在数据类型中。

1. 常量

常量是在程序执行中其值不能被改变的量。

1) 直接常量和符号常量

常量有两种形式:直接常量和符号常量。

直接常量是指在程序中直接引用的常量,包括数值型常量和非数值型常量。其中,数值型常量称为常数,包括整数和浮点数,如 10、-10.16 等;非数值型常量有字符常量、字符串常量和布尔常量,如'X'、"abc"、true 等。

符号常量是以标识符形式出现的常量,符号常量必须先声明后使用。声明符号常量可以提高程序的可读性,使程序易于修改。

2) 常量声明

常量声明形式与变量声明形式基本一样,需要使用关键字 final。

格式如下:

```
[修饰符] final 数据类型 常量标识符[=常量值];
```

修饰符：用于表示数据属性的权限，如 public、默认值等。

final：用于变量和类的声明。

数据类型：可以是基本数据类型。

例如：

```
public final int MAX=10;
final float PI=3.14f;
```

Java 语言约定常量标识符全部用大写字母表示。标识符一旦被声明为常量，就不能再做他用。声明常量的好处有两点：一是增加可读性，从常量名可知常量的含义；二是增强可维护性，只要在常量声明处修改常量值，即可改变程序中多处使用的常量值。

2. 变量

Java 中的变量具有 4 个基本属性：变量名、数据类型、存储单元和变量值。变量名是变量的名称，用合法的标识符表示；数据类型可以是基本数据类型和引用类型；每个数据都有一个存储单元，大小由数据类型决定；变量值是指在变量存储单元中存放的值。

前面提到了 Java 是强类型语言，必须对数据先声明后使用。变量也是这样，必须先声明后使用。变量在声明时可以初始化，也可以在使用时进行赋值或者调用时传参数。变量声明的格式如下：

[修饰符] 类型 变量标识符 [=初始化表达式];

例如：

```
public String str1="姓名";
private String str2;
int x,y,z=10;
double a,b,c=5.5;
```

【例 3-4】 变量声明(VariableDeclaration.java)

```
/*
    功能简介：声明一些变量并初始化值，然后输出这些值。
*/
public class VariableDeclaration
{
    public static void main(String args[]){
        boolean b=true;
        byte b1=1;
        short s=2;
        int i=3;
        long l=4;
        float f=1.11f;
        double d=2.222d;
        char c='x';
        String s1="我爱学习 Java 课程!";
        System.out.println("b="+b);
```

```
        System.out.println("b1="+b1);
        System.out.println("s="+s);
        System.out.println("i="+i);
        System.out.println("l="+l);
        System.out.println("f="+f);
        System.out.println("d="+d);
        System.out.println("c="+c);
        System.out.println("s1="+s1);
    }
}
```

3.1.6 运算符与表达式

程序中对数据的操作实际上是指对数据的运算。表示运算类型的符号称为运算符，参与运算的数据称为操作数。运算符把操作数按 Java 语法规则连接起来组成表达式。运算符和表达式构成了程序中完成各种运算任务的语句，是程序设计的基础。

1. 运算符

Java 提供了多种运算符。按参与运算的操作数的数目可分为一元运算符、二元运算符和三元运算符。按照运算符的功能可分为算术运算符、关系运算符、布尔逻辑运算符、位运算符以及赋值、条件、实例等其他运算符。

1) 算术运算符

算术运算符包括＋、－、＊、/、＋＋、－－和％，使用整型、字符或浮点型操作数。其中＋和－分别具有正和负、加和减两种不同含义，并根据不同含义分别属于一元或二元运算符。其他算术运算符都是二元运算符。

对于除法运算(/)，若操作数均为整数，结果也为整数；若有浮点数参与运算，结果为浮点数。例如，3/6 的结果为 0，3.0/6 的结果为 0.5。

取模运算(％)可用于整型或浮点型操作数，运算结果的符号与第一个操作数的符号相同。例如：

```
10%-3;    //结果为 1
-10%3;    //结果为-1
```

自增运算符(＋＋)和自减运算符(－－)是一元运算符，分别用于实现将变量的值增 1 和减 1，要求操作数必须是整型或字符型变量。自增、自减运算符的前置运算是先实施自增、自减运算，再使用自增、自减后的操作数的值；后置运算是先使用操作数的值，再对操作数实施自增、自减运算。例如：

```
int x=2;            //定义变量 x,并赋初值为 2
                    //最后 y、x 的值分别为 2、3
int y=x++;          //定义变量 y,先用 x 的值给 y 赋值,再令 x 增 1
int z=--x;          //定义变量 z,先令 x 减 1,再用 x 的值给 z 赋值
                    //最后 z、x 的值均为 2
```

Java 对加运算进行了扩展，使它能够进行字符串的连接，如"abc"＋"de"，可得到字符