第3章 Java程序设计基础

学习目的与要求

要想应用Java语言进行程序设计必须学习其基本语言知识。通过本章的学习将能够掌握Java语言的基本语法、基本编程结构和方法，学会简单的Java语言程序设计。本章是程序设计的基本知识和技巧，应该牢固掌握。

本章主要内容

- 标识符、关键字；
- Java语言的数据类型；
- Java语言的运算符和表达式；
- 程序控制流程(顺序、选择和循环)。

3.1 标识符、关键字及数据类型

任何程序设计语言，都是由语言规范和一系列开发库组成的。如标准C，除了语言规范外，还有很多函数库；Java语言也不例外，也是由Java语言规范和Java开发类库组成的。

学习任何程序设计语言，都要从这两方面着手，关于常用的Java开发类，会在后面章节介绍，这里主要介绍Java基础语法。本章讲解Java标识符、关键字与数据类型，运算符与表达式以及程序设计使用的控制语句。

3.1.1 标识符

程序中所用到的每一个变量、方法、类和对象的名称都是标识符(Identifier)。都应该有相应的名称作为标识。把给程序中的实体——变量、常量、方法、类和对象等所起的

名称为标识符。简单地说,标识符就是一个名称。

Java 语言的标识符应遵循如下规则:

- 只能由英文字母 A～Z 或 a～z、下划线(_)、美元符号($)和数字 0～9 组成,而且这样的字符序列开头必须是英文字母、下划线或美元符号($)。变量名不能使用空格、加号“+”、减号“-”、逗号“,”等符号。
- 标识符区分大小写。如 sum、Sum 和 SUM 是三个不同的标识符。为避免引起混淆,程序中最好不要出现仅靠大小写字母区分的相似标识符。
- 不允许使用 Java 关键字来命名。因为关键字是系统已经定义过的具有特殊含义的标识符(Java 语言的关键字参见 3.1.2 节)。另外,还有一些名称虽然不是关键字,但是系统已把它们留作特殊用途,如系统使用过的函数名等,用户也不要使用它们作为标识符(如 main),以免引起混乱。
- 标识符没有长度限制,但不建议使用太长的标识符。
- 标识符命名原则应以直观且易于拼读为宜,即做到“见名知意”,最好使用英文单词及其组合,这样便于记忆和阅读。
- Java 中标识符命名约定:常量用大写字母,变量用小写字母开始,类以大写字母开始。

例如,下面是合法的标识符:

b, a3, _switchstudentName,Student_Name,_my_value, $address

下面是非法的标识符:

ok?,"abc", a.b, 2teacher,a+b, room#,abstract(这是一个关键字)

请读者自己分析。

注意:为了提高 Java 程序的可读性,Java 源程序有一些约定俗成的命名规定。

(1) 包名:包名是全小写的名词,中间可以由点分隔开,如 java.awt.event。

(2) 类名:首字母大写,通常由多个单词合成一个类名,要求每个单词的首字母也要大写,如 class HelloWorldApp。

(3) 接口名:命名规则与类名相同,如 interface Collection。

(4) 方法名:往往由多个单词合成,第一个单词通常为动词,首字母小写,中间的每个单词的首字母都要大写,如 balanceAccount、isButtonPressed。

(5) 变量名:全小写,一般为名词,如 length。

(6) 常量名:基本数据类型的常量名为全大写,如果是由多个单词构成,可以用下划线隔开,例如,int YEAR, int WEEK_OF_MONTH;如果是对象类型的常量,则是大小写混合,由大写字母把单词隔开。

3.1.2 关键字

关键字又称作保留字(reserved word),它们具有专门的意义和用途,不能当作一般的标识符使用。

Java 保留字包括：

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

注意：

(1) true、false 和 null 通常为小写，而不是像在 C++ 语言中那样大写。严格地讲，它们不是关键字，而是文字。然而，这种区别是理论上的。

(2) Java 没有 sizeof 运算符；所有类型的长度和表示是固定的。

(3) 在 Java 编程语言中不使用 goto 和 const 作为关键字，尽管它们在其他语言中常用，但不能用 goto、const 作为变量名。

3.1.3 数据类型

Java 语言提供了丰富的数据类型，分为基本类型（又叫做原始类型）和引用类型两大类，如图 3-1 所示。

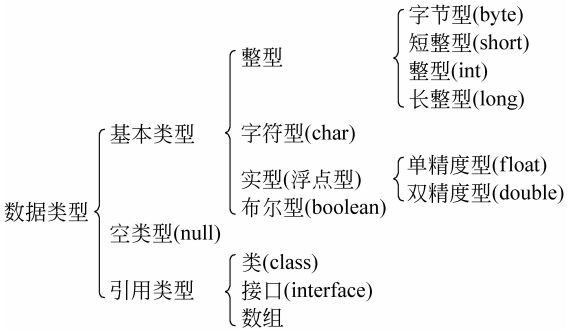

图 3-1 数据类型

1. 常量

数值不能再变化的变量叫做常量，需要使用关键字 final 修饰。其定义格式为：

```
final Type varName=value [, varName [=value] …];
```

例如：

```
final  int   MAXLEN=1000;
final  double  PI=3.1415926;
```

一般情况下，常量名用大写字母标识，如果由几个单词组成，不同单词之间使用“_”连接，例如，NAME，STUDENT_NAME 等。

2. 变量

在程序中使用变量之前必须对变量预先定义或赋值，一方面符合程序运行的要求，此外还使得程序的可读性更强。

变量是程序中的基本存储单元，其定义包括变量名、变量类型和作用域几个部分。格式如下：

```
Type varName=value [, varName [=value] …];
```

其中：

(1) Type 代表变量数据类型。Java 的数据类型分为基本类型(又叫做原始类型)和引用类型两种。

(2) varName 代表变量名，也就是上面讲到的标识符。在一定的作用域内，变量名必须是唯一的。

(3) value 代表变量值。对变量的定义实际上分为两步：

第一步是变量声明(Declaration)，例如：int x;。

第二步是变量赋值(Assignment)，例如：x=10;。第一次赋值也叫做初始化。

有时会合并这两步，例如：int x=10; int y=x;。

(4) 变量的作用域：变量的作用域是指变量在程序中的作用范围，它分为全局变量和局部变量。全局变量是作用于全程序范围的变量(即在整个程序中均有效)，它在方法体外声明。也就是说，在同一个页面的所有脚本都可以随时使用它。局部变量是在方法内定义的变量，它仅在该方法内的语句中起作用。

局部变量和全局变量可以同名，当在方法中定义局部变量与全局变量同名时，方法内的变量名引用指的是该方法内的局部变量，而不是全局变量。而局部变量定义以外的变量引用则指全局变量。

3. 基本类型

Java 定义了 8 个基本数据类型：字节型(byte)、短整型(short)、整型(int)、长整型(long)、字符型(char)、单精度型(float)、双精度型(double)和布尔型(boolean)。

这些类型可分为 4 组。

- 整型：该组包括字节型(byte)、短整型(short)、整型(int)和长整型(long)，它们代表有符号整数。
- 浮点型：该组包括单精度型(float)和双精度型(double)，它们代表有小数精度要求的数字。
- 字符型：该组包括字符型(char)，它代表字符集的符号，例如字母和数字。
- 布尔型：该组包括布尔型(boolean)，它是一种特殊的类型，表示真/假值。

可以按照定义使用它们，也可以构造数组或类的类型（后面会讲解）来使用它们。这样，它们就构成了用来创建所有其他类型数据的基础。

基本数据类型代表单值，而不是复杂的对象。Java 是完全面向对象的，但基本数据类型不是对象，它们类似于其他大多数非面向对象语言的基本数据类型。这样做的原因是出于效率方面的考虑。在面向对象中引入基本数据类型不会对执行效率产生太多的影响。当然，Java 也提供了对基本数据类型的封装类型（Wrapper Class），这在后面会详细介绍。

所有基本类型所占的位数都是确定的，并不因操作系统的不同而不同如表 3-1 所示。

表 3-1　基本数据类型的位数和范围

数据类型	所占位数	数的取值范围	举　例
char	16	0～65 535	int x1，x2；
byte	8	$-2^7 \sim 2^7-1$（−128～127）	unsigned int y1，y2；
short	16	$-2^{15} \sim 2^{15}-1$（−32 768～32 767）	short z1，z2；
int	32	$-2^{31} \sim 2^{31}-1$	unsigned short f1，f2；
long	64	$-2^{63} \sim 2^{63}-1$	long h1，h2；
float	32	$-3.4e^{038} \sim 3.4e^{+038}$	unsigned long k1，k2；
double	64	$-1.7e^{308} \sim 1.7e^{+308}$	double x，y；
boolean	8	true，false	boolean t；

下面依次讨论每种数据类型。

1）整型

计算机中的数据都以二进制形式存储。在 Java 程序中，为了便于表示和使用，整型数据可以用以下几种形式表示，编译系统会自动将其转换为二进制形式存储。

（1）整型常量

整数可能是在典型的程序中最常用的类型。用来表达整数的方式有 3 种：十进制（通常使用的方式，基数是 10）、八进制（octal，基数是 8）和十六进制（hexadecimal，基数是 16）。

十进制整型数的数字由 0～9 表示，十进制无前缀。八进制是一种常用的表示形式。表示八进制数时，前边加 0，即加前缀 0。八进制数的数字由 0～7 表示。表示十六进制数时，前缀加 0x 或 0X。表示十六进制数的数字由 0～9 和 a～f 或 A～F 组成，如表 3-2 所示。

表 3-2　整型常量按进制分类

分　类	表 示 方 法	说　明	举　例
十进制整数	一般表示形式	逢十进一	100 表示十进制数 100
八进制整数	以 0 开头	逢八进一	0100 表示八进制数 100
十六进制整数	以 0x 开头	逢十六进一	0x100 表示十六进制数 100

十进制整数形式：与数学上的整数表示相同。例如：

```
123, -567, 0
```

八进制整数形式：在数码前加数字 0。

以 0 开头，如 0123 是八进制数表示十进制数 83，－011 是八进制数表示十进制数－9，012 表示十进制数 10。

十六进制整数形式：以 0x 或 0X 开头，如 0x123 表示是十六进制数表示十进制数 291，－0X12 表示十进制数－18。

(2) 整型变量

类型为 byte、short、int 或 long，其中 byte 在机器中占 8 位，short 占 16 位，int 占 32 位，long 占 64 位。如果没有明确指出一个整数值的类型，那么它默认为 int 类型。

整型变量的定义如下：

```
byte by1;                          //定义变量 by1 为 byte 型
short  a;                          //定义变量 a 为 short 型
int x=123;                         //定义变量 x 为 int 型,且赋初值为 123
long y=123L; 或 long z=1231;       //定义变量 y 为 long 型,且赋初值为 123
```

由此可见，一个整数值可以被赋给一个 long 变量。但是，定义一个 long 变量，用户需要明白地告诉编译器变量的值是 long 型，可以通过在变量的后面加一个大写或小写的 L。长整型必须以 L 做结尾，如 9L、156L。

2) 实型

(1) 实型常量

由于计算机中的实型数据是以浮点形式表示的，即小数点的位置可以是浮动的，因此，实型常量既可以称为实数，也可以称为浮点数。浮点数常量有 float(32 位)和 double(64 位)两种类型，分别叫做单精度浮点数和双精度浮点数，表示浮点数时，要在后面加上 f(F)或者 d(D)。

Java 语言的实型常量有两种表示形式：标准记数法形式和科学记数法形式。

① 标准记数法形式由数字和小数点组成。小数点前表示整数部分，小数点后表示小数部分，具体格式如下：

```
<整数部分>.<小数部分>
```

② 科学记数法形式即指数形式。该种表示形式包含数值部分和指数部分。数值部分表示方法同十进制小数形式，指数部分是一个可正可负的整型数，这两部分用字母 e 或 E 连接起来。具体格式如下：

```
<整数部分>.<小数部分>e<指数部分>
```

其中，e 左边部分可以是＜整数部分＞.＜小数部分＞，也可以只是＜整数部分＞，还可以是".＜小数部分＞"；e 右边部分可以是正整数或负整数，但不能是浮点数，如表 3-3 所示。

说明：10e2 表示 10 乘以 10 的 2 次幂，1.56e3 表示 1.56 乘以 10 的 3 次幂。

注意：在表示指数形式时，e 前必须有数字，e 后必须为整数。例如：10e2 也可写为 10e＋02。

表 3-3　实型常量表示方法

表 示 方 法	说　　明	举　　例
标准记数法形式	由数字和小数点组成	0.123、123.79
科学记数法形式	由尾数、字母 e 或 E 和指数组成	1.23e3 或 1.23E3

提示：使用指数形式来表示很大或很小的数比较方便。

(2) 实型变量

实型数据类型为 float(单精度型)或 double(双精度型)，其中 float 在机器中占 32 位，double 占 64 位。

实型变量的定义如下：

```
double x=0.123;                          //定义变量 x 为 double 型,且赋初值为 0.123
float y=0.123F;  或 float z=0.123f;      //定义变量 x 为 float 型,且赋初值为 0.123
```

各种类型的实数的范围及所占字节数如表 3-4 所示。

表 3-4　实型变量类型

类型	所占字节数	数的取值范围	举　　例
float	4	$-3.4e^{038}\sim3.4e^{+038}$	float x1,x2;
double	8	$-1.7e^{308}\sim1.7e^{+308}$	double y1,y2;

注意：Java 中的实型变量默认是双精度型(double)。为了指明一个单精度型变量，必须在数据后面加 F 或 f，若加 d 或 D 为 double 类型。

例如：3.6d、2e3f、.6f、1.68d、7.012e+23f 都是合法的。

3) 字符型数据

(1) 字符常量

字符常量是由英文字母、数字、转义序列、特殊字符等字符所表示，它的值就是字符本身。字符常量是用单引号括起来的单个字符，Java 中的字符占用两个字节。例如：'J'、'@'、'1'。另外 Java 中还有以\开头的具有特殊含义的字符，称为转义字符。常用的转义字符如表 3-5 所示。

表 3-5　Java 中的转义字符

字符形式	说　　明
\n	换行
\t	横向跳格(即跳到下一个 tab 位置)
\b	退格
\r	回车
\f	走纸换页
\\	反斜杠字符"\"

续表

字符形式	说　明
\'	单引号(撇号)字符
\"	双引号字符
\0	空字符
\ddd	3 位八进制数所代表的字符(d 介于 0～7 之间)
\uxxxx	4 位十六进制数所代表的字符(x 介于 0～f 之间)

表中列出的转义字符,意思是将反斜杠(\)后面的字符转换成另外的意义。如'\n'中的 n 不代表字母 n 而作为"换行"符。

表中的最后两行是用 Unicode 码(八进制和十六进制)表示的一个字符,例如'\101'和'\u0041'都代表 Unicode 码(十进制)为 65 的字符"A"。请注意'\0'或'\000'是代表 Unicode 码为 0 的控制字符,即"空操作"字符,它将用在字符串中。

字符串常量与字符常量的区别是,字符串常量是由双引号括起来的字符序列,如"abc"、"a"、"Let's learn java!"。

注意:不要将字符常量与字符串常量混淆。'a'是字符常量,"a"是字符串常量,二者不同。后面可以使用字符串常量初始化一个 String 类的对象。关于字符串处理将在后面的章节中详细介绍。

(2) 字符变量

在 Java 语言的字符变量只有一种定义形式:

char 变量名;

字符变量用来存放字符常量,注意只能存放一个字符,在机器中占 16 位,如表 3-6 所示。

表 3-6　字符型变量类型

类型	所占字节数	说　明	数据的取值范围	举 例
char	2	存放单个字符	0～65 535	char c1,c2='a';

例如字符型变量的定义如下:

```
char  c='a';              //定义变量 c 为 char 型,且赋初值为'a'
char  c1='\n';            //定义变量 c 为 char 型,且赋初值为'\n',转义字符,表示换行
```

Java 的 char 与 C 或 C++ 中的 char 不同。在 C/C++ 中,char 的位宽是 8 位整数。但 Java 不同,Java 使用 Unicode 码代表字符。Unicode 定义的国际化字符集能表示迄今为止人类语言的所有字符集。它是几十个字符集的统一,如拉丁文、希腊语、阿拉伯语、古代斯拉夫语、希伯来语、日文片假名、匈牙利语等,因此,它要求每个字符占 16 位。这样,Java 中的 char 类型是 16 位,其范围是 0～65 536,没有负数的 char。人们熟知的标准字符集 ASCII 码的范围仍然是 0～127,扩展的 8 位字符集 ISO-Latin-1 的范围是 0～255。

下面的程序演示了字符型变量的使用。

【例 3-1】 字符型变量的使用。

```
package sample;
public class CharTest1{
  public static void main(String args[]){
    char ch1,ch2;
    ch1=65;                                          //A 字符的 Unicode 代码值
    ch2='B';
    System.out.println("ch1 and ch2: "+ch1+" "+ch2);
  }
}
```

该程序的输出结果如下：

```
ch1 and ch2: A B
```

变量 ch1 被赋值 65，它是 ASCII 码（Unicode 码也一样）用来代表字母 A 的值。前面已提到，ASCII 字符集占用了 Unicode 字符集的前 127 个值，因此以前使用过的一些字符概念在 Java 中同样适用。

尽管 char 不是整数，但在许多情况下可以对它们进行运算操作。例如，可以将 2 个字符相加，或者对一个字符变量值进行增量操作。

下面的程序演示了字符型变量的运算。

【例 3-2】 字符型变量的运算。

```
package sample;
public class CharTest2{
  public static void main(String args[]){
    char ch1;
    ch1='A';
    ch1++;
    System.out.println("ch1 is: "+ch1);
  }
}
```

程序的输出结果如下：

```
ch1 is: B
```

在该程序中，变量 ch1 首先被赋值为 A，然后递增 1。结果是 ch1 将代表字符 B，即在 ASCII（以及 Unicode）字符集中 A 的下一个字符。

上面介绍了几种基本类型，在一个程序中可能会出现多种类型，下面来看一个实例。

【例 3-3】 几种基本类型变量的定义与使用。

```
package sample;
public class AssignTest{
  public static void main(String args []){
     int x, y;                                       //定义整型变量
     float z=3.414f;                                 //定义单精度型变量并赋初值
```

```
    double w=3.1415;                          //定义双精度型变量并赋初值
    char c;                                   //定义字符型变量
    String str;                               //定义字符串变量
    String str1="bye";                        //定义字符串变量并赋初值
    c='A';                                    //赋初值
    str="Hello! Welcome!";
    x=6;
    y=1000;
    System.out.println("int x="+x);
    System.out.println("int y="+y);
    System.out.println("float z="+z);
    System.out.println("double w="+w);
    System.out.println("char c="+c);
    System.out.println("string str="+str);
  }
```

程序的输出结果如下：

```
int x=6
int y=1000
float z=3.414
double w=3.1415
char c=A
string str=Hello! Welcome!
```

4）布尔型

布尔型数据只有两个值 true 和 false，且它们不对应于任何整数值。

布尔型变量的定义如下：

```
boolean b1=true;      //语句声明变量 b1 为 boolean 类型,它被赋予的初值为 true
```

同样：

```
boolean b2=false;     //语句声明变量 b2 为 boolean 类型,它被赋予的值为 false。
```

下面的程序说明了布尔型数据的使用。

【例 3-4】 布尔型数据的使用。

```
package sample;
public class BooleanTest{
  public static void main(String args[]){
    boolean a;
    a=true;
    System.out.println("It is true.");
    a=false;
    System.out.println("It is false.");
  }
}
```