第3章 chapter 3

Java 语言基础

本章要点

本章介绍 Java 语言编程技术。主要内容包括 Java 语言简介、Java 语言的基本语法、Java 语言的类与对象。

3.1 Java 语言简介

本章主要介绍 Java 语言编程技术，通过本章的学习，读者可以掌握 Java 语言的基本语法、Java 语言的类与对象，并结合案例加以练习，最终实现在 HTML 页面中熟练编写 Java 脚本，完成 JSP 页面的开发。

3.1.1 Java 的由来

自计算机问世以来，计算机技术的发展可谓日新月异。尤其是进入 21 世纪以后互联网技术异军突起，使得基于网络的应用更加普遍。Java 出现之前，Internet 上的信息只是一些简单的 HTML 文档，节点之间仅可以传送一些文本和图片，交互性能较差。同时，Internet 的发展又使得 Internet 的安全问题显得极为突出。这些都是传统的编程语言所无法解决的。

Java 语言起初是 Sun 公司开发的一种与平台无关的软件技术，是为一些消费性电子产品而设计的一个通用环境。后来 Sun 公司将这种技术应用于 Web 上并于 1994 年开发出了 Hot Java(Java 的第一个版本)。1995 年 Sun 公司正式推出了 Java 语言。

3.1.2 Java 的特点

Java 语言的出现实现了页面的互动，而且 Java 语言以其平台无关性、强安全性、语言简洁、面向对象以及适用于网络等特点，已成为 Internet 网络编程语言事实上的标准。

1. 平台无关性

Java 语言引进虚拟机原理，并运行于虚拟机，用 Java 编写的程序能够运行于不同平台上。

2. 简单性

Java语言去掉了C++语言的许多功能,又增加了一些很有用的功能,使得Java语言的功能更加精准。

3. 面向对象

Java语言类似于C++语言,继承了C++语言的面向对象技术,是一种完全面向对象的语言。

4. 安全性

Java语言抛弃了C++的指针运算,程序运行时,内存由操作系统分配,从而避免病毒通过指针侵入系统,同时,Java语言对程序提供了安全管理器,防止程序的非法访问。

5. 分布性

Java语言建立在扩展TCP/IP网络平台上。库函数提供了用HTTP和FTP协议传送和接收信息的方法。这使得程序员使用网络上的文件和使用本机文件一样容易。

6. 动态性

Java语言适应动态变化的环境。Java程序需要的类能动态地被载入到运行环境,也可以通过网络来载入所需要的类。这也有利于软件的升级。另外,Java语言中的类有一个运行时刻的表示,能进行运行时刻的类型检查。

7. 健壮性

Java语言的强类型机制、异常处理、废料的自动收集等是Java程序健壮性的重要保证。对指针的丢弃是Java的明智选择。Java致力于检查程序在编译和运行时的错误。类型检查可帮助查出许多开发早期出现的错误。Java自己操纵内存减少了内存出错的可能性。Java的安全检查机制使得Java更具健壮性。

8. 多线程性

Java环境本身就是多线程的。若干系统线程运行,负责必要的无用单元回收、系统维护等系统级操作;另外,Java语言内置多线程控制,可以大大简化多线程应用程序的开发。Java提供了一个类Thread,由它负责启动运行,终止线程,并可检查线程状态。

9. 可移植性

Java主要靠Java虚拟机(JVM)在目标码级实现平台无关性。用Java编写的应用程序不用修改就可在不同的软硬件平台上运行。

3.1.3 Java语言程序简介

Java语言程序实际上有两种：一种是Java应用程序(Application)，它是一种独立的程序，不需要任何Web浏览器来执行，可运行于任何具备Java运行环境的机器中。另一种是Java小应用程序(Applet)，是运行于Web浏览器中的一个程序，它通常由浏览器下载到客户端，并通过浏览器运行。Applet通常较小，下载时间较短，通常嵌入HTML页面中。

1. 应用程序

Java应用程序是在本地机上运行的程序，它含有一个main()方法，并以该方法作为程序的入口，可以通过解释器(java.exe)直接独立运行，文件名必须与main()所在类的类名相同。

下面我们给出一个简单的Java程序ex3-1.java，其源程序如下所示。

ex3-1.java源程序

```
class ex3-1
{
    public  static  void   main(String args[])
{
    System.out.print("This  is  a  Java  Example!");
}
}
```

我们在DOS命令提示符下输入“javac ex3-1.java”，将编译ex3-1.java程序，如果成功将生成FirstJavaPrg.class字节码文件。然后输入“java ex3-1”运行该程序，结果显示如图3-1所示。

```
D:\Java>java ex3-1
This  is  a  Java  Example!
```

图3-1 ex3-1的运行结果

2. 小应用程序

下面我们给出一个简单的Applet小程序ex3-2.java，其源程序如下所示。

ex3-2.java源程序

```
import java.awt.*;
import  java.applet.*;
public  class  ex3-2  extends  Applet
{
public  void  paint(Graphics  g)
    {
    g.drawString("This is Second  Java  Example!",100,20);
    }
}
```

我们在DOS命令提示符下输入“javac ex3-2.java”，将编译该程序，如果成功将生成SecondJavaPrg.class字节码文件。然后建立一个与类名ex3-2相同但扩展名为html的文件ex3-2.html，内容如下：

```
<HTML>
<HEAD>
<TITLE>SecondJavaPrg.html</TITLE>
<BODY>
<applet  code="SecondJavaPrg.class"  width=200  height=100 >
</applet>
</BODY>
</HTML>
```

在DOS方式下输入“appletviewer ex3-2.html”后按Enter键,将执行上述Applet程序,结果如图3-2所示。

图3-2 ex3-2.html的运行结果

3.2 Java的基本语法

3.2.1 Java语言的标识符与关键字

1. 标识符

与其他高级语言类似,Java语言中的标识符可以定义变量、类或方法等。Java语言中规定标识符是以字母、下画线(_)或美元符号($)开始的,其后也可跟数字、字母、下画线或美元符号组成的字符序列。

注意:

(1) 标识符区分大小写,例如a和A是两个不同的变量名,没有长度限制,我们可以为标识符取任意长度的名字。

(2) 标识符不能是关键字,但是它可以包含关键字作为它的名字的一部分。例如,thisone是一个有效标识符,但this却不是,因为this是一个Java关键字。

下面是合法的标识符:score,stu_no,sum,$123,total1_2_3。下面是非法的标识符:stu.name,2sum,class。

Java语言采用的是Unicode编码字符集(即统一编码字符集),在这种字符集中,每个字符用2字节即16位来表示,包含65535个字符。其中前256个字符表示ASCII码,使其对ASCII码具有兼容性,后面26000个字符用来表示汉字、日文片假名、平假名和朝鲜文等。但Unicode编码字符集只用于Java平台内部,当涉及打印、屏幕显示、键盘输入等外部操作时,仍由具体计算机操作系统决定表示方法。

2. 关键字

在Java语言中,有些标识符已经具有固定的含义和专门的用途,不能在程序中当作

一般的标识符来随意使用，这样的标识符称为关键字。Java语言中的保留字均用小写字母表示。

Java语言中的关键字如下：

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

注意：

(1) true、false和null必须为小写。

(2) 无sizeof运算符，因为Java语言的平台无关性，所有数据类型的长度和表示是固定的。

(3) goto和const不是Java语言的关键字。

3.2.2 Java语言的基本数据类型

Java语言数据类型有简单数据类型和复合数据类型两大类。简单数据类型包括整数类型(byte，short，int，long)、浮点类型(float，double)、字符类型char、布尔类型(boolean)；复合数据类型包括类(class)、接口(interface)、数组。

1. 整数类型

在Java语言中有4种整数类型，可分别使用关键字byte，short，int和long来进行声明(见表3-1)。

表3-1 Java语言的整数类型

数据类型	所占位数	数的范围
byte	8	$-2^7 \sim 2^7-1$
short	16	$-2^{15} \sim 2^{15}-1$
int	32	$-2^{31} \sim 2^{31}-1$
long	64	$-2^{63} \sim 2^{63}-1$

整数类型的数字有十进制、八进制和十六进制三种表示形式。首位为“0”表示是八进制的数值，首位为“0x”表示是十六进制的数值。例如12表示十进制数12；012是八进制整数，表示十进制数10；0x12是十六进制整数，表示十进制数18。

2. 浮点类型

Java语言有两种浮点类型：float(单精度)和double(双精度)(见表3-2)。如果一个数字后带有字母F或f为float类型；带有字母D或d，则该数为double类型；如果不明确指明浮点数的类型，默认为double类型。例如：

2.5876(double 型浮点数);
1.5E12(double 型浮点数);
3.14f(float 型浮点数)。

表 3-2 Java 语言的浮点类型

数 据 类 型	所 占 位 数	数 的 范 围
float	32	3.4e−038～3.4e+038
double	64	1.7e−308～1.7e+308

3. 字符类型 char

使用 char 类型可表示单个字符,字符型数据代表 16 位的 Unicode 字符,字符常量是用单引号括起来的一个字符,如'c','F'等。与 C、C++ 相同,Java 也提供转义字符(见表 3-3),以反斜杠(\)开头,将其后的字符转变为另外的含义。

表 3-3 Java 语言的转义字符

转 义 字 符	含　义
\b	退格
\t	Tab 制表
\n	换行
\r	硬回车
\"	双引号
\'	单引号
\\	反斜杠
\ddd	1～3 位八进制数据所表示的字符
\uxxxx	1～4 位十六进制数据所表示的 Unicode 字符

4. 布尔类型 boolean

布尔类型只有两个值:false 和 true。

注意:在 Java 语言中不允许将数字值转换成逻辑值,这一点与 C 语言不同。

5. 常量

Java 语言中的常量是用关键字 final 来定义的。其定义格式为:

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

例如:

```
final int StuNum = 50,Cnum=6;
```

6. 变量

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

```
Type varName [= value ][{, varName [=value]}];
```

例如：

```
int n = 3, n1 = 4;
char c='a';
```

3.2.3 Java语言的运算符与表达式

1. 运算符

在程序设计中，我们经常要进行各种运算，而运算符正是用来表示某一种运算的符号。按照运算符功能来分，Java语言所提供的运算符可分为算术运算符（见表3-4）、赋值运算符、关系运算符、逻辑运算符、位运算符、条件运算符等。

表3-4 Java语言的算术运算符

运 算 符	功 能	运 算 符	功 能
+（一元）	取正值	/	除法
-（一元）	取负值	%	求模运算（即求余数）
+	加法	++	加1
-	减法	--	减1
*	乘法		

1）算术运算符

（1）自增运算符在操作数左右出现的不同。i++，i--先使用变量的值，然后再自增或自减，而++i，--i先自增或自减然后再使用变量的值。

（2）Java语言中，运算符“+”除完成普通的加法运算外，还能够进行字符串的连接。如“ab”+“cd”，结果为“abcd”。

（3）与C、C++语言不同的是取模运算（%）其操作数可以是浮点数，例如13.4%4=1.4。

2）赋值运算符

赋值运算符就是用来为变量赋值的。最基本的赋值运算符就是等号“=”，其格式为：

```
变量名=值
```

例如：

```
int a=1;
```

在 Java 语言中,还提供了一种叫算术赋值运算符(见表 3-5)。

表 3-5 Java 语言的赋值运算符

运 算 符	实 例	说 明
+=	X+=2	等价于 X=X+2
-=	X-=2	等价于 X=X-2
*=	X*=2	等价于 X=X*2
/=	X/=2	等价于 X=X/2
%=	X%=2	等价于 X=X%2

3) 关系运算符

在使用 Java 语言进行程序设计时,也常常需要对两个对象进行比较,关系运算符用来比较两个值。关系运算符都是二元运算,关系运算的结果返回布尔类型的值 true 或 false(注意不是 C 或 C++ 中的 1 或 0)。关系运算符常与布尔逻辑运算符一起使用,作为流控制语句的判断条件。

Java 语言提供了 6 种关系运算符:>、>=、<、<=、==、!=。Java 语言中,任何数据类型的数据(包括基本类型和组合类型)都可以通过==或!=来比较是否相等(这与 C 或 C++ 不同)。

4) 逻辑运算符

逻辑运算符(见表 3-6)又称为布尔运算符,是用来处理一些逻辑关系的运算符,它最常应用于流程控制。Java 语言中逻辑运算符包括与运算符"&&"、或运算符"||"、非运算符"!"。&&、|| 为二元运算符,实现逻辑与、逻辑或;!为一元运算符,实现逻辑非。

表 3-6 Java 语言的逻辑运算符

op1	op2	op1 && op2	op1 || op2	!op1
false	false	false	false	true
false	true	false	true	true
true	false	false	true	false
true	true	true	true	false

对于布尔逻辑运算,先求出运算符左边的表达式的值。对或运算如果为 true,则整个表达式的结果为 true,不必对运算符右边的表达式再进行运算;同样,对与运算,如果左边表达式的值为 false,则不必对右边的表达式求值,整个表达式的结果为 false。

5) 位运算符

位运算符用来对二进制位进行操作。Java 中提供的位运算符有:&(按位与)、|(按位或)、^(按位异或)、~(按位取反)、>>(向右移位)、<<(向左移位)、>>>(向右移位,用零来填充高位)。位运算符中,除~以外,其余均为二元运算符,操作数只能为整型和字符型数据。

Java语言使用补码来表示二进制数,在补码表示中,最高位为符号位,正数的符号位为0,负数的符号位为1。补码的规定如下:对正数来说,最高位为0,其余各位代表数值本身(以二进制表示),如+42的补码为00101010。对负数而言,把该数绝对值的补码按位取反,然后对整个数加1,即得该数的补码。如-42的补码为11010110(即+42的补码00101010按位取反11010101,加1即为11010110)用补码来表示数,0的补码是唯一的,都为00000000。

(1) 按位与运算(&):参与运算的两个值,如果两个相应位都为1,则该位的结果为1,否则为0。按位或运算(|):参与运算的两个值,如果两个相应位都是0,则该位结果为0,否则为1。按位异或运算(^):参与运算的两个值,如果两个相应位的某一个是1,另一个是0,那么按位异或(^)在该位的结果为1;也就是说,如果两个相应位相同,输出为0,否则为1。

(2) 按位取反运算(~):按位取反生成与输入位相反的值——若输入0,则输出1;输入1,则输出0。

(3) 右移位运算符>>:执行一个右移位(带符号),左边按符号位补0或1。例如:

```
int a=16,b;
b=a>>2;          //b=4
```

(4) 运算符>>>:同样是执行一个右移位,只是它执行的是不带符号的移位。也就是说,对以补码表示的二进制数操作时,在带符号的右移中,右移后左边留下的空位中填入的是原数的符号位(正数为0,负数为1);在不带符号的右移中,右移后左边留下的空位中填入的一律是0。

(5) 左移位运算符(<<):执行一个左移位。做左移位运算时,右边的空位补0。在不产生溢出的情况下,数据左移1位相当于乘以2。例如:

```
int a=64,b;
b=a<<1;          //b=128
```

6) 条件运算符

在Java语言中,条件运算符(?:)使用的形式是:

```
表达式1 ?表达式2 : 表达式3;
```

其运算规则与C语言中的完全一致:先计算表达式1的值,若为真,则整个表达式的结果是表达式2的值,否则,整个表达式的结果取表达式3的值。

7) 其他运算符

(1) 分量运算符.:用于访问对象实例或者类的类成员函数。

(2) 下标运算符[ ]:是数组运算符。

(3) 对象运算符instanceof:用来判断一个对象是不是某一个类或者其子类的实例。如果对象是该类或者其子类的实例,返回true;否则返回false。

(4) 内存分配运算符new:new运算符用于创建一个新的对象或者新的数组。

(5) 强制类型转换运算符(类型):强制类型转换的格式是:

```
(数据类型)变量名
```

经过强制类型转换,将得到一个在“()”中声明的数据类型的数据,该数据是从指定变量所包含的数据转换而来的。值得注意的是,指定变量本身不会发生任何变化。

将占用位数较长的数据转化成占用位数较短的数据时,可能会造成数据超出较短数据类型的取值范围,造成“溢出”。如:

```
long  i=10000000000;
int  j=(int)i;
```

因为转换的结果已经超出了 int 型数据所能表示的最大整数(4294967295),造成溢出,产生了错误。

方法调用运算符():表示方法或函数的调用。

下面给出 Java 语言中各种运算符的优先级,如表 3-7 所示。

表 3-7 Java 语言的运算符优先级

优先次序	运算符
1	. [] ()(方法调用)
2	! ~ ++ -- +(一元) -(一元) instanceof
3	(类型) new
4	* / %
5	+ -
6	<< >> >>>
7	< <= >= >
8	== !=
9	&
10	^
11	|
12	&&
13	||
14	?:
15	= += -= *= /= &= |= ^= <<= >>= . >>=

2. 表达式与数值类型的相互转换

1) 表达式

表达式是由操作数和运算符按一定的语法形式组成的符号序列。一个常量或一个变量名是最简单的表达式,其值即该常量或变量的值;表达式的值还可以用作其他运算的操作数,形成更复杂的表达式。例如:

```
y=--x;
```

表达式的计算方法可以归纳成以下两点。

(1) 有括号先算括号内的,有乘除先乘除,最后算加减。

(2) 存在多个加减或多个乘除,则从左到右进行。

2) 数值类型的互相转换

当不同数据类型的数据参加运算时,会涉及不同数据类型的转换问题。Java 程序里,类型转换有两种:自动类型转换(或称隐含类型转换)和强制类型转换。

在实际中常会将一种类型的值赋给另外一种变量类型,如果这两种类型是兼容的,Java 将执行自动类型转换。Java 语言赋值运算的自动类型转换规则如下:

byte→short→int→long→float→double 或者 byte→char→int→long→float→double。

以上规则表明 byte 可以转换成 char、short、int、long、float 和 double 类型。short 可以转换成 int、long、float 和 double 类型。不是所有的数据类型都允许自动(隐含)转换。例如,下面的语句把 long 型数据赋值给 int 型数据,在编译时就会发生错误:

```
long  a=100;
int  b=a;
```

这是因为当把占用位数较长的数据转换成占用位数较短的数据时,会出现信息丢失的情况,因而不能够自动转换。这时就需要利用强制类型转换,执行非兼容类型之间的类型转换。上面的语句写成下面的形式就不会发生错误:

```
long  a=100;
int  b=(int)a;
```

3.2.4 Java 语言的基本控制语句

与 C、C++ 相同,Java 程序通过控制语句来执行程序。语句可以是单一的一条语句(如 c=a+b;),也可以是复合语句。

Java 中的流控制语句包括以下几种。

(1) 分支语句:if-else,break,switch,return。

(2) 循环语句:while,do-while,for,continue。

(3) 例外处理语句:try-catch-finally,throw。

(4) 注释语句。

1. 分支语句

分支语句是在多条执行路径中选择一条执行的控制结构。

1) 条件语句 if-else

if-else 语句根据判定条件的真假来执行两种操作中的一种,格式为:

```
if(条件表达式)
{语句序列 1;}
[else
{语句序列 2;}]
```

注意: 条件表达式是任意一个返回布尔型数据的表达式(这比 C、C++ 的限制要严格)。

(1) 每个单一的语句后都必须有分号。

(2) 语句序列 1、语句序列 2 可以为复合语句,这时要用花括号{}括起来。{}外面不加分号。

(3) else 子句是任选的。

(4) 若条件表达式的值为 true,则程序执行语句序列 1,否则执行语句序列 2。

(5) else 子句不能单独作为语句使用,它必须和 if 配对使用。else 总是与离它最近的 if 配对。可以通过使用花括号{}来改变配对关系。

如:

```
if(x>0)     y=1;
else        y=-1;
```

if-else 语句的一种特殊形式为:

```
if(条件表达式 1){
语句序列 1
}else if(条件表达式 2){
语句序列 2
}
…
}else if(条件表达式 M){
语句序列 M
}else{
语句序列 N
}
```

如:

```
if(x<0)     y=-1;
else
    if(x>0)    y=1;
    else       y=0;
```

2) 多分支语句 switch

switch 语句(又称开关语句)是和 case 语句一起使用的,其功能是根据某个表达式的值在多个 case 引导的多个分支语句中选择一个来执行,它的一般格式如下:

```
switch(表达式){
case 值 1: 语句序列 1;break;
case 值 2: 语句序列 2;break;
…
case 值 N: 语句序列 N;break;
[default: 语句序列 N+1;]
}
```

注意:

(1) 表达式的值必须是符合 byte,char,short,int 类型的常量表达式,而且所有 case 子句中的值是不同的。

(2) default 子句是任选的。当表达式的值与任一 case 子句中的值都不匹配时,程序

执行 default 后面的语句。如果表达式的值与任一 case 子句中的值都不匹配且没有 default 子句，则程序不作任何操作，而是直接跳出 switch 语句。

(3) case 表达式只是起语句标号作用，并不是在该处进行条件判断，因此应该在执行一个 case 分支后，可以用 break 语句来使流程跳出 switch 结构。在一些特殊情况下，多个不同的 case 值要执行一组相同的操作，这时可以不用 break 语句。

(4) case 分支中包括多个执行语句时，可以不用花括号{}括起。

下面我们给出使用 switch 语句的例程 ex3-3.java，该程序能够根据考试成绩的等级打印出百分制分数段。其源程序如下。

ex3-3.java 源程序

```
public class ex3-3
{
public static void main(String args[])
 {
char grade='B';
switch(grade)
  {
    case 'A':System.out.println("85-100");break;
    case 'B':System.out.println("70-84");break;
    case 'C':System.out.println("60-69");break;
    case 'D':System.out.println("<60"); break;
    default:System.out.println("error");
  }
 }
}
```

运行结果如图 3-3 所示。

```
D:\Java>javac  ex3-3.java

D:\Java>java  ex3-3
70-84
```

图 3-3 ex3-3.java 的运行结果

上述程序如果写成下面形式：

```
public class ex3-3
{
public static void main(String args[])
 {
char grade='B';
switch(grade)
  {
    case 'A':System.out.println("85-100");
    case 'B':System.out.println("70-84");
    case 'C':System.out.println("60-69");
    case 'D':System.out.println("<60");
    default:System.out.println("error");
  }
 }
}
```

运行结果如图 3-4 所示。

```
D:\Java>javac  ex3-3.java

D:\Java>java  ex3-3
70-84
60-69
<60
error
```

图 3-4　改写后的 ex3-3.java 的运行结果

2. 循环语句

在程序中经常需要在一定的条件下反复执行某段程序,这时可以通过循环结构来控制实现。Java 语言中有三种循环控制语句,分别是 while、do-while 和 for 语句。

1) while 语句

while 语句的格式如下:

```
while(条件表达式)
   {
       循环体语句组;
   }
```

在循环刚开始时,先计算一次"条件表达式"的值,当结果为真时,便执行循环体,否则,将不执行循环体,直接跳转到循环体外执行后续语句。每执行完一次循环体,都会重新计算一次条件表达式,当结果为真时,便继续执行循环体,直到结果为假时结束循环。下面我们给出用 while 语句实现 1～100 累计求和的例程 ex3-4.java,其源程序如下。

ex3-4.java 源程序

```
public class ex3-4{
public static void main(String args[]){
int n=1,sum=0;
while(n<101){
sum+=n++;
}
System.out.println("\nThe sum is "+sum);
}
}
```

运行结果如图 3-5 所示。

```
D:\Java>javac  ex3-4.java

D:\Java>java  ex3-4

The sum is 5050
```

图 3-5　ex3-4.java 的运行结果

2) do-while 语句

do-while 语句的格式如下:

```
do
{
    循环体语句组;
}while(条件表达式);
```

do-while循环与while循环的不同在于：它先执行循环中的语句，然后再判断结果是否为真，如果为真则继续循环；如果为假，则终止循环。因此，do-while循环至少要执行一次循环语句。下面我们给出用do-while语句实现1～100累计求和的例程ex3-5.java，其源程序如下。

ex3-5.java源程序

```
public class ex3-5
{
 public static void main(String args[])
  {
int n=1,sum=0;
 do{
     sum+=n++;
    }while(n<101);
  System.out.println("\nThe sum is"+sum);
  }
}
```

3）for语句

for语句的格式如下：

```
for(表达式1;表达式2;表达式3)
  {
    循环体语句组;
  }
```

表达式1一般是一个赋值语句，用来给循环控制变量赋初值；表达式2是一个布尔类型的表达式，用来决定什么时候终止循环；表达式3一般用来修改循环变量，控制变量每循环一次后如何变化。这三部分之间用“;”分开。

for语句的执行过程如下。

（1）在循环刚开始时，先计算表达式1。

（2）根据表达式2的值来决定是否执行循环体。表达式2是一个返回布尔值的表达式，若该值为假，将不执行循环体，并退出循环；若该值为真，将执行循环体。

（3）执行完一次循环体后，计算表达式3。

（4）转入第(2)步继续执行。

for语句通常用来执行循环次数确定的情况(如对数组元素进行操作)，也可以根据循环结束条件执行循环次数不确定的情况。初始化、终止以及迭代部分都可以为空语句(但分号不能省)，三者均为空时，相当于一个无限循环。for语句是三个循环语句中功能最强，使用最广泛的一个。下面我们给出用for语句实现1～100累计求和的例程ex3-6.java，其源程序如下。

ex3-6.java源程序

```
public class ex3-6
{
public static void main(String args[])
  {  int sum=0;
```

```
    for(int i=1;i<=100;i++)
       {
         sum+=i;
       }
    System.out.println("\n The sum is" +sum);
   }
}
```

4）跳转语句

跳转语句用来实现循环执行过程中的流程转移。在 Java 语言中有两种跳转语句：break 语句和 continue 语句。break 语句用于强行退出循环，不再执行循环体中剩余的语句。而 continue 语句用来结束本次循环，跳过循环体中剩余的语句，接着进行循环条件的判断，以决定是否继续循环。下面我们给出用 continue 语句计算 1～100 的奇数和的例程 ex3-7.java，其源程序如下。

ex3-7.java 源程序

```
public class ex3-7
{
public static void main(String args[])
{
   int sum=0;
for(int i=1;i<100; i++)
   {
   if(i%2==0) continue;
   sum+=i;
    }
System.out.println("\nThe sum of odd numbers(1~100)is:"+sum);
}
}
```

运行结果如图 3-6 所示。

```
D:\Java>javac  ex3-7.java

D:\Java>java   ex3-7

The sum of odd numbers(1～100)is:2500
```

图 3-6　ex3-7.java 的运行结果

3. 例外处理语句

例外处理语句包括 try、catch、finally 以及 throw 语句。与 C、C++ 相比，例外处理语句是 Java 所特有的。

4. 注释语句

Java 语言中的注释语句有以下三种形式。

//：用于单行注释，注释从//开始，终止于行尾。

/*...*/：用于多行注释，注释从/*开始，到*/结束，且这种注释不能互相嵌套。

/**...*/：是Java所特有的doc注释，它以/**开始，到*/结束。这种注释主要是为支持JDK工具javadoc而采用的。

3.3 Java语言的类与对象

Java语言是完全面向对象的程序设计语言，Java语言程序的基本单位是类。Java语言中不允许有独立的变量、常量或函数，即Java程序中的所有元素都要通过类或对象来访问。

3.3.1 Java语言的类

1. 类声明

Java语言类声明的一般格式如下：

```
[修饰符] class 类名 [extends 父类] [implements 接口名]
{
      类体}
```

其中，class是声明类的关键字，类名与变量名的命名规则一样。修饰符是该类的访问权限，如public、private等。extends项表明该类是由其父类继承而来的，implements项用来说明当前类中实现了哪个接口定义的功能和方法。

例如：定义两个类student和catclass：

```
class  student
{…
}

public class catclass extends animalclass{
…
}
```

2. 类体

每个类中通常都包含数据与函数两种类型的元素，我们一般把它叫作属性和成员函数(也称为方法)，所以类体中包含了该类中所有成员变量的定义和该类所支持的所有方法。

```
class 类名
{   类成员变量声明
    类方法声明
}
```

1）成员变量的声明

简单的成员变量声明的格式为：

```
[修饰符]   变量类型   变量名 [=变量初值];
```

例如：

```
int  x,y;
float  f=5.0;
```

2）类方法的声明

类方法相当于C或C++中的函数。类方法声明的格式为：

```
[修饰符]  返回值类型  方法名(参数列表)
    {方法体}
```

注意：每一个类方法必须返回一个值或声明返回为空(void)。

3）构造方法

在Java程序设计语言中，每个类都有一个特殊的方法即构造方法。构造方法名必须与类名相同，且没有返回类型。构造方法的作用是构造并初始化对象，构造方法不能由编程人员显式地直接调用，是在创建一个类的新对象的时候，由系统来自动调用的。另外，Java语言支持方法重载，所以类可以有多个构造方法，它们可以通过参数的数目或类型的不同来加以区分。

例如：

```
class  MyAnimals
    {
        int dog,cat;
        count()
            {
                dog=10;
                cat=20;
            }
        count(int dognumber,int catnumber)
            {
          this.dog=dognumber;
          this.cat=catnumber;
            }
        }
```

这里的关键字this是用来在方法中引用创建它的对象，this.dog应用的是当前对象的成员变量dog。

3.3.2 Java语言的对象

当我们创建了自己的类之后，通常需要使用它来完成某种工作。这时可以通过定义类的实例——对象来实现这种需求。

1. 对象的生成

对象的生成包括声明、实例化和初始化三方面。首先必须为对象声明一个变量，其格式为：

```
类型  对象名;
```

其中，类型为复合数据类型(包括类和接口)。

例如：

```
student  st1;
```

但仅有对象的声明是不够的，还要为对象分配内存空间，即实例化。创建类实例，要使用关键字运算符 new。用 new 可以为一个类实例化多个不同的对象，这些对象分别占用不同的内存空间。例如：

```
st1=new student();
```

也可表示为：

```
student st1=new student();
student st2=new student("zhang",68,72,85);
```

2. 对象的引用

1) 引用对象的成员变量

其引用格式为：

```
objectReference.variable;
```

其中，objectReference 是一个已生成的对象，例如上例中生成了对象 st1，就可以用 st1.name 来访问其成员变量了。

2) 调用对象的方法

其引用格式为：

```
objectReference.method();
```

下面给出一个有关对象成员变量的引用及方法调用的例程 ex3-8.java。

ex3-8.java 源程序

```
class ex3-8{
String name;
float sc1,sc2,sc3,ave;
Student(){
name="wang";
sc1=60;
sc2=82;
sc3=74;
}
Student(String str,float a,float b,float c){
name=str;
sc1=a;
sc2=b;
sc3=c;
}
```

```
float average(){
ave=(sc1+sc2+sc3)/3;
return ave;
}
public static void main(String args[]){
Student st1=new Student();
Student st2=new Student("zhang",78,89,92);
System.out.println("\nName  score1  score2  score3  average");
System.out.println(st1.name+" "+st1.sc1+" "+st1.sc2+" "+st1.sc3+"
"+st1.average());
System.out.println(st2.name+"  "+st2.sc1+"  "+st2.sc2+"  "+st2.sc3+"
  "+st2.average());
}
}
```

运行结果如图 3-7 所示。

```
D:\Java>javac  ex3-8.java

D:\Java>java  ex3-8

Name  score1  score2  score3  average
wang   60.0    82.0    74.0     72.0
zhang   78.0    89.0    92.0     86.333336
```

图 3-7　ex3-8.java 的运行结果

本章小结

Java 是由 Sun 公司推出的 Java 程序设计语言和 Java 平台的总称。用 Java 实现的 HotJava 浏览器(支持 Java Applet)显示了 Java 的魅力：跨平台、强安全性、动态的 Web、语言简洁、Internet 计算、面向对象以及适用于网络，已成为 Internet 网络编程语言事实上的标准。本章要求重点理解 Java 语言的语法基础，掌握 Java 面向对象编程的思想，了解 Java Applet 编程技术。Java 技术的出现推动了 Web 的迅速发展，常用的浏览器现在均支持 Java Applet。

习题及实训

1. 简述 Java 面向对象程序设计语言的特点。

2. Java 语言包含了哪些基本数据类型？各自是如何规定的？

3. Java 语言包括几种不同类型的控制语句？

4. 什么是类和对象？请说出 Java 中类和对象之间的关系。

5. 请定义一个 Java 类。

6. 给出如下一个 Java 源程序 MyJavaFile.java。请检查其正确性，在 JDK 环境下编译运行该程序并说出该程序实现的功能。