第3章　表达式和流程控制语句

3.1　Java 中常用的运算符有哪些？它们的含义分别是什么？

解：Java 运算符按功能可分为：算术运算符、关系运算符、逻辑运算符、位运算符、赋值运算符、条件运算符；除此之外，还有几个特殊用途运算符，如数组下标运算符等。

常用的运算符及其意义如表 3-1 所示。

表 3-1　运算符及其意义

类型	运算符	意　义
算术运算符	＋	加法或字符串的连接
	－	减法
	*	乘法
	/	除法，如果参加运算的两个操作数都是整数，则为整除运算，否则为浮点数除法
	%	整数求余运算
	＋＋	加 1 运算
	－－	减 1 运算
	－	求相反数
关系运算符	＝＝	等于
	!＝	不等于
	＞	大于
	＜	小于
	＞＝	大于等于
	＜＝	小于等于
逻辑运算与位运算	&&	逻辑与
	||	逻辑或
	!	取反
	&	按位与
	|	按位或
	^	按位异或
	～	按位取反

续表

类型	运算符	意　义
逻辑运算与位运算	>>	右移
	>>>	右移,并用0填充高位
	<<	左移
赋值运算符	+=	加法赋值
	-=	减法赋值
	*=	乘法赋值
	/=	除法赋值
	%=	取余赋值
	&=	按位与赋值
	|=	按位或赋值
	^=	按位异或赋值
	<<=	左移赋值
	>>=	右移赋值
	>>>=	右移赋值
其他运算符	[]	数组下标
	.()	方法调用
	?:	三目条件运算
	instanceof	对象运算

3.2 Java中操作符优先级是如何定义的?

解:Java中,各操作符的运算优先顺序如表3-2所示。

表3-2　操作符的运算优先顺序

优先级	运算符	运　算	结合律
1	[]	数组下标	自左至右
	.	对象成员引用	
	(参数)	参数计算和方法调用	
	++	后缀加	
	--	后缀减	
2	++	前缀加	自右至左
	--	前缀减	
	+	一元加	

续表

优先级	运算符	运　　算	结合律
2	－	一元减	自右至左
	～	位运算非	
	！	逻辑非	
3	new	对象实例	自右至左
	(类型)	转换	
4	*	乘法	自左至右
	/	除法	
	%	取模	
5	＋	加法	自左至右
	＋	字符串连接	
	－	减法	
6	＜＜	左移	自左至右
	＞＞	用符号位填充的右移	
	＞＞＞	用 0 填充的右移	
7	＜	小于	自左至右
	＜＝	小于等于	
	＞	大于	
	＞＝	大于等于	
	instanceof	类型比较	
8	＝＝	相等	自左至右
	！＝	不等于	
9	&	位运算与	自左至右
	&	布尔与	
10	^	位运算异或	自左至右
	^	布尔异或	
11	|	位或	自左至右
	|	布尔或	
12	&&	逻辑与	自左至右
13	||	逻辑或	自左至右
14	?：	条件运算符	自右至左

续表

优先级	运算符	运　　算	结合律
15	=	赋值	自右至左
	+=	加法赋值	
	+=	字符串连接赋值	
	-=	减法赋值	
	*=	乘法赋值	
	/=	除法赋值	
	%=	取余赋值	
	<<=	左移赋值	
	>>=	右移(符号位)赋值	
	>>>=	右移(0)赋值	
	&=	位与赋值	
	&=	布尔与赋值	
	^=	位异或赋值	
	^=	布尔异或赋值	
	|=	位或赋值	
	|=	布尔或赋值	

3.3　>>>与>>有什么区别？试分析下列程序段的执行结果：

```
int b1=1;
int b2=1;

b1 <<=31;
b2 <<=31;

b1 >>=31;
b1 >>=1;

b2 >>>=31;
b2 >>>=1;
```

解：>>>与>>都是右移运算符，它们的不同之处在于使用不同位填充左侧的空白。>>运算使用符号位填充左侧的空位，而>>>使用零填充空位。也就是说，>>运算保持操作数的符号不变，而>>>运算则可能改变原数的符号。

分析上面的程序段。初始时，b1 和 b2 都是 int 型的变量，占 32 位，初值均为 1，两个变量分别向左移动 31 位，则两个值变为：

```
10000000 00000000 00000000 00000000
```

在计算机内部，这个值是：－2147483648。

下一步，b1 再向右移动 31 位，这里使用的是＞＞运算符。b1 是一个负数，其最高位为 1，右移时使用 1 填充左侧的空位。右移 31 位后 b1 的值为：

```
11111111 11111111 11111111 11111111
```

在计算机内部，这个值是：－1。b1 继续向右移动 1 位，此时值不变，仍为－1。实际上，此后使用＞＞运算符将 b1 向右移动任何位，它的值都不会再变了。

使用＞＞＞运算符则有所不同。＞＞＞使用零填充左侧的空位，所以将 b2 向右移动 31 位后，它的值为：

```
00000000 00000000 00000000 00000001
```

即 b2 的值为 1。再向右移动 1 位，则它的值为：

```
00000000 00000000 00000000 00000000
```

即 b2 的值为 0。

测试这些语句的程序如下所示：

```
import java.util.*;

public class Test
{   public static void main(String[] args)
    {   int b1=1;                                   //b1 赋初值
        int b2=1;                                   //b2 赋初值
        System.out.println("b1="+b1);
        System.out.println("b2="+b2);

        b1 <<=31;                                   //b1 左移 31 位
        b2 <<=31;                                   //b2 左移 31 位
        System.out.println("b1="+b1);
        System.out.println("b2="+b2);

        b1 >>=31;                                   //b1 右移 31 位
        System.out.println("b1="+b1);
        b1 >>=1;                                    //b1 再右移 1 位
        System.out.println("b1="+b1);

        b2 >>>=31;                                  //b2 右移 31 位
        System.out.println("b2="+b2);
        b2 >>>=1;                                   //b2 再右移 1 位
        System.out.println("b2="+b2);
    }
}
```

相应的执行结果如图 3-1 所示。

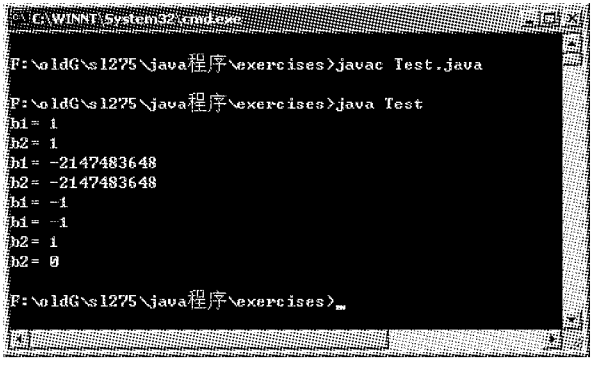

图 3-1　右移操作的执行结果

【拓展思考】

给出下面的说明，下列每个赋值语句会得到什么结果？

```
int iResult, num1=25, num2=40, num3=17, num4=5;
double fResult, val1=17.0, val2=12.78;
a. iResult=num1/num4;
b. fResult=num1/num4;
c. iResult=num3/num4;
d. fResult=num3/num4;
e. fResult=val1/num4;
f. fResult=val1/val2;
g. iResult=num1/num2;
h. fResult=(double) num1/num2;
i. fResult=num1/(double) num2;
j. fResult=(double) (num1/num2);
k. iResult=(int) (val1/num4);
l. fResult=(int) (val1/num4);
m. fResult=(int) ((double) num1/num2);
n. iResult=num3%num4;
o. iResult=num2%num3;
p. iResult=num3%num2;
q. iResult=num2%num4;
```

3.4　设 n 为自然数，

$$n! = 1 \times 2 \times 3 \times \cdots \times n$$

称为 n 的阶乘，并且规定 0!＝1。试编制程序计算 2!，4!，6!，8!和 10!，并将结果输出到屏幕上。

解：阶乘函数在数学上的定义为：

$$n! = \begin{cases} 1 & (n=0) \\ n(n-1)! & (n>0) \end{cases}$$

这是一个递归定义，因为阶乘本身又出现在阶乘的定义中。对于所有的递归定义，一定要有一个递归结束的出口，这既是定义的最基本情况，也是程序执行递归结束的地方。本定义中的第一行即是递归出口。

当一个函数使用递归定义的时候，往往直接使用递归方法实现它。阶乘的递归实现如下所示：

```
import java.util.*;

public class Factorial
{   public static void main(String[] args)
    {   Factorial ff=new Factorial();
        for (int i=0; i<5; i++)                    //共计算 5 个阶乘结果
        {   ff.setInitVal(2*(i+1));                //计算哪个值的阶乘
            ff.result=Factorial(ff.initVal);       //计算
            ff.print();                            //输出结果
        }
    }
    public static int Factorial(int n)
    {   if (n==0) return 1;                        //递归出口
        return  n*Factorial(n-1);                  //递归计算
    }
    public void print()
    {   System.out.println(initVal+"!="+result);
    }
    public void setInitVal(int n)
    {   initVal=n;
    }
    private int result, initVal;
}
```

程序运行结果如图 3-2 所示。

图 3-2　使用递归算法计算阶乘的执行结果

另一方面，根据阶乘的定义，$n!=n(n-1)(n-2)\cdots 3\cdot 2\cdot 1$，因此完全可以用循环来计算阶乘，而不必使用递归。因为递归毕竟多次调用同一个方法，函数调用所花的时间较

长，递归调用的效率较低。

阶乘的非递归实现如下所示：

```
import java.util.*;

class Factorial2
{   private int result, initVal;
    public static void main(String[] args)
    {   Factorial2 ff=new Factorial2();
        for (int i=0; i<5; i++)
        {   ff.setInitVal( 2*(i+1));                    //计算初值
            ff.result=1;                                //连乘运算的初值为 1
            for (int j=2; j <=ff.initVal; j++)          //循环计算连乘结果
                ff.result*=j;
            ff.print();
        }
    }
    public void print()
    {   System.out.println(initVal+"!="+result);
    }
    public void setInitVal(int n)
    {   initVal=n;
    }
}
```

3.5 使用 java.lang.Math 类，生成 100 个 0～99 之间的随机整数，找出它们之中的最大者及最小者，并统计大于 50 的整数个数。

提示：

Math 类支持 random 方法：

```
public static synchronized double random()
```

该方法返回一个 0.0～1.0 之间的小数，如果要得到其他范围的数，需要进行相应的转换。例如想得到一个 0～99 之间的整数，可以使用下列语句：

```
int num=(int) (100*Math.random());
```

解：提示中已经说明了，可以使用 Math.random()方法得到随机数，但这个随机数是一个 0.0～1.0 之间的浮点数，首先需要将数的范围变化到 0～99 之间，然后再将得到的数转换为整数。

程序中，使用了两个变量 MAXof100、MINof100 分别记录这 100 个整数中的最大值和最小值。先生成前两个随机整数，较大者放入 MAXof100 中，较小者放入 MINof100 中。随后使用一个循环生成剩余的 98 个随机整数，然后分别与 MAXof100 和 MINof100 进行比较，新生成的数如果大于 MAXof100，则将 MAXof100 修改为新的数。同样如果新生成的数小于 MINof100，则让 MINof100 记下这个数。程序中使用 count 记录大于 50

的随机数的个数，初始时，它的值为 0。

程序如下所示：

```
import java.util.*;

public class MathRandomTest
{   public static void main(String[] args)
    {   int count=0, MAXof100, MINof100;
        int num,i;

        MAXof100=(int) (100*Math.random());          //生成的第一个随机数
        MINof100=(int) (100*Math.random());          //生成的第二个随机数
        System.out.print(MAXof100+"   ");
        System.out.print(MINof100+"   ");
        if (MAXof100>50) count++;                    //记录下大于 50 的个数
        if (MINof100>50) count++;                    //记录下大于 50 的个数

        if (MINof100>MAXof100)                       //比较前两个随机数
        {   num=MINof100;
            MINof100=MAXof100;                       //较小者记入 MINof100
            MAXof100=num;                            //较大者记入 MAXof100
        }

        for (i=0; i<98; i++)                         //接下来生成其余的 98 个随机数
        {   num=(int) (100*Math.random());
            //控制每输出 10 个数即换行
            System.out.print(num+((i+2)%10==9?"\n":"   "));
            if (num>MAXof100)
                MAXof100=num;                        //更大的数记入 MAXof100
            else if (num<MINof100)
                MINof100=num;                        //更小的数记入 MINof100
            if (num>50) count++;                     //记录下大于 50 的个数
        }
        System.out.println("The MAX of 100 random integers is:  "+MAXof100);
        System.out.println("The MIN of 100 random integers is:  "+MINof100);
        System.out.println("The number of random more than 50 is:  "+count);
    }
}
```

程序运行结果如图 3-3 所示。

【拓展思考】

(1) 哪个包包含 Scanner 类？String 类又在哪个包内？Random 类呢？Math 类呢？

解：Scanner 类和 Random 类属于 java.util 包。String 和 Math 类属于 java.lang 包。

(2) 为什么不需要在程序中引入 Math 类？

解：Math 类属于 java.lang 包，这个包自动引入到任一个 Java 程序中，所以不需要

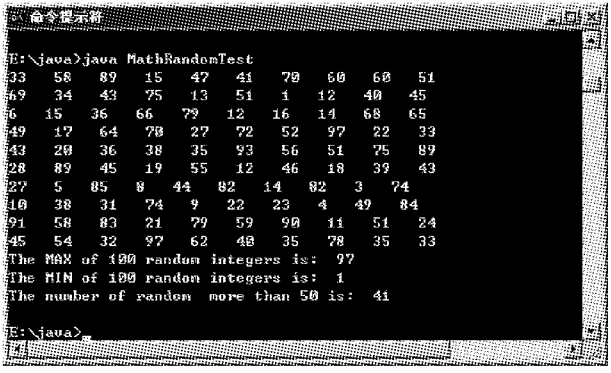

图 3-3　随机整数的执行结果

使用单独的 import 语句来说明。

(3) 给定一个 Random 对象 rand,调用 rand. nextInt()将返回什么?

解:调用 Random 对象的 nextInt()方法返回 int 值范围内的一个随机整数,包括正数和负数。

(4) 给定一个 Random 对象 rand,调用 rand. nextInt(20)将返回什么?

解:给 Random 对象的 nextInt()方法传递一个正整数参数 x,返回 0～x－1 范围内的一个随机数。所以调用 nextInt(20)将得到 0～19(含)之间的一个随机数。

(5) 写一个语句,打印 1.23 弧角的正弦值。

解:下列语句打印 1.23 弧度的正弦值:

```
System.out.println (Math.sin(1.23));
```

(6) 说明一个 double 类型变量 result,初始化为 $5^{2.5}$。

解:下列说明创建了变量 double,并初始化为 $5^{2.5}$。

```
double result=Math.pow(5, 2.5);
```

3.6　下列表达式中,找出每个操作符的计算顺序,在操作符下按次序标上相应的数字。

```
a+b+c-d
a+b/c-d
a+b/c*d
(a+b)+c-d
(a+b)+(c-d)%e
(a+b)+c-d%e
(a+b)%e%c-d
```

解:在 Java 中,在对一个表达式进行计算时,如果表达式中含有多种运算符,则要按运算符的优先顺序依次从高向低进行,同级运算符则从左向右进行。括号可以改变运算次序。运算符的优先次序参见 3.2 题答案。

各个表达式中运算符的优先次序如下: