第3章　控制语句

Java的流程控制有三种，即顺序、选择和循环。熟悉C/C++的读者可以发现，本章所介绍的控制语句的语法格式与C/C++的类似。熟悉C/C++的读者可以粗略地浏览第3.1节和第3.2节，直接学习第3.3节带标号的break和continue语句。

本章的学习目标：

- 学会使用Java的分支语句
- 学会使用循环控制语句
- 学会使用break和continue语句
- 学会使用带标号的break和continue语句

3.1　分支语句

Java中的分支语句有两个，一个是if-else语句；另一个是switch-case语句。

3.1.1　if语句

if语句的语法格式如下：

```
if(Expression)
    Statement
[else
    Statement]
```

【注意】 if语句的条件表达式Expression与C/C++不同，Java要求该条件表达式必须是布尔类型，否则无法通过编译。在C/C++中条件表达式可以是一个数值，用非0代表真，0代表假，而Java不允许这样。

例如，下面的写法在C/C++中是合法的：

```
int  i=10;
    ⋮
if(i)
    ⋮
```

但在Java中却是错误的。必须将条件表达式转换为一个boolean值。例如，将if(i)语句修改为if(i !=0)。

if语句中含有else时，要注意else与if的匹配关系，例如这样写代码：

```
if(x<100)
    if(y==60)
    {
```

```
            System.out.println(x+y);
            y--;
        }
    else
        y++;
```

从程序缩排格式上,可以推断出这段程序含义,但由于 Java 规定:else 与最靠近自己的、上面的一个 if 语句匹配,因此 else 与第二个 if 匹配,若要与第一个 if 匹配,需要加括号改变匹配关系:

```
if(x<100)
{
    if(y==60)
    {
        System.out.println(x+y);
        y--;
    }
}else      //此时 else 匹配第一个 if
    y++;
```

因此,写程序时不但要注意缩排格式,而且还要注意 if-else 之间的匹配关系。

【例 3-1】 采用 Java Applet 小程序从文本框中获取数据,然后显示比较结果。

```
package chapter3;                                    //第 3 章的例子程序,都放在 chapter3 包中
import java.awt.*;
import java.applet.*;

public class compareNumbers extends Applet
{
    Label  lab1, lab2;
    TextField  input1, input2;
    int  num1, num2;

    public void init()
    {
        lab1=new Label("输入第 1 个整数");       //产生第 1 个标签对象
        input1=new TextField(10);                //产生第 1 个文本框对象
        lab2=new Label("输入第 2 个整数");
        input2=new TextField(10);

        add(lab1);       //将标签 lab1 对象放到网页上
        add(input1);     //将文本框 input1 对象放到网页上
        add(lab2);
        add(input2);
    }

    public boolean action(Event e , Object o)
    {
```

```
        if(e.target==input1||e.target==input2)
        {
            num1=Integer.parseInt(input1.getText());        //获取文本框中的数值
            num2=Integer.parseInt(input2.getText());
            if(num1<num2)
                showStatus(num1+"<"+num2);
            else  if(num1>num2)
                showStatus(num1+">"+num2);
            else  showStatus(num1+"=="+num2);
        }
        return  true;
    }
}
```

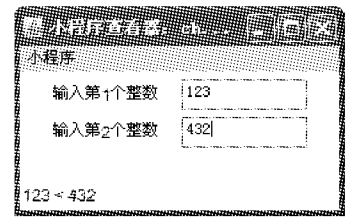

图 3-1　采用小程序输出比较结果

【程序运行结果】 假设输入 123 和 432 两个数，程序运行结果如图 3-1 所示。

【程序解析】 程序中的 init()和 action()均是系统规定的方法名，在此进行了覆盖(见第 4 章)。程序运行时，先执行 init()方法，若用户在文本框中按了 Enter 键，将调用 action()方法。主类中定义的 lab1 和 lab2 均是标签类型的引用，input1 和 input2 是文本框类型的引用。在 init()方法中，产生了 4 个对象，并将它们放置到屏幕上。

在 action()方法中，if(e. target＝＝input1 ‖ e. target＝＝input2)语句用于确定是否在 input1 或 input2 中按下了 Enter 键。input1. getText()是获得文本框 input1 中的文本信息，然后采用系统提供的 Integer. parseInt()方法将这个文本转换为一个整数，并传递给变量 num1。

程序中的 if-else 语句用于比较 num1 和 num2 的大小，showStatus()方法是将其括号中的参数在状态栏输出。若 num1 的值是 123，num2 的值是 432，num1＋"＜"＋num2 就是将 num1 的值后面跟上一个字符串"＜"，然后再跟上 num2 的值，共同构成了一个字符串，此处的＋是一个 String 连接符，而不是一个算术运算符。

3.1.2　switch 语句

Java 的 switch 语句的语法结构如下：

```
switch(integral-expression)
{
    case  integral-value1:
                              statement;
                              break ;
        …
    case  integral-valuen:
                              statement;
                              break ;
    default:
                              statement;
                              break ;
}
```

switch 语句中的 integral-expression 表达式必须是 int、byte、char 和 short 这几种类型之一。integral-value 表达式的值必须是对应类型的常量,并且常量不能重复。switch 把 integral-expression 的值和每个 integral-value 逐一比较,若找到相同者,就执行相应的 statement,若找不到相同者,就执行 default 中的 statement。每个 case 语句后面都有一个 break 语句,若缺少了该语句,便会继续执行其后的 case 语句。虽然 default 语句的 break 是多余的,不过考虑到编程风格问题,带上也无妨。下面看一个 switch 语句的举例:

【例 3-2】 switch 语句应用举例。

```
package  chapter3;
public class justVowels       //判断一个字母是否为元音
{
    public static void main(String args[])
    {
        char c;
        for(int i=0;i<100;i++)       //随机产生 100 个字母,进行判断
        {
            c=(char)(Math.random() * 26+'a');
            System.out.print(c);
            switch(c){
                case  'a':
                case  'e':
                case  'i':
                case  'o':
                case  'u':
                      System.out.println("是元音");
                      break;
                case  'y':
                case  'w':
                      System.out.println("有时是元音");
                      break;
                default:
                      System.out.println("不是元音");
                      break;
            }
        }
    }
}
```

【程序运行结果】 编译运行后,一个可能的结果是:

```
i 是元音
w 有时是元音
k 不是元音
w 有时是元音
v 不是元音
```

```
n 不是元音
b 不是元音
d 不是元音
k 不是元音
w 有时是元音
```

【程序解析】 程序中的变量 i 定义在 for 循环中，其作用域为该循环语句，当循环结束时，变量 i 就消失。Math.random()产生的是一个[0,1) 之间的随机值，将该值乘以 26 加上 'a'，然后取整，就是 26 个小写字母的 ASCII 码。需要说明的是，类型强制转换是临时的，并且不会进行四舍五入运算，而是截断，例如(char)(97.89)，结果就是 97，即'a'的 ASCII，而不是 98。程序最后利用 switch-case 语句判断是否为元音字母。

【例 3-3】 采用 Java Applet 小程序实现将学生的百分制成绩转换为优秀、良好、中等、及格和不通过 5 个等级。

```
package  chapter3;
import java.awt.*;
import java.applet.*;

public class scoreConvert extends Applet
{
    Label  prompt;
    TextField  input;
    int  Excellent, Good, Middle, Pass,Failure;

    public void init()
    {
        prompt=new Label("输入成绩");
        input=new TextField(2);
        add(prompt);
        add(input);
    }

    public void paint(Graphics g)
    {
        g.drawString("各等级的人数:",25,40);
        g.drawString("优秀   : "+Excellent,25,55);
        g.drawString("良好   : "+Good,25,70);
        g.drawString("中等   : "+Middle,25,85);
        g.drawString("及格   : "+Pass,25,100);
        g.drawString("不通过: "+Failure,25,115);
    }
    public boolean action(Event e , Object o)
    {
            //从当前引发事件的对象 input 获取一个整数
        int  score=Integer.parseInt(input.getText());
        showStatus("");
```

```
            input.setText("");
            switch(score/10)
            {
                case  10:
                case  9:
                            Excellent++;
                            break;
                case  8:
                            Good++;
                            break;
                case  7:
                            Middle++;
                            break;
                case  6:
                            Pass++;
                            break;
                case  5:
                case  4:
                case  3:
                case  2:
                case  1:
                case  0:
                            Failure++;
                            break;
                default:
                           showStatus("输入有误,请重新输入!");      //显示错误信息
            }
            repaint();       //注意:容易忘记的地方
            return  true;
        }
    }
```

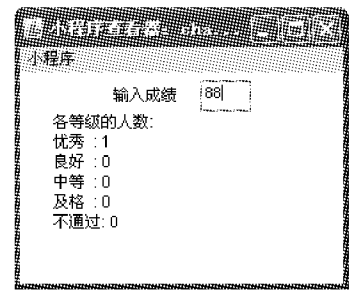

图 3-2　采用小程序对百分制成绩转换为五分制进行统计

【程序运行结果】 如图 3-2 所示。

【程序解析】 在主类 scoreConvert 中定义的 Excellent、Good、Middle、Pass 和 Failure 分别用于统计各个等级的人数,由于它们是类内的成员变量,并且是整型,因此初始值默认为 0。程序首先执行 init()方法,然后调用 paint()方法,所以程序的开始界面显示这 5 个变量的值是 0。

当用户在文本框中输入一个整型数据并且按了 Enter 键后,自动调用 action()方法。在 action()方法中,调用 input.getText()方法从文本框对象获取对应的字符串,Integer.parseInt()方法将该字符串转换为一个整数,因此 score 的值就是用户在文本框中所输入的值。showStatus("")是将状态栏显示的信息清空,input.setText("")是将文本框对象显示的内容清空。若用户在文本框中输入有误,例如输入了 567,或者是-12 等,都会在状态栏显示错误信息。

【注意】 repaint()方法是一个系统方法,它自动调用 paint()方法(后续章节将分析其

具体调用),从而实现了对网页的刷新。若漏写了这一行,就看不到正确结果了。此外,由于action()方法是 boolean 类型,所以该方法最后要返回一个布尔值。

3.2　循环控制语句

Java 中的循环控制语句有 3 种,分别是 while、do-while 和 for 语句。循环体内的语句会反复执行,直到用于控制循环的布尔表达式的值变为 false 为止。

3.2.1　while 语句

while 语句的循环形式是:

```
while(Boolean-Expression)
    statements;
```

循环控制条件 Boolean-Expression 会在循环开始时判断一次,在循环体执行结束以后,再次判断 Boolean-Expression,以确定是否还执行循环体。

【例 3-4】 产生 10 个 60～100 之间的随机整数。

```
package  chapter3;
public class GenerateNumbers
{
    public static void main(String  args[])
    {
        int  i=0, Int_val=0;
        while(i<10)
        {
            Int_val=(int)(Math.random() * (100-60)+60);
            System.out. printf("%5d",Int_val);
            i++;
        }
    }
}
```

【程序运行结果】 由于采用了随机数,故一次可能的运行结果如下:

```
66   99   95   68   78   75   87   69   66   91
```

【程序解析】 程序通过循环控制变量 i,保证循环执行 10 次,每次产生一个整数,输出时的域宽为 5。

3.2.2　do-while 语句

do-while 语句的语法格式如下:

```
do{
    statement;
}while(Boolean-Expression);
```

【注意】 do-while 语句和 while 语句类似,它们的区别是 while 语句是先判断后执行,若 Boolean-Expression 为假,整个循环体一次也不执行;而 do-while 语句是先执行后判断,所以循环体至少要执行一次。

【例 3-5】 直到产生一个大于 0.9 的随机数为止。

```
package  chapter3;
public class GenerateDoubleNumbers
{
    public static void main(String  args[])
    {
        double  d;
        do{
            d=Math.random();
            System.out. println(d);
        }while(d<0.9);
    }
}
```

【程序运行结果】 某次运行结果如下:

```
0.418710        0.524779        0.038600        0.936135
```

【程序解析】 由于要求产生一个大于 0.9 的随机数,因此程序先产生一个随机数,而后判断是否满足条件,若满足条件就停止循环,否则继续产生下一个随机数。

3.2.3 for 语句

for 语句的语法格式如下:

```
for(ForInitopt; Boolean-Expression; ForUpdateopt)
    Statement;
```

在第一次循环前,先执行初始化语句 ForInitopt;再进行条件测试,若 Boolean-Expression 为真,就执行循环体;然后执行 ForUpdateopt 部分,转入条件测试;若 Boolean-Expression 为假,整个循环结束。

【注意】 for 语句中的 ForInitopt、Boolean-Expression 和 ForUpdateopt 都可以为空。

【例 3-6】 编写一个 applet,输出一个倒三角形图案。

```
package  chapter3;
import java.awt.*;
import java.applet.Applet;
public class printGraphics extends Applet
{
    public void paint(Graphics g)
    {
        int xpos,ypos=80;
```

```
        for(int row=6;row>=1;row--)
        {
            xpos=25;
            ypos+=10;
            for(int column=1;column<=row;column++)
            {
                g.drawString("*",xpos,ypos);
                xpos+=7;
            }
        }
    }
}
```

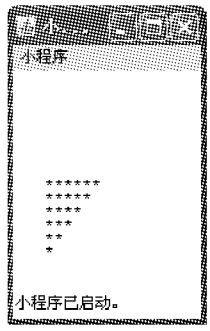

图 3-3　采用小程序输出一个倒三角形

【程序运行结果】 如图 3-3 所示。

【程序解析】 程序中的 g. drawString(" * ", xpos, ypos)是在指定的 xpos 和 ypos 位置输出一个 * 。ypos＋＝10 是对输出 * 的 Y 轴坐标增量，即行距设定为 10 个像素；xpos＋＝7 是对 X 轴坐标增量，即字符 * 之间的距离设定为 7 个像素。

【思考】 如果将程序中的“ypos＋＝10”改为“ypos－－10”，你还知道结果吗？

3.3　break 语句和 continue 语句

将 Java 中的 break 语句和 continue 语句分为两类讨论，一类是不带标号的语句；另一类是带标号的语句。

3.3.1　不带标号的 break 语句和 continue 语句

采用 break 语句可以控制循环的流程，break 语句可以跳出包含它的最内层的循环，不再执行剩余的语句，continue 语句会停止执行当前的循环，回到循环处，开始执行下一轮的循环。这些特性和 C/C++ 中的 break 语句和 continue 语句的功能一样。

【例 3-7】 基本的 break 语句和 continue 语句。

```
package  chapter3;
public  class breakANDcontinue
{
      public static void main(String args[])
      {
          for(int i=1;i<20;i++)
          {
              if(i%9==0)
                    break;
              if(i%3==1)
                    continue;
              System.out.printf("%5d",i);
```

```
            }
        }
    }
```

【程序运行结果】 输出如下：

```
2       3       5       6       8
```

【程序解析】 当 i 的值等于 9 时，break 语句就终止了当前的循环，所以输出的值是到 8 为止。在 1～8 之间，若满足条件 i%3==1，将转入下一次循环，其后面的语句也不再执行，所以 i 的值等于 1、4、7 时，均未输出。

3.3.2 带标号的 break 语句和 continue 语句

Java 不但保留了与 C/C++ 相似的 break 语句和 continue 语句特性，而且还引入了带标号的 break 语句和 continue 语句。当在循环体中执行带标号的 break 语句时，可以立即退出任意多个嵌套循环。从程序设计语言原理的角度讲，这种带标号的语句是 goto 语句的一种变形，它使程序具有一定的灵活性。

带标号的 break 语句的语法格式如下：

```
break  Identifier;
```

带标号的 continue 语句与 break 语句类似：

```
continue  Identifier;
```

其中 Identifier 是一个标识符。这种带标号的语句类似于 C/C++ 中的 goto 语句，尽管 goto 是 Java 的保留字，但 Java 并未使用它。

【例 3-8】 带标号的 break 语句和 continue 语句。

```
package  chapter3;
public  class hello
{
    public static void main(String args[])
    {
        int i,j=0;
outer:
        for(i=0;i<3;i++)
            for(j=0;j<5;j++)
            {
                System.out.println(i+"  "+j);
                if(j==1)
                    break  outer;       //注意该语句
            }
        System.out.println("最终值："+i+"  "+j);       //注意该语句的位置
    }
}
```