第3章

程序流程控制

计算机程序中，有些代码在满足条件时执行，有些代码需要反复执行，程序设计语言通过程序流程控制语句完成这些任务。Java 语言提供了选择结构和循环结构实现程序流程控制。Java 程序流程控制的语法结构是否与 C/C++ 一样？它有哪些新特点？编程实践中，该如何选择流程控制语句？本章将为读者一一解答这些问题。

本章内容

（1）if 语句、switch 语句和条件运算符等 3 种选择结构。

（2）while 语句、do…while 语句和 for 语句等 3 种循环结构。

（3）break 语句和 continue 语句。

3.1 选择结构


选择结构

现实生活中，常常需要根据具体情况做出不同决定。例如，交通警察查酒驾，如果驾驶人每 100mL 血液中，20mg≤酒精含量＜80mg 的情况下驾驶机动车属于饮酒驾车，如果酒精含量≥80mg 的情况下驾驶机动车属于醉酒驾车，饮酒驾车和醉酒驾车的处罚标准不同；教师根据课程考试分数赋成绩等级，分数≥90 为优秀，80≤分数＜90 为良好等。这些情况需要使用 Java 提供的选择结构语句完成任务。

与 C/C++ 等语言一样，Java 语言提供了 if 和 switch 两种选择结构语句，这两种语句根据条件决定执行的代码块。

3.1.1 if 语句


if 语句

if 语句是选择结构的基本语句，根据判断条件执行代码块，包括简单 if、if…else 和 if…else if…else 等 3 种。

1. 简单 if 语句

【语法格式 3-1】 if 语句。

```
if(判断条件){
    语句块                                    //语句主体
}
```

简单 if 语句的执行流程如图 3-1 所示，如果判断条件结果为 true，执行语句

块,否则不执行语句块。如果语句块仅有一条语句,可省略花括号。

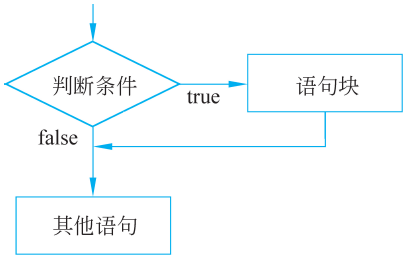

图 3-1　简单 if 语句的执行流程

程序案例 3-1 演示输入一个整数,判断该整数是否小于、大于或等于 0。例如,输入－25,输出“－25 小于 0”;输入 38,输出“38 大于 0”。

【程序案例 3-1】　简单 if 语句。

```
1   package chapter03;
2   import java.util.Scanner;
3   public class Demo0301 {
4       public static void main(String[] args) {
5           //声明输入流,从标准输入设备(键盘)输入数据
6           Scanner scan = new Scanner(System.in);
7           int aInt;
8           System.out.println("请输入一个整数:");
9           aInt = scan.nextInt();                    //输入一个整数
10          if (aInt > 0) {                           //如果输入的整数大于 0
11              System.out.println(aInt + "大于 0!");
12          }
13          if (aInt == 0) {                          //如果输入的整数等于 0
14              System.out.println(aInt + "等于 0!");
15          }
16          if (aInt < 0) {                           //如果输入的整数小于 0
17              System.out.println(aInt + "小于 0!");
18          }
19      }
20  }
```

简单 if 语句执行,判断条件为 true 时执行 if 语句块,该语句只能实现一种分支。实际问题中还存在判断条件为 false 的情况,这时需要使用 if…else 语句。


if…else 语句

2. if…else 语句

为了解决简单 if 语句不能在判断条件为 false 时执行语句块问题,Java 提供了 if…else 语句。判断条件为 true 时,执行 if 后面的语句块;判断条件为 false 时,执行 else 后面的语句块,执行流程如图 3-2 所示。

【语法格式 3-2】　if…else 语句。

```
if(判断条件){
    语句块 1;
}else{
    语句块 2;
}
```

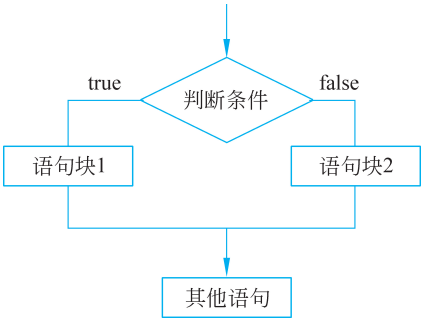

图 3-2　if…else 语句的执行流程

程序案例 3-2 演示 if…else 语句的使用方法。判断学生成绩是否及格，如果成绩≥60 分，输出成绩及格的提示信息；如果成绩＜60 分，输出成绩不及格的提示信息。第 5 行满足 score＞＝60 时执行第 6 行的语句，否则执行第 9 行的语句。

【程序案例 3-2】　if…else 语句。

```
1   package chapter03;
2   public class Demo0302 {
3       public static void main(String[] args) {
4           double score=55;
5           if(score>=60) {                          //如果成绩大于或等于 60
6               System.out.println("恭喜你,你的成绩及格了");
7           }
8           else {                                   //如果成绩小于 60
9               System.out.println("很遗憾,你的成绩不及格!");
10          }
11      }
12  }
```

3. if…else if…else 语句


if…else if…else 语句

实际应用中，有些场景使用 if…else 语句就能解决，但在涉及多条件判断的复杂场景时，if…else 的能力有些不足。例如，水果店销售水果，春、夏、秋、冬四季销售的主要时令水果不同。这些场景需要根据多个不同条件执行不同操作，采用 if…else if…else 语句比较适合。该语句的执行流程如图 3-3 所示。如果判断条件 1 为 true，执行语句块 1；否则执行判断条件 2，如果为 true，执行语句块 2；以此类推，否则执行判断条件 n。

【语法格式 3-3】　if…else if… else 语句。

```
if(判断条件 1){
    语句块 1;
}else if(判断条件 2){
    语句块 2;
}else if(判断条件 3){
    语句块 3;
}
…//n 个 else if 语句
else{
    语句块 n+1;
}
```

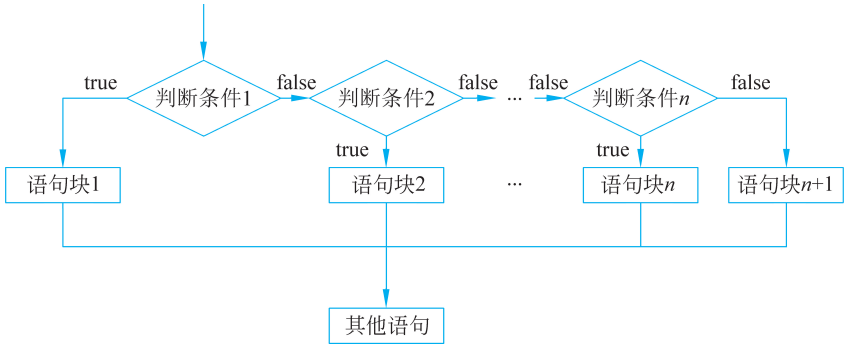

图 3-3　if…else if …else 语句的执行流程

程序案例 3-3 演示了 if…else if…else 多条件选择结构语句。输入季节序号,输出对应季节出产的主要水果。第 6 行建立输入流,键盘作为输入设备;第 8 行接收从键盘输入的 int 数据;第 9～18 行是 if…else if…else 语句。

【程序案例 3-3】　if…else if…else 语句。

```
1   package chapter03;
2   import java.util.Scanner;
3   public class Demo0303 {
4       public static void main(String[] args) {
5           int season;                                    //季节
6           Scanner scan = new Scanner(System.in);         //键盘作为输入设备
7           System.out.println("请输入季节(1~4:春夏秋冬)");
8           season = scan.nextInt();                       //输入季节
9           if (1 == season)                               //春季
10              System.out.println("春季主要水果:荔枝、枇杷、青枣等!");
11          else if (2 == season)                          //夏季
12              System.out.println("夏季主要水果:杧果、西瓜、葡萄等!");
13          else if (3 == season)                          //秋季
14              System.out.println("秋季主要水果:石榴、无花果、菠萝等!");
15          else if (4 == season)                          //冬季
16              System.out.println("冬季主要水果: 橙子、草莓、牛奶蕉等!");
17          else                                           //没按要求输入
18              System.out.println("输入错误,请重新输入!");
19      }
20  }
```

程序运行结果如图 3-4 所示。

图 3-4　程序案例 3-3 运行结果

3.1.2　switch 语句

if…else if…else 语句能处理多条件分支,如果条件分支比较多,这种处理方式比较烦

琐。例如，根据月份输出该月的主要水果则要写 11 条 else if 语句，这种方式使程序显得比较复杂。Java 语言提供了便捷的开关语句 switch 实现多分支选择，该语句使程序的多重条件判断结构清晰，容易阅读。switch 语句的执行流程如图 3-5 所示。如果表达式的值等于选择值 1，执行语句块 1，break 语句退出该 switch。否则，如果表达式的值等于选择值 2，执行语句块 2，break 语句退出 switch；如果表达式的值不等于选择值 n，执行 default 中的语句块。

【语法格式 3-4】 switch 语句。

```
switch(表达式){
    case 选择值 1:    语句块 1;
                    break;
    case 选择值 2:    语句块 2;
                    break;
    …
    case 选择值 n:    语句块 n;
                    break;
    default:        语句块;
}
```

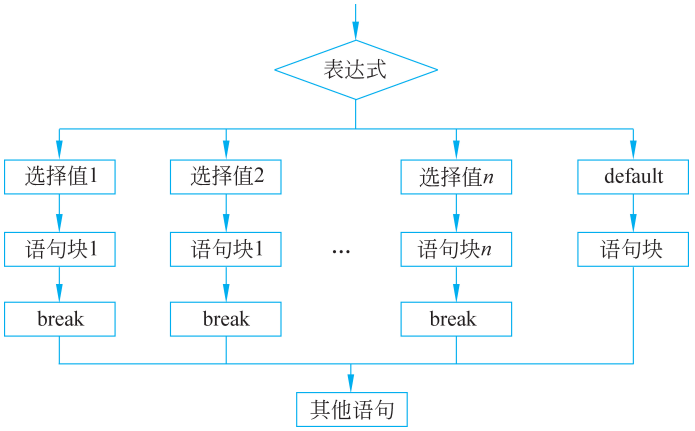

图 3-5　switch 语句的执行流程

使用 switch 语句需要注意 3 个问题：①switch 语句的选择表达式包括 byte、char、short、int、String 和枚举等类型，不能是浮点数或者 long 类型；②switch 语句先计算表达式，然后根据表达式的值检测是否匹配 case 后面的选择值，若不匹配所有 case 的选择值，则执行 default 语句块，执行完毕后离开 switch 语句；③如果某个 case 的选择值匹配表达式的结果，执行该 case 的语句块，一直遇到 break 语句后才退出 switch 语句，若 case 语句块中没有 break 语句，则程序一直执行到 switch 语句尾。

程序案例 3-4 演示了 switch 语句的使用方法。第 9 行 week 是 byte 类型，第 13、17、21、25 和 29 行是 break 语句，作用是退出 switch 语句。程序运行结果如图 3-6 所示。

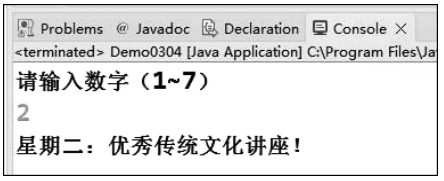

图 3-6　程序案例 3-4 运行结果

【程序案例 3-4】 switch 语句。

```
1   package chapter03;
2   import java.util.Scanner;
3   public class Demo0304 {
4       public static void main(String[] args) {
5           byte week;                                  //星期几
6           Scanner scan = new Scanner(System.in);      //键盘作为输入设备
7           System.out.println("请输入数字(1~7)");
8           week = scan.nextByte();                     //输入星期几
9           switch (week) {
10          case 1:                                     //星期一
11              System.out.print("星期一:");
12              System.out.println("华为成长史讲座!");
13              break;                                  //退出 switch 语句
14          case 2:                                     //星期二
15              System.out.print("星期二:");
16              System.out.println("优秀传统文化讲座!");
17              break;                                  //退出 switch 语句
18          case 3:                                     //星期三
19              System.out.print("星期三:");
20              System.out.println("软件需求工程!");
21              break;                                  //退出 switch 语句
22          case 4:                                     //星期四
23              System.out.print("星期四:");
24              System.out.println("软件测试!");
25              break;                                  //退出 switch 语句
26          case 5:                                     //星期五
27              System.out.print("星期五:");
28              System.out.println("劳动教育!");
29              break;                                  //退出 switch 语句
30          case 6:                                     //星期六
31          case 7:                                     //星期日
32              System.out.println("周末了,应该休息哟!");
33              break;                                  //退出 switch 语句
34          default:                                    //没有按要求输入
35              System.out.println("输入错误!");
36          }
37      }
38  }
```


switch 语句

程序案例 3-5 演示了输入一个简单四则表达式(例如,a+b),输出计算结果。第 13 行接收输入的一个四则运算表达式(字符串),第 15、17、19、21 行获得运算符的位置,第 26 行 expression.charAt(operationLocation)取出运算符,第 28、30 行从字符串取得两个操作数并转换为 double 类型,第 31 行使用 switch 语句匹配运算符并进行相应计算。程序运行结果如图 3-7 所示。

图 3-7 程序案例 3-5 运行结果

【程序案例 3-5】 switch 语句的应用。

```
1   package chapter03;
2   import java.util.Scanner;
3   public class Demo0305 {
```

```
4       public static void main(String args[]) {
5           String expression;                              //四则运算表达式为字符串
6           double firstNum;                                //第一个数
7           double secondNum;                               //第二个数
8           char operation;                                 //操作符
9           double result = 0.0;                            //运算结果
10          int operationLocation = -2;                     //运算符的位置
11          Scanner scan = new Scanner(System.in);          //键盘作为输入设备
12          System.out.println("请输入四则运算表达式:");
13          expression = scan.nextLine();                   //输入一个四则运算表达式
14          if (expression.indexOf("+") >= 1)               //判断加号的位置
15              operationLocation = expression.indexOf("+");
16          else if (expression.indexOf("-") > 0)           //判断减号的位置
17              operationLocation = expression.indexOf("-");
18          else if (expression.indexOf("*") >= 1)          //判断乘号的位置
19              operationLocation = expression.indexOf("*");
20          else if (expression.indexOf("/") >= 1)          //判断除号的位置
21              operationLocation = expression.indexOf("/");
22          else {                                      //没有输入所要求的四则运算表达式
23              System.out.println("输入错误,请输入正确的四则运算表达式!");
24              System.exit(0);                             //退出系统
25          }
26          operation = expression.charAt(operationLocation);     //取出运算符
27          //取得第一个数
28          firstNum = Double.parseDouble (expression.substring(0,
                                        operationLocation));
29          //取得第二个数
30          secondNum = Double.parseDouble (expression.substring
                                          (operationLocation + 1));
31          switch (operation) {                            //匹配运算符
32          case '+':                                       //运算符+
33              result = firstNum + secondNum;
34              break;
35          case '-':                                       //运算符-
36              result = firstNum - secondNum;
37              break;
38          case '*':                                       //运算符*
39              result = firstNum * secondNum;
40              break;
41          case '/':                                       //运算符/
42              result = firstNum / secondNum;
43          }
44          System.out.println(expression + "=" + result);  //输出运算结果
45      }
46  }
```

3.1.3　条件运算符

使用 if…else 语句解决问题时,在 if 和 else 语句块只有一条语句的特殊情况下,能够使用条件运算符代替该语句。条件运算符“?:”是三目运算符。

【语法格式 3-5】 条件运算符。

```
变量=布尔表达式?表达式 1 :表达式 2;
```

该语句执行过程：如果布尔表达式为 true，把表达式 1 的运算结果赋给变量，否则把表达式 2 的运算结果赋给变量。条件运算符与 if…else 语句的关系如图 3-8 所示。

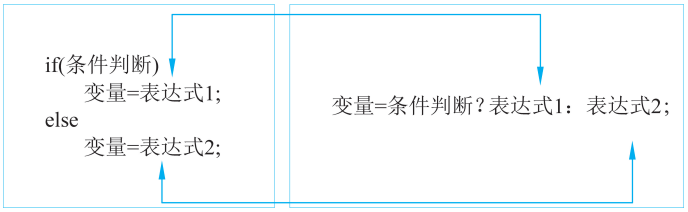

图 3-8　条件运算符与 if…else 语句的关系

下面代码段中，如果销售额大于销售目标，薪资 salary ＝salary * （1＋0.3），否则薪资 salary ＝salary * （1－0.3）。第 5 行使用条件运算符计算薪资，第 8～11 行使用 if…else 语句计算薪资。

```
1   double target=100;                                    //销售目标
2   double yourSales=140;                                 //销售额
3   double salary = 3000;                                 //薪资基数
4   //条件运算符计算薪资
5   salary = yourSales > target ? salary * (1 + 0.3) : salary * (1 - 0.3);
6   System.out.println("你的薪资:" + salary);
7   //if…else 语句计算薪资
8   if(yourSales > target)
9       salary=salary * (1 + 0.3) ;
10  else
11      salary=salary * (1 -0.3) ;
```


循环结构

3.2　循环结构

日起日落，秋去冬来，自然界普遍存在一些规律性现象。在现实生活中，周期性出现的事情比比皆是，如国家每五年制定下一个五年发展规划、超市收银员扫描顾客购买的若干商品条形码录入商品信息、高速公路电子警察对路过车辆拍照记录、统计文档中某个字符出现的次数等。循环结构语句能有效处理重复出现的事情。Java 提供了 while、do…while 和 for 等 3 种循环结构语句，如果事先不知道循环次数，一般采用 while 语句和 do…while 语句，for 语句一般用于事先知道循环次数的情况。

3.2.1　while 语句

如果不能确定循环体语句块的执行次数，而仅仅知道循环结束条件，while 语句比较适合这种情况。

【语法格式 3-6】　while 语句。

```
while(布尔表达式){
    循环体语句块
}
```

while 语句的执行流程如图 3-9 所示。

while 语句的执行过程：首先计算循环条件，如果为 true，执行循环体语句块（循环体语句块一般包含使循环条件为 false 的语句），并继续计算循环条件；如果循环条件为 false，退出循环结构，执行该循环语句后面的其他语句。

该语句的显著特点是先判断后执行，极端情况下循环体语句块可能没有执行机会。

程序案例 3-6 演示了 while 语句的使用方法。计算 1＋2＋…＋99。第 6 行 while 语句，当 a ＜＝ 99 为 true 时重复执行第 7、8 行代码组成的语句块。第 8 行修改循环变量 a 的值，每循环一次 a 的值增加 2，当 a 的值大于 99 时结束 while 语句，执行第 10 行。程序运行结果如图 3-10 所示。

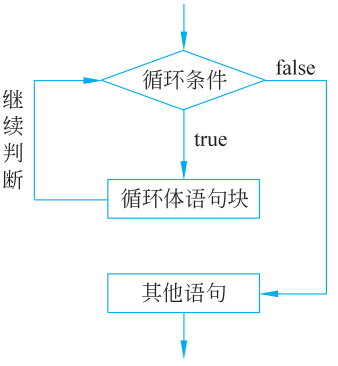

图 3-9　while 语句的执行流程

while 语句

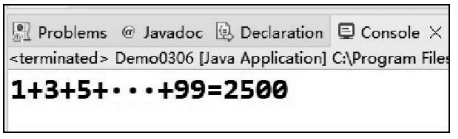

图 3-10　程序案例 3-6 运行结果

【程序案例 3-6】　while 语句。

```
1   package chapter03;
2   public class Demo0306 {
3       public static void main(String[] args) {
4           int a = 1;                          //记录循环变量
5           int sum = 0;                        //记录累加和
6           while (a <= 99) {                   //满足 a<= 99 时,执行语句块
7               sum = sum + a;                  //累加
8               a = a + 2;                      //修改循环变量
9           }
10          System.out.println("1+3+5+…+99=" + sum);
11      }
12  }
```

3.2.2　do…while 语句

实际生活中，存在先完成一个任务，然后判断是否需要重复执行的情况。例如，击鼓传花游戏，击鼓人开始击鼓时，游戏开始，道具在人群中传送，击鼓人停止击鼓时，拿道具的人按要求表演节目。游戏时，道具在人群中重复传送（相当于重复执行一段代码），击鼓人停止击鼓时拿道具的人表演节目（表示循环条件为 false，退出循环，开始表演节目）。

成功的产品都有规律可循，它们的诞生并不源于颠覆式的发明创造，而是源于“微创新”。这种场景使用 do…while 语句比较合适，该语句是 while 语句的改进版。do…while 是另一种不需要知道循环次数的循环结构语句，它的特点是先判断后执行。

【语法格式 3-7】 do…while 语句。

```
do{
    循环体语句块
}while(判断条件);
```

do…while 语句的执行流程如图 3-11 所示。

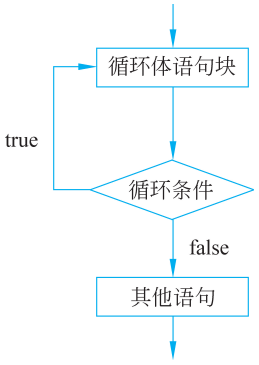

图 3-11 do…while 语句的执行流程

do…while 语句执行过程：进入 do…while 循环语句，执行循环体语句块；如果循环条件为 true，继续执行循环体语句块；如果循环条件为 false，退出循环，执行 do…while 之后的其他语句。

while 语句与 do…while 语句都不需要知道循环执行次数，使用时存在区别：①while 语句先判断循环条件，然后执行循环体语句块，存在循环体语句块没有被执行的可能性；②do…while 语句先执行循环体语句块，然后判断循环条件，至少执行一次循环体语句块。


do…while 语句

程序案例 3-7 演示了 do…while 语句的基本使用方法。计算 2＋22＋222＋…＋2222222。先执行第 7、8 行的语句块，然后判断第 9 行的循环条件。程序运行结果如图 3-12 所示。

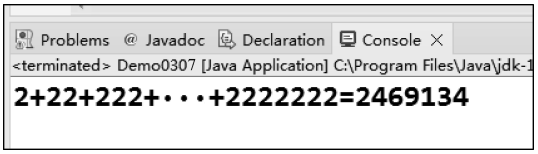

图 3-12 程序案例 3-7 运行结果

【程序案例 3-7】 do …while 语句。

```
1   package chapter03;
2   public class Demo0307 {
3       public static void main(String[] args) {
4           long a = 2;                              //循环变量初始值 2
5           long sum = 0;                            //累加和初始值 0
6           do {
7               sum = sum + a;                       //累加
8               a = a * 10 + 2;                      //修改循环变量
9           } while (a <= 2222222);                  //循环条件
10          System.out.println("2+22+222+…+2222222=" + sum);
11      }
12  }
```

前面学习了选择结构语句和 while、do…while 等两种循环结构语句。软件开发的实际情况往往比较复杂，需要综合运用循环语句和选择语句才能解决比较复杂的问题。

程序案例 3-8 综合运用 do…while 和 if…else 两种语句解决一个较复杂问题。模拟汽车行驶情况，使用随机数模拟汽车行驶速度，在跟踪时间段内，记录超速和正常驾驶的汽车数量。第 12、22 行获取系统当前时间；第 23 行计算程序运行时间，当运行时间小于 30ms 时继续跟踪，否则停止 do…while 语句；第 15 行产生不大于 200 的整数模拟汽车行驶速度；

第 16 行判断汽车速度，如果超速给出提示信息。程序运行结果如图 3-13 所示。

图 3-13　程序案例 3-8 运行结果

【程序案例 3-8】　选择结构语句和循环结构语句综合案例。

```
1   package chapter03;
2   import java.util.Random;
3   public class Demo0308 {
4       public static void main(String args[]) {
5           int speed;                                  //汽车行驶速度
6           double upNum = 0;                           //超速车辆数
7           double normalNum = 0;                       //正常行驶车辆数
8           double totalAuto = 0;                       //车辆总数
9           long startTime;                             //开始时间
10          long endTime;                               //结束时间
11          long tranceTime;                            //跟踪时间
12          startTime = System.currentTimeMillis();  //获取开始时间
13          Random random = new Random();               //产生随机数对象
14          do {                                        //直接进入循环体
15              speed = random.nextInt(200);  //随机产生不大于 200km/h 的汽车行驶速度
16              if (speed >= 120) {           //如果速度超过 120km/h,表示超速
17                  upNum++;                            //超速车辆数加 1
18                  System.out.println("你已超速,请遵守交通规则!");
19              } else {                                //没有超速
20                  normalNum++;                        //正常行驶车辆数加 1
21              }
22              endTime = System.currentTimeMillis();  //记录当前时间
23              tranceTime = endTime - startTime;      //计算跟踪时间(程序运行时间)
24          } while (tranceTime <= 30);          //如果跟踪时间小于 30ms,则继续跟踪
25          totalAuto = upNum + normalNum;              //统计跟踪汽车数量
26          System.out.println("跟踪时间:" + tranceTime + " ms");
27          System.out.println("汽车总数:" + totalAuto);
28          System.out.println ("超速车辆:" + upNum + ",超速比例:" + upNum /
                        totalAuto);
29          System.out.println ("正常行驶车辆:" + normalNum + ",正常行驶比例:" +
                        normalNum / totalAuto);
30      }
31  }
```

3.2.3 for 语句

while 语句和 do…while 语句的使用场景各有侧重,合理运用它们能解决所有需要循环处理的问题。但是,对于已经事先知道循环次数的场景,Java 提供的 for 语句能比较方便地解决这类问题,for 语句包含初值表达式、循环条件和循环过程表达式,使用起来比较简洁,深受程序员们喜爱。

【语法格式 3-8】 for 语句。

```
for(初值表达式;循环条件;循环过程表达式){
    循环语句块;
}
```

for 语句的执行流程如图 3-14 所示。

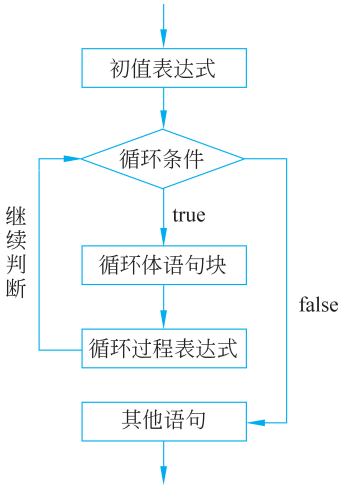

图 3-14 for 语句的执行流程

for 语句的执行流程分为 3 步:①第一次进入 for 循环时,执行初值表达式,为循环控制变量赋初始值。②根据循环条件决定是否执行循环体语句块,如果循环条件为 true,继续执行循环体语句块;如果循环条件为 false,退出循环,执行 for 后面的其他语句。③执行循环体语句块后,执行循环过程表达式,然后再回到步骤②判断是否继续循环。


for 语句

程序案例 3-9 演示了 for 语句的使用方法。输入一个字符串,统计字符串中的字母、数字以及其他符号的个数。第 15 行的 for 语句逐个取得字符串中的每个字符,第 16 行 myString.charAt(i)取得字符串位置 i 的字符,第 17 行判断字符类别。程序运行结果如图 3-15 所示。

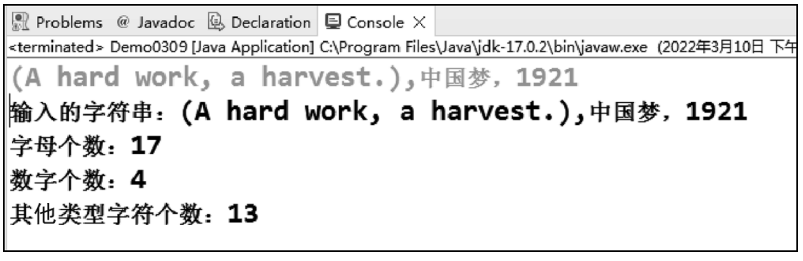

图 3-15 程序案例 3-9 运行结果

【程序案例 3-9】 for 语句。

```
1  package  chapter03;
2  import java.util.Scanner;
3  public class Demo0309 {
4      public static void main(String[] args) {
5          String myString = null;                    //字符串
6          int stringLength = 0;                      //字符串的长度
7          int letterNum = 0;                         //字母个数
8          int decimalNum = 0;                        //数字个数
```

```
9           int otherNum = 0;                                   //其他字符个数
10          char currentChar;                                   //当前字符
11          Scanner scan = new Scanner(System.in);              //键盘作为输入设备
12          System.out.println("请输入字符串:");
13          myString = scan.nextLine();                         //输入字符串
14          stringLength = myString.length();                   //取得字符串长度
15          for (int i = 0; i < stringLength; i++) {            //逐个取得字符串中的每个字符
16              currentChar = myString.charAt(i);               //取得字符串位置 i 的字符
17              if (currentChar >= '0' && currentChar <= '9') //如果字符是数字
18                  decimalNum++;                               //数字个数加 1
19              //如果字符是字母
20              else if ((currentChar >= 'a' && currentChar <= 'z') ||
                        (currentChar >= 'A' && currentChar <= 'Z'))
21                  letterNum++;                                //字母个数加 1
22              else                                            //字符不是数字也不是字母
23                  otherNum++;                                 //其他字符个数加 1
24          }
25          System.out.println("输入的字符串:" + myString);
26          System.out.println("字母个数:" + letterNum);
27          System.out.println("数字个数:" + decimalNum);
28          System.out.println("其他类型字符个数:" + otherNum);
29          scan.close();                                       //关闭输入流
30      }
31  }
```

运用循环语句时为了防止出现死循环的情况，要合理设置循环结束条件。循环控制有计数器控制和标记控制两种方法：计数器控制必须确保每执行一次循环，循环控制变量(计数器)要进行增量或减量计算，程序案例 3-9 中，for 循环语句采用计数器控制循环，每循环一次，计数器 i 的值就执行一次增量计算(增加 1)，使计数器 i 的值逐步接近循环条件为假的情况；标记控制循环需要确保标记在某种情况下能使循环条件为假，程序案例 3-7 中，标记就是循环变量 a，当 a>2222222 时循环条件为假。

如果没有正确设置循环退出条件，或者退出循环条件不可能成立，循环控制流程一直重复执行循环体语句块，无法退出循环，即产生死循环。如下面程序出现死循环。

```
1   package chapter03;
2   public class EndlessLoop {
3       public static void main(String[] args) {
4           int flag=1;                         //循环标记
5           while(flag>0){                      //循环标记大于 0 执行循环，小于 0 退出循环
6               System.out.println("地球呼唤绿色");
7               flag++;                         //循环标记加 1，循环标记不可能小于 0
8           }
9       }
10  }
```

第 4 行设置循环标记 flag=1，第 7 行使循环标记 flag 加 1，flag 总是大于 0，因此第 5 行对该循环标记的判断永远为真，该程序将一直执行。

3.2.4　嵌套循环

while、do…while 和 for 是 3 种各有特点的循环结构语句，可以互相替换，但使用场景

的侧重点不同。作为语句,它们能出现在程序中任何可以使用语句的位置,如 while 循环体内可以使用 while、do…while 和 for 3 种语句。

嵌套循环指一条循环语句内包含另一条循环语句的情况,内嵌循环还可以嵌套循环语句,称为多层循环。while、do…while 和 for 3 种循环语句能互相嵌套,也可以自己嵌套自己。图 3-16 显示了嵌套循环结构,外层 while 语句嵌套了一条 for 语句。


嵌套循环

程序案例 3-10 演示了嵌套循环使用方法。输入一个整数,例如输入 4,计算 1+(1+2)+(1+2+3)+(1+2+3+4),使用嵌套循环处理该问题(当然不用嵌套也能解决该问题)。第 5 行 i 控制外层循环,第 6 行 j 控制内层循环,外层循环变量 i 取 0～x 的某个值,内层循环变量 j 从 1～i 进行循环,如 i=3 时,j 从 1～3 循环;第 10 行表示内层循环结束后,输出换行。程序运行结果如图 3-17 所示。

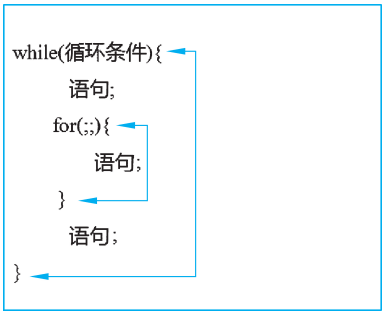

图 3-16　嵌套循环结构

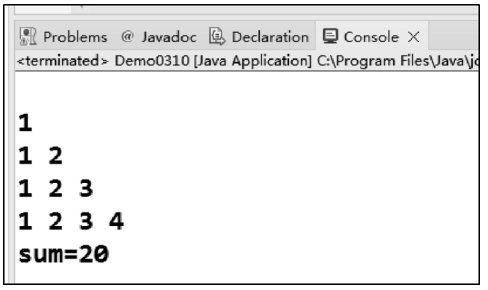

图 3-17　程序案例 3-10 运行结果

【程序案例 3-10】　嵌套循环。

```
1   package chapter03;
2   public class Demo0310 {
3       public static void main(String[] args){
4           int x = 4,sum = 0;
5           for(int i = 0 ; i<=x ; i++){            //外层循环
6               for(int j = 1 ; j<=i ; j++){        //内层循环,控制从 1 到某个整数
7                   sum = sum+j;                    //累加
8                   System.out.print(j+" ");        //输出当前数
9               }
10              System.out.println();               //换行
11          }
12          System.out.println("sum="+sum);
13      }
14  }
```

选择结构语句 if、if…else 和 switch 以及循环结构语句 while、do…while 和 for 都是 Java 语句,它们能在任何可以使用语句的位置出现。也就是说,循环结构语句之间可以嵌套,选择结构语句之间也可以嵌套,并且循环结构语句与选择结构语句之间可以混合嵌套。

下面代码段中,第 1 行的 while 循环语句包含了 if…else 语句(第 6～14 行)、do…while 语句(第 15～19 行)以及其他若干不同类型的语句。

```
1   while(循环条件){
2       语句 1;
```

```
3       语句 2;
4       ...
5       语句 n;
6       if(循环条件){
7           语句 1;
8           ...
9           语句 m;
10          }else{
11              语句 1;
12              ...
13              语句 k;
14          }
15      do{
16          语句 1;
17          ...
18          语句 s;
19      }while(循环条件);
20  }
```

3.3　跳转语句


跳转语句

已经证明，由顺序、选择和循环 3 种基本结构构成的算法能解决任何复杂问题，但在具体程序设计实践中，为了提高代码效率，少数情况下需要使用跳转语句改变程序执行流程。例如，在包含 1 万个元素的集合中查找某个特定数据，如果第 1000 次比较时找到该元素，不用比较余下的元素，退出查找元素算法，这种情况需要使用跳转语句退出查找循环。

Java 提供了 break、continue 和 return 3 种程序流程跳转语句，它们能改变程序执行流程，break 作为单独一条语句使用，continue 只能使用在循环结构中，return 退出当前方法并返回一个值。根据结构化程序设计原则，程序开发中不鼓励使用 break 和 continue 语句，因为随意改变程序执行流程，增加了调试和阅读程序的难度。

3.3.1　break 语句


break 语句

break 语句用于下面两种情况。

(1) switch 中使用 break 语句。执行 break 语句时，程序跳出 switch 语句，执行 switch 后面的语句。

(2) 循环结构中使用 break 语句。执行 break 语句时，中断当前循环结构，执行该循环语句的下一条语句，如果 break 语句出现在嵌套循环的内层循环，break 语句跳出当前层的循环。

【语法格式 3-9】 break 语句。

```
break;
```

图 3-18 显示了 break 语句应用在循环结构中的执行流程。

程序案例 3-11 演示了 break 语句的使用方法。计算 sum＝1＋2＋3＋…＋x，当 sum≥1949 时，输出 x。第 5 行循环条件为 true，循环体内需要设置退出循环体的语句，否则是死循环；第 8 行满足 sum ＞＝ 1949 时，执行第 9 行的 break 语句退出第 5 行的 while 循环语

句。程序运行结果如图 3-19 所示。

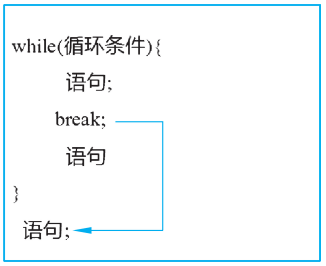

图 3-18　break 语句应用在 while 循环语句情况

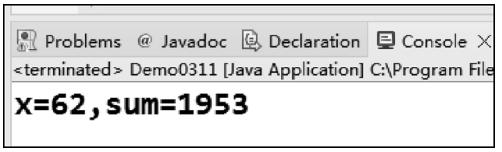

图 3-19　程序案例 3-11 运行结果

【程序案例 3-11】　break 语句案例。

```
1   package chapter03;
2   public class Demo0311 {
3       public static void main(String[] args) {
4           int x = 0, sum = 0;
5           while (true) {              //循环条件为 true,循环体内使用 break 退出循环
6               x += 1;
7               sum = sum + x;
8               if (sum >= 1949)
9                   break;              //退出当前循环语句
10          }
11          System.out.println("x=" + x + ",sum=" + sum);
12      }
13  }
```


continue 语句

3.3.2　continue 语句

continue 是另一条跳转语句,它只能在循环语句中使用。程序执行 continue 语句时,会停止运行该语句之后的循环体内语句,回到循环开始处继续运行。

【语法格式 3-10】　continue 语句。

```
continue;
```

图 3-20 显示循环语句中使用 continue 语句的程序执行流程。

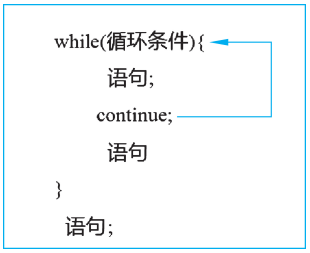

图 3-20　while 语句中使用 continue 语句的程序执行流程

程序案例 3-12 演示了 continue 语句的使用方法。输出 1～100 不能被 3 整除的整数,

每行输出 10 个。第 7 行判断当前整数 x 如果能被 3 整除，跳转到第 6 行 for 循环语句的开始位置。程序运行结果如图 3-21 所示。

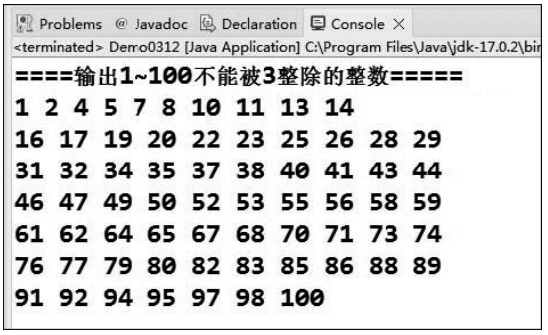

图 3-21　程序案例 3-12 运行结果

【程序案例 3-12】　continue 语句案例。

```
1   package chapter03;
2   public class Demo0312 {
3       public static void main(String[] args) {
4           int x = 1, count = 0;
5           System.out.println("====输出 1~100 不能被 3 整除的整数===== ");
6           for (; x <= 100; x++) {
7               if (0 == x % 3)                  //该数能被 3 整除，跳转到循环语句开始处
8                   continue;
9               System.out.print(x + " ");
10              count++;
11              if (0 == count % 10)             //如果输出了 10 个数
12                  System.out.println();        //输出换行
13          }
14      }
15  }
```

3.3.3　return 语句

与 C/C++ 语言一样，Java 提供的 return 语句返回一个方法的值，并把控制权交给调用它的语句。

【语法格式 3-11】　return 语句。

```
return  [表达式];
```

表达式是可选参数，表示返回值，它的数据类型要与方法声明的返回值类型兼容或者一致，可通过强制类型转换实现。

使用 return 语句要注意两点：①return 能放在选择结构语句和循环结构语句的任意位置；②如果 return 没有放在选择结构语句和循环结构语句中，只能放在方法最后位置，放在其他位置将出现编译错误。

下面代码段演示了 return 语句的使用方法。第 1 行定义方法 exam()；第 5 行，return 放置在 while 循环结构语句内，通过编译；第 7 行，return 语句不在循环或者选择结构语句

内,不能通过编译,因为第 7 行 return 使程序没有机会执行后面代码;第 9 行 return 放置在 if 语句内,通过编译;第 10 行,return 放置在方法最后,通过编译。

```
1   public static int exam(int a, int b) {
2       int i = 1;
3       while (i < 10) {
4           i++;
5           return 1;      //return 语句放在循环结构语句中
6       }
7        return i;         //return 不在选择结构语句和循环结构语句中,只能放在最后位置
8       if (a > b)
9           return a - b; //return 语句放在选择结构语句中
10      return b - a;      //return 语句放在方法最后位置
11  }
```

3.4 小　　结

本章主要知识点如下。

(1) 程序结构包括顺序结构、选择结构和循环结构。

(2) 当根据不同条件执行不同语句时,需要采用选择结构语句,选择结构语句包括 if、if…else、if…else if…else 和 switch。

(3) 当反复执行程序中的某些代码块时,需要采用循环结构语句,循环结构语句包括 while、do…while 和 for,使用循环结构语句时要避免死循环。

(4) break 语句使程序跳出 switch 语句和当前循环结构语句。continue 语句使程序跳转到循环结构语句的起始处。

(5) return 语句返回方法的值,并把控制权交给调用它的语句。

3.5 习　　题

3.5.1 填空题

1. (　　)语句用来强制退出循环结构语句。

2. 简单 if…else 语句可以使用(　　)运算符来代替。

3. (　　)和(　　)循环结构语句一般用在事先不能确定循环次数的情况下。

4. 取得字符串中某个位置的字符应该使用 string 类的(　　)方法。

5. 根据提示补全程序空白处,使程序能够正确运行。

```
//计算 1+2!+3!+4!+5!+6!
public class Demo11 {
    public static void main(String[] args) {
            long sum=0;
            long fac=1;                          //阶乘项的计算结果
            int i=1;
            while(    ①    ) {
                  fac=(    ②    );              //计算阶乘
                  i++;
```

```
                sum= (    ③    );            //累加
            }
            System.out.println("sum="+sum);
        }
    }
```

3.5.2　选择题

1. Java 语言中，下面的(　　)取得系统当前时间(ms)。

A. getTime();　　B. System.currentTimeMillis();

C. getDate();　　D. time();

2. Java 语言中，下面的(　　)产生一个不大于 100 的随机整数。

A. Random random=new Random();
aInt=random.nextInt(100);

B. getInteger(100);

C. System.getInt(100);

D. Random.nextInt(100);

3. Java 语言中，采用(　　)从键盘输入一行字符串。

A. scanf("%s",&ch);

B. System.out.println();

C. Scanner scan=new Scanner(System.in);
String expression=scan.nextLine();

D. getString();

3.5.3　简答题

1. 简述 break 语句在 switch 语句中的作用。
2. 画出 for 语句的执行流程图并简要说明执行过程。
3. 简述如何避免死循环。

3.5.4　程序设计题

1. 分别使用 3 种不同的循环语句，计算 1～1000 能被 7 和 11 同时整除的整数之和。
2. 有如下函数，编写程序，输入 x，然后输出 y。

$$y=\begin{cases}5+6x & x<0\\ -1 & x=0\\ 5-6x & x>0\end{cases}$$

3. 水仙花数是指一个三位整数的个位、十位、百位 3 个数的立方和等于该数本身，求出所有水仙花数。例如，$153=1^3+5^3+3^3$，153 是水仙花数。