第3章

结构化编程

Java是面向对象编程语言，支持面向对象基本特征，同时它也吸收了结构化编程的思想精华，支持结构化编程方法。本章主要讨论Java对结构化编程的支持，主要包括选择结构和循环结构。关于Java面向对象的概念，我们从第4章开始讨论。

本章内容要点

- 选择结构。
- switch语句和switch表达式。
- 循环结构。
- break语句和continue语句。
- 无限循环及嵌套循环。

扫一扫

视频讲解

3.1 选择结构

结构化编程(structured programming)方法于20世纪60年代提出，是软件发展的一个重要的里程碑。在结构化编程中，只允许三种基本的程序结构，它们是**顺序结构**、**选择结构**和**循环结构**。这三种基本结构的共同特点是只允许有一个入口和一个出口。仅由这三种基本结构组成的程序称为**结构化程序**。顺序结构比较简单，程序按语句的顺序依次执行。本节介绍选择结构，下节介绍循环结构。

Java有几种类型的选择语句：单分支if语句、双分支if-else语句、嵌套if语句、多分支if-else语句、switch语句和switch表达式。

3.1.1 单分支if语句

单分支if语句的一般格式如下：

```
if (<条件>){
      语句(组);
}
```

其中，<条件>是一个布尔表达式，布尔表达式应该使用圆括号括住，它的值为true或false。程序执行的流程是：首先计算条件表达式的值，若其值为true，则执行语句(组)的语句序列，否则转去执行if结构后面的语句。if语句的执行流程如图3-1所示。

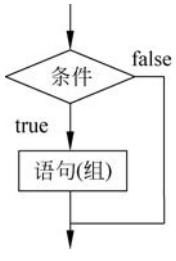

图3-1 if语句的执行流程

程序3.1要求用户从键盘输入两个整数，分别存入变量a与b，如果a大于b，则交换a和b的值，也就是保证a小于或等于b，最后输出a和b的值。

程序 3.1　ExchangeDemo.java

```
package com.boda.xy;
import java.util.Scanner;
public class ExchangeDemo{
    public static void main(String[ ] args){
      Scanner input = new Scanner(System.in);
      System.out.print("请输入整数 a:");
      int a = input.nextInt();
      System.out.print("请输入整数 b:");
      int b = input.nextInt();
      if(a > b){
         int t = b;
         b = a;
         a = t;
       }
       System.out.println("a = " + a);
       System.out.println("b = " + b);
   }
}
```

交换 a 与 b 的值，t 是临时变量

执行程序，输入 a 为 30，b 为 20，运行结果如下：

```
请输入整数 a:30
请输入整数 b:20
a = 20
b = 30
```

在 if 语句中，如果花括号内只有一条语句，则可以省略花括号。

```
if(num % 5 == 0){
    System.out.println(num + " 能被 5 整除.");
}
```

与下面的代码等价：

```
if(num % 5 == 0)
  System.out.println(num + " 能被 5 整除.");
```

提示　省略花括号可以使代码更整洁，但也更容易产生错误。将来需要为代码块增加语句时，容易忘记加上花括号，这是初学者常犯的错误。

3.1.2　双分支 if…else 语句

双分支 if…else 语句是最常用的选择结构，它根据条件是真是假，决定执行的路径。if-else 结构的一般格式如下：

```
if (<条件>){
     语句(组)1;
}else{
     语句(组)2;
}
```

该结构的执行流程是：首先计算<条件>表达式的值，如果为 true，则执行语句(组)1 的语句序列，否则执行语句(组)2 的语句序列。if…else 语句的执行流程如图 3-2 所示。

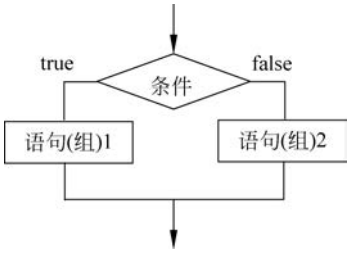

图 3-2　if…else 语句的执行流程

考虑下面的代码：

```
radius = input.nextDouble();
```

```
if(radius >= 0){
     area = Math.PI * radius * radius;
     System.out.println("圆的面积为:" + area);
}else{
     System.out.println("半径不能为负值");
}
```

上述代码实现只有当半径(radius)值大于或等于 0 时才计算圆的面积；否则，当半径值小于 0 时，程序给出错误提示。

当 if 或 else 部分只有一条语句时，大括号可以省略，但推荐使用大括号。

3.1.3 条件运算符

条件运算符(conditional operator)的格式如下：

<条件表达式> ? <表达式 1> : <表达式 2>

因为有三个操作数，又称为**三元运算符**。<条件表达式>为关系或逻辑表达式，其计算结果为布尔值。如果该值为 true，则计算<表达式 1 >的值，并将其结果作为条件表达式的结果；如果该值为 false，则计算<表达式 2 >的值，并将其结果作为条件表达式的结果。

条件运算符可以实现 if-else 结构。例如，若 max、a、b 是 int 型变量，求 a 与 b 的最大值并将结果存储到 max 变量中，可以使用条件运算符实现，也可以使用 if-else 结构，如下所示。

```
max = ( a > b) ? a : b ;
```

等价于

```
if (a > b) {
     max = a;
}else {
     max = b;
}
```

从上面可以看到使用条件运算符会使代码简洁，但是不容易理解。现代的编程，程序的可读性变得越来越重要，因此推荐使用 if-else 结构，毕竟并没有多输入多少代码。

3.1.4 嵌套的 if 语句和多分支的 if…else 语句

if 或 if…else 结构中语句可以是任意合法的 Java 语句，甚至可以是其他的 if 或 if…else 结构。内层的 if 结构称为嵌套在外层的 if 结构中。内层的 if 结构还可以包含其他的 if 结构。嵌套的深度没有限制。例如，下面就是一个嵌套的 if 结构，其功能是求 a、b 和 c 中最大值并将其保存到 max 中。

```
if(a > b){
     if(a > c)
          max = a;
     else
          max = c;
}else{
  if(b > c)
          max = b;
     else
          max = c;
}
```

一个嵌套的if-else结构

一个嵌套的if-else结构

注意 把每个 else 同与它匹配的 if 对齐排列，这样很容易辨别嵌套层次。

如果程序逻辑需要多个选择，可以在 if 语句中使用一系列的 else if 语句，这种结构称为**多分支的 if…else 结构**或者称为**阶梯式 if…else 结构**。

编写程序，要求输入一个人的身高和体重，计算并打印出他的 BMI，同时显示 BMI 是高还

是低。对于一个成年人，BMI 值的含义如下：

- BMI＜18.5，表示偏瘦；
- 18.5≤BMI＜25.0，表示正常；
- 25.0≤BMI＜30.0，表示超重；
- BMI≥30.0，表示过胖。

程序 3.2　ComputeBMI.java

```
package com.boda.xy;
import java.util.Scanner;
public class ComputeBMI{
   public static void main(String[ ]args){
        Scanner input = new Scanner(System.in);
        double weight,height;
        double bmi;
        System.out.print("请输入你的体重(单位:公斤):");
        weight = input.nextDouble();
        System.out.print("请输入你的身高(单位:米):");
        height = input.nextDouble();
        bmi = weight / (height * height);
        System.out.println("你的身体质量指数是:" + bmi);
        if(bmi < 18.5){
            System.out.println("你的体重偏瘦.");
        }else if(bmi < 25.0) {
            System.out.println("你的体重正常.");
        }else if(bmi < 30.0) {
            System.out.println("你的体重超重.");
        }else {
            System.out.println("你的体重过胖.");
        }
   }
}
```

下面是程序的一次运行结果。

```
请输入你的体重(单位:公斤):70
请输入你的身高(单位:米):1.75
你的身体质量指数是:22.857142857142858
你的体重正常.
```

扫一扫

视频讲解

3.2　switch 语句与 switch 表达式

Java 语言从一开始就提供了 switch 语句实现多分支结构。从 Java 12 开始对 switch 语句进行了修改并支持 switch 表达式。尽管 Java 仍然支持旧的 switch 结构，但建议读者熟悉并使用新的 switch 语句和 switch 表达式。

3.2.1　switch 语句

如果需要从多个选项选择其中一个，可以使用 switch 语句。switch 语句主要实现多分支结构，一般格式如下：

```
switch (<表达式>){
   case 值 1 -> 语句(组)1;
   case 值 2 -> 语句(组)2;
   …
   case 值 n -> 语句(组)n;
```

```
    [default -> 语句(组)n+1;]
}
```

其中,<表达式>的计算结果与 case 子句的值进行比较。表达式的值必须是 byte、short、int、char、enum 类型或 String 类型。case 子句用来指定每一种情况,后面的值是常量或常量表达式,它的值必须与表达式值类型相容。switch 语句的执行流程如图 3-3 所示。

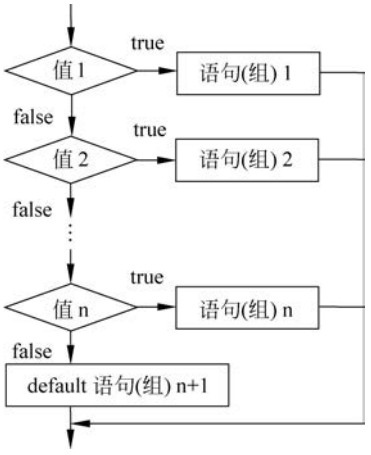

图 3-3 switch 语句的执行流程

当程序执行到 switch 语句,首先计算<表达式>的值,然后用该值依次与每个 case 中的常量值(或常量表达式)进行比较,如果等于某个值,则执行该 case 子句中后面的语句(组),之后结束 switch 结构。每个 case 后面使用箭头号(->)指定要执行的语句(组),这里可以是一条语句,也可以包含多条语句。如果包含多条语句,需使用花括号。

default 子句是可选的,当<表达式>的值与每个 case 子句中的值都不匹配时,就执行 default 后的语句。如果表达式的值与每个 case 子句中的值都不匹配,且又没有 default 子句,则程序不执行任何操作,而是直接跳出 switch 结构,执行后面的语句。

用 switch 结构实现多重选择。程序 3.3 要求从键盘输入一个季节数字(1,2,3,4),程序根据输入的数输出一句话。

程序 3.3 SwitchDemo.java

```
package com.boda.xy;
import java.util.Scanner;
public class SwitchDemo{
   public static void main(String[ ] args) {
       Scanner input = new Scanner(System.in);
       System.out.print("输入一个季节(1,2,3,4):");
       int season = input.nextInt();
       switch (season) {
           case 1 -> System.out.println("春雨惊春清谷天");
           case 2 -> System.out.println("夏满芒夏暑相连");
           case 3 -> System.out.println("秋处露秋寒霜降");
           case 4 -> System.out.println("冬雪雪冬小大寒");
           default ->
                System.out.println("季节数字输入非法.");
         }
    }
}
```

下面是程序的一次运行结果:

```
输入一个季节(1,2,3,4):4
冬雪雪冬小大寒
```

从 Java 7 开始,可以在 switch 语句的表达式中使用 String 对象。下面的程序根据英文季节字符串名称(Spring、Summer、Autumn 和 Winter)输出中文季节名。

程序 3.4 StringSwitchDemo.java

```
package com.boda.xy;
import java.util.Scanner;
public class StringSwitchDemo {
    public static void main(String[ ] args) {
```

```
        String season = "";
        Scanner input = new Scanner(System.in);
        System.out.print("请输入英文季节名称:");
        season = input.next();
        switch (season.toLowerCase()) {
            case "spring" -> System.out.println("春天");
            case "summer" -> System.out.println("夏天");
            case "autumn" -> System.out.println("秋天");
            case "winter" -> System.out.println("冬天");
            default       -> System.out.println("输入名称错误!");
        }
    }
}
```

下面是程序的一次运行结果。

```
请输入英文季节名称:Autumn
秋天
```

程序中 season.toLowerCase()用于将字符串转换成小写字符串。switch 表达式中的字符串与每个 case 中的字符串进行比较。

3.2.2 switch 表达式

在 switch 结构中，除了可以执行语句外，还可以使用 switch 表达式，即通过 switch 结构返回一个值，并将该值赋给变量。例如，下面代码根据 day 的值返回一个数值赋给变量 numLetters。

```
DayOfWeek day = DayOfWeek.SATURDAY;
int numLetters = switch(day){
    case MONDAY, FRIDAY, SUNDAY -> 6;
    case TUESDAY                -> 7;
    case THURSDAY, SATURDAY     -> 8;
    case WEDNESDAY              -> 9;
};
System.out.println(numLetters);                    // 输出:8
```

根据表示星期几的枚举常量返回单词的字母数

分号是赋值语句的结束

程序 3.5 从键盘输入一个年份（如 2000 年）和一个月份（如 2 月），用 switch 表达式返回该月的天数（29），将其存入一个变量。

程序 3.5　SwitchExprDemo.java

```
package com.boda.xy;
import java.util.Scanner;
public class SwitchExprDemo {
    public static void main(String[ ] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("输入一个年份:");
        int year = input.nextInt();
        System.out.print("输入一个月份:");
        int month = input.nextInt();
        int numDays = switch (month) {
            case 1, 3, 5, 7, 8, 10, 12 -> 31;
            case 4, 6, 9, 11 -> 30;
            // 对 2 月需要判断是否是闰年
            case 2 -> {
              if (((year % 4 == 0) && !(year % 100 == 0)) || (year % 400 == 0))
                yield 29;
              else
```

switch表达式

yield是受限关键字，返回一个值

```
                yield 28;
            }
            default -> 0;
            };          ← 分号是赋值语句的结束
            System.out.println("该月的天数为:" + numDays);
        }
    }
```

下面是程序的一次运行结果。

```
输入一个年份:2020
输入一个月份:2
该月的天数为:29
```

yield 语句用于返回一个值，并跳出 switch 块。yield 和 return 的区别在于：return 会直接跳出当前方法或循环，而 yield 只会跳出当前 switch 块。注意，在使用 yield 时，需要有 default 条件。

另外注意，如果某个 case 直接返回一个值，不能直接使用 yield 语句，但可以在大括号中使用，如下所示。

```
case 4, 6, 9, 11 -> { yield 30;}
```

switch 表达式还可以用在方法返回值中，如下所示。

```
private static String descLang(String name){
    return switch(name){
        case "Java" ->{yield "object-oriented, platform independent.";}
        case "Ruby" ->{yield "a programmer's best friend.";}
        default ->{yield name + " is a good language.";}
    }
}
```

提示 在 Java 语言规范中，yield 不属于关键字，它在 switch 结构中用于产生一个值，在其他地方仍然可以作为标识符使用。

Java 仍然支持传统的 switch 语法结构，传统 switch 结构如下：

```
switch (expression){
   case 值 1: 语句(组)1; [break;]      ← 不建议使用这种结构
   case 值 2: 语句(组)2; [break;]
   …
   case 值 n: 语句(组)n; [break;]
   [default: 语句(组)n+1;]
}
```

使用这种语法结构很容易忘记 break 语句而产生错误，也没有新的 switch 结构功能强大，建议使用新的 switch 结构。

编写程序，要求从一副纸牌中任意抽取一张，并打印出抽取的是哪一张牌。一副牌有 4 种花色，黑桃、红桃、方块和梅花。每种花色有 13 张牌，共有 52 张牌。可以将这 52 张牌编号，从 0 到 51。规定编号 0 到 12 为黑桃，13 到 25 为红桃，26 到 38 为方块，39 到 51 为梅花。

可以使用整数的除法运算来确定是哪一种花色，用求余数运算确定是哪一张牌。例如，假设抽出的数是 n，计算 n/13 的结果：若商为 0，则牌的花色为黑桃；若商为 1，则牌的花色为红桃；若商为 2，则牌的花色为方块；若商为 3，则牌的花色为梅花。计算 n%13 的结果，可得到第几张牌。

程序 3.6　PickCards.java

```
package com.boda.xy;
public class PickCards {
   public static void main(String[ ] args){
     int card = (int) (Math.random() * 52);
     String suit = "",rank = "";
     suit = switch(card / 13){                    // 确定牌的花色
       case 0 -> "♠";
       case 1 -> "♥";
       case 2 -> "♦";
       case 3 -> "♣";
       default -> "";          ← 在switch表达式中必须包含default语句
     };
     rank = switch(card % 13){                    // 确定是第几张牌
       case 0 -> "A";
       case 10 -> "J";
       case 11 -> "Q";
       case 12 -> "K";
       default -> "" + (card %13 +1);
     };
     System.out.println("你抽取的牌是:" + suit + rank);
   }
}
```

下面是程序的一次运行结果。

```
你抽取的牌是: ♥ 10
```

扫一扫

视频讲解

3.3　案例学习——两位数加减运算

1. 问题描述

开发一个让小学生练习两位整数加减法的程序，要求程序随机生成两个两位数及加减号（要保证减法算式的被减数大于减数），显示题目让学生输入计算结果，程序判断结果是否正确。

2. 设计思路

该案例的设计思路主要如下：

（1）要实现加减法运算，首先应该随机产生两个两位整数。随机生成整数有多种方法，可以使用 Math.random()方法生成一个随机浮点数，然后将它扩大再取整。random()方法返回 0.0～1.0（不包括）的浮点数，要得到 10～99 的整数，可以使用下面的表达式：

```
int number1 = 10 + (int)(Math.random() * 90);
```

（2）确定加或减运算。这也可以通过产生 2 个随机数（如 0 和 1，0 表示加法，1 表示减法）确定。

```
int operator = (int)(Math.random() * 2);
```

（3）假设学生没有学过负数概念，如果做减法运算，要保证第一个数大于第二个数。也就是如果 number1 小于 number2，应该交换这两个数。

```
if(number1 < number2){
    int temp = number2;
    number2 = number1;
    number1 = temp;
}
```

(4) 最后根据运算符决定做何种运算。将计算结果保存到 result 变量中,然后与用户输入的答案 answer 比较,判断用户答题是否正确。

3. 代码实现

两位数加减法运算代码如程序 3.7 所示。

程序 3.7 Calculator.java

```
package com.boda.xy;
import java.util.Scanner;
public class Calculator{
   public static void main(String[ ] args) {
       Scanner input = new Scanner(System.in);
       int number1 = 10 + (int)(Math.random() * 90);
       int number2 = 10 + (int)(Math.random() * 90);
       int operator = (int)(Math.random() * 2);
       int result, answer;
       if(operator == 0){          ← operator为0表示加法
         result = number1 + number2;
         System.out.print(number1 + " + " + number2 + " = ");
       }else{                      ← operator不为0表示减法
         if(number1 < number2){
           int temp = number2;
           number2 = number1;      ← 交换number1和number2
           number1 = temp;
         }
         result = number1 - number2;
         System.out.print(number1 + " - " + number2 + " = ");
       }
       answer = input.nextInt();   ← 接收用户输入答案
       if(answer == result){
         System.out.print("恭喜你,回答正确!");
       }else{
         System.out.print("对不起,回答错误!");
       }
    }
}
```

4. 运行结果

程序 3.7 的运行结果如图 3-4 所示,这里产生一个减法题目。

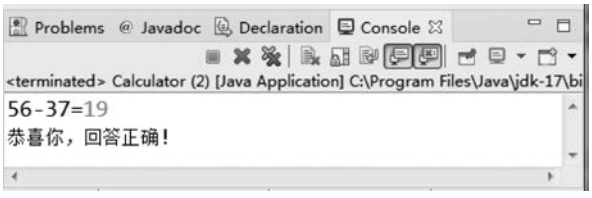

图 3-4 程序 3.7 的运行结果

要产生随机数除了可以使用 Math.random()方法外,Java 语言还提供了一个 java.util.Random 类,使用该类的实例也可以随机产生整数、浮点数或布尔型值。下面的代码随机产生一个两位整数。

```
Random rand = new Random();
int number = 10 + rand.nextInt(90);
```

Random 类除定义了 nextInt()方法,还定义了 nextBoolean()、nextDouble()、nextLong()等方法,分别产生随机布尔值、浮点数和长整数等。

扫一扫 视频讲解

3.4 循环结构

在程序设计中，有时需要反复执行一段相同的代码，这时就需要使用循环结构来实现。Java 语言提供了 4 种循环结构：while 循环、do…while 循环、for 循环和增强的 for 循环。

一般情况下，一个循环结构包含以下 4 部分内容。

（1）初始化部分：设置循环开始时的变量初值。

（2）循环条件：一般是一个布尔表达式。当表达式值为 true 时，执行循环体；当表达式值为 false 时，退出循环。

（3）迭代部分：改变变量的状态。

（4）循环体部分：需要重复执行的代码。

3.4.1 while 循环

while 循环是 Java 最基本的循环结构。这种循环是在某个条件为 true 时，重复执行一个语句或语句块。它的一般格式如下：

```
[初始化部分]
while(<条件>){
      // 循环体
      [迭代部分]
}
```

大括号内为循环体和迭代部分

其中，初始化部分用于初始化循环中需要的变量，<条件>为一个布尔表达式，它是循环条件。中间的部分为循环体，用一对花括号定界。迭代部分用于改变循环变量值。

该循环首先判断循环条件，当条件为 true 时，一直反复执行循环体。这种循环一般称为**“当型循环”**。一般用在循环次数不确定的情况下。while 循环的执行流程如图 3-5 所示。

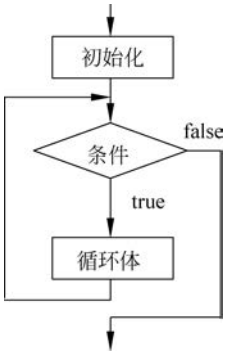

图 3-5 while 循环的执行流程

编写一个正确的循环对编程初学者来说并不是件容易的事，编写循环时应该考虑三个步骤：①确定需要重复的语句。②将这些语句放入一个循环体。③编写循环继续条件，并添加适合的语句控制循环。

下面一段代码使用 while 结构计算 1～100 之和。

```
int n = 1;                    ← 初始化部分
int sum = 0;
while(n <= 100){              ← 循环条件
    sum = sum + n;            ← 重复执行的语句
    n = n + 1;                ← 迭代部分
}
System.out.println("sum = " + sum);          // 输出 sum = 5050
```

程序 3.8 随机产生一个 100～200 的整数，用户从键盘上输入所猜的数，程序显示是否猜中的消息，如果没有猜中要求用户继续猜，直到猜中为止。

程序 3.8 GuessNumber.java

```
package com.boda.xy;
import java.util.Scanner;
public class GuessNumber{
   public static void main(String[ ] args){
```

```
        int magic = (int)(Math.random() * 101) + 100;          ← 随机生成的数
        Scanner sc = new Scanner(System.in);
        System.out.print("请输入你猜的数:");
        int guess = sc.nextInt();          ← 用户猜的数
        while(guess != magic){
            if(guess > magic)
              System.out.print("错误!太大,请重猜:");
            else
              System.out.print("错误!太小,请重猜:");

            guess = sc.nextInt();          ← 输入下一次猜的数
        }
        System.out.println("恭喜你,答对了!\n该数是:" + magic);
    }
}
```

本程序仍然使用 java.lang.Math.random()方法生成随机数。程序中该方法乘以 101 再转换为整数,得到 0～100 的整数,再加上 100,则 magic 的范围就为 100～200 的整数。

3.4.2 do…while 循环

do…while 循环的一般格式如下:

```
[初始化部分]
do{
   // 循环体
   [迭代部分]
}while(<条件>);          ← 注意,这里以分号(;)结束
```

do…while 循环的执行过程如图 3-6 所示。该循环首先执行循环体,然后计算条件表达式。如果表达式的值为 true,则返回到循环的开始继续执行循环体,直到<条件>的值为 false 时循环结束。这种循环一般称为“**直到型**”循环。该循环结构与 while 循环结构的不同之处是,do…while **循环至少执行一次循环体**。

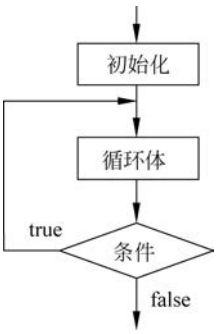

图 3-6 do…while 循环的执行过程

使用 do…while 循环也可以计算 1～100 之和,代码如下:

```
int n = 1;
int sum = 0;
do{
    sum = sum + n;
    n = n + 1;
}while(n <= 100);
System.out.println("sum = " + sum);          // 输出:sum = 5050
```

程序 3.9 要求用户从键盘输入若干 double 型数(输入 0 则结束),程序计算并输出这些数的总和与平均值。

程序 3.9 DoWhileDemo.java

```
package com.boda.xy;
import java.util.Scanner;
public class DoWhileDemo {
     public static void main(String[ ] args) {
       double sum = 0,avg = 0;
       int n = 0;
       double number;
       Scanner input = new Scanner(System.in);
       do{
```

```
            System.out.print("请输入一个数(输入 0 结束):");
            number = input.nextDouble();
            if(number != 0){
                sum = sum + number;
                n = n + 1;
            }
        }while(number!= 0);
        avg = sum / n;
        System.out.println("sum = " + sum);
        System.out.println("avg = " + avg);
    }
}
```

扫一扫

视频讲解

3.4.3 for 循环

for 循环是 Java 语言中使用最广泛的，也是功能最强的循环结构。它的一般格式如下：

```
for (初始化部分; 条件; 迭代部分){
    // 循环体
}
```

其中，初始化部分、循环条件和迭代部分用分号隔开。for 循环的执行流程如图 3-7 所示。

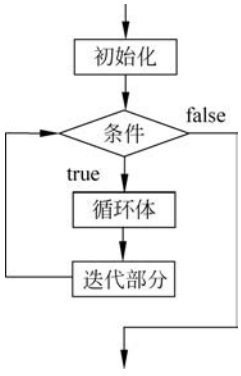

图 3-7 for 循环的执行流程

循环开始执行时首先执行初始化部分。该部分在整个循环中只执行一次。在这里可以定义循环变量并赋初值。接下来判断循环条件，若为 true，则执行循环体部分；否则，退出循环。当循环体执行结束后，程序控制返回到迭代部分，执行迭代，然后再次判断循环条件，若为 true，则反复执行循环体。

下面的代码使用 for 循环计算 1～100 之和。

```
int sum = 0;
for(int i = 1; i <= 100; i++){
   sum = sum + i;
}
System.out.println("sum = " + sum);            // 输出:sum = 5050
```

在初始化部分可以声明多个变量，中间用逗号分隔，它们的作用域在循环体内。在迭代部分也可以有多个表达式，中间也用逗号分隔。下面循环中声明了两个变量 i 和 j。

```
for(int i = 0, j = 10; i < j; i++, j--) {
    System.out.println("i = "+ i + ",j = " + j);
}
```

for 循环中的一部分或全部可为空，循环体也可为空，但分号不能省略。例如：

```
for ( ; ; ){
    // 这实际是一个无限循环,循环体中应包含结束循环代码
}
```

对于无限循环，在循环体中应该包含结束循环的代码，否则可能产生**死循环**(dead loop)。可以使用 break 语句或带标签的 break 语句结束循环。同样，对 while 和 do…while 循环，如果循环条件永远为 true，也会产生无限循环。

for 循环和 while 循环及 do…while 循环有时可相互转换。例如 for 循环

```
for(int i = 0, j = 10; i < j; i++, j--){
    System.out.println("i = "+ i + " ,j = " + j);
}
```

可以转换为等价的 while 循环结构

```
int i = 0, j = 10;
while(i < j){
    System.out.println("i = " + i + " ,j = " + j);
    i ++;
    j --;
}
```

提示 在 Java 5 中增加了一种新的循环结构，称为增强的 for 循环，它用于对数组和集合元素迭代。关于增强的 for 循环在 5.1.4 节中讨论。

3.4.4 循环的嵌套

在一个循环的循环体中可以嵌套另一个完整的循环，称为**循环的嵌套**。内嵌的循环还可以嵌套循环，这就是**多层循环**。同样，在循环体中也可以嵌套另一个选择结构。

编写程序，用嵌套的 for 循环打印输出 8 行由星号构成的三角形图形。其中，第一行输出一个星号，第二行输出 3 个星号，……，第 8 行输出 15 个星号。

程序 3.10 PrintStars.java

```
package com.boda.xy;
public class PrintStars {
    public static void main(String[ ] args) {
        for (int n = 1; n <= 8; n++) {          ← n 记录行数
            for (int k = 1; k <= (8 - n); k++) {
                System.out.print(" ");          ← 打印每行前导空格
            }
            for(int j = 1; j <= (2 * n - 1); j++) {
                System.out.print(" * ");        ← 每行打印 2*n-1 个星号
            }
            System.out.println();               // 换行
        }
    }
}
```

程序运行的输出结果如下所示。

```
        *
       ***
      *****
     *******
    *********
   ***********
  *************
 ***************
```

这个程序在一个 for 循环内嵌套了另外两个 for 循环。第一个 for 循环用于打印若干空格，每行打印的空格数为 8－n 个。第二个 for 循环用于打印若干星号，每行打印 2×n－1 个星号。

3.4.5 break 语句和 continue 语句

在 Java 循环体中可以使用 break 语句和 continue 语句。

1. break 语句

break 语句是用来跳出 while、do、for 或 switch 结构的执行。该语句有两种格式：

```
break;
break 标签名;
```

break 语句的功能是结束本次循环，控制转到其所在循环的后面执行。对各种循环均直接退出，不再计算循环控制表达式。程序 3.11 演示了 break 语句的使用。

程序 3.11　BreakDemo.java

```
package com.boda.xy;
public class BreakDemo{
   public static void main(String[ ] args){
      int n = 1;
      int sum = 0;
      while(n <= 100){
         sum = sum + n;
         if(sum > 100){
             break;                              // 若条件成立,退出循环
         }
         n = n + 2;
      }
      System.out.println("n = " + n);
      System.out.println("sum = " + sum);
   }
}
```

程序输出结果如下：

```
n = 21
sum = 121
```

使用 break 语句只能跳出当前的循环体。如果程序使用了多重循环，又需要从内层循环跳出或者从某个循环开始重新执行，此时可以使用带标签的 break。

观察下面的代码：

```
start:
for(int i = 0; i < 3; i++){
   for(int j = 0; j < 4; j++){
       if(j == 2){
         break start;                            // 跳出 start 标签标识的循环
       }
       System.out.println(i +":" + j);
    }
}
```

这里，标签 start 用来标识外层的 for 循环，因此语句 break start；跳出了外层循环。上述代码的运行结果如下：

```
0 : 0
0 : 1
```

2. continue 语句

continue 语句与 break 语句类似，但它只终止执行当前的迭代，导致控制权从下一次迭代开始。该语句有下面两种格式：

```
continue;
continue 标签名;
```

下面代码演示了 continue 语句的使用。代码会输出 0～9 除了 5 以外的数字。

```
for(int i = 0; i < 10; i++){
    if(i == 5){
        continue;
    }
    System.out.print(i + " ");                    // 输出:0 1 2 3 4 6 7 8 9
}
```

当 i 等于 5 时,if 语句的表达式运算结果为 true,使得 continue 语句得以执行。因此,后面的输出语句不能执行,控制权从下一次循环处继续,即 i 等于 6 的时候。

continue 语句也可以带标签,用来标识从哪一层循环继续执行。下面是使用带标签的 continue 语句的例子。

```
start:
for(int i = 0; i < 3; i++){
    for(int j = 0; j < 4; j++){
        if(j == 2){
            continue start;                       // 返回到 start 标签标识的循环的条件处
        }
        System.out.println(i +" : " + j);
     }
}
```

这段代码的运行结果如下:

```
0 : 0
0 : 1
1 : 0
1 : 1
2 : 0
2 : 1
```

注意

(1) 带标签的 break 可用于循环结构和带标签的语句块,而带标签的 continue 只能用于循环结构。

(2) 标签命名遵循标识符的命名规则,相互包含的块不能用相同的标签名。

(3) 带标签的 break 和 continue 语句不能跳转到不相关的标签块。

提示 有些编程语言可以使用 goto 语句从内层循环跳到外层循环,在 Java 语言中尽管将 goto 作为关键字,但不能使用,也没有意义。

扫一扫

视频讲解

3.5 案例学习——求最大公约数

1. 问题描述

两个整数的最大公约数(Greatest Common Divisor,GCD)是能够同时被两个数整除的最大整数。例如,4 和 2 的最大公约数是 2,16 和 24 的最大公约数是 8。

编写程序,要求从键盘输入两个整数,计算并输出这两个数的最大公约数。

2. 设计思路

求两个整数的最大公约数有多种方法。一种方法是,假设求两个整数 m 和 n 的最大公约数,显然 1 是一个公约数,但它可能不是最大的。可以依次检查 k(k=2,3,4,…)是否 m 和 n 的最大公约数,直到 k 大于 m 或 n 为止。

本案例的设计思路主要如下:

（1）从键盘输入两个整数，分别存入变量 m 和变量 n。

（2）用下面的循环结构计算能同时被 m 和 n 整除的数，循环结束后得到的 gcd 就是最大公约数。

```
while(k <= m && k <= n){
    if(m % k == 0 && n % k == 0)
        gcd = k;
    k++;
}
```

3. 代码实现

求最大公约数的代码如程序 3.12 所示。

程序 3.12　GCDDemo.java

```
package com.boda.xy;
import java.util.Scanner;
public class GCDDemo{
  public static void main(String[ ] args){
      Scanner input = new Scanner(System.in);
      System.out.print("请输入第 1 个整数:");
      int m = input.nextInt();
      System.out.print("请输入第 2 个整数:");
      int n = input.nextInt();
      // 求 m 和 n 的最大公约数
      int gcd = 1;
      int k = 2;
      while(k <= m && k <= n){
        if(m % k == 0 && n % k == 0)          ← 判断k是否能同时被m和n整除
          gcd = k;
        k++;
      }
      System.out.println(m +" 与 "+ n +" 的最大公约数是"+ gcd);
    }
}
```

4. 运行结果

案例运行结果如图 3-8 所示，这里输入第 1 个整数 16，第 2 个整数 24，它们的最大公约数为 8。

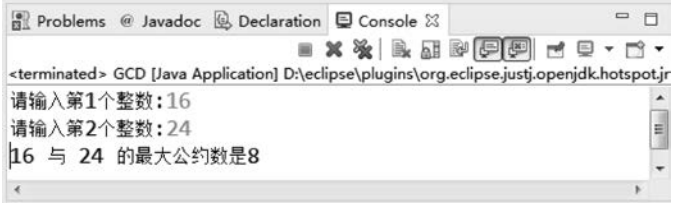

图 3-8　程序 3.12 的运行结果

计算两个整数 m 与 n 的最大公约数还有一个更有效的方法，称为**辗转相除法**或称**欧几里得算法**，其基本步骤如下：计算 r＝m％n，若 r＝＝0，则 n 是最大公约数。若 r!＝0，则执行 m＝n，n＝r，再次计算 r＝m％n，直到 r＝＝0 为止，最后一个 n 即为最大公约数。请读者自行编写程序实现上述算法。

扫一扫
视频讲解

3.6　案例学习——打印输出若干素数

1. 问题描述

素数(prime number)又称**质数**，有无限个，在计算机密码学中有重要应用。素数定义为

在大于 1 的自然数中，除了 1 和它本身以外不再有其他因数的数。编写程序计算并输出前 50 个素数，每行输出 10 个。

2. 设计思路

该案例的设计思路主要如下：

(1) 要输出 50 个素数，首先用 while 循环对素数计数(用 count 变量)。

(2) 要判断的数 number 从 2 开始，这里用一个 for 循环。根据定义，使用从 2 开始的数(divisor)去除要判断的数，如果直到 divisor=number−1 仍不能整除，则 number 就是一个素数。

(3) 打印输出素数。为了输出美观，可以使用格式化输出方法，这里使用了 System 类的 printf()方法，它可以对输出数据进行格式化。

```
System.out.printf("%5d%n",number);
```

该语句用宽度为 5 个字符位置(%5d)输出 number，输出数字右对齐，输出后换行(%n)。

3. 代码实现

计算并输出前 50 个素数的代码如程序 3.13 所示。

程序 3.13　PrimeNumber.java

```
package com.boda.xy;
public class PrimeNumber{
    public static void main(String[ ] args){
      int count = 0;                              // 记录素数个数
      int number = 2;
      boolean isPrime;
      System.out.printf("前 50 个素数如下：%n");
      while(count < 50){
        isPrime = true;
        for(int divisor = 2; divisor <= number; divisor ++){
          if(number % divisor == 0){
              isPrime = false;
              break;
            }
          }
          if(isPrime){
              count ++;
              if(count % 10 == 0)
                 System.out.printf("%5d%n",number);
              else
                 System.out.printf("%5d",number);
          }
          number ++;
      }
    }
}
```

判断 number 是否素数，若循环结束 number 仍不能被 divisor 整除，则 number 是一个素数

如果 number 是素数，计数器加 1，并打印素数

格式化输出

判断下一个数

4. 运行结果

程序 3.13 的运行结果如图 3-9 所示。

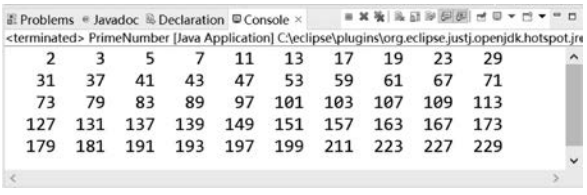

图 3-9　程序 3.13 的运行结果

说明：这里的循环条件 divisor <= number 表示一直检测到 divisor = number。实际上，根据数论的理论，判断一个数是否素数，其因子只需判断到 number 的平方根即可。因此这里的条件也可以写成如下形式：

```
divisor <= Math.sqrt(number)
```

或者

```
divisor * divisor <= number
```

3.7 小结

本章介绍了 Java 语言的结构化编程方法，主要知识点包括 Java 的选择结构，单分支、双分支和多分支结构；switch 语句和 switch 表达式；Java 的循环结构，while 循环、do-while 循环和 for 循环以及循环的嵌套；各种结构的综合应用等。

下一章将讨论 Java 语言的类、对象和方法设计，内容包括面向对象基本概念，类的定义，对象的创建，方法设计，构造方法，静态成员，对象初始化和销毁等重要内容。

编程练习

1. 编写程序，要求用户从键盘上输入一个正整数，程序判断该数是奇数还是偶数。

2. 编写程序，要求用户从键盘上输入一个年份，输出该年是否闰年。符合下面两个条件之一的年份即为闰年：①能被 4 整除，但不能被 100 整除；②能被 400 整除。下面是程序的一次运行。

```
请输入年份:2017
2017 年不是闰年。
```

3. 编写程序，要求用户从键盘输入 4 个整数，找出其中最大值和最小值并打印输出。要求使用尽可能少的 if(或 if-else)语句实现。提示：4 条 if 语句就够了。

4. 可以使用下面的公式求一元二次方程 $ax^2+bx+c=0$ 的两个根：

$$x_1=\frac{-b+\sqrt{b^2-4ac}}{2a} \quad 和 \quad x_2=\frac{-b-\sqrt{b^2-4ac}}{2a}$$

b^2-4ac 称为一元二次方程的判别式，如果它是正值，那么方程有两个实数根；如果它为 0，方程就只有一个根；如果它是负值，方程无实根。

编写程序，提示用户输入 a、b 和 c 的值，程序根据判别式显示方程的根。如果判别式为负值，显示“方程无实根”。提示：使用 Math.sqrt(n)方法计算数 n 的平方根。

5. 编写程序，要求从键盘输入一个月份数字(值为 1—12)，程序输出月所在的季节，2—4 月为春季，5—7 月为夏季，8—10 月为秋季，11、12、1 月为冬季，要求使用多分支的 if-else 结构实现。

6. 从键盘输入一个百分制的成绩，输出五级制的成绩，如输入 85，输出“良好”，要求使用 switch 结构实现。

7. 编写程序，接收用户从键盘输入 10 个整数，比较并输出其中的最大值和最小值。

8. 编写程序，显示 100～1000 所有能被 5 和 6 整除的数，每行显示 10 个。数字之间用一个空格字符隔开。

9. 编写程序，分别使用 while 循环、do-while 循环和 for 循环结构，计算并输出 1～1000 含有 7 或者 7 的倍数的整数之和及个数。下面是部分输出结果。

```
……
994
997
总个数 = 374
总和 = 206191
```

10. 编写程序，要求用户从键盘输入一个年份，程序输出该年出生的人的生肖。中国生肖基于 12 年一个周期，每年用一个动物代表：鼠(rat)、牛(ox)、虎(tiger)、兔(rabbit)、龙(dragon)、蛇(snake)、马(horse)、羊(sheep)、猴(monkey)、鸡(rooster)、狗(dog)和猪(pig)。通过 year%12 确定生肖，1900 年属鼠。

11. 编写程序，模拟“石头、剪刀、布”游戏。程序随机产生一个数，这个数为 2、1 或 0，分别表示石头、剪刀和布。提示用户输入值 2、1 或 0，然后显示一条消息，表明用户和计算机谁赢了游戏。下面是运行示例：

```
你出什么?(石头(2)、剪刀(1)、布(0)):2
计算机出的是:剪刀,你出石头,你赢了。
```

12. 编写程序，从键盘输入一个整数，计算并输出该数的各位数字之和。例如：

```
请输入一个整数:8899123
各位数字之和为:40
```

13. 编写程序，提示用户输入一个十进制整数，然后显示对应的二进制值。在这个程序中不要使用 Integer.toBinaryString(int)方法。

14. 编写程序，计算下面的级数之和：

$$\frac{1}{3}+\frac{3}{5}+\frac{5}{7}+\frac{7}{9}+\frac{9}{11}+\frac{11}{13}+\cdots+\frac{95}{97}+\frac{97}{99}$$

15. 求解“鸡兔同笼问题”：鸡和兔在一个笼里，共有腿 100 条，头 40 个。问：鸡兔各有几只？

16. 编写程序，求出所有的水仙花数。水仙花数是这样的三位数，它的各位数字的立方和等于这个三位数本身，例如 $371=3^3+7^3+1^3$，371 就是一个水仙花数。

17. 从键盘输入两个整数，计算这两个数的最小公倍数和最大公约数并输出。

18. 编写程序，求出 1～1000 所有的完全数。完全数是其所有因子(包括 1 但不包括该数本身)的和等于该数。例如 $28=1+2+4+7+14$，28 就是一个完全数。

19. 编写程序读入一个整数，显示该整数的所有素数因子。例如，输入整数为 120，输出应为 2、2、2、3、5。

20. 编写程序，计算当 n=10000，20000，…，100000 时 π 的值。求 π 的近似值公式如下。

$$\pi=4\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\frac{1}{13}+\cdots+\frac{1}{2n-1}-\frac{1}{2n+1}\right)$$