第3章 结构化编程

本章主要学习 Java 语言的流程控制结构，包括选择结构(if-else 结构、switch 结构)、循环结构(while 循环、do-while 循环和 for 循环)及其应用。

本章主要内容：

- 编程方法
- 选择
- 循环
- 循环嵌套
- 跳转语句(break 和 continue)

3.1 编程方法


视频讲解

随着计算机硬件技术的不断发展，编程方法也在不断地发展和改进。有多种编程方法，如结构化编程方法、面向对象编程方法、函数式编程、反应式编程。本节简要介绍结构化编程基本概念。

结构化编程(structured programming)方法是在 1965 年提出的，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(包括多选择结构)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。顺序结构表示程序中的各操作是按照它们出现的先后顺序执行的。如图 3-1 所示，其中，A、B 可以是一个语句(甚至是空语句)，也可以是图 3-1～图 3-3 所示的 7 种结构中的一种结构。

图 3-1 顺序结构

- 选择结构。程序的处理步骤出现了分支，它需要根据某一特定的条件选择其中的一个分支执行。简单的选择结构如图 3-2(a)和图 3-2(b)所示，这里 P 表示谓词条件。它们通常称为单选择、双选择。图 3-2(c)一般称为多选择，它可以通过嵌套的图 3-2(b)得到。
- 循环结构。程序反复执行某个或某些操作，直到某条件为假(或为真)时才可终止循环。在循环结构中最主要的是：什么情况下执行循环？哪些操作需要循环执行？循

环结构有三种，如图 3-3 所示。这三种结构依次为 while 循环、repeat 循环和 N+1/2 循环。前两种循环可以看作第三种循环的特殊情形。这三种循环虽然形式不同，但它们有一个共同点，即在一定条件下，将控制转移到本结构的入口点。

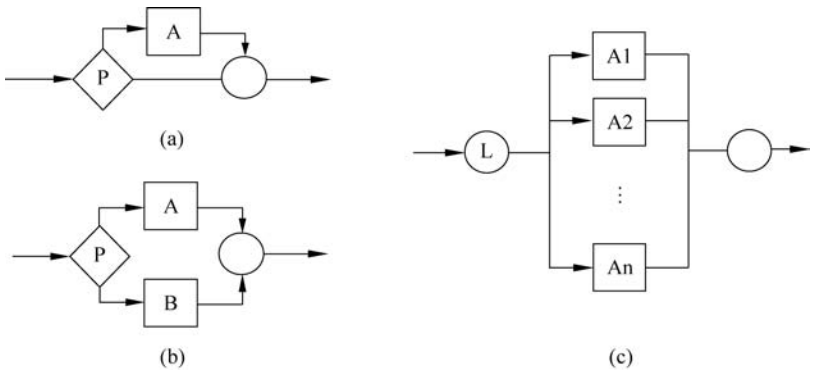

图 3-2　选择结构

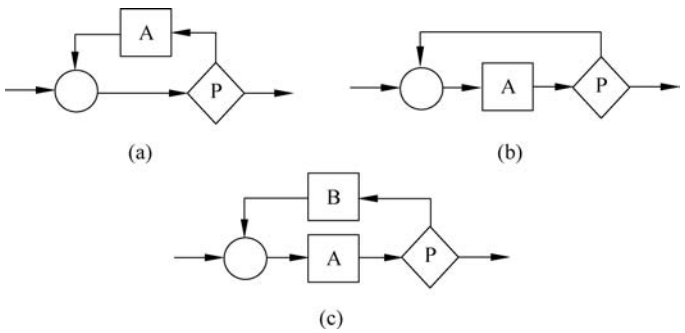

图 3-3　循环结构

按照结构化程序设计的观点，任何算法功能都可以通过由程序模块组成的三种基本程序结构的组合来实现。

采用结构化编程方法的软件开发需要遵循的主要原则如下。

- 自顶向下。程序设计时，应先考虑总体，后考虑细节；先考虑全局目标，后考虑局部目标。不要一开始就过多追求众多的细节，先从最上层总目标开始设计，逐步使问题具体化。
- 逐步求精。对复杂问题，通常应设计一些子目标作为过渡，然后逐步求精。
- 模块化设计。一个复杂问题，是由若干简单的问题构成。模块化是把程序要解决的总目标分解为子目标，再进一步分解为具体的小目标，把每一个小目标称为一个模块。
- 限制 goto 语句使用。很多编程语言支持 goto 语句，但使用 goto 语句是有害的，是造成程序混乱的根源，程序的质量与 goto 语句的数量成反比。

前面介绍了结构化编程的三种基本结构，其中顺序结构比较简单，程序按语句的顺序依次执行。前面章节编写的程序都是顺序结构的，本章重点讨论选择结构和循环结构。

3.2　选择

视频讲解

Java是面向对象编程语言，它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。关于Java面向对象的概念，将从第4章开始讨论。

Java有多种类型的选择语句：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语句和条件表达式。

3.2.1　if语句

if语句的格式如下。

```
if (条件){
    语句(组);
}
```

其中，条件为一个布尔表达式，它的值为true或false。布尔表达式应该用括号括住，后面是一对大括号。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)”语句序列，否则转去执行if结构后面的语句，如图3-4所示。

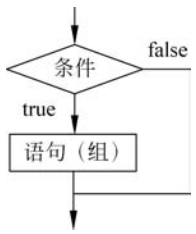

图3-4　if语句执行流程

【案例3-1】 if语句的使用。编写程序，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;        ←— 交换a与b的值
      a = t;
    }
    System.out.println(" a = " + a) ;
    System.out.println(" b = " + b) ;
  }
}
```

执行程序输入a为30，b为20，运行结果如图3-5所示。

【注意】 在if语句中，如果大括号内只有一条语句，则可以省略大括号。省略大括号可

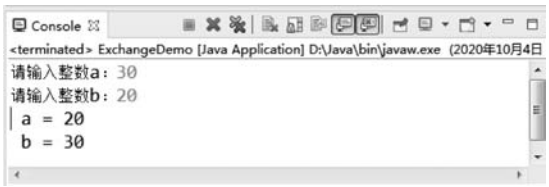

图 3-5　案例运行结果

以使代码更简洁，但也容易产生错误。将来需要为代码块增加语句时，容易忘记加上括号。这是初学者常犯的错误。

3.2.2　if-else 语句

if-else 语句是最常用的选择结构，它根据条件是 true 还是 false 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：首先计算条件的值，如果为 true，则执行“语句(组)1”，否则执行“语句(组)2”，如图 3-6 所示。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。

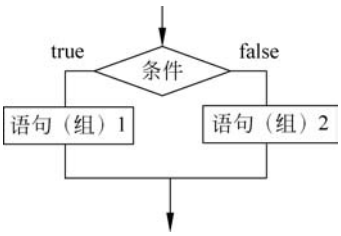

图 3-6　if-else 语句

【案例 3-2】 if-else 语句的使用。该案例要求用户从键盘输入一个年份，输出该年是否是闰年。符合下面两个条件之一的年份即为闰年：①能被 400 整除；②能被 4 整除，但不能被 100 整除。

【程序 3-2】 LeapYear.java

```
package com.boda.xy;
import java.util.Scanner;
public class LeapYear{
   public static void main(String[] args){
      Scanner scan = new Scanner(System.in);
      System.out.print("请输入年份: ");
      int year = scan.nextInt() ;
      if(year % 400 == 0||(year % 4 == 0 && year % 100 != 0)){
         System.out.println(year + " 年是闰年.") ;
      }else{
         System.out.println(year + " 年不是闰年.") ;
      }
   }
}
```

运行程序输入年份 2020，结果如图 3-7 所示。

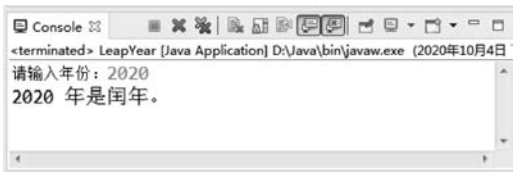

图 3-7　案例运行结果

3.2.3　多分支 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，这样做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，这种结构有时称为多分支 if-else 语句或阶梯式 if-else 结构。

【案例 3-3】　多分支 if-else 语句的使用。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 3-3】　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：kg)：");
    weight = input.nextDouble();
```

```
        System.out.print("请输入你的身高(单位: m): ");
        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.");
        }
    }
}
```

程序的一次运行结果如图 3-8 所示。

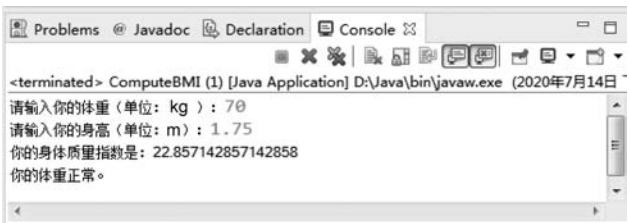

图 3-8　案例运行结果

3.2.4　条件运算符

条件运算符(conditional operator)的格式如下。

```
条件 ? 表达式 1 : 表达式 2
```

因为有三个操作数,又称为三元运算符。这里条件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,并将计算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,并将计算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 和 b 是 int 型变量,要计算 a 与 b 的最大值 max,下面左侧条件运算符结构与右侧使用 if-else 结构等价。

```
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。

3.3　案例研究：两位整数加减运算

视频讲解

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，要求程序运行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. 运行结果

案例运行结果如图 3-9 所示，这里产生一个减法题目。

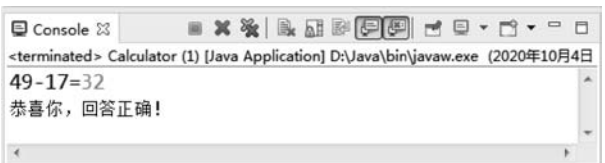

图 3-9　案例运行结果

3. 任务目标

（1）学会分析“两位整数加减运算”案例的实现思路。

（2）根据设计思路独立完成“两位整数加减运算”案例代码的编写、编译和运行。

（3）掌握如何使用 Math.random()方法生成随机数，并掌握如何将随机数转换成整数。

（4）掌握 if-else 结构的使用。

4. 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。Math.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。这也可以通过产生两个随机数（例如，0 和 1，0 表示加法，1 表示减法）确定。

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

（3）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＜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。

5. 代码实现

两位数加减法运算代码如程序 3-4 所示。

【程序 3-4】 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!");
      }
   }
}
```

✍多学一招

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。

视频讲解

3.4　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。

3.4.1　switch 语句

如果需要从多个选项中选择其中一个，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;]
}
```

switch 语句表达式的值必须是 byte、short、int、char、enum 类型或 String 类型。case 子句用来设定每一种情况，后面的值必须与表达式值类型相容。switch 语句的执行流程如图 3-10 所示。

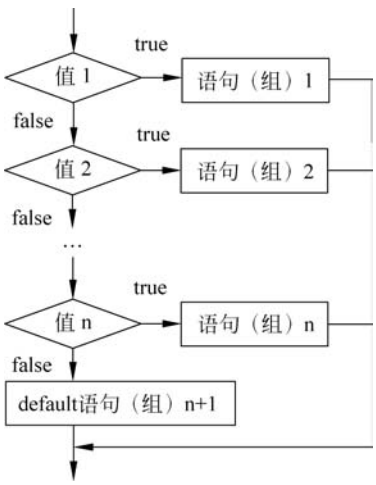

图 3-10　switch 语句的执行流程

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（或常量表达式）的值进行比较，如果等于某个值，则执行该 case 子句中后面的语句，之后结束 switch 结构。每个 case 后面使用箭头符号(->)指定要执行的语句，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，需使用大括号。

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。

【案例 3-4】　用 switch 结构实现多重选择。该案例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 3-5】　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("夏满忙夏暑相连");
           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("季节输入非法.");
        }
    }
}
```

图 3-11 是程序的一次运行结果。

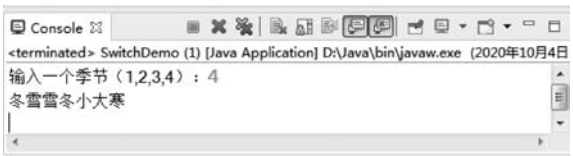

图 3-11 案例运行结果

【案例 3-5】 switch 使用字符串表达式。从 Java 7 开始，可以在 switch 语句的表达式中使用 String 对象，本案例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 3-6】 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!");
        }
    }
}
```

运行结果如图 3-12 所示。

程序中 season.toLowerCase()是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。

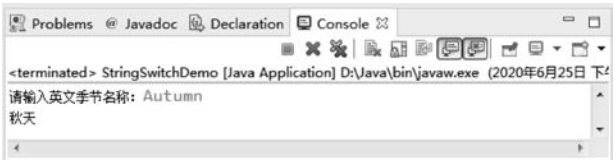

图 3-12　案例运行结果

3.4.2　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，下面的代码根据 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;      ←┐ 根据表示星期的枚举常量返
    case THURSDAY, SATURDAY      -> 8;       │ 回单词的字母数
    case WEDNESDAY               -> 9;      ←┘
  };                                        ←┐ 分号是赋值语句的结束
System.out.println(numLetters);             //输出 8
```

【案例 3-6】　switch 表达式应用。下面的程序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 3-7】　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) {   ←┐ switch 表达式
       case 1, 3, 5, 7, 8, 10, 12 -> 31;
       case 4, 6, 9, 11 -> 30;
       //对 2 月需要判断是不是闰年
       case 2 -> {
          if (((year % 4 == 0) && !(year % 100 == 0)) || (year % 400 == 0))
             yield 29;                  ←┐ yield 是受限标识符，生成一个值
          else
             yield 28;
       }
       default -> 0;
     };                                 ←┐ 分号是赋值语句的结束
     System.out.println("该月的天数为: " + numDays);
  }
}
```

图 3-13 是程序的一次运行结果。

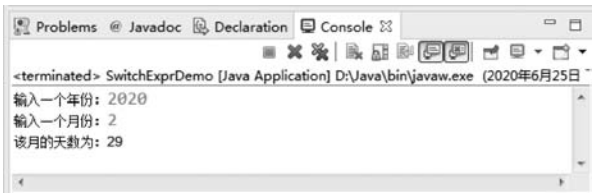

图 3-13 案例运行结果

yield 关键字用于跳出 switch 块,主要用于返回一个值。yield 和 return 的区别在于:return 会直接跳出当前循环或方法,而 yield 只会跳出当前 switch 块,同时在使用 yield 时,需要有 default 条件。

另外注意,如果某个 case 直接返回一个值,不能使用 yield 关键字,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,如以下方法所示。

```
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.";}
   }
}
```

Java 仍然支持传统的 switch 语法结构,传统的 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,建议使用新的 switch 结构。

【案例 3-7】 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 张牌编号为 0～51。规定编号 0～12 为黑桃,13～25 为红桃,26～38 为方块,39～51 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 3-8】　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);
  }
}
```

运行结果如图 3-14 所示。

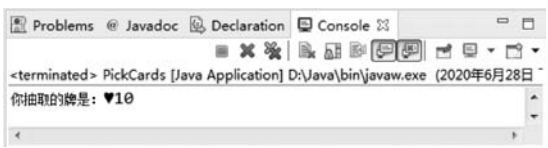

图 3-14　案例运行结果

3.5　循环


视频讲解

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四部分内容。

(1) 初始化部分: 设置循环开始时变量初值。

(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,为 false 时退出循环。

(3) 迭代部分: 改变变量的状态。

(4) 循环体: 需要重复执行的代码。

3.5.1　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

其中,初始化部分通常定义变量并赋初值,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,用一对大括号定界。迭代部分也是可选的。

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 3-15 所示。

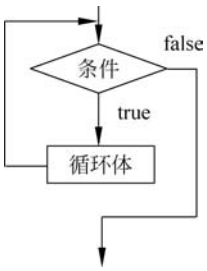

图 3-15 while 循环执行流程

【案例 3-8】 使用 while 循环计算 1～100 之和。

【程序 3-9】 WhileDemo.java

```
package com.boda.xy;
public class WhileDemo{
   public static void main(String[] args){
      int n = 1;
      int sum = 0;                    ← 初始化部分
      while(n <= 100){
         sum = sum + n;
         n = n + 1;                   ← 迭代语句
      }
      System.out.println("sum = " + sum);       //输出 sum = 5050
   }
}
```

运行结果如图 3-16 所示。

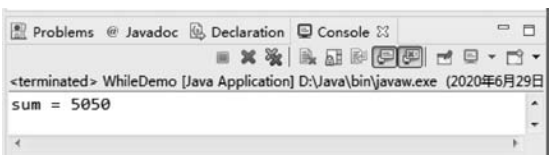

图 3-16 案例运行结果

【案例 3-9】 本案例要求程序运行随机产生一个 1～100 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 3-10】 GuessNumber.java

```
package com.boda.xy;
import java.util.Scanner;
public class GuessNumber{
  public static void main(String[] args){
    int magic = (int)(Math.random() * 100) + 1;   ← 程序产生的数
```

```
    Scanner sc = new Scanner(System.in);
    System.out.print("我想出一个 1～100 的数,请你猜: ");
    int guess = sc.nextInt();          ←┤ 用户猜的数
    while(guess != magic){
      if(guess > magic)
        System.out.print("对不起!太大了,请重猜: ");
      else
        System.out.print("对不起!太小了,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);
  }
}
```

运行结果如图 3-17 所示。

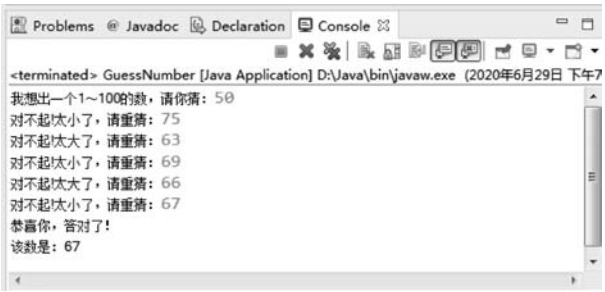

图 3-17　案例运行结果

程序中使用了 java.lang.Math 类的 random()方法,该方法返回一个 0.0～1.0(不包括 1.0)的 double 型的随机数。程序中该方法乘以 100 再转换为整数,得到 0～99 的整数,再加上 1,则 magic 的范围就为 1～100 的整数。

✍多学一招

在上面猜数游戏中,最多猜多少次应该猜中随机生成的数呢?实际上,这个猜数游戏是一个查找过程,并且是二分查找。它是在有序列表中查找指定的数(67),由于每次查找都去掉一半的元素,所以查找的次数最多是 $\log_2 N$,这里 N 是 100,$\log_2 N$ 的结果是 6.6439,所以最多不超过 7 次就应该猜中。

3.5.2　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]          ←┤ 大括号内为循环体
}while(条件);
```

do-while 循环执行过程如图 3-18 所示。

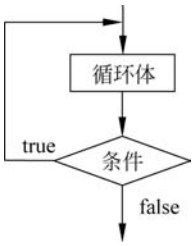

图 3-18　do-while 循环结构

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 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。

【案例 3-10】 用 do-while 循环计算 1～100 之和。

【程序 3-11】 Sum100.java

```
package com.boda.xy;
public class Sum100 {
  public static void main(String[] args) {
     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
  }
}
```

运行结果如图 3-19 所示。

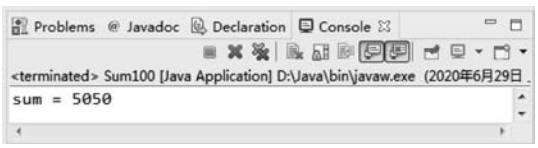

图 3-19　案例运行结果


视频讲解

3.5.3　for 循环

for 循环是 Java 程序中使用最广泛的,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; 循环条件; 迭代部分){
    //循环体
}
```

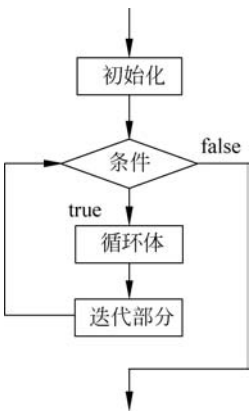

图 3-20　for 循环执行流程

这里,初始化部分、循环条件和迭代部分用分号隔开,大括号内为循环体。for 循环的执行流程如图 3-20 所示。

循环开始时首先执行初始化部分,该部分在整个循环中只执行一次。在这里通常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,为 false 则退出循环。循环体执行结束后,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 n = 1; n <= 100; n++){
   sum = sum + n;
}
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，如下所示。

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

for 循环和 while 循环及 do-while 循环有时可相互转换，例如，有下面的 for 循环：

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

可以转换为下面等价的 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，它主要用于对数组和集合元素迭代。关于增强的 for 循环在 5.1.4 节讨论。

3.5.4　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，选择结构中也可以嵌套循环。

【案例 3-11】 用嵌套的 for 循环打印输出如图 3-21 所示图形。这里，第 1 行输出 1 个星号，第 2 行输出 3 个星号，…，第 8 行输出 15 个星号。

【程序 3-12】 PrintStars.java

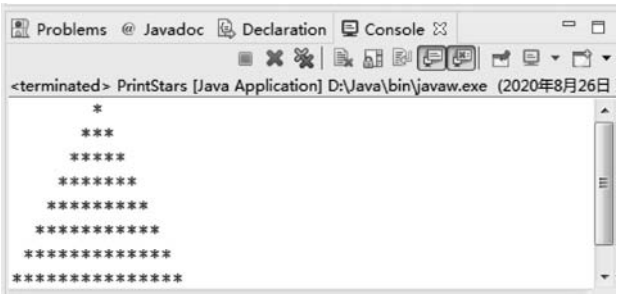

图 3-21　输出若干星号

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

这个程序在一个 for 循环内嵌套了两个 for 循环，第一个 for 循环用于打印若干个空格，每行打印的空格数为 8－n 个。第二个 for 循环用于打印若干个星号，每行打印 2n－1 个星号。

3.5.5　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 语句是用来结束 while、do、for 结构的执行，该语句有以下两种格式。

```
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。

【案例 3-12】　break 语句的使用。

【程序 3-13】　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);
  }
}
```

运行结果如图 3-22 所示。

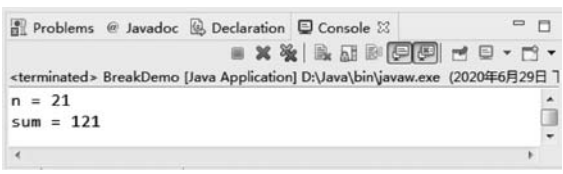

图 3-22　案例运行结果

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。

考虑下面的代码：

```
start:                          ←┤ 定义一个标签
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图 3-23 所示。

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。

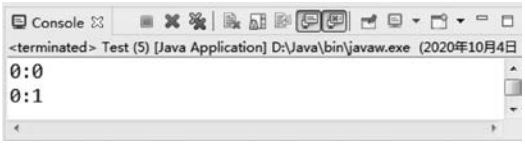

图 3-23　代码运行结果

```
continue;
continue  标签名;
```

以下代码会输出 0～9 的数字,但不会输出 5。

```
for(int i = 0; i < 10; i++){
  if(i == 5){
     continue;     ←┤ 控制转到迭代部分,即 i + + 处
  }
  System.out.println(i);
}
```

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:                    ←┤ 定义一个标签
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,即 i++处
     }
     System.out.println(i + " : " + j);
  }
}
```

代码运行结果如图 3-24 所示。

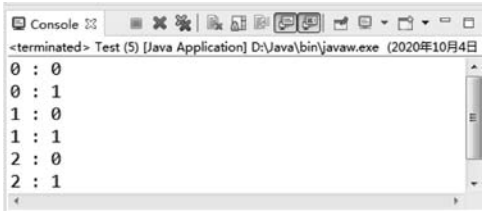

图 3-24　代码运行结果

【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。

3.6　案例研究：求最大公约数

视频讲解

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,程序计算并输出这两个数的最大公约数。

2. 运行结果

案例运行结果如图 3-25 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 3-25　案例运行结果

3. 任务目标

(1) 学会分析“求最大公约数”案例的实现思路。

(2) 根据设计思路独立完成“求最大公约数”案例代码的编写、编译和运行。

(3) 掌握如何用 k(k≥2,k≤min(m,n))去同时除两个整数的方法求两个整数的最大公约数。

(4) 掌握 while 循环结构的使用。

4. 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,…)是否是 m 和 n 的最大公约数,直到 k 大于 m 或 n 为止。

该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++;
}
```

5. 代码实现

求最大公约数的代码如程序 3-14 所示。

【程序 3-14】 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);
  }
}
```

✍多学一招

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 $r!=0$,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。


视频讲解

3.7 案例研究: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. 运行结果

案例运行结果如图 3-26 所示。

3. 任务目标

(1) 学会分析“打印输出若干素数”案例的实现思路。

(2) 根据设计思路独立完成“打印输出若干素数”案例代码的编写、编译和运行。

(3) 掌握如何判断一个正整数是素数的方法。

(4) 掌握使用 System.printf()方法对输出进行格式化。

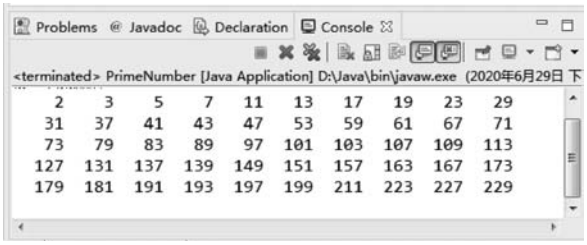

图 3-26　案例运行结果

4. 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 5 个字符位置(%5d)输出 number，输出数字右对齐，输出后换行(%n)。

5. 代码实现

计算并输出前 50 个素数的代码如程序 3-15 所示。

【程序 3-15】 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){          ← 判断 number 是否是素数，
          isPrime = false;                  ← 若循环结束 number 仍不能被 divisor 整除，
          break;                            ← 则 number 是一个素数。
        }
      }
      if(isPrime){
        count ++;                           ← 如果 number 是素数，计数器加 1，并打印素数
        if(count % 10 == 0)
          System.out.printf("%5d%n",number);  ← 格式化输出
        else
          System.out.printf("%5d",number);
```

```
            }
            number ++;  ←┤ 判断下一个数
        }
    }
}
```

【说明】 这里的循环条件 divisor＜number 表示一直检测到 divisor＝number－1。实际上,根据数论的理论,判断一个数是不是素数,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
```

小结

本章首先介绍了结构化编程方法和面向对象编程方法,然后重点介绍了结构化编程的三种控制结构,通过案例演示了这些结构的使用。

第 4 章将学习面向对象编程的基本概念。主要包括类的定义、对象的创建、方法设计以及对象初始化和销毁。

习题与实践

一、填空题

1. 编写一个 if 语句,在变量 x 的值大于 0 时,将 1 赋给变量 y:_____。

2. 编写一个 if-else 语句,若变量 x 大于 y,将 x 赋给变量 max,否则将变量 y 赋给变量 max:_____。

3. 编写布尔表达式,当体重 weight 大于 50 或者身高 height 大于 1.70 时结果为 true:_____。

4. do-while 循环和 while 循环,_____将至少被执行一次。

5. 用 while 写一个无限循环结构:_____。

6. 在嵌套的循环结构的内层循环中,要结束外层循环应该使用语句:_____。

7. 在 for 循环结构 for(int i ＝ 0; i ＜100; i＋＋)循环体中,若包含一个 continue 语句,当执行到该语句时,程序控制转到_____。

8. 下面的 if-else 结构用条件运算符可写为_____。

```
if (age >= 16) ticketPrice = 20;
else  ticketPrice = 10;
```

9. 下面代码中输出语句将打印_____个星号(＊)。

```
for(int i = 0; i < 10; i++)
```

```
   for(int j = i; j < 10; j ++)
      System.out.print (" * ");
```

二、判断题

1. if 语句的条件可以是 1 或 0。(　　)

2. 下面的 if 语句在 Java 语言中是不合法的。(　　)

```
if(0 < n < 10){
     System.out.println("n 的值介于 0 和 10 之间");
}
```

3. break 语句只能用在循环结构中。(　　)

4. if 语句可以嵌套在循环结构中,但循环结构不能嵌套在 if 语句中。(　　)

5. 当 if 语句或循环体只有一条语句时,可以省略大括号。(　　)

三、选择题

1. 假设 x=2 以及 y=3,下面代码的输出结果是(　　)。

```
if (x > 2)
    if(y > 2){
        int z = x + y;
        System.out.println("z is " + z) ;
    }
else{
     System.out.println("x is " + x) ;
}
```

A. x is 2　　B. x is 5　　C. x is 6　　D. 无输出

2. 给定下面代码段,变量 i 可使用的数据类型有(　　)。

```
switch (i) {
   default -> System.out.println("Hello");
}
```

A. char　　B. byte　　C. float　　D. double

E. Object　　F. enum

3. 给定下面程序段,输出结果为(　　)。

```
int i = 1, j = 0 ;
switch(i){
    case 1 -> j += 6;
    case 2 -> j += 1;
    default -> j += 2;
 }
System.out.println(j);
```

A. 2　　B. 6　　C. 7　　D. 9

4. 下面程序段执行后,i、j 的值分别为(　　)。

```
int i = 1, j = 10;
```

```
do{
    if(i++> --j)
       continue;
}while(i < 5);
System.out.println("i = " + i + "  j = " + j);
```

A. i=6 j=5　　B. i=5 j=5　　C. i=6 j=4　　D. i=5 j=6

5. 下面程序的执行结果为(　　)。

```
public class FooBar{
    public static void main(String[] args){
        int i = 0, j = 5;
        tp: for(; ; i++){
              for(; ; --j)
                  if(i > j)  break tp;
              }
        System.out.println("i = " + i + ",j = " + j);
    }
}
```

A. i=1,j=-1　　B. i=0,j=-1　　C. i=1,j=4　　D. i=0,j=4

E. 在第 4 行产生编译错误

四、讨论题

1. 下面代码有什么错误?

```
if (score >= 60.0)
   System.out.println("及格") ;
else if(score >= 70.0)
   System.out.println("中等") ;
else if(score >= 80.0)
   System.out.println("良好") ;
else if(score >= 90.0)
   System.out.println("优秀") ;
else
   System.out.println("不及格") ;
```

2. 下面程序有两处编译错误,请指出。

```
public class IfWhileTest {
    public static void main (String []args)   {
        int x = 1, y = 6;
        if(x = y)
            System.out.println("相等");
        else
            System.out.println("不相等");
        while (y--)  { x++; }
        System.out.println("x = " + x+ "  y = " + y);
    }
}
```

若使程序输出下面的结果,应如何修改程序?

```
不相等
x = 7  y = -1
```

3. 下面代码的输出结果是什么？解释原因。

```
int x = 80000000;
while (x > 0)
  x ++;
System.out.println("x is " + x);
```

4. 若编写代码实现如下功能：如果年龄(age)大于 40 岁，并且工资(salary)小于 5000 元，工资增加 1000 元；如果年龄不大于 40 岁，将工资增加 500 元。讨论该段代码是否正确。

```
if( age > 40)
    if( salary < 5000)
        salary = salary + 1000;
else
    salary = salary + 500;
```

五、简答题

1. 根据下面的要求编写语句。

(1) 产生一个随机整数 i，使得 0≤i<20。

(2) 产生一个随机整数 i，使得 10≤i<20。

(3) 产生一个随机整数 i，使得 0≤i≤50。

(4) 编写一个表达式，随机返回 0 或者 1。

2. 使用 if-else 语句重写下面的表达式。

```
tax  = (income > 10000) ? income * 0.2 : income * 0.17 + 1000;
```

3. 编写布尔表达式，当年龄 age 大于 13 且小于 16 时结果为 true。

4. 编写布尔表达式，当体重 weight 大于 50 或者身高 height 大于 1.70，但不同时满足时结果为 true。

5. 编写一个 switch 语句，如果 day 是 0、1、2、3、4、5、6，分别显示 Sunday、Monday、Tuesday、Wednesday、Thursday、Friday 和 Saturday。

6. 编写一个表达式，要求这个表达式根据 x 是否大于、小于或等于 y，分别取值 1，－1 或 0。

7. 将下面的 for 循环语句转换为 while 循环和 do-while 循环。

```
long sum = 0;
for(int i = 0; i <= 1000; i++)
     sum = sum + i;
```

8. 下面的程序输出 2～100 的所有素数，请填空。

```
public class PrimeDemo {
    public static void main(String[] args){
        int i = 0, j = 0;
        for(i = 2; i <= 100; i++){
```

```
            for(j = 2; j < i; j++){
                if(i % j == 0)
                    ________
                }
            if(________)
                 System.out.print(i + "  ");
        }
    }
}
```

9. 写出下面程序的运行结果。

```
public class Foo{
   public static void main(String[] args){
      int i = 1;
      int j = i++;
      if((i > ++j) && (i++ == j)){
         i += j;
      }
      System.out.println("i = " + i + ", j = " + j) ;
   }
}
```

10. 假设输入是 2 3 4 5 0,那么下面程序的输出结果是什么?

```
public class Test {
   public static void main (String []args)   {
     Scanner input = new Scanner(System.in);
     int number, max;
     number = input.nextInt();
     max = number;

     while(number!= 0){
        number = input.nextInt();
        if(number > max)
           max = number;
     }
     System.out.println("max is " + max);
     System.out.println("number is " + number);
  }
}
```

11. 给出下面程序的执行结果。

```
public class LabelDemo{
   public static void main(String[] args){
     outer:
       for(int i = 0; i < 3; i++){
          inner:
          for(int j = 0; j < 100; j++){
             if(j == 20){
                break outer;}
```

```
                if(j % 3 == 0){
                    continue inner;
                }
                System.out.print(j + "  ");
            }
        }
    }
}
```

六、编程题

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. 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。

3. 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。

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

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()方法计算数的平方根。

4. 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 中含有 7 或者是 7 倍数的整数之和及个数。运行结果如图 3-27 所示。

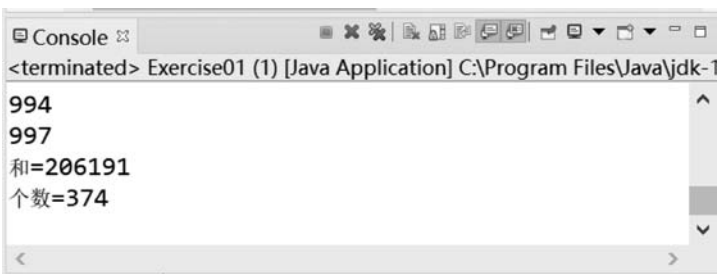

图 3-27　运行结果

5. 编写程序，打印输出如图 3-28 所示的九九乘法表。

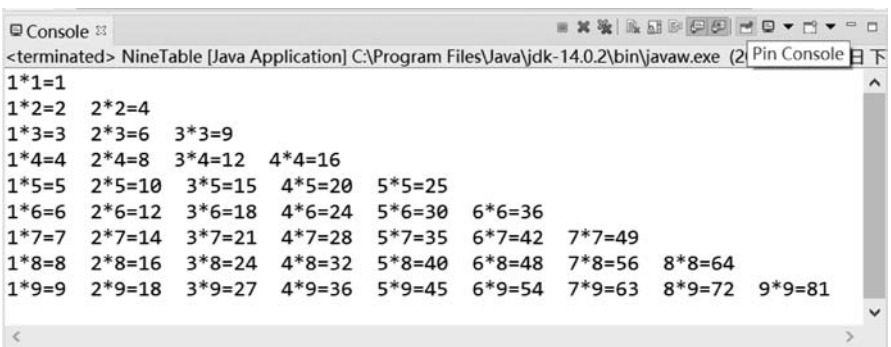

图 3-28　九九乘法表

6. 编写程序,显示 100～1000 中所有能被 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。

7. 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。运行结果如图 3-29 所示。

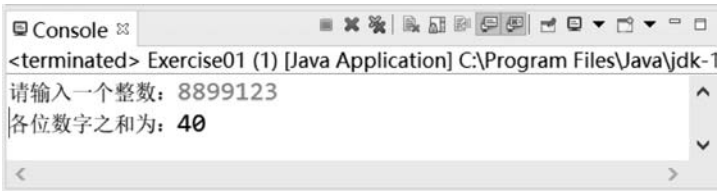

图 3-29　运行结果

8. 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。

9. 编写程序,求出 1～1000 的所有完全数。完全数是指其所有因子(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。

10. 编写程序,读入一个正整数,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。