第 3 章

Java 程序控制结构

学习目标

1. 熟悉流程图的相关概念和基本知识
2. 掌握绘制流程图的基本方法和技巧
3. 掌握顺序、分支和循环结构的流程和语法
4. 熟练使用程序结构控制语法进行相关程序设计

教学方式

本章以理论讲解、案例演示、代码分析为主。借助流程图进行问题分析，并进行相应程序的设计。

重点知识

1. 三种基本类型流程图的绘制
2. 多种分支结构语法规则，根据实际需求选择合适的分支结构
3. 三种典型的循环结构语法规则，根据实际需求选择合适的循环结构

3.1 案例 3-1 商品竞价

3.1.1 案例描述

给出一种商品，要求用户竞猜其价格，如果猜对了，则给出商品价格，并提示一共猜测了几次；如果猜错了，则提示猜高了或猜低了，用户进行下一次尝试，一共限定 5 次机会。

要求使用流程图描述商品竞价过程，并设计相应程序。

3.1.2 案例关联知识

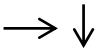
1. 问题解决与流程图

在现实工作和生活中，人们一直在试图解决各种问题。对于不同类型的问题，人们研究

出了不同的解决办法，通常用算法来描述解决问题的过程。为了更清晰地表达算法，可以将解决问题的步骤整理成流程图。

流程图是一种被广泛用于描述算法的工具，它使用美国国家标准化学会（American National Standards Institute，ANSI）规定的一些图框、线条来形象、直观地描述算法过程。常用的流程图符号如表 3-1 所示。

表 3-1　常用的流程图符号

名　称	图 形 元 素	功　能
开始或结束框		表示算法的开始或结束
处理框		表示算法的一般处理操作
判断框		表示对一个给定的条件进行判断
流程线		表示算法的执行走向
输入或输出框		表示算法的输入或输出操作

2．基本结构流程图

在解决问题的过程中，有三种基本控制结构，即顺序结构、分支结构和循环结构。

1）顺序结构

顺序结构是问题描述中简单常用的一种结构，即按照算法的流程，自上而下，依次执行。

例 3-1　用流程图描述去图书馆借书的过程。

分析：在图书馆借书需要先在检索台查询图书条形码；再根据条形码查找图书位置并找到图书；最后在借书处登记借书。顺序结构流程图如图 3-1 所示。

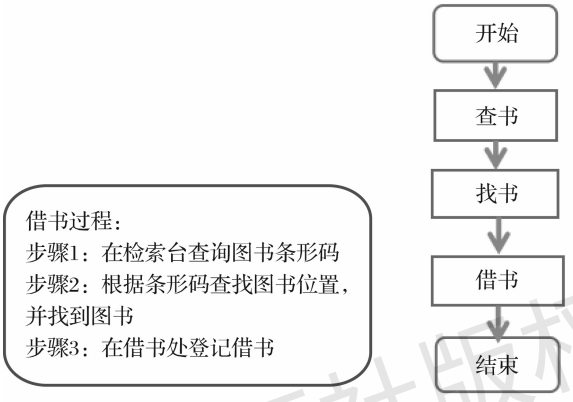


图 3-1　顺序结构流程图

借助如图 3-1 所示的流程图，可以很清晰地看出图书馆借书的步骤，顺序结构流程图也是几种流程图中相对简单且常见的一种，顺序流程图主要由起始框和处理框组成。

例 3-2　请根据下述公式，将用户录入的摄氏温度转换为华氏温度。

华氏温度 = 摄氏温度×（9/5）+32

分析：将用户录入的温度数据根据上式进行转换即可，因此，可用顺序结构流程图表示。例 3-2 的顺序结构流程图如图 3-2 所示。

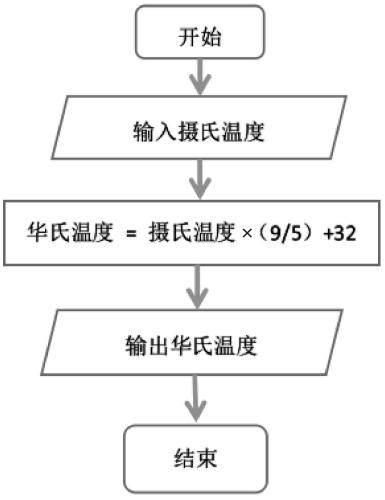


图 3-2　例 3-2 的顺序结构流程图

2）分支结构

分支结构又称选择结构，此种结构在处理问题时需要根据条件进行判断。根据要处理的分支不同，分支结构可分为单分支结构和多分支结构。分支结构流程图如图 3-3 所示。

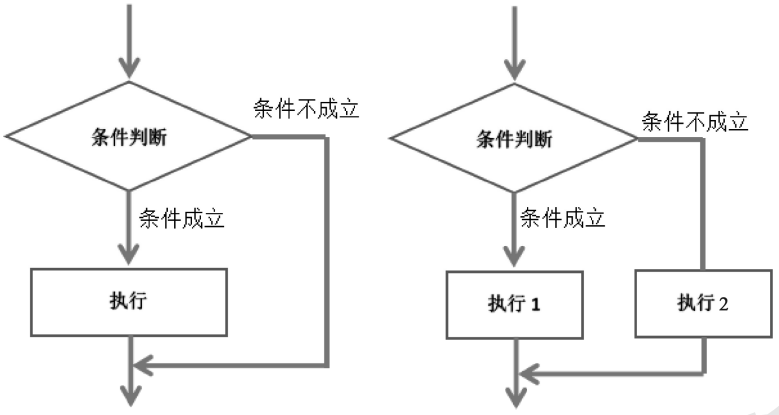


图 3-3　分支结构流程图

例 3-3　在用户输入合法（0～100）的语文、数学、英语成绩后，计算并输出三科的平均成绩。

分析：首先判断输入的语文、数学、英语成绩是否合法，合法则计算平均成绩，否则不计算。例 3-3 的分支结构流程图如图 3-4 所示。

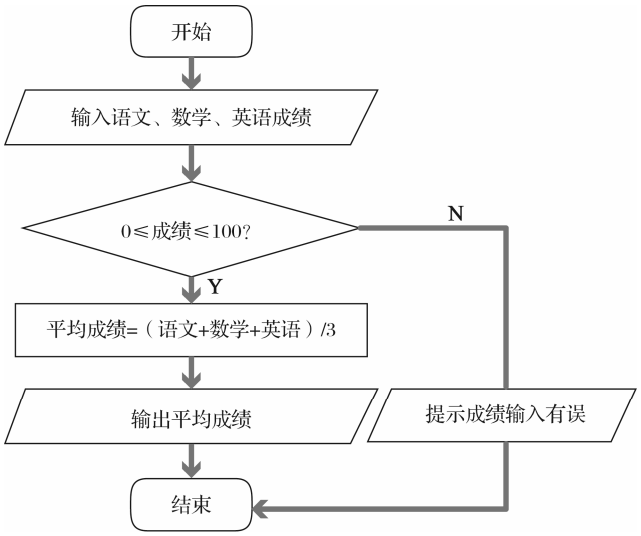


图 3-4　例 3-3 的分支结构流程图

例 3-4　某物业公司收取物业费，房屋面积小于或等于 80m² 时物业费为 3 元/m²；如果房屋面积大于 80m²，则超过 80m² 的部分的物业费为 5 元/m²，请根据住户的房屋面积计算需要交多少物业费？

分析：当房屋面积 0m²<*S*≤80m² 时，物业费为 *M*=*S*×3 元；当房屋面积 *S*>80m² 时，物业费为 *M*=80×3+(*S*−80)×5 元。例 3-4 的分支结构流程图如图 3-5 所示。

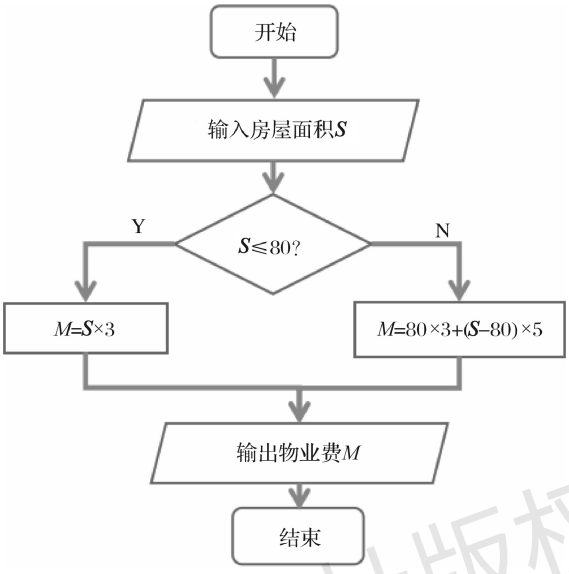


图 3-5　例 3-4 的分支结构流程图

例 3-5　输入 3 个数然后将其按照从大到小的顺序输出，请用流程图描述其算法。

分析：此问题为三个数的降序排序问题，三个数需要两两进行大小比较，然后根据比较结果进行排序。例 3-5 的分支结构流程图如图 3-6 所示。

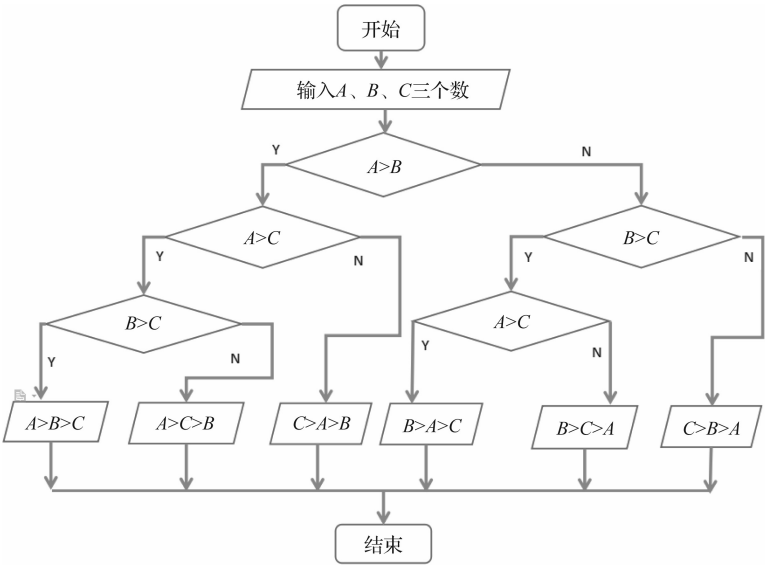


图 3-6　例 3-5 的分支流程图

3）循环结构

循环结构适用于处理根据给定条件重复执行某一部分的操作的情况，其流程图与顺序结构、分支结构比较大的区别就是，执行过程不是依次顺序向下，而是根据条件判断结构确定流程后重新回到流程判断的地方。

例 3-6　计算 1+2+…+100 的结果。

分析：此问题是求解 100 以内所有整数的和，如果加数小于或等于 100，则重复进行累加求和，加数递增 1，继续求和；如果加数大于 100，则结束。此过程是一个典型的循环结构。例 3-6 的循环流程图如图 3-7 所示。

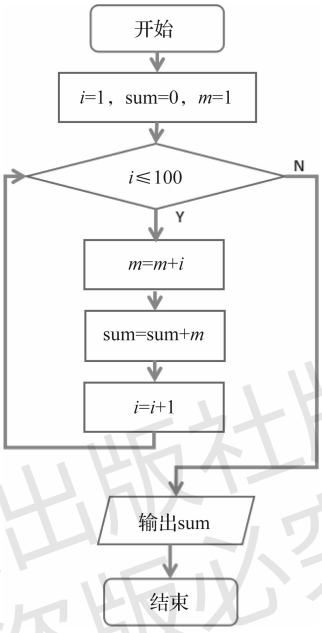


图 3-7　例 3-6 的循环流程图

3.1.3 案例分析

有了案例关联知识的铺垫，接下来我们分析一下商品竞猜案例的基本过程。

（1）竞猜价格和真正的商品价格需要进行比较，如果二者相等，竞猜正确，则执行提示结果正确输出商品价格的分支；否则，重复进行下一次商品价格的竞猜操作。

（2）在每次竞猜前判定是否已经达到最大竞猜次数，如果已经达到最大竞猜次数，则提示竞猜失败；如果没有达到最大竞猜次数，则可以进行下一次判断。

此案例的流程图涉及分支、循环两种结构，借助流程图可以更清晰地分析出竞猜的基本过程。

3.1.4 案例实现

商品价格竞猜流程图如图 3-8 所示。

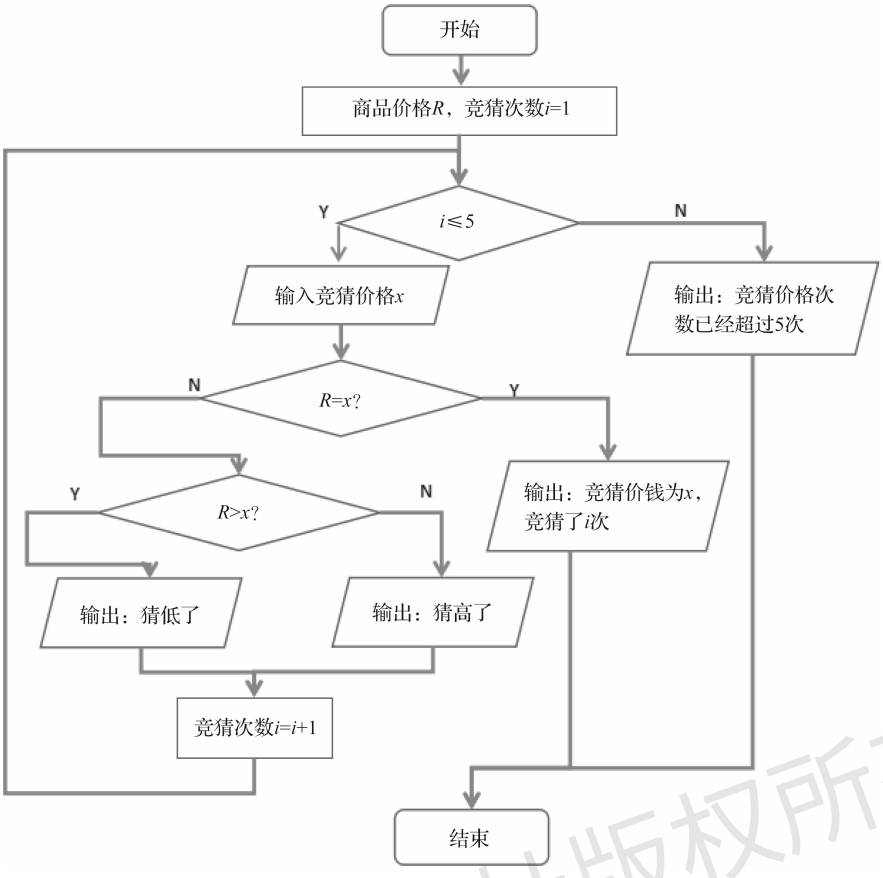


图 3-8　商品价格竞猜流程图

3.1.5 案例小结

本案例主要侧重于对分支结构和循环结构的流程图进行设计，设计过程中需要注意的是，流程图以开始框开始，以结束框结束；判断框需要对成立和不成立分支进行明确绘制，不可遗漏。

3.1.6　案例拓展

同学们可以进一步在本案例的基础上考虑如下情况：如果现有商品一共是 5 种，需要分别竞猜价格，那么应如何用流程图描述竞猜过程呢？

3.2　案例 3-2 出租车计费

3.2.1　案例描述

上海出租车计费标准如下所示。

白天（5:00—23:00）：起步价为 14 元（3 公里以内）；超过 3 公里后，每公里价格为 2.4 元；超过 10 公里后，每公里价格为 3.6 元。

夜间（23:00—次日 5:00）：起步价为 18 元（3 公里以内）；超过 3 公里后，每公里价格为 3.1 元；超过 10 公里后，每公里价格为 4.7 元。

时速低于 12 公里时（等候费）每 5 分钟收费 2 元。

请根据乘客输入的公里数、乘车时间（白天或夜间）、是否等候等数据来计算乘客的出租车费。

3.2.2　案例关联知识

1．Java 顺序结构

例 3-7　根据用户输入的半径来计算圆面积，请给出程序设计实现。

文件名：Demo3_1.java

程序代码：

```
import java.util.Scanner;                          //导入 Scanner 类所在的包
public class Demo3_1 {
   public static void main(String[] args) {
      double r;                                    //声明半径
      double area;                                 //声明面积
      Scanner sc = new Scanner(System.in);         //创建键盘输入类对象
      System.out.println("请输入圆的半径：");
      r = sc.nextDouble();                         //用户输入半径
      area = 3.14*r*r;                             //计算面积
      System.out.println("该圆的面积为："+area);
   }
}
```

Demo3_1.java 的运行结果如图 3-9 所示。

程序解析：上述代码按照语句的先后顺序逐句执行，即顺序结构程序。上例程序流程如下所示。

（1）通过键盘输入圆的半径。

（2）根据圆的半径计算圆的面积。

（3）显示该半径对应的圆的面积值。

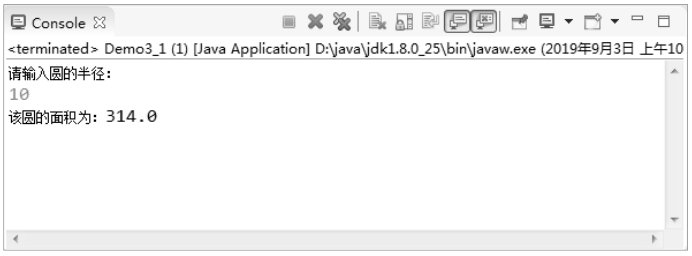


图 3-9　Demo3_1.java 的运行结果

例 3-8　实现对于例 3-2 的代码，即将摄氏温度转换为华氏温度。

文件名：Demo3_2.java

程序代码：

```
import java.util.Scanner;                          //导入 Scanner 类所在的包
public class Demo3_2 {
    public static void main(String[] args) {
        double C;                                  //声明摄氏温度
        double F;                                  //声明华氏温度
        Scanner sc = new Scanner(System.in);       //创建键盘输入类对象
        System.out.println("请输入要转换的摄氏温度：");
C = sc.nextDouble();                               //用户输入摄氏温度
F=C*9/5.0+32;                                      //计算华氏温度
        System.out.println("该摄氏温度："+C+" 对应的华氏温度为："+F);
    }
}
```

Demo3_2.java 的运行结果如图 3-10 所示。

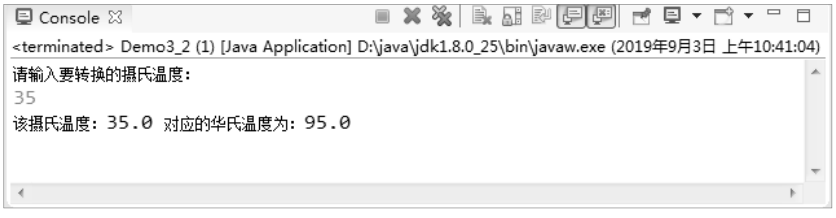


图 3-10　Demo3_2.java 的运行结果

程序解析：上述代码结构与例 3-7 相似，为顺序结构程序设计实现。

2．Java 分支结构

Java 提供了两种基本的分支选择语句，分别为 if 语句和 switch 语句。

1）if 语句（单分支）

```
if(条件表达式)
    语句
```

上述语句也称单分支结构，由关键字 if 及括号内的条件表达式和执行语句组成，如果条件表达式的运行结果为 true，则执行语句；如果条件表达式的运行结果为 false，则跳过该语句

执行后续语句。单分支结构流程图如图 3-11 所示。

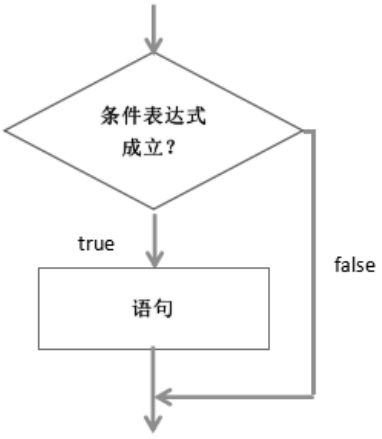


图 3-11　单分支结构流程图

例 3-9　根据用户输入的半径进行圆的面积的计算，增加用户输入的合法性判断。

文件名：Demo3_3.java

程序代码：

```
import java.util.Scanner;                    //导入 Scanner 类所在的包
public class Demo3_3 {
   public static void main(String[] args) {
      double r;
      double area;
      Scanner sc = new Scanner(System.in);
      System.out.println("请输入圆的半径：");
      r = sc.nextDouble();
      if(r>=0)
      {
      area = 3.14*r*r;
      System.out.println("该圆的面积为："+area);
      }
   }
}
```

Demo3_3.java 的运行结果如图 3-12 所示。

图 3-12　Demo3_3.java 的运行结果

程序解析：与例 3-7 相比，上述代码使用了 if 语句来判断用户输入的半径的合法性，如果输入的数据合法，则进行圆面积计算。该代码属于单分支结构。

例 3-10　完成例 3-3 的代码设计实现，即求输入合法的三科成绩的平均分。

文件名：Demo3_4.java

程序代码：

```
import java.util.Scanner;                    //导入 Scanner 类所在的包
public class Demo3_4 {
    public static void main(String[] args) {
        int iChinese,iMath,iEnglish;
        iChinese = 0;
        iMath = 0;
        iEnglish = 0;
        double avg = 0;
        Scanner sc = new Scanner(System.in);
        System.out.println("请分别输入语文、数学、英语成绩(0~100): ");
        iChinese = sc.nextInt();
        iMath = sc.nextInt();
        iEnglish = sc.nextInt();
        if((iChinese>=0 && iChinese<=100) &&(iMath>=0 && iMath<=100) &&
                (iEnglish>=0 && iEnglish<=100)){
            avg = (iChinese+iMath+iEnglish)/3.0;
            System.out.printf("对应的平均成绩为: %.2f",avg);
        }
    }
}
```

Demo3_4.java 的运行结果如图 3-13 所示。

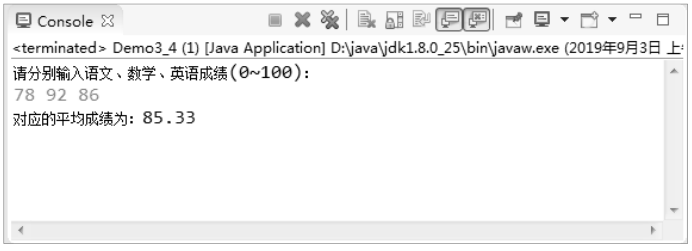


图 3-13　Demo3_4.java 的运行结果

程序解析：上述代码在计算平均成绩前，需要有成绩录入的合法性判断，限定成绩为 0～100 分，再进行平均成绩求解。代码中使用了 printf 输出，限定输出的小数点后的位数为 2。

2）if 语句（双分支）

```
if(条件表达式)
    语句 1
 else 语句 2
```

上述语句也称为双分支结构，if()内部的条件表达式的结果必须是布尔型的。当需要某个条件表达式的值为 true 执行某语句，其值为 false 执行另一语句时，可以在 if 语句中增加一个 else 子句构建一条 if…else 语句来处理这种情况。双分支程序结构流程图如图 3-14 所示。

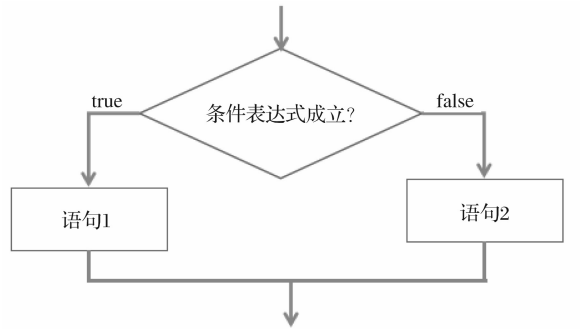


图 3-14　双分支程序结构流程图

注意　图 3-14 中的语句 1 和语句 2 可以是一条语句，也可以是多条语句构成的语句块，还可以是嵌套的 if…else 语句。

例 3-11　根据用户输入的半径计算圆的面积，增加用户输入的合法性判断，如果用户输入的半径值大于或等于 0，则进行圆面积计算；否则，输出“您输入的圆半径不合法”。

文件名：Demo3_5.java

程序代码：

```
import java.util.Scanner;                    //导入 Scanner 类所在的包
public class Demo3_5 {
   public static void main(String[] args) {
      double r;
      double area;
      Scanner sc = new Scanner(System.in);
      System.out.println("请输入圆的半径：");
      r = sc.nextDouble();
      if(r>=0)
      {
      area = 3.14*r*r;
      System.out.print("该圆的面积为："+area);
      }
      else
      {
System.out.println("您输入的圆半径不合法");
      }
   }
}
```

Demo3_5.java 的运行结果如图 3-15 所示。

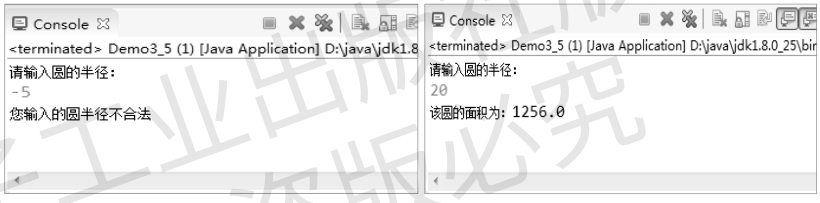


图 3-15　Demo3_5.java 的运行结果

程序解析：与例 3-9 相比，上述代码使用了 if 语句来判断用户输入的半径的合法性，合法则计算圆半径；否则，输出“您输入的半径不合法”的提示。

例 3-12　完成例 3-5，即输入 3 个数并按照从大到小的顺序将其输出的程序设计实现。

文件名：Demo3_6.java

程序代码：

```
import java.util.Scanner;                              //导入 Scanner 类所在的包
public class Demo3_6 {
    public static void main(String[] args) {
        int a,b,c;
        a=b=c=0;
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入 a,b,c 三个整数：");
        a = sc.nextInt();
        b = sc.nextInt();
        c = sc.nextInt();
        if(a>b){                                       //1
            if(a>c){                                   //2
                if(b>c){                               //3
                    System.out.println("a>b>c");
                }
                else{                                  //与 3 配对
                    System.out.println("a>c>b");
                }
            }
            else{                                      //与 2 配对
                System.out.println("c>a>b");
            }
        }
        else{                                          //与 1 配对
            if(b>c){                                   //4
                if(a>c){                               //5
                    System.out.println("b>a>c");
                }
                else{                                  //与 5 配对
                    System.out.println("b>c>a");
                }
            }
            else{                                      //与 4 配对
                System.out.print("c>b>a");
            }
        }
    }
}
```

Demo3_6.java 的运行结果如图 3-16 所示。

图 3-16　Demo3_6.java 的运行结果

程序解析：上述代码属于双分支结构，该代码的每个分支结构中又嵌套了分支结构。需要注意的是，在这种多分支嵌套的情况下，为避免嵌套的 if…else 语句的二义性，Java 规定，else 总是与在其之前未配对的最近的 if 配对，例 3-12 中各个 else 配对的 if 参见其中给出的注释。

3）if 语句（多分支）

```
if(条件表达式 1)
    语句 1
else if（条件表达式 2）
    语句 2
else if（条件表达式 3）
    语句 3
…
else if（条件表达式 m）
    语句 m
else
    语句 n
```

上述语句也称为多分支结构，在执行该语句时，先计算 if 的表达式 1 的值，如果表达式 1 为 true，则执行语句 1；否则，执行 else if 的表达式 2 的值。如果 else if 的表达式 2 为 true 成立，则执行语句 2；否则，依次计算后面的 else if 的表达式 3 的值，以此类推。如果所有表达式的值都是 false，则执行 else 后面的语句，结束整体判断。

使用 if…else if…else 语句的时候，需要注意以下几点。

（1）if 语句至多有 1 个 else 语句，else 语句在所有 else if 语句之后。

（2）if 语句可以有若干个 else if 语句，它们必须在 else 语句之前。

（3）一旦有一个 else if 语句检测为 true，其他 else if 及 else 语句都将跳过执行。

例 3-13　请根据用户从键盘输入的月份，判断该月份属于一年中哪个季度。

文件名：Demo3_7.java

程序代码：

```
import java.util.Scanner;                    //导入 Scanner 类所在的包
public class Demo3_7 {
   public static void main(String[] args) {
      int month;
      Scanner sc = new Scanner(System.in);
      System.out.println("请输入一个月份（1~12 的整数）");
      month = sc.nextInt();
      if(month>=1 && month<=3){
         System.out.println("该月是第一季度");
```

```
        }
        else if(month>=4 && month<=6){
            System.out.println("该月是第二季度");
        }
        else if(month>=7 && month<=9){
            System.out.println("该月是第三季度");
        }
        else if(month>=10 && month<=12){
            System.out.println("该月是第四季度");
        }
        else{
            System.out.println("您输入的月份不合法");
        }
    }
}
```

Demo3_7.java 的运行结果如图 3-17 所示。

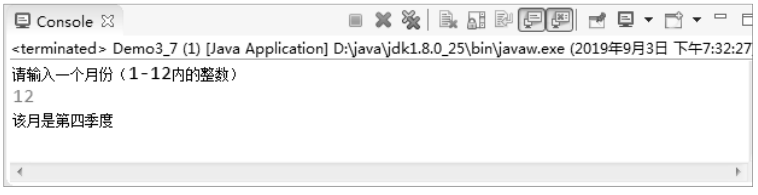


图 3-17　Demo3_7.java 的运行结果

程序解析：通过对键盘输入的月份值进行判断，哪个分支成立就执行哪个语句块。

例 3-14　请根据表 3-2 计算出个人所得税。

表 3-2　个人所得税率计算表

工资、薪金所得适用个人所得税累进税率表			
级数	全月应纳税所得额	税率%	速算扣除数（元）
一	不超过 3000 元的	3	0
二	超过 3000 元至 12000 元的部分	10	210
三	超过 12000 元至 25000 元的部分	20	1410
四	超过 25000 元至 35000 元的部分	25	2660
五	超过 35000 元至 55000 元的部分	30	4410
六	超过 55000 元至 80000 元的部分	35	7160
七	超过 80000 元的部分	45	15160

文件名：Demo3_8.java

程序代码：

```
import java.util.Scanner;                    //导入 Scanner 类所在的包
public class Demo3_8 {
    public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
```

```
        System.out.println("应发工资为：");
            double salary = sc.nextDouble();
            double tax = 0.0;
            double money = salary-3000;
            if(money<0)
                System.out.println("你不需要缴税，努力吧");
            else if(money<=3000)
                tax = money*0.03;
            else if(money<=12000)
                tax = money*0.1 -210;
            else if(money<=25000)
                tax = money*0.2-1410;
            else if(money<=35000)
                tax =money*0.25-2660;
            else if(money<=55000)
                tax =money*0.3-4410;
            else if(money<=80000)
                tax =money*0.35-7160;
            else
                tax =money*0.45-15160;
            System.out.println("实发工资为："+(salary-tax)+",应缴税为："+tax);
        }
    }
```

Demo3_8.java 的运行结果如图 3-18 所示。

图 3-18　Demo3_8.java 的运行结果

程序解析：主要解决个人所得税计算问题，共包括七个判断区间，需要根据用户输入的收入进入对应分支进行税收计算。需要注意的是，多分支判断 else if 之间是互斥的，不可以同时进入多个分支操作，即在练习时，注意不可写成如下判断形式：

```
if(money<0)
            ...
        if(money<=3000)
            ...
        if(money<=12000)
            ...
```

若写成这种形式，如果月收入是 5 万元，那么就会输出之前的“你不需要缴税，努力吧”，且会重复多次计算 tax，导致结果不正确。

4）switch 语句

switch 语句是 Java 中的另一种条件语句，执行过程为：先判断一个变量与一系列值中的

某个值是否相等，每一个值构成一个分支，如果相等，则从多条分支中选择对应相等的分支来执行。使用 if 多分支语句也可以实现同样的效果，但是相较而言使用 switch 语句会使得代码的可读性更强。

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

switch 语句先计算一个表达式的值，然后将该值和几个可能的 case 子句取值进行匹配，每种取值都有与之关联的执行语句。当计算出表达式的值后，将执行与表达式值相匹配的 case 子句。如果没有与 case 子句的值相匹配的值，则流程会执行由 default 指定的默认语句。

需要特别注意的是：

（1）switch 语句中开始的表达式运算结构必须是 char、byte、short 或 int 类型。

（2）每一个 case 子句中的表达式必须为常量，不能为变量或其他表达式。

（3）当执行到每个 case 子句结束处的 break 语句时，会退出 switch 语句中对应的 case 子句执行。如果没有 break 语句，则会继续执行后面 case 指示的若干语句。

例 3-15　用 switch 语句实现用户从键盘输入一个代表月份的整数，程序输出该月份所属的季度。

文件名：Demo3_9.java

程序代码：

```
import java.util.Scanner;                    //导入 Scanner 类所在的包
public class Demo3_9 {
    public static void main(String[] args) {
        int month;
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入一个月份（1~12 的整数）");
        month = sc.nextInt();
            switch(month){
                case 1:
                case 2:
                case 3:
                    System.out.println("该月是第一季度");
                    break;
                case 4:
```

```
                case 5:
                case 6:
                    System.out.println("该月是第二季度");
                    break;
                case 7:
                case 8:
                case 9:
                    System.out.println("该月是第三季度");
                    break;
                case 10:
                case 11:
                case 12:
                    System.out.println("该月是第四季度");
                    break;
                default:
                    System.out.println("您输入的月份不合法");
            }
        }
    }
```

Demo3_9.java 的运行结果如图 3-19 所示。

图 3-19　Demo3_9.java 的运行结果

程序解析：上述代码首先对月份进行判断，根据用户输入的月份执行和表达式匹配的 case 子句。如果整型变量 month 的值为 1、2、3，则“显示该月是第一季度”，跳过后续的判断；否则，跳过 case 1、case 2、case 3 子句，进入后续 case 判断。

如果上述代码没有 break 语句，程序如下面所示。

```
        switch(month){
        case 1:
        case 2:
        case 3:
    System.out.println("该月是第一季度");
        case 4:
        case 5:
        case 6:
    System.out.println("该月是第二季度");
        case 7:
        case 8:
        case 9:
    System.out.println("该月是第三季度");
        case 10:
```

```
        case 11:
        case 12:
    System.out.println("该月是第四季度");
default:
    System.out.println("您输入的月份不合法");
    }
```

例 3-15 中没有 break 语句时的运行结果如图 3-20 所示。

```
请输入一个月份（1~12的整数）
8
该月是第三季度
该月是第四季度
您输入的月份不合法
```

图 3-20　例 3-15 中没有 break 语句时的运行结果

从图 3-20 中可以看出，从当前 case8 开始，后续所有 case 的值都会被输出，如果后续的 case 语句块有 break 语句，那么程序将会跳出判断。

3.2.3　案例分析

有了关联知识的铺垫，接下来我们编写代码来实现出租车计费。通过对案例需求的分析可知，出租车收费时涉及几个典型的分支判断过程，包括：判断是白天乘车还是夜晚乘车、乘车距离是否超过起步价的里程、是否有候车情况等。

3.2.4　案例实现

文件名：Demo3_10.java

程序代码：

```
import java.util.Scanner;                    //导入 Scanner 类所在的包
public class Demo3_10 {
    public static void main(String[] args) {
double money_dis,money_tim;
        double distance;
        int waittime=0;
        Scanner t=new Scanner(System.in);
        System.out.println("输入出租车等车时间");
        waittime=t.nextInt();
        money_tim=2*waittime/5;
        System.out.println("请输入白天还是夜晚行车（白天输入 1，夜晚输入其他数值：）");
        int tt=t.nextInt();
        if(tt==1){
            System.out.println("输入白天出租车行驶里程：");
            distance=t.nextDouble();
            if(distance<=3) {money_dis=12;}
            else if(distance<=10) {money_dis=12+2.4*(distance-3);}
            else{money_dis=+12+7*2.4+3.6*(distance-10);}
```

```
        }
        else {
            System.out.println("输入夜晚出租车行驶里程");
            distance=t.nextDouble();
            if(distance<=3){money_dis=16;}
            else if(distance<=10){money_dis=16+3.1*(distance-3);}
            else{money_dis=16+3.1*7+4.7*(distance-10);}
        }
        double money= money_dis+money_tim;
        System.out.println("***********消费清单*************");
        System.out.println("     亲，一共行驶"+(distance)+"公里");
        System.out.println("       等待时间"+(waittime)+"分钟");
        System.out.println("          收取费用￥"+(money)+"元");
        System.out.println("*************************");
    }
}
```

Demo3_10.java 的运行结果如图 3-21 所示。

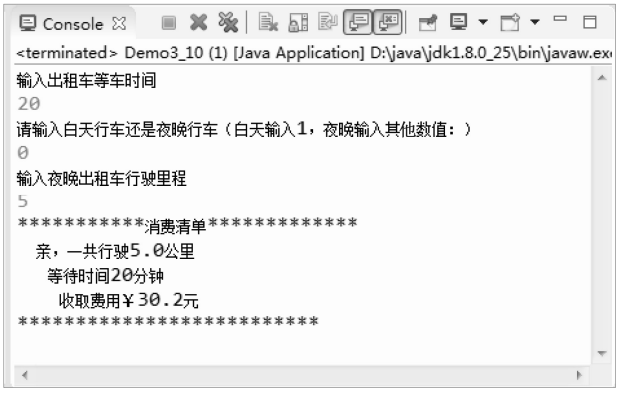


图 3-21　Demo3_10.java 的运行结果

3.2.5　案例小结

本案例主要用来练习分支语句的使用，包括双分支结构、多分支结构实现，代码实现可以采用 if…else 语句或嵌套的 if 语句，具体选用什么语句视实际情况而定。

3.2.6　案例拓展

通过本案例相关知识及案例实现，同学们对如何设计 Java 中的分支结构程序有了比较深入的理解。如果某位乘客是白天乘车，行程较长，直到夜间才到地点，请同学们进一步完善该程序。

3.3 案例 3-3 闰年求解

3.3.1 案例描述

请求解出 20 世纪一共有多少个闰年，并将结果输出显示出来。

3.3.2 案例关联知识

顺序结构的程序语句只能被执行一次，如果同样的语句被执行多次，那么就需要使用循环结构。Java 中有三种主要的循环结构：while 循环、do…while 循环、for 循环。

1．while 循环

```
while(表达式){
    语句或语句块
}
```

while 循环是基本的循环，像 if 语句一样先计算布尔表达式的值，当值为 true 时执行循环体语句，循环体执行完毕后再次计算表达式的值。当表达式的值不为 true 时，结束 while 语句。

例 3-16　使用 while 语句计算 100 以内的奇数的和。

文件名：Demo3_11.java

程序代码：

```
import java.util.Scanner;                    //导入 Scanner 类所在的包
public class Demo3_11 {
   public static void main(String[] args) {
      int sum = 0;
      int i = 1;
      while(i<100){
      sum+=i;
      i+=2;
    }
      System.out.println("sum:"+sum);
```

Demo3_11.java 的运行结果如图 3-22 所示。

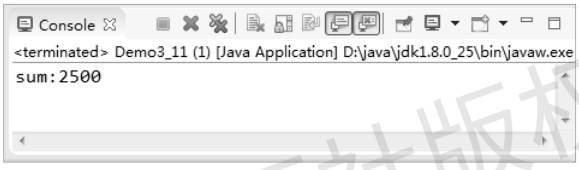


图 3-22　Demo3_11.java 的运行结果

程序解析：例 3-16 通过 while 循环实现了 100 以内的奇数的和的计算，只要判断条件 i<100 为真，就会进行加和计算，并且循环变量每次增量为 2，之后循环判断下一个数，直到不满足判断条件为止。需要特别注意的是，程序中“i+=2”语句用于修改循环变量，不可缺少；否则，程序将会一直循环执行，不断加和，无法退出。

2．do…while 循环

```
do{
    语句或语句块
}while(表达式);
```

do…while 也是循环的一种形式，do…while 循环中的循环体至少被执行一次，与 while 循环的区别可以通过如图 3-23 所示的流程图进一步区别开。

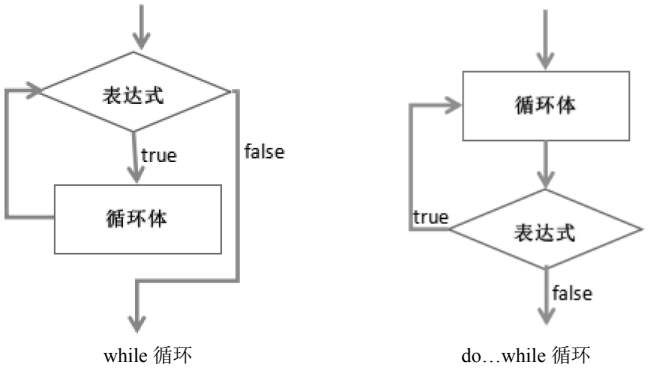


图 3-23　while 循环和 do…while 循环流程图对比

从图 3-23 中可以看出，在 do…while 循环中，循环体至少会被执行一次；while 的循环体在表达式不成立的时候，一次都不会被执行。

例 3-17　使用 do…while 语句计算 100 以内的奇数的和。

文件名：Demo3_12.java

程序代码：

```
import java.util.Scanner;                    //导入 Scanner 类所在的包
public class Demo3_12 {
    public static void main(String[] args) {
       i=1;
       sum=0;
       do{
             sum+=i;
             i+=2;
       }while(i<100);
       System.out.println("sum:"+sum);
    }
}
```

Demo3_12.java 的运行结果如图 3-24 所示。

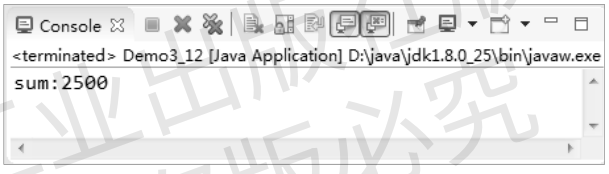


图 3-24　Demo3_12.java 的运行结果

程序解析：例 3-17 通过 do…while 循环实现了 100 以内的奇数的和的计算，与 while 循环不同，do…while 循环是先执行循环体再进行判断的，只要判断条件为真，就会进行加和计算，直到不满足判断条件为止。由图 3-24 可以看出，使用 do…while 语句的程序运行结果与使用 while 语句的程序运行结果是一致的。

例 3-18　对比 while 循环和 do…while 循环。

首先使用 while 循环进行实现，代码如下所示。

文件名：Demo3_13.java

程序代码：

```
public class Demo3_13 {
   public static void main(String[] args) {
      int x = 10;
        while( x <=10 &&x>=0) {
          if(x%2==0){
            System.out.println("偶数 x=: " + x );
          }
          x--;
        }
      }
}
```

在 Demo3_13.java 基础上使用 do…while 循环进行实现，代码如下所示。

文件名：Demo3_14.java

程序代码：

```
public class Demo3_14 {
   public static void main(String[] args) {
       int x = 10;
        do{
           if(x%2==0){
              System.out.println("偶数 x=: " + x ); }
             x--;
        }while( x<=10&&x>=0 );
   }
}
```

此时 Demo3_13.java 和 Demo3_14.java 的运行结果相同，如图 3-25 所示。

图 3-25　运行结果

程序解析：如果将 Demo3_13.java 和 Demo3_14.java 中 x 的初始值修改为 11，那么因为

while 的判断条件不为真，所以不进入循环，不输出任何数据；而 do…while 循环因为先执行了一次循环体语句，将 x 变量进行了减一操作，所以其判断条件为真，进入循环，输出结果与初始值为 10 时的结果一样。

因此，在程序设计过程中，while 循环和 do…while 循环的判断条件采用的是同一个语句。需要注意的是当 while 循环内的判断条件不成立时，是不会进行循环操作的，但是，do…while 循环内无论 while 条件成立与否，都会执行一次循环体。

3．for 循环

当循环次数无法确定时，最好使用 while 循环或者 do…while 循环。如果具体循环次数确定，那么通常使用 for 循环更合适。for 循环的一般格式如下：

```
for(表达式 1;表达式 2;表达式 3) {
    循环语句;
}
```

for 循环的控制头中包含如下三个由分号隔开的部分。

（1）表达式 1 用于初始化，在循环过程中只执行一次；

（2）表达式 2 是一个布尔表达式，在执行循环体之前需要进行判断，如果其值为 true，则继续执行循环语句；

（3）表达式 3 用来修改循环变量，改变循环条件。

需要注意的是，for 循环中三个表达式都可以省略，即 for(;;)这种形式。for 循环执行过程流程图如图 3-26 所示。

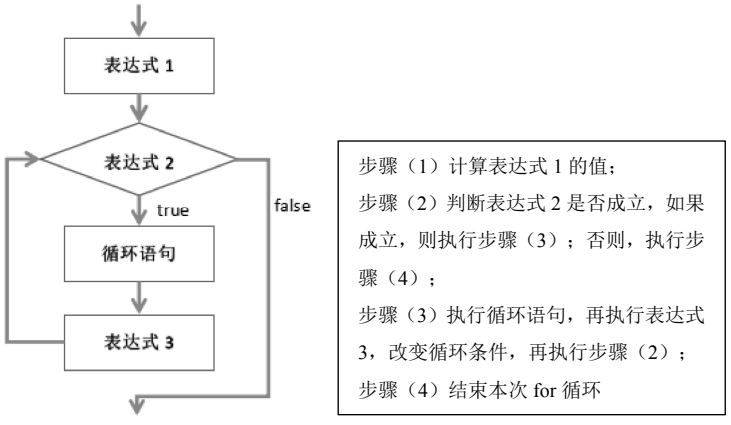


图 3-26　for 循环执行过程流程图

例 3-19　使用 for 循环计算出 100 以内的所有奇数的和。

文件名：Demo3_15.java

程序代码：

```
public class Demo3_15 {
   public static void main(String[] args) {
int sum = 0;
      for(int i = 1;i<100;i+=2){
         sum+=i;
      }
```

```
        System.out.println("sum:"+sum);
    }
}
```

Demo3_15.java 的运行结果如图 3-27 所示。

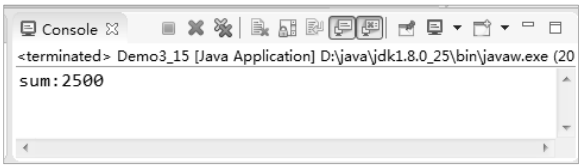


图 3-27　Demo3_15.java 的运行结果

程序解析：例 3-19 使用了 for 循环，将循环变量初始化、循环条件判断、循环变量修改放在 for 循环控制语句中。与 for 循环相比，while 循环和 do…while 循环程序更简洁。

例 3-20　使用 for 循环输出一个三角形，要求每行“*”的数量逐步递增输出。

文件名：Demo3_16.java

程序代码：

```
public class Demo3_16 {
   public static void main(String[] args) {
final int MAX_NUM = 10;
      for(int i=1;i<=MAX_NUM;i++){
         for(int j=1;j<=i;j++){
            System.out.print("*");}
System.out.println();}
   }
}
```

Demo3_16.java 的运行结果如图 3-28 所示。

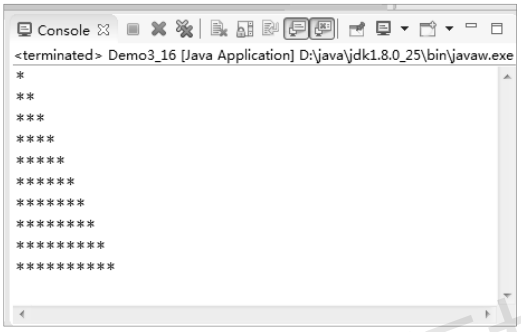


图 3-28　Demo3_16.java 的运行结果

程序解析：例 3-20 的实现使用了双重 for 循环，外层 for 循环用来控制三角形中的“*”的输出行数，内层 for 循环用来控制每行输出的“*”的个数。

4．跳转语句

Java 中用来控制程序跳转的语句主要是 break 语句或 continue 语句。

break 语句用在 switch 结构中，其作用是强制退出 switch 结构，执行 switch 结构后的语句。break 语句用在单层循环结构的循环体中，其作用是强制退出循环体。

continue 语句也称短路语句。在循环结构中，当程序执行到 continue 语句时就返回到循环体的入口处，执行下一次循环，而使循环体内的 continue 语句后面的语句不被执行。

例 3-21　比较 break 语句和 continue 语句的输出结果。

文件名：Demo3_17.java

程序代码：

```
public class Demo3_17 {
    public static void main(String[] args) {
        int stop = 4;
        for(int i=1;i<=10;i++){
            if(i==stop) break;
                System.out.println("i="+i);
          }
    }
}
```

文件名：Demo3_18.java

程序代码：

```
public class Demo3_18 {
    public static void main(String[] args) {
int skip = 4;
          for(int i=1;i<=10;i++){
              if(i==skip) continue;
                System.out.println("i="+i);
      }
    }
}
```

Demo3_17.java 的运行结果如图 3-29 所示，当循环到 i=4 时，退出循环。

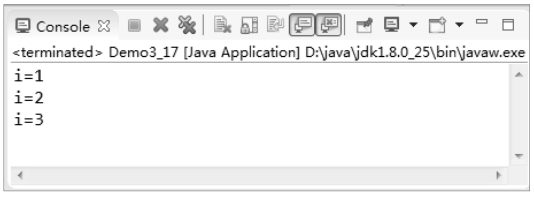


图 3-29　Demo3_17.java 的运行结果

Demo3_18.java 的运行结果如图 3-30 所示，当循环到 i=4 时，跳过了此次打印。

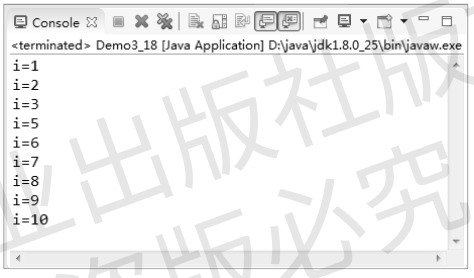


图 3-30　Demo3_18.java 的运行结果

例 3-22　商品价格竞猜程序设计实现。

文件名：Demo3_19.java

程序代码：

```
import java.util.Scanner;                    //导入 Scanner 类所在的包
public class Demo3_19 {
   public static void main(String[] args) {
int price = (int)(Math.random()*901)+100;
       System.out.println("商品价格在 100~1000 元之间，请输入你的竞猜价格：");
       Scanner sc = new Scanner(System.in);
       double guess = sc.nextInt();
       int time = 1;
       while(guess!=price){
          if(time>5)
             break;
          if(guess>price){
             System.out.println("你给的价格太高了");
          }
          else{
             System.out.println("你给的价格太低了");
          }
          guess = sc.nextInt();
          time++;
       }
       if(time<=5){
          System.out.println("你竞猜的价格正确，为："+price+" 竞猜次数为："+time);
       }
       else{
          System.out.println("你已经连续 5 次竞猜的价格不正确，商品价格为："+price);
       }
   }
}
```

Demo3_19.java 的运行结果如图 3-31 所示。

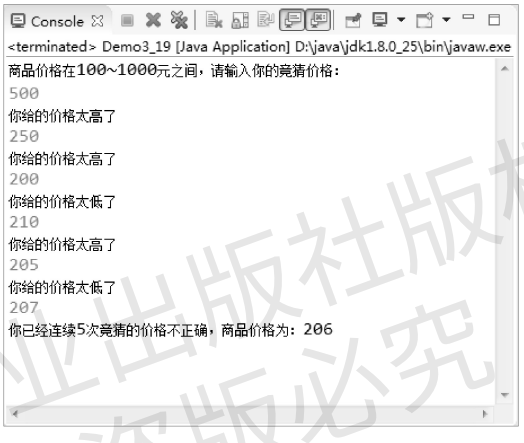


图 3-31　Demo3_19.java 的运行结果

3.3.3　案例分析

20 世纪是从公元 1901 年到公元 2000 年，循环判断下限是 1901 年，上限是 2000 年。闰年是指能被 400 整除的年份，或者不能被 100 整除能被 4 整除的年份。判断语句为：

```
if(  (year%400==0)   ||  (  (year%100!=0)  &&  (year%4==0)    )     )
```

3.3.4　案例实现

文件名：Demo3_20.java

程序代码：

```
public class Demo3_20 {
   public static void main(String[] args) {
int year=0;
       int sum=0;
       for(year=1901;year<=2000;year++){
          if((year%400==0)||((year%100!=0)&&(year%4==0)))
{
              sum++;
              System.out.println("year:"+year+"是闰年");
          }}
       System.out.println("20 世纪一共有"+sum+"个闰年");
   }}
```

Demo3_20.java 的运行结果如图 3-32 所示。

图 3-32　Demo3_20.java 的运行结果

3.3.5 案例小结

本案例重点练习的内容是循环结构和分支结构的综合使用，建议同学们在编码时，重点练习 if 多分支、switch 结构、循环结构等内容，从而熟练掌握本章知识内容。

3.3.6 案例拓展

通过学习本案例，同学们应该对循环结构设计有更为清晰的认识，可以进一步进行案例拓展内容设计实现。例如，求解出生年份是否为闰年，这些闰年中有哪些年份的生肖是一致的。