第 3 章　程序流程控制

主要内容：本章主要介绍 Java 程序设计过程中的主要程序结构、程序控制语句和数组相关知识。

教学目标：掌握程序流程结构及主要的程序控制语句，即选择语句、循环语句及循环控制语句；掌握数组的基础知识和使用。

3.1　案例：摄氏温度到华氏温度对照表的实现

【例 3-1】 华氏温度到摄氏温度对照表的实现：华氏温度(Fahrenheit)和摄氏温度(Centigrade)都是用来计量温度的单位，华氏温度＝摄氏温度×9/5＋32，华氏温标规定在一个大气压下水的冰点为 32 度，沸点为 212 度。程序运行结果如图 3.1 所示。

摄氏、华氏温度对照表

摄氏	华氏	摄氏	华氏	摄氏	华氏	摄氏	华氏	摄氏	华氏
1.0	33.8	2.0	35.6	3.0	37.4	4.0	39.2	5.0	41.0
6.0	42.8	7.0	44.6	8.0	46.4	9.0	48.2	10.0	50.0
11.0	51.8	12.0	53.6	13.0	55.4	14.0	57.2	15.0	59.0
16.0	60.8	17.0	62.6	18.0	64.4	19.0	66.2	20.0	68.0
21.0	69.8	22.0	71.6	23.0	73.4	24.0	75.2	25.0	77.0
26.0	78.8	27.0	80.6	28.0	82.4	29.0	84.2	30.0	86.0
31.0	87.8	32.0	89.6	33.0	91.4	34.0	93.2	35.0	95.0
36.0	96.8	37.0	98.6	38.0	100.4	39.0	102.2	40.0	104.0

图 3.1　摄氏、华氏温度对照表

完整程序代码：

```
1.  public class CelsiusConverterTable {
2.    public static void main(String[] args) {
3.        System.out.println();
4.        System.out.print("\t\t\t");
5.        System.out.println("摄氏、华氏温度对照表");
6.        System.out.print("-----------------------------------------");
7.        System.out.println("----------------------------------");
8.        for(int i=0;i<5;i++){
9.            System.out.print("摄氏"+" "+"华氏"+"\t");
10.       }
11.       System.out.println();
12.       for(int i=1;i<=40;i++){
```

```
13.            double celsius=i;
14.            double fahrenheit=celsius * 9/5+32;
15.            if(i%5==0)
16.            {
17.            System.out.print(celsius+"  "+fahrenheit+"\t");
18.            System.out.println();
19.            }
20.            else
21.            System.out.print(celsius+"  "+fahrenheit+"\t");
22.        }
23.    }
24. }
```

本例输出了摄氏温度值 1～40 对应的华氏温度值。其中，第 3～7 行输出表头；第 8～10 行输出表的列标题；第 12～22 行输出对照表中具体的温度值。在本案例中存在 3 种程序运行流程，即顺序结构、分支结构和循环结构。

在 Java 语言中，顺序结构最简单，按出现的顺序执行代码；分支结构用于实现根据条件选择性地执行某段代码；循环结构则用于根据循环条件重复执行某段代码。Java 提供了 if 和 switch 两种分支语句，并提供了 while、do…while 和 for 共 3 种循环语句，除此之外，JDK1.5 还提供了一种新的循环——for…each 循环，能以更简单的方式遍历集合、数组的元素。同时，Java 还提供了 break 和 continue 来控制程序的循环结构。

数组是大部分编程语言都支持的数据结构，Java 也不例外。Java 的数组类型是一种引用类型的变量，Java 程序通过数组引用变量来操作数组，包括获得数组的长度、访问数组元素的值等。

本章将详细介绍 Java 程序的运行流程及数组的相关知识，包括各种程序运行结构的定义，程序控制语句的运行流程及使用，如何定义、初始化数组等基础知识，并深入介绍数组在内存中的运行机制。

3.2 顺序结构

在任何编程语言中最常见的程序结构就是顺序结构。顺序结构是按照语句出现的顺序依次执行的程序结构，中间没有任何判断和跳转。

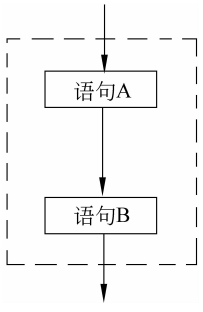

图 3.2 顺序结构程序流程图

如果 main 方法中多行代码之间没有任何流程控制，则程序总是从上向下依次执行，排在前面的代码先执行，排在后面的代码后执行。这意味着如果没有流程控制，Java 方法中的语句是一个顺序执行流，从上向下依次执行每条语句。顺序结构的程序流程图如图 3.2 所示，按照语句 A 和语句 B 的出现顺序从上到下依次执行程序代码。

例如在例 3-1 中，程序代码的第 3～7 行即为顺序结构，负责输出对照表的头部信息，如图 3.3 所示，具体程序

代码如下所示。第 3 行代码用来输出一个空行，主要为了程序运行结果的美观；第 4 行代码输出“摄氏、华氏温度对照表”前的空格，使文字尽量居中显示；第 6、7 行代码输出连续的小横线。

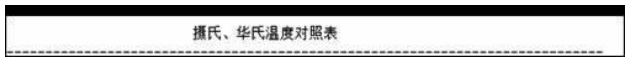

图 3.3　对照表的头部信息效果图

```
1.  public class CelsiusConverterTable {
2.    public static void main(String[] args) {
3.        System.out.println();
4.        System.out.print("\t\t\t");
5.        System.out.println("摄氏、华氏温度对照表");
6.        System.out.print("-----------------------------------------");
7.        System.out.println("---------------------------------");
8.    }
9.  }
```

3.3　分 支 语 句

分支结构也称选择结构，这种结构用来有条件地执行或跳过特定的语句或语句块，实现有选择的流程控制。Java 主要有 3 种分支结构，即单路分支结构、双路分支结构和多路分支结构，这 3 种结构的程序流程图如图 3.4 所示。

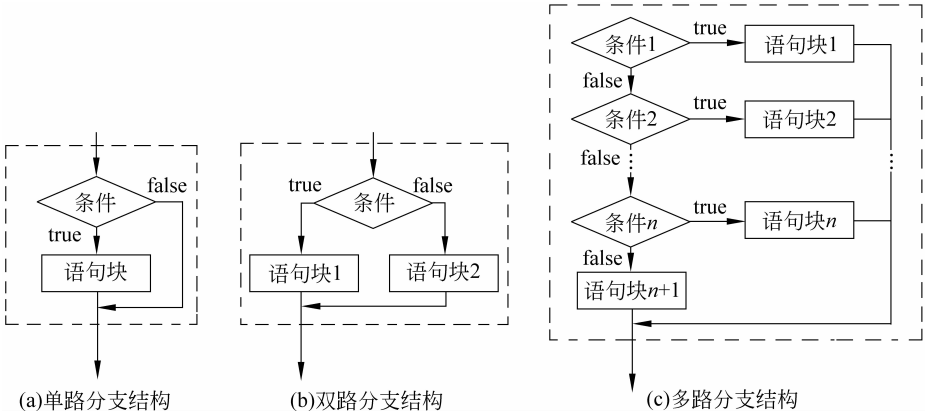

图 3.4　3 种分支结构流程图

条件语句是逻辑选择的核心，也是所有流程控制结构中最基础的控制语句。条件语句主要包括两种重要语句，即 if…else 语句和 switch 语句，使用它们可以实现程序流程的分支控制。下面具体介绍这两种语句。

3.3.1 if…else 语句

if…else 语句用于实现分支结构，其中的 else 子句不是必需的。if…else 语句又可细分为 3 种形式，即单路分支结构、双路分支结构和多路分支结构。

1. 用 if 语句实现单路分支结构

1）语法格式

```
if(<boolean类型表达式>)
    <statement>
```

2）举例

【例 3-2】 利用铁路将行李从甲地托运到乙地，行李不超过 10kg 时，运费是 10 元，如果超过 10kg，超过部分的运费为 2.00 元/kg。设行李重 $x$kg，请编写程序计算运费 a。

这是一个关于运费计算的数学应用题，要解决这个问题，首先要考虑应该如何构建计算公式。根据题意可写出如下计算公式：

$$a=\begin{cases}10 & x\leqslant 10\\ 10+2(x-10) & x>10\end{cases}$$

根据以上计算公式可以构建类 chapter3_2 的程序代码。本程序的设计思想是：第 3 行代码首先定义了一个存储行李托运费用的整型变量 a 并赋初值 10，因为无论行李多重，都要收取 10 元的基本费用；第 4 行代码定义存放行李重量的变量 x，这里将行李重量处理成了整型值，读者可以尝试改为其他合理类型；第 5、6 行代码用 if 语句判断当行李重量超过基本重量时，运用前面得到的公式计算实际的行李托运费用；第 7 行代码将最终的行李托运费用输出。程序运行流程如图 3.5 所示，程序运行结果如图 3.6 所示。

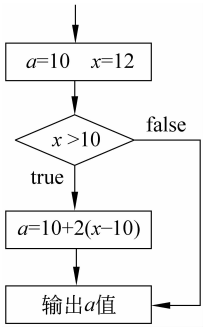

图 3.5 例 3-2 程序的流程图

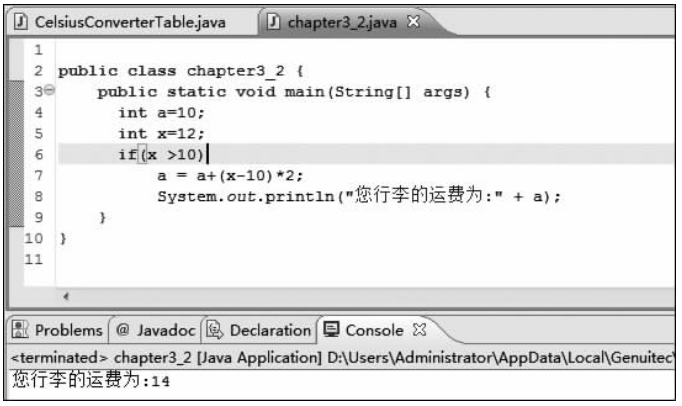

图 3.6 例 3-2 程序的运行结果

完整程序代码：

```
1.  public class chapter3_2 {
2.      public static void main(String[] args) {
```

```
3.          int a=10;
4.          int x=12;
5.          if(x >10)
6.              a=a+ (x-10) * 2;                          //受 if 条件影响,条件成立时执行
7.              System.out.println("您行李的运费为:" +a);     //无条件执行
8.       }
9.   }
```

2. 用 if…else 语句实现双路分支结构

1）语法格式

```
if(<表达式>)
    <statement1>
else
    <statement2>
```

2）应用举例

【例 3-3】 利用铁路将行李从甲地托运到乙地，行李不超过 50kg 时，运费是 1.50 元/kg，如果超过 50kg，超过部分的运费为 2.00 元/kg。设行李重 $x$kg，请编写程序计算运费 a。

例 3-3 仍然是一个关于行李托运费用的应用问题，不同的是行李重量和托运费用之间的关系发生了一些变化。根据题意，可以得出以下计算公式：

$$a=\begin{cases}1.5x & x\leqslant 50\\ 1.5\times 50+2(x-50) & x>50\end{cases}$$

根据以上计算公式可以构建类 chapter3_3 的程序代码。本程序的设计思想是：第 3 行代码首先定义了一个存储行李托运费用的浮点类型变量 a 并赋初值 0；第 4 行代码定义存放行李重量的变量 x 并赋初值 56，这里将行李重量处理成了整型值，读者可以尝试改为其他合理类型；第 5～8 行代码用 if…else 双路分支结构语句判断当前行李重量处于哪个取值范围，并运用前面得到的公式计算实际的行李托运费用；第 9 行代码将最终的行李托运费用输出。程序运行流程如图 3.7 所示，程序运行结果如图 3.8 所示。

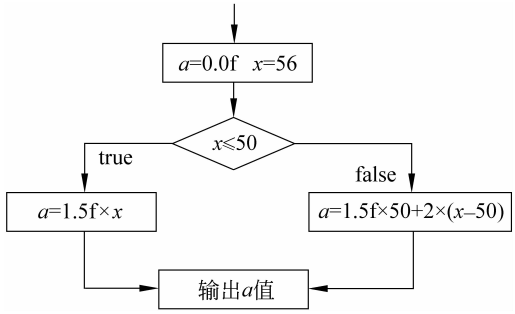

图 3.7　例 3-3 程序的流程图

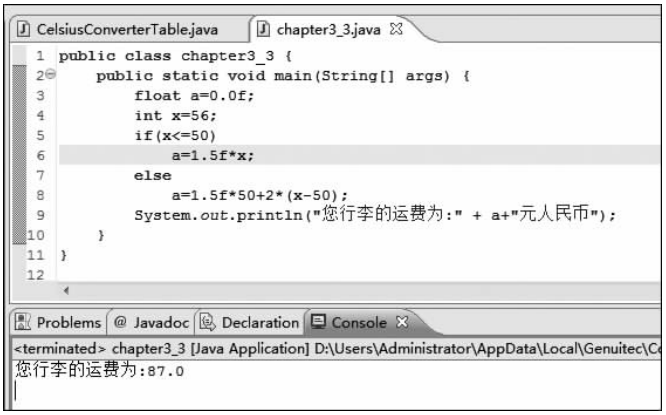

图 3.8　例 3-3 程序的运行结果

完整程序代码：

```
1.  public class chapter3_3 {
2.    public static void main(String[] args) {
3.        float a=0.0f;
4.        int x=56;
5.        if(x<=50)
6.            a=1.5f*x;
7.        else
8.            a=1.5f*50+2*(x-50);
9.        System.out.println("您行李的运费为:"+a+"元人民币");
10.   }
11. }
```

3）案例分析

本章开头案例摄氏、华氏温度对照表在具体实现过程中用到了 if…else 语句，程序代码从类 CelsiusConverterTable 的第 16 行代码开始到第 22 行代码结束，具体代码如图 3.9 所示。if…else 语句在本部分程序中的功能是控制每行输出的摄氏温度和华氏温度值的对数，具体地用第 16 行代码 $i\%5==0$ 控制每行输出 5 对摄氏温度和华氏温度值。

```
if(i%5==0)
{
System.out.print(celsius+"    "+fahrenheit+"\t");
System.out.println();
}
else
System.out.print(celsius+"    "+fahrenheit+"\t");
```

图 3.9　if…else 语句在例 3-1 中的使用截图

3. 用 if…else 语句实现多路分支结构

1）语法格式

```
if(<表达式 1>)
<语句 1>
```

```
else if(<表达式 2>)
<语句 2>
  …
else if(<表达式 n>)
<语句 n>
[else
<语句 n+1>]
```

该语法格式实际上是对基本 if…else 语句的一个嵌套，在基本语法格式的 statement1 和 statement2 处再次添加 if…else 语句。

2）例题

【例 3-4】 按照表 3.1 所示的标准，根据成绩所属范围给出相应的级别。

表 3.1　成绩评定标准

成　绩	60≤score＜75	75≤score＜85	85≤score＜＝100
级别	中	良	优

本例如果用简单 if 语句实现，思路如下：

```
if(score<60 || score>100)
    不参加评级;
if(score>=60 && score<75)
    等级为中;
if(score>=75 && score<85)
    等级为良;
if(score>=85 && score<=100)
    等级为优;
```

可以看出，当成绩值在不同的取值范围时等级是不同的，可以考虑将成绩划分为几个互斥的范围，这样就可以使用 if…else 的嵌套解决问题，具体思路如图 3.10 所示。

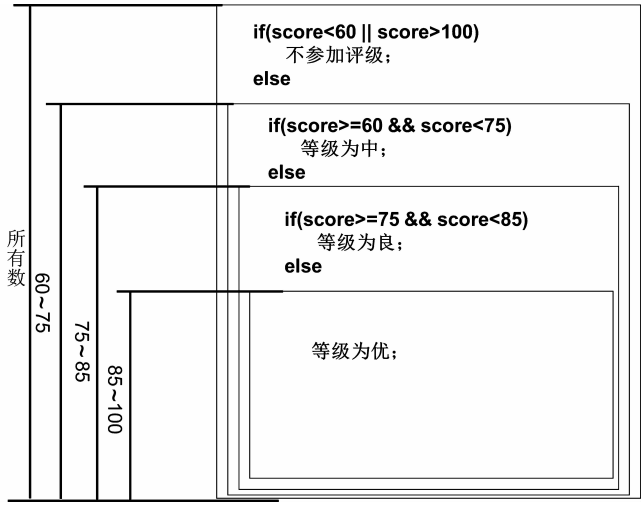

图 3.10　成绩等级划分示意图

通过以上分析，可以构造类 chapter3_4 所示的程序代码。第 3 行代码定义了一个待判断等级的成绩值 78；第 4、5 行代码处理位于(－∞，60)和(100，＋∞)范围内的成绩值；第 6 行代码“否定”的是第 4 行代码的范围，即以下处理的成绩值都位于［60，100］的范围内；第 7、8 行代码则具体处理［60，75)这一范围内的成绩值，依此类推。程序流程图和程序运行结果分别如图 3.11 和图 3.12 所示。

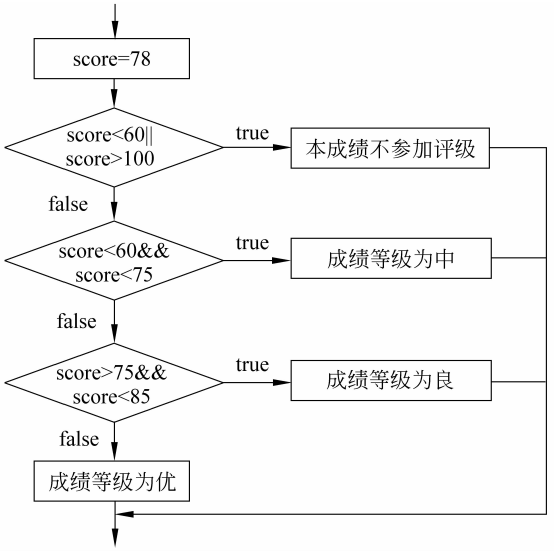

图 3.11　类 chapter3_4 的程序流程图

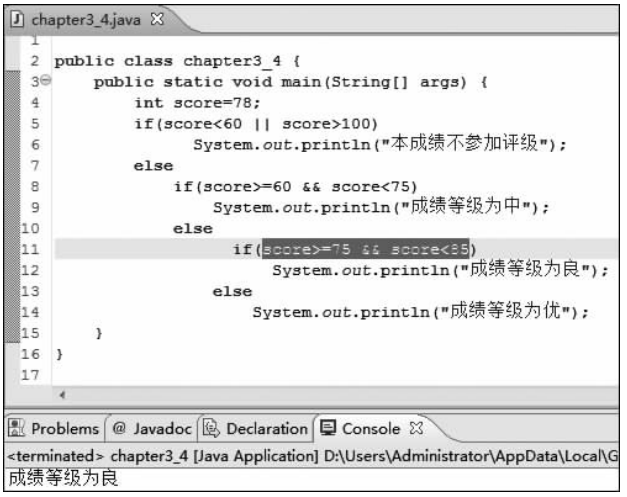

图 3.12　类 chapter3_4 的程序运行结果

完整程序代码：

```
1.  public class chapter3_4 {
2.    public static void main(String;] args) {
3.        int score=78;
```

```
4.          if(score<60 || score>100)
5.              System.out.println("本成绩不参加评级");
6.          else
7.              if(score>=60 && score<75)
8.                  System.out.println("成绩等级为中");
9.              else
10.                 if(score>=75 && score<85)
11.                     System.out.println("成绩等级为良");
12.                 else
13.                     System.out.println("成绩等级为优");
14.     }
15. }
```

3）说明

（1）并非每个嵌套都需要使用完整的 if…else 语句，可以只使用单个 if 语句来实现嵌套。例如：

```
System.out.println("请输入一个学生成绩");
Scanner sc=new Scanner(System.in);
int score=sc.nextInt();
if(Score>=60)
{ if(Score<=100)
    System.out.println("该学生成绩及格了!"); }
  else
    System.out.println("该学生成绩不及格!");
```

（2）if…else 配对原则：省略{ }时，else 总是和它上面最近的未配对的 if 配对。例如：

```
if(…)
    if(…)
        if(…)
        else…
    else…
else…
```

3.3.2　switch 语句

【例 3-5】 按照表 3.2 所示的标准，根据成绩所属范围给出相应的级别。

表 3.2　成绩评定标准

分数	0～59	60～69	70～79	80～89	90～100	＞100
级别	E	D	C	B	A	错误

例 3-5 和例 3-4 类似，属于多路分支的成绩评定问题。通过对 if 语句的学习，我们知道这类问题可以使用 if…else 的语句嵌套实现，但在分支过多的情况下，使用 if…else 嵌

套不仅代码的可读性不高，而且执行效率偏低。在这种情况下，使用 switch 语句可以解决 if…else 的弊端。

1. 语法格式

```
switch(expression)
  {
    case case1:
        statement1;
        break;
    case case2:
        statement2;
        break;
            ⋮
    case caseN:
        statementN;
        break;
    default:
        default statement;
        break;
}
```

在程序执行 switch 语句时会自动检测 expression 并且与 case 后面的 case1、case2、…、case*N* 按顺序依次进行比较。如果与 case*i*（其中 $i=1,2,\cdots,N$）相等，则从 statement*i* 开始执行；如果全都不符合，则从 default statement 开始执行。该语句的程序流程图如图 3.13 所示。

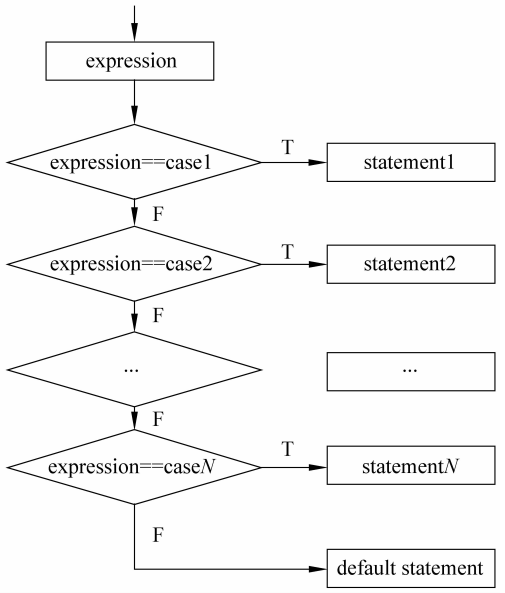

图 3.13　switch 语句的执行流程图

下面有几点说明：

- switch 后的＜表达式＞必须是 int、byte、char、short，枚举类型和封装类类型。
- case 子句中的常量值不得重复。
- 执行完一个 case 后面的语句后，程序将转移到下一个 case 继续执行。
- 鉴于此，多个 case 可以共用一组执行语句。

例如：switch(level)

```
case 'A':
case 'B':
case 'C':
System. out. println("score＞60\n");
break;
```

- default 子句是可选的。
- 各个 case 及 default 出现的次序不影响执行结果。
- break 语句用来在执行完一个 case 分支后使程序跳出 switch 语句块。

2. switch 语句应用

1）switch 语句举例 1

根据输入字母输出字符串。

```
public class SwitchEx1{
public static void main(String args[]){
    int c;
    System.out.println("输入 m 或 n 或 h 或其他:");
    c=scanner.nextChar();
    switch(c)
     {  case 'm': System.out.println("Good morning!");break;
        case 'n': System.out.println("Good night!"); break;
        case 'h': System.out.println("Hello!"); break;
        default: System.out.println("????????");}}}
```

2）switch 语句举例 2

根据整数值输出不同的信息。

```
public class SwitchEx2{
    public static void main(String args[]){
        int num=3;
        switch(num)
         {      case    5: System.out.println("Very good!"); break;
                case    4: System.out.println("Good!"); break;
                case    3: System.out.println("Pass!"); break;
                case    2: System.out.println("Fail!"); break;
                default  : System.out.println("data error!"); }
    }
}
```

3）用 switch 语句解决例 3-5 问题

通过以上对 switch 语句的学习和使用，可以构建类 chapter3_5 来解决例 3-5 提出的问题。

完整程序代码：

```
1.  public class chapter3_5 {
2.    public static void main(String[] args) {
3.        int score=78;
4.        int dec=score/10;
5.        switch(dec)
6.        {
7.        case 10:
8.        case 9:
9.            System.out.println("该学生成绩对应等级为 A");
10.           break;
11.       case 8:
12.           System.out.println("该学生成绩对应等级为 B");
13.           break;
14.       case 7:
15.           System.out.println("该学生成绩对应等级为 C");
16.           break;
17.       case 6:
18.           System.out.println("该学生成绩对应等级为 D");
19.           break;
20.       default:
21.           System.out.println("该学生成绩对应等级为 E");
22.       }
23.   }
24. }
```

第 3 行代码定义了一个存放成绩值的整型变量 score 并赋初值 78；第 4 行代码定义了一个整型变量 dec 存放成绩值整除数字 10 的商，因为 score、dec 与成绩等级之间有如表 3.3 所示的关系，这种关系为程序的编写带来方便；第 5～22 行代码利用 switch 语句判断整数位 dec 的值，根据结果从 6～10 输出不同的成绩等级，如果与 6～10 都不对应，则执行 default 后面的语句。程序运行结果如图 3.14 所示。

表 3.3　score、dec 与成绩等级之间的关系

成绩值 score		商 dec	成绩等级
0～9	被 10 除	0	E
10～19		1	E
20～29		2	E
30～39		3	E
40～49		4	E
50～59		5	E

续表

成绩值 score		商 dec	成绩等级
60～69		6	D
70～79		7	C
80～89	被 10 除	8	B
90～99		9	A
100		10	A
＞100		其他值	不考虑

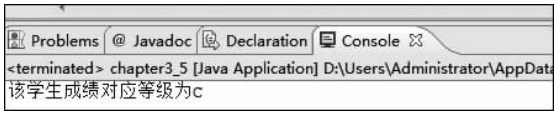

图 3.14　例 3-5 程序的运行结果

3.4　循 环 语 句

循环结构是在一定的条件下重复执行特定代码的程序结构。使用循环结构能够多次执行同一个任务，直到完成另一个比较大的任务。循环结构可以减少源程序重复书写的工作量，用来描述重复执行某段算法的问题，这是程序设计中最能发挥计算机特长的程序结构。Java 提供了多种循环语句用来实现重复性的任务，包括 for 循环、while 循环、do…while 循环和 for…each 循环。

3.4.1　for 循环

1. 语法格式

```
for(<初始化表达式>; <判断表达式>; <递增(递减)表达式>)
    <语句>
```

说明：

- 初始化表达式：初始化表达式的意义在于定义循环之前变量的值是多少，如果没有这一项，则不知道该从哪个值开始循环。
- 判断表达式：判断表达式的作用在于规定循环的终点，如果没有判断表达式，那么此循环就成了死循环。
- 递增(递减)表达式：这一项规定每执行一次程序，变量以多少增量或减量进行变化。
- 语句：需要重复执行的程序，也可以称为循环体。

2. 运行流程

语句是反复执行的部分，即循环体，循环只是语句进行循环，后面的语句都是循环体

外的内容。for 循环语句的执行过程为：首先执行初始化表达式，然后执行判断表达式，若它的值为 true，则执行循环语句，接着执行递增（递减）表达式，最后再次回到判断表达式进行逻辑判断，直到其值为 false，结束循环。其具体循环过程如图 3.15 所示。

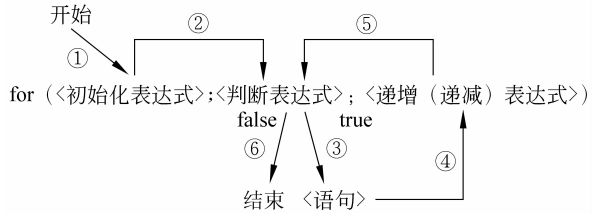

图 3.15　for 循环的执行过程

3. 应用举例 1

【例 3-6】 打印 5 行*****。

完整程序代码：

```
1. public class chapter3_6 {
2.     public static void main(String;] args) {
3.         for(int i=1;i<=5;i=i+1)
4.             System.out.println("*****");
5.     }
6. }
```

本例用 for 循环打印输出*****，在初值表达式中将变量 i 赋予初始值 1，判断条件表达式限定 i 的最大值为 5，增量表达式使变量 i 不断增加 1，可控制循环执行 5 次，每执行一次循环语句输出一行*****。例 3-6 程序的执行流程图和运行结果分别如图 3.16 和图 3.17 所示。

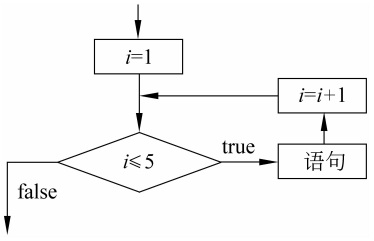

图 3.16　例 3-6 程序的执行流程图

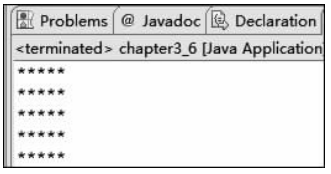

图 3.17　例 3-6 程序的运行结果

4. 等价形式

例 3-6 中使用的是 for 循环的标准形式，从标准形式中还可以变换出以下 3 种等价形式。

1）初值表达式为空

当初值表达式为空时，for 循环可以采用如下形式实现：

等价形式 1：

```
初值表达式;
for(;判断表达式;增量(减量)表达式)
    <语句>
```

根据以上变化形式,例 3-6 的 for 循环可以变化如下:

```
int i=1;
for(;i<=5;i++)
System.out.println("*****");
```

2) 判断表达式为空

```
等价形式 2:
for(初值表达式;; 增量(减量)表达式) {
    <语句>
    if(判断表达式的反命题)
    break;       //用于退出 for 循环   }
```

根据以上变化形式,例 3-6 的 for 循环可以变化如下:

```
for(int i=1;;i++)
  {System.out.println("*****");
    if(i>5)
      break;}
```

3) 增量(减量)表达式为空

```
等价形式 3:
for(初值表达式;判断表达式;;) {
    <语句>
    增量(减量)表达式;  }
```

根据以上变化形式,例 3-6 的 for 循环可以变化如下:

```
for(int i=1;i<=5;;) {
    System.out.println("*****");
      i=i+1;    }
```

5. 应用举例 2

【例 3-7】 求 1+2+3+4+…+100 的值。

本例求 100 以内整数的总和,可以考虑先定义一个存储总和的变量并赋初值 0,然后依次向其中加入 1～100 共 100 个整数,这是一个固定次数循环问题,可以用 for 循环实现。

完整程序代码:

```
1.  public class chapter3_7 {
2.
3.    public static void main(String[] args) {
```

```
4.          int sum=0;
5.          for(int i=0;i<=100;i++)
6.              sum=sum+i;
7.          System.out.println("1+2+3+4+…+100="+sum);
8.      }
9. }
```

在类 chapter3_7 程序中，第 3 行代码定义了一个存放总和的整型变量 sum 并赋初值 0 作为被加数，第 4、5 行利用一个 100 次的循环依次将 $i(i=1,2,\cdots,100)$ 的值作为加数与 sum 相加，第 6 行输出最终结果。程序运行结果如图 3.18 所示。

for 循环语句不仅可以灵活地调整表达式部分，还可以没有循环体，这是因为空语句在 Java 语法中是合法的。没有循环体的循环在某些情况下也是很有用的。例如，要完成例 3-7 的累加计算也可以使用下面的代码：

```
1. public class chapter3_7_1 {
2.      public static void main(String[] args) {
3.          int sum=0;
4.          for(int i=0;i<=100;sum+=i++)
5.              ;
6.          System.out.println("1+2+3+4+…+100="+sum);
7.      }
8. }
```

在类 chapter3_7_1 程序代码中，for 循环语句后直接使用分号表示该语句结束，而 for 的循环表达式 sum+=i++ 的计算顺序是首先计算 sum+=i，然后计算 i++。编译并运行类 chapter3_7_1，运行结果如图 3.19 所示。

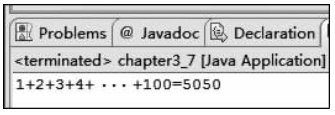

图 3.18　例 3-7 程序的运行结果

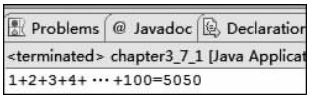

图 3.19　类 chapter3_7_1 程序的运行结果

for 循环使用灵活，它可以像 if…else 语句一样进行嵌套使用。嵌套使用的 for 循环语句可以完成更为复杂的任务。

6. 案例分析

在例 3-1 输出摄氏、华氏温度对照表程序中两次用到了 for 循环，第一次为第 9～11 行，这是一个 5 次的固定次数循环，循环体只有一个输出语句，不再赘述。在这里主要解析 for 循环在本案例中的第二次应用，具体代码如图 3.20 所示。

第 13 行代码是 for 循环开始，循环控制变量为 i，初始值为 1，最大值为 40，每次循环增加 1；第 14 行代码定义了一个 double 类型变量 celsius 用来存放摄氏温度值，第 15 行代码利用公式将摄氏温度值 celsius 转化为华氏温度值并存储在 double 类型变量 fahrenheit 中；第 16～22 行使用 if…else 语句控制对照表的具体输出。

```
13          for(int i=1;i<=40;i++){
14              double celsius=i;
15              double fahrenheit=celsius*9/5+32;
16              if(i%5==0)
17              {
18              System.out.print(celsius+"     "+fahrenheit+"\t");
19              System.out.println();
20              }
21              else
22              System.out.print(celsius+"     "+fahrenheit+"\t");
23          }
```

图 3.20　for 循环在例 3-1 中的第二次应用

3.4.2　while 循环

while 结构循环为当型循环(when type loop),是循环结构的一种,一般用于不知道循环次数的情况。维持循环的是一个条件表达式,若条件成立执行循环体,若条件不成立退出循环。

1. 语法格式

```
while(循环条件表达式)
    {
        循环体
    }
```

其中,循环条件表达式是循环能否继续下去的条件;循环体是需要反复执行的语句。当 while 循环体有且只有一个语句时,可以将大括号省略。while 循环控制流程图如图 3.21 所示。

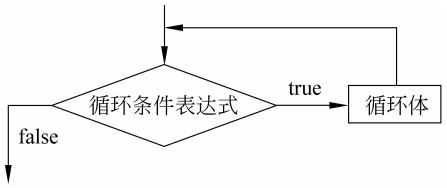

图 3.21　while 循环控制流程图

while 循环的流程:

(1) 在第一次进入 while 循环前,必须为循环控制变量(或表达式)赋初值。

(2) 根据循环条件表达式决定是否继续执行循环,如果条件判断值为真,继续执行循环体语句;否则跳出循环执行其他语句。

(3) 执行完循环体内的语句后,重新为循环控制变量(或表达式)赋值(增加或减少),由于 while 循环不会自行更改循环控制变量(或表达式)的内容,所以 while 循环中为循环控制变量赋值的工作要由设计者自己来做,完成后再回到步骤(2)重新判断是否继续执行循环。

2. 应用举例

【例 3-8】 输出 5 行********。

完整程序代码:

```
1. public class chapter3_8 {
2.     public static void main(String[] args) {
3.         int i=1;
```

```
4.          while(i<=5)
5.          {
6.          System.out.println("* * * * * * * * ");
7.          i=i+1;
8.          }
9.      }
10. }
```

在类 chapter3_8 程序中，第 3 行代码首先定义一个整型变量 i 作为循环控制变量，初始值为 1；第 4～8 行代码为 while 循环结构，其中第 4 行代码中的循环条件表达式“i<=5”限定了变量 i 的最大值为 5，即当 i 超过 5 时结束循环；第 6、7 行代码为循环体语句，每执行一次，输出一行********，并将 i 的值增加 1，以使 i 的值不断接近最大值 5。若省略第 7 行代码，则 i 的值始终为 1，将造成死循环。程序运行结果如图 3.22 所示。

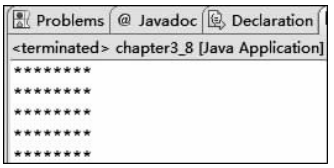

图 3.22　例 3-8 程序的运行结果

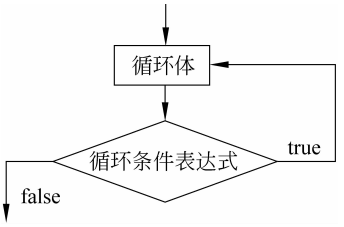

图 3.23　do…while 循环的控制流程图

3.4.3　do…while 循环

do…while 循环是 Java 语言中的一种循环控制语句，主要由一个代码块（作为循环体）和一个表达式（作为循环条件）组成。一般情况下，do…while 循环与 while 循环相似。

1. 语法格式

```
do
    循环体
while(<循环条件表达式>);
```

其中，表达式为布尔（boolean）型，循环体内的代码执行一次后程序会去判断循环条件表达式的返回值，如果返回值为 true（即满足循环条件），则循环体内的代码会反复执行，直到表达式的返回值为 false（即不满足循环条件）时终止。程序会在每次循环体执行一次后进行一次表达式的判断。do…while 循环的程序控制流程图如图 3.23 所示。

2. 应用举例

【例 3-9】 输出 5 以下的整数值。

```
1. public class chapter3_9 {
2.     public static void main(String[] args) {
3.         int i=1;
4.         do
```

```
5.        {
6.            System.out.println(i);
7.            i=i+1;
8.        }while(i<=5);
9.    }
10. }
```

在类 chapter3_9 程序中，第 3 行代码首先定义一个整型变量 i 作为循环控制变量，初始值为 1；第 4～8 行代码为 do…while 循环结构，其中第 8 行代码中的循环条件表达式“i＜＝5”限定了变量 i 的最大值为 5，即当 i 超过 5 时结束循环；第 6、7 行代码为循环体语句，每执行一次输出一个整数值 i（i 依次为 1、2、3、4、5），并将 i 的值增加 1，以使 i 的值不断接近最大值 5。程序运行结果如图 3.24 所示。

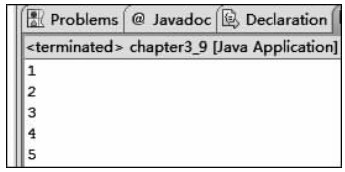

图 3.24　例 3-9 程序的运行结果

3. while 和 do…while 的区别

do…while 循环与 while 循环很相似，两者唯一的区别是：do…while 循环将先执行一次循环体内的代码，再去判断循环条件。所以无论循环条件是否满足，do…while 循环体内的代码至少会执行一次。因此，do…while 循环属于后测循环（post-test loop）。

3.4.4　for…each 循环

for…each 循环是 JDK5.0 新增加的一个循环结构，可以用来逐个处理（遍历）数组或集合中的所有元素。for…each 循环使得代码更加简短，也让代码更加易懂。

1. 语法格式

```
for(<迭代变量声明>: <数组或集合>)
    <语句>
```

其中，迭代变量声明是指在使用 for…each 循环语句时首先要定义一个变量用于暂存集合中的每一个元素，显然其数据类型要与后面的数组或集合中的元素类型一致。程序执行时，依次将数组或集合中的元素赋给迭代变量，然后执行循环语句，直到数组或集合中的所有元素读取完毕。

2. 应用举例

【例 3-10】 输出数组 a 的值，其中 a＝{3,5,67,98,56}。

完整程序代码：

```
1. public class chapter3_10 {
2.     public static void main(String[] args) {
3.         int[] a={3,5,67,98,56};
4.         for(int k: a){
5.         System.out.println(k);
6.         }
```

```
7.     }
8. }
```

在类 chapter3_10 程序中，第 3 行代码首先定义并初始化了一个数组 a；第 4～6 行代码为 for…each 循环结构，其中第 4 行代码定义了一个迭代变量 k 用于存放数组 a 中的每一个元素，第 5 行代码输出迭代变量 k。程序运行结果如图 3.25 所示。

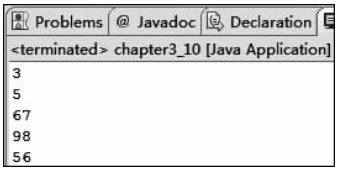

图 3.25　例 3-10 程序的运行结果

3.5　控 制 语 句

在循环程序设计中，可能会有一些特殊情况需要中断整个循环或某些循环，Java 中提供了 break 语句和 continue 语句来完成程序设计者的这一要求。

1. break 语句

break 语句用于终止所在 switch 块或循环语句的运行。当程序执行到 break 语句时就会离开循环，继续执行循环外的下一个语句，如果 break 语句出现在嵌套循环中的内层循环，则 break 语句只会跳出当前层的循环。

break 语句用在 switch 语句和循环语句中，break 语句的格式如下：

```
break;
```

【例 3-11】 用 break 语句中断 for 循环的执行。

完整程序代码：

```
1. public class chapter3_11 {
2.     public static void main(String[] args) {
3.         for(int i=0; i<10; i++){
4.             if(i==3)
5.                 break;
6.             System.out.println(" i=" +i);
7.             }
8.             System.out.println("Game Over!");
9.     }
10. }
```

在类 chapter3_11 程序中，第 3～7 行代码是一个 for 循环结构，第 3 行代码用循环变量 i 构造了一个 10 次的循环，但第 5、6 行代码用一个 if 语句和 break 语句使程序的实际循环次数只有 3 次。当程序运行时，首先会判断 i 值是否等于 3，如果 i 不等于 3，则将当