第3章　流程控制

一般来说，程序的执行通常是按照语句书写的顺序从前往后一条一条执行的，流程控制语句可以用来改变程序执行的顺序。程序利用流程控制语句有条件地执行语句、循环性地执行语句，或者跳转性地执行语句，从而实现一些复杂的算法。Java 语言的流程控制语句分为分支语句、循环语句和跳转语句。

3.1　分支语句

分支结构是根据布尔条件表达式的值采取不同的动作，应用在程序中能够使程序更灵活。在 Java 语言中使用的分支语句有 if…else 语句和 switch 语句。

3.1.1　if…else 语句

if…else 语句可以使程序根据条件执行不同的语句。if…else 语句包括单独的 if 语句、if…else 语句和嵌套的 if 语句。

1. if 语句

if 语句的语法形式如下：

```
if (表达式){
    语句块;
}
```

这是最简单的 if 语句，为单分支结构。当表达式的值为 true 时，执行语句块，否则忽略语句块，执行其后的语句，如图 3.1 所示。

2. if…else 语句

if…else 语句的语法形式如下：

```
if (表达式) {
    语句块 1;
}
else {
    语句块 2;
}
```

当 if 表达式为 true 时，程序执行语句块 1，执行完后跳出 if 语句；如果表达式为 false，程序跳过语句块 1，执行 else 后的语句块 2，执行完后结束 if 语句，如图 3.2 所示。

例 3.1　if…else 应用实例，根据成绩判断并输出是否及格(ch03\IfDemo.java)。

```
public class IfDemo  {
    public static void main(String[] args)  {
```

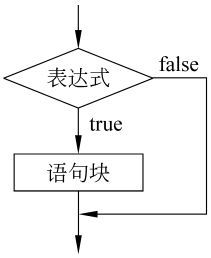

图 3.1　if 语句

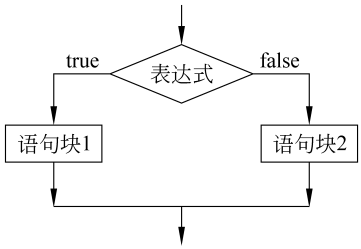

图 3.2　if…else 语句

```
        int score=71;
        if (score>=60)
            System.out.println("成绩及格!");
        else
            System.out.println("成绩不及格");
    }
}
```

运行结果：

```
成绩及格!
```

3. 嵌套的 if 语句

对于超过二分支情况的选择，可以使用嵌套的 if…else 语句。语法形式如下：

```
if (表达式 1)  { 语句块 1; }
else if (表达式 2)  {语句块 2; }
    else if (表达式 3)  {语句块 3; }
    …
    else if (表达式 m)  {语句块 m; }
    else  {语句块 m+1; }
```

这种嵌套的 if 语句，将会依次计算表达式的值，如果某个表达式的值为 true，则执行其后的语句块，执行完后跳出 if 语句；若所有表达式都为 false，则执行最后一个 else 后的语句块，如图 3.3 所示。

例 3.2　嵌套的 if 语句的应用，根据成绩判断并输出所在级别（ch03\IfElseDemo.java）。

```
public class IfElseDemo  {
public static void main(String[] args)  {
    int score=83;
    if (score<60) { System.out.println("成绩为不及格!"); }
    else if (score<70)  { System.out.println("成绩为及格!"); }
    else if (score<80)  { System.out.println("成绩为中!"); }
    else if (score<90)  { System.out.println("成绩为良!"); }
    else if (score<100)  { System.out.println("成绩为优!"); }
    else if (score==100)  { System.out.println("成绩为满分!"); }
    else
```

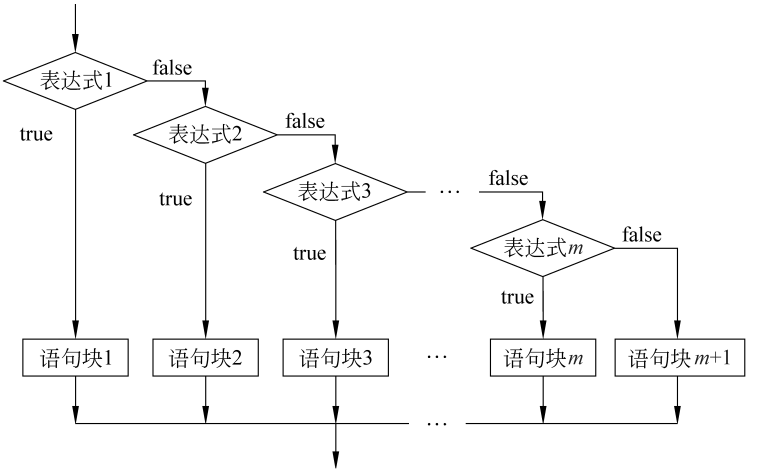

图 3.3　嵌套的 if 语句

```
            System.out.println("Error!");
        }
    }
```

运行结果：

```
成绩为良!
```

3.1.2　switch 语句

虽然嵌套的 if 语句可以实现多分支处理，但是语句较为复杂，并且容易混乱和出错，因此可以使用 switch 语句来实现多分支情况的处理。switch 语句也叫开关语句，当条件有很多选项时，switch 语句可以很容易写出判断条件，使得编写的程序结构较为清晰。

switch 的语法形式如下：

```
switch(表达式) {
    case  常量值 1: 语句块 1; [ break; ]
    case  常量值 2: 语句块 2; [ break; ]
    case  常量值 3: 语句块 3; [ break; ]
    …
    case  常量值 m: 语句块 m; [ break; ]
    [default: 语句块 m+1;[break;]]
  }
```

其中，switch 表达式的类型可以是整型(byte、short、int、long)、字符类型、枚举类型，在 JDK 1.7 版本中又新增了表达式的类型可以是 String 类型。各 case 后面的常量值必须与 switch 后面的表达式类型一致，而且每个 case 后面的值不能有重复；switch 语句中的 break 语句为可选项，根据需要进行设置。switch 语句的执行顺序是：首先计算表达式的值，如果表达式的值和某个 case 后的常量值相同，就执行该 case 后的语句块，直到遇到 break 语句结束整个 switch 语句，如果该 case 后没有设置 break 语句，则程序将继续执行下面的各 case 语

句里的语句块，直到遇到 break 语句跳出 switch 语句。若没有一个 case 的常量值与表达式的值相同，则执行 default 后面的语句块。default 语句是可选项，根据需要选用。如果程序没有设置 default，则当没有一个 case 的常量值与表达式的值相同时，那么 switch 语句将不做任何处理。switch 语句可以嵌套使用，即在一个 case 语句块中可以使用另一个 switch 语句结构，对另一个表达式的值进行判断。

例 3.3 switch 分支结构的应用，根据数值判断并输出为星期几（ch03\SwitchDemo.java）。

```
public class SwitchDemo {
    public static void main(String[] args) {
    int day=6;
        switch(day){
            case 1:
                System.out.println("Monday");
                break;
            case 2:
                System.out.println("Tuesday");
                 break;
             case 3:
                 System.out.println("Wednesday");
                 break;
             case 4:
                 System.out.println("Thursday");
                 break;
             case 5:
                 System.out.println("Friday");
                 break;
             case 6:
                 System.out.println("Saturday");
                 break;
             case 7:
                 System.out.println("Sunday");
                 break;
             default:
                 System.out.println("Error!");
         }
    }
}
```

运行结果：

```
Saturday
```

在 JDK 1.7 之前的版本中，switch 表达式的类型不能是 String 类型，而在 JDK 1.7 中新

增了 switch 表达式可以是 String 类型这一新特性，如例 3.4 所示。

例 3.4 从控制台输入星期几，输出该日处于一周的哪个阶段（ch03\SwitchStringDemo.java）。

```
public class SwitchStringDemo  {
    public static void main(String[] args)  {
        String typeOfDay;
        switch(args[0]) {
            case "Monday":
                typeOfDay="Start of work week";
                break;
            case "Tuesday":
            case "Wednesday":
            case "Thursday":
                typeOfDay="Midweek";
                break;
            case "Friday":
                typeOfDay="End of work week";
                break;
            case "Saturday":
            case "Sunday":
                typeOfDay="Weekend";
                break;
            default:
                typeOfDay="Invalid day of the week: "+args[0];
        }
        System.out.println(typeOfDay);
    }
}
```

以上例子可以使用“case L－＞”形式来简化，不再需要 break 语句来中断分支，default 也是可选的。如例 3.3 可以改为如下代码片段：

```
int day=6;
switch(day){
    case 1->System.out.println("Monday");
    case 2->System.out.println("Tuesday");
    case 3->System.out.println("Wednesday");
    case 4->System.out.println("Thursday");
    case 5->System.out.println("Friday");
    case 6->System.out.println("Saturday");
    case 7->System.out.println("Sunday");
    default->System.out.println("Error!");
}
```

代码中可以同时判断多个条件，如下为判断某个月有多少天的代码片段（其中 out 为静态导入的 System.out）：

```
int month=2;
switch(month) {
    case 1, 3, 5, 7, 8, 10, 12-> {
        out.println(31);
        out.println("多行代码");
    }
    case 2->out.println(28);
    case 4, 6, 9, 11->out.println(30);
    default->throw new IllegalArgumentException("没有这个月份:" +month);
}
```

3.1.3 switch 表达式

switch 可以用作表达式计算值，形式如下：

```
case label_1, label_2, …, label_n ->表达式;|throw 语句;|代码块
```

注意：

- 不需要 break 中断分支。
- 箭头右边如果是一个表达式，则该表达式的值为该分支的值。
- 必须要有 default，除非为 enum 类型且列举了所有值。
- 代码块中可以用 yield 为该分支指定值。

如下为计算某个月有多少天的代码片段：

```
int month=2;
int days=2 * switch(month) {
    case 1, 3, 5, 7, 8, 10, 12->31;
    case 2->28;
    case 4, 6, 9, 11->30;
    default->throw new IllegalArgumentException("没有这个月份 " +month);
};
out.println(days);
```

如果箭头右边为代码块，可以用 yield 来返回该分支的值，代码如下：

```
int month=3;
int days=2 * switch(month) {
    case 1, 3, 5, 7, 8, 10, 12->{
        out.println("仅做示例");
        yield 31;
    }
    case 2->28;
    case 4, 6, 9, 11->30;
    default->throw new IllegalArgumentException("没有这个月份:" +month);
```

```
};
out.println(days);
```

用在 enum 类型的例子见第 7 章枚举相关内容。

3.2 循环语句

循环结构可以根据一定的条件使某一段程序能够重复执行多次，直到满足终止条件为止。Java 提供了 3 种形式的循环语句：while 语句、do…while 语句和 for 语句。

3.2.1 while 语句

while 语句的语法形式如下：

```
while(表达式){
    循环体
}
```

while 语句首先计算表达式的值，它将返回一个布尔值 true 或 false。如果值为 true，执行循环体的语句，然后计算表达式的值，判断是否执行循环体，直到表达式的值为 false，跳出 while 循环，执行其后的程序，如图 3.4 所示。

例 3.5 应用 while 循环语句，求 1＋2＋3＋…＋100 的值(ch03\WhileDemo.java)。

```
public class WhileDemo  {
    public static void main(String[] args)  {
        int i, sum=0;
        i=1;
        while (i<=100) {
            sum+=i++;
        }
        System.out.println("1~100 的和为"+sum);
    }
}
```

运行结果：

```
1~100 的和为 5050
```

使用 while 循环语句时应注意，在循环体内要有使循环趋于结束的语句，即每执行一次循环，循环条件要发生相应的变化，使得表达式的值能够最终为 false，否则将出现死循环的情况，这在程序设计应尽量避免。

3.2.2 do…while 语句

do…while 循环与 while 循环不同，do…while 循环先执行循环体中的语句，然后计算表达式的值，若值为 true，则继续执行下一轮循环，当表达式的值为 false 时，跳出循环，如图 3.5 所示。

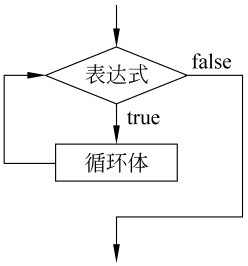

图 3.4　while 循环语句

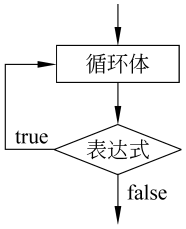

图 3.5　do…while 循环语句

do…while 循环的语法形式如下：

```
do  {
    循环体
}  while(表达式);
```

例 3.6　使用 do…while 循环语句计算 1＋2＋3＋…＋100 的值(ch03\DoWhileDemo.java)。

```
public class DoWhileDemo  {
    public static void main(String[] args)  {
        int i, sum=0;
        i=1;
        do {
            sum+=i++;
        } while (i<=100);
        System.out.println("1~100 的和为"+sum);
    }
}
```

运行结果：

```
1~100 的和为 5050
```

由此可见，while 循环与 do…while 循环可以实现同一个程序功能，但是它们也有区别，前者是先判断后执行，后者是先执行后判断。do…while 循环可以保证循环至少执行一次，而 while 循环有可能一次都不执行，这是 do…while 循环与 while 循环最大的区别。

3.2.3　for 语句

for 语句提供了一种更为灵活的循环方式，可以使程序按照指定的次数进行循环。其语法形式如下：

```
for (初始表达式; 逻辑表达式; 增量表达式)  {
    循环体;
  }
```

for 循环语句执行时，首先执行初始表达式，然后计算逻辑表达式来判断是否终止，当逻辑表达式的值为 true 时，执行循环体中的语句，然后执行增量表达式。完成一次循环后，重

新判断终止条件，直到终止表达式的值为 false 时，跳出整个 for 循环，如图 3.6 所示。

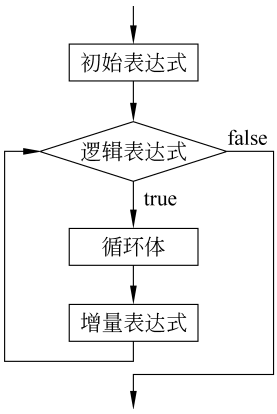

图 3.6　for 循环语句

因此，上述程序中计算 1～100 的和的循环可设计如下：

```
int i, sum=0;
for (i=1;i<=100;i++) {
    sum+=i;
}
System.out.println("1 到 100 的和为"+sum);
```

for 循环语句通常用来执行循环次数确定的情况。初始表达式在循环开始的时候被执行一次；逻辑表达式决定什么时候终止循环，该表达式在每次循环的过程中都被执行一次；增量表达式决定了循环一次增加的步长值。以上三部分都可以按照情况设置为空语句(但“;”不能省)，三者均为空的时候，相当于一个无限循环，在程序设计中应避免出现无限循环的情况。

例 3.7　应用 for 语句，对数组中的数据进行冒泡排序(从小到大)并输出(ch03\ForDemo.java)。

```
public class ForDemo{
    public static void main(String[] args){
        int  i;
        int intArray[]={39, 12, 5, 102,76};
        int  n=intArray.length;
        for (i=0; i<n-1; i++){
            for (int j=0; j<n-1; j++){
                if (intArray[j]>intArray[j+1]) {
                    int temp=intArray[j];
                    intArray[j]=intArray[j+1];
                    intArray[j+1]=temp;
                }
            }
        }
        for (i=0; i<n; i++) {
            System.out.println(intArray[i]);
        }
    }
}
```

运行结果：

```
5
12
39
76
102
```

3.2.4 for…each 语句

for…each 语句是 for 语句的特殊简化版本，也称为增强的 for 循环，是 JDK 1.5 的新特性之一，在遍历数组和集合方面，为开发人员提供了极大的方便。for…each 语句并不能完全取代 for 语句，但所有 for…each 语句都可以改写为 for 语句。

for…each 不是一个关键字，其语法格式如下：

```
for (部分类型 变量名:整体){
    循环体;
}
```

其中，整体由多个值组成，每次从整体中取出一部分，并赋值给冒号前的变量。例如：

```
int[] months={1,3,5,7,8,10,12};
for (int month:months){        //months 由多个 int 组成，每次从中取出一个，赋值给 month
    System.out.println(month);
}
```

for…each 语句经 JDK 编译后成为普通的 for 语句。

例 3.8 应用 for…each 语句，遍历数组，输出其中各元素(ch03\ForEachDemo.java)。

```
public class ForEachDemo{
    public static void main(String[] args){
        String[] str={"a","b","c"};
        int arr[][]={{9, 6}, {3, 1}};
        for (String s: str){
            System.out.print(s+" ");                    //逐个输出数组元素的值
        }
        System.out.print("\n");
        for (int x[] : arr) {
            for (int i : x) {
                System.out.print(i+"  ");
            }
            System.out.print("\n");
        }
    }
}
```

运行结果：

```
a b c
9  6
3  1
```

for…each 语句的局限性：当要引用数组或者集合的指定元素(例如给某个数组元素赋值等)时，for…each 语句无法做到，仍需使用基本的 for 语句。

3.2.5 嵌套循环

嵌套循环是指在某个循环语句的循环体中又包含另一个循环语句，也称多重循环。Java 语言提供的 3 种循环语句可以互相嵌套使用。例 3.9 即为一种嵌套循环。

例 3.9 应用嵌套循环，输出九九乘法表(ch03\WhileAndForDemo.java)。

```
public class WhileAndForDemo {
    public static void main(String[] args) {
        for (int i=1;i<=9;i++) {
            int j=1;
            while (j<=i) {
                System.out.print(j+"*"+i+"="+j*i+"\t");
                j++;
            }
            System.out.print("\r\n");                    //输出一个回车换行符
        }
    }
}
```

运行结果：

```
1*1=1
1*2=2  2*2=4
1*3=3  2*3=6   3*3=9
1*4=4  2*4=8   3*4=12  4*4=16
1*5=5  2*5=10  3*5=15  4*5=20  5*5=25
1*6=6  2*6=12  3*6=18  4*6=24  5*6=30  6*6=36
1*7=7  2*7=14  3*7=21  4*7=28  5*7=35  6*7=42  7*7=49
1*8=8  2*8=16  3*8=24  4*8=32  5*8=40  6*8=48  7*8=56  8*8=64
1*9=9  2*9=18  3*9=27  4*9=36  5*9=45  6*9=54  7*9=63  8*9=72  9*9=81
```

设计嵌套循环时应注意，内层循环语句必须完整地被包含在外层循环的循环体中，切不可出现内外层循环交叉的情况。

3.2.6 循环语句的对比

Java 的 3 种循环语句可以解决同一问题，在很多情况下，3 种循环语句可以相互转换使用，但在实际应用中，也有一定的区别。

(1) while 语句和 do…while 语句只在 while 后面指定循环条件，但是需要在循环体中包括使循环趋于结束的语句，而 for 语句则可以在增量表达式中包含使循环趋于结束的语句。

(2) 用 while 语句和 do…while 语句时，对循环变量的初始化操作应该放在 while 语句和 do…while 语句之前，而 for 语句则可以在初始表达式中完成。

(3) while 语句和 do…while 语句实现的功能相同，唯一的区别就是 do…while 语句先执行后判断，无论表达式的值是否为 true，都将执行一次循环；而 while 语句则是首先判断

表达式的值是否为 true，如果为 true 则执行循环语句，否则不执行循环语句。

(4) for 循环语句一般用在对于循环次数已知的情况，而 while 语句和 do…while 语句则一般用在对于循环次数不确定的情况。

3.3 跳转语句

跳转语句可以无条件地改变程序的执行顺序。Java 语言提供了三种跳转语句，分别为 break 语句、continue 语句和 return 语句。

3.3.1 break 语句

1. 无标签 break 语句

break 语句应用于 switch、for、while 及 do 语句中，可以立即终止执行包含 break 语句所在的一个程序块。

例 3.10 使用无标签的 break 语句(ch03\BreakDemo.java)。

```
public class BreakDemo {
    public static void main(String[] args) {
        boolean test=true;
        int i=1;
        while (test) {
            System.out.println("i="+i);
            if (i==5)  break;
            i++;
        }
        System.out.println("i 为 5 时结束循环!");
    }
}
```

运行结果：

```
i=1
i=2
i=3
i=4
i=5
i 为 5 时结束循环!
```

在上述程序中，尽管 while 循环的判断条件一直为 true，但是当 i 等于 5 时，break 语句结束整个循环，接着执行循环后续的代码。因此，在循环语句中，可以使用 break 语句中断正在执行的循环。

在实际的代码中，结构往往会因为逻辑比较复杂，而存在循环语句的嵌套，如果 break 语句出现在循环嵌套的内部时，则只结束 break 语句所在的循环，对于其他的循环没有影响，示例代码如下：

```
for (int i=0;i<2;i++) {
    for (int j=0;j<5;j++) {
        System.out.println(i+","+j);
        if (j==2) break;
    }
}
```

运行结果：

```
0,0
0,1
0,2
1,0
1,1
1,2
```

该 break 语句因为出现在循环变量为 j 的循环内部，则执行到 break 语句时，只中断循环变量为 j 的循环，而对循环变量为 i 的循环没有影响。

使用 break 语句应注意，一个循环中可以有一个以上的 break 语句，但是太多的 break 语句会破坏代码结构。另外，switch 语句中的 break 仅影响所在的 switch 语句，不会影响外层的任何循环。

2. 带标签的 break 语句

break 语句只终止执行包含它的最小语句块，如果希望终止更外层的语句块，可以用带标签的 break 语句。

带标签的 break 语句的格式如下：

```
break  标签;
```

当 break 语句以这种形式执行时，可控制传递出指定的代码块。被加标签的代码块必须包含 break 语句，但是它不需要是直接包含 break 块。因此可以使用一个加标签的 break 语句退出一系列的嵌套块。

要为一个代码块加标签，在其开头加一个标签即可。标签可以是任何合法有效的 Java 标识符，标签后跟一个“:”。给一个块加上标签后，就可以使用这个标签作为 break 语句的对象。

例 3.11　使用带标签的 break 语句(ch03\LabelBreakDemo.java)。

```
public class LabelBreakDemo {
    public static void main(String[] args) {
        loop:
        for (int i=1;i<=9;i++) {
            for (int j=1;j<=i;j++) {
               if (j*i==16) {
                   break loop;
               }
               System.out.print(j+"*"+i+"="+j*i+"  ");
```

```
            }
            System.out.print("\r\n");                    //输出一个回车换行符
        }
    }
}
```

运行结果：

```
1*1=1
1*2=2   2*2=4
1*3=3   2*3=6    3*3=9
1*4=4   2*4=8    3*4=12
```

3.3.2 continue 语句

continue 语句只能用于循环结构中，用于跳过当次循环的剩余语句，重新开始下一轮循环。continue 语句也有两种形式，即无标签的和带标签的。

1. 无标签的 continue 语句

无标签的 continue 语句可以结束当次循环，即跳过 continue 语句后面剩余部分，根据条件判断是否进入下一轮循环。

例 3.12 使用无标签的 continue 语句，输出 100 以内能够被 7 整除的数（ch03\ContinueDemo.java）。

```
public class ContinueDemo {
    public static void main(String[] args) {
        for (int i=1;i<100;i++) {
            if (i%7!=0) {                          //当 i 的值不能被 7 整除时
                continue;
            }
            System.out.print(i+"  ");              //输出 i 的值
        }
    }
}
```

运行结果：

```
7  14  21  28  35  42  49  56  63  70  77  84  91
```

2. 带标签的 continue 语句

continue 语句和 break 语句一样，也可以和标签搭配使用。其作用也是用于跳出深层循环。带标签的 continue 语句也是通常用在嵌套循环的内循环中，其语句格式如下：

```
continue 标签;
```

continue 后的标签必须标识在循环语句之前，使程序的流程在遇到 continue 之后，立即结束当次循环，转入标签所标识的循环层次，进行下一轮循环。

例 3.13 使用带标签的 continue 语句（ch03\LabelContinueDemo.java）。

```
public class LabelContinueDemo {
    public static void main(String[] args) {
        loop:
        for (int i=1;i<=9;i++){
            for (int j=1;j<=i;j++){
                if (j==2){
                    continue loop;                          //跳出双层 for 循环语句
                }
                System.out.print(j+"*"+i+"="+j*i+"\t");
             }
            System.out.print("\r\n");                       //输出一个回车换行符
        }
    }
}
```

运行结果：

```
1*1=1
1*2=2   1*3=3   1*4=4   1*5=5   1*6=6   1*7=7   1*8=8   1*9=9
```

3.3.3 return 语句

return 语句用来返回方法的值，其功能是从当前方法退出，返回调用该方法的语句，语句格式如下：

```
return 表达式;
```

return 语句通常位于一个方法体的最后一行，return 语句退出方法并返回一个值。由 return 返回的表达式的值必须与方法的返回值类型一致。当方法用 void 声明时，说明方法无返回值，return 语句省略。

例 3.14 使用 return 语句，计算圆面积(ch03\ReturnDemo.java)。

```
public class ReturnDemo {
    final static double PI=3.14;
    public static void main(String[] args) {
        double r1=3.0, r2=5.0;
        System.out.println("半径为"+r1+"的圆面积为"+area(r1));
        System.out.println("半径为"+r2+"的圆面积为"+area(r2));
    }
    static double area(double r) {
        return (PI*r*r);
    }
}
```

运行结果：

```
半径为 3.0 的圆面积为 28.259999999999998
半径为 5.0 的圆面积为 78.5
```

本章小结

本章介绍了 Java 流程控制语句，包括分支结构的 if…else 语句和 switch 语句，循环结构的 while 语句、do…while 语句和 for 语句，以及跳转语句 break、continue 和 return，通过对语句的介绍，说明了语句的特点及应用。

习 题 3

1. 有如下函数，要求输入 x 的值，求 y 的值。（要求用 if…else 语句实现）编辑、编译、运行该程序，分别使用数据 −5、0、5、10、100 做测试，写出相应的结果。

$$y=\begin{cases}x^2, & x<0\\ 2x-1, & 0\leqslant x<10\\ 3x-11, & x\geqslant 10\end{cases}$$

2. 求一元二次方程 $ax^2+bx+c=0$ 的根（考虑 b^2-4ac 的值的 3 种情况）。

3. 将一个数组中的值按逆序重新存放。

4. 求两个整数的最大公约数和最小公倍数。

5. 求 100～200 的全部素数。

6. 输出 100～999 所有的“水仙花数”。

7. 求 Fibonacci 数列的前 40 个数。即 $F_1=1, F_2=1$，$F_n=F_{n-1}+F_{n-2}(n\geqslant 3)$。

8. 用 $\frac{\pi}{4}\approx 1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots$ 公式求 π 的近似值，直到最后一项的绝对值小于 10^{-6}。

9. 一个数如果恰好等于它的因子之和，这个数就是完数。例如 6 的因子为 1、2、3，而 6＝1＋2＋3，因此 6 是一个完数。编程求出 1000 之内的所有完数。并按如下格式输出：

```
6,因子是 1,2,3
…
```

10. 用迭代法求出 $x=\sqrt{a}$。求平方根的迭代公式为

$$x_{n+1}=\frac{1}{2}\left(x_n+\frac{a}{x_n}\right)$$

其中，$x_0=a/2$，要求前后两次求出的 x 的差的绝对值小于 10^{-5}。

11. 一球从 100m 的高度自由落下，每次落地后反跳回原来高度的一半，再落下。求它在第 10 次落地时共经过多少米，第 10 次反弹多高。

12. 有一堆零件（100～200 个），如果以 4 个零件为一组进行分组，则多 2 个零件；如果以 7 个零件为一组进行分组，则多 3 个零件；如果以 9 个零件为一组进行分组，则多 5 个零件。编程求解这堆零件的总数。

13. 已知大公鸡三文钱一只，大母鸡两文钱一只，小鸡一文钱买三只。现有 100 文钱，想买 100 只鸡，请编写程序解决这个问题。

14.《孙子算经》中记载了这样的一道题：“今有雉兔同笼，上有三十五头，下有九十四

足，问雉兔各几何？”这四句话的意思就是：有若干只鸡和兔在同一个笼子里，从上面数，有35个头；从下面数，有94只脚。求笼中各有几只鸡和兔。

15. 猴子吃桃问题。猴子第一天摘了若干桃子，当即吃了一半，还不过瘾，又多吃了一个，第二天早上又吃了剩下的一半多一个，以后每天早上如此，到第10天早上想再吃时，只剩下一个桃子了。求第一天共摘了多少桃子。

16. 洗碗（中国古题）：有一位妇女在河边洗碗，过路人问她为什么洗这么多碗？她回答说：家中来了很多客人，他们每两人合用一个饭碗，每三人合用一个汤碗，每四人合用一个菜碗，共用了65个碗。你能通过她家的用碗情况算出她家来了多少客人吗？