第 3 章 选择与循环

本章学习目标

- 理解结构化程序设计的三种基本结构;
- 学会选择结构的使用，包括单分支和双分支结构;
- 理解嵌套 if-else 结构的用法;
- 了解条件运算符的用法，会用 if-else 结构重写条件表达式;
- 学会使用 switch 结构实现多分支，熟悉 switch 中可使用的表达式类型;
- 了解循环结构的应用场景和循环结构的类型;
- 能够区分 while 循环和 do-while 循环的区别;
- 熟悉 for 循环的基本格式和使用场景;
- 掌握 break 语句和 continue 语句的使用;
- 了解无限循环及结束循环的方法;
- 掌握嵌套循环的执行流程。

3.1 选　　择


教学视频

结构化程序设计有三种基本结构：顺序结构、选择结构和循环结构。顺序结构比较简单，程序按语句的顺序依次执行。本节介绍选择结构，下节介绍循环结构。

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

3.1.1 单分支 if 语句

单分支的 if 结构的一般格式如下：

```
if (condition){
    statements;
}
```

其中 condition 为布尔表达式，它的值为 true 或 false。布尔表达式应该使用括号括住。程序执行的流程是：首先计算 condition 表达式的值，若其值为 true，则执行 statements 语句序列，否则转去执行 if 结构后面的语句，如图 3-1 所示。

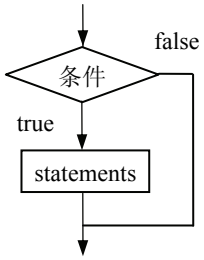

图 3-1　单分支结构

编写程序，从键盘上读取一个整数，检查该数是否能同时被 5 和 6 整除，是否能被 5 或被 6 整除，是否只能被 5 或只能被 6

整除。

程序 3.1　CheckNumber.java

```
import java.util.Scanner;
public class CheckNumber{
public static void main(String[] args){
    Scanner input = new Scanner(System.in);
    System.out.print("请输入一个整数：");
    int num = input.nextInt() ;
    if(num % 5 == 0 && num % 6 == 0){
       System.out.println( num + " 能被5和6同时整除。") ;
    }
    if(num % 5 == 0 || num % 6 == 0){
       System.out.println( num + " 能被5或6整除。") ;
    }
    if(num % 5 == 0 ^ num % 6 == 0){
       System.out.println( num + " 只能被5或只被6整除。") ;
    }
  }
}
```

下面是程序运行的一个结果：

```
请输入一个整数：12
12 能被5或6整除。
12只能被5或只被6整除。
```

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。

```
if(num % 5 == 0 && num % 6 == 0){
     System.out.println( num + " 能被5和6同时整除。") ;
}
```

与下面代码等价。

```
if(num % 5 == 0 && num % 6 == 0)
  System.out.println( num + " 能被5和6同时整除。") ;
```

注意：省略大括号可以使代码更整洁，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。

3.1.2　双分支 if-else 语句

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

```
if (condition){
```

```
    statements1;
}else{
    statements2;
}
```

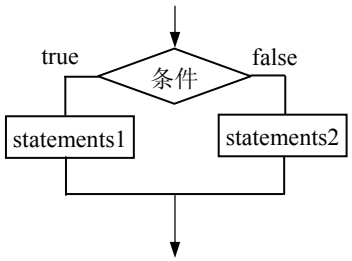

图 3-2　双分支结构

该结构的执行流程是：首先计算 condition 的值，如果为 true，则执行 statements1 语句序列，否则执行 statements2 语句序列，如图 3-2 所示。

例如：

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

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

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

假设要开发一个让一年级学生练习一位数加法的程序。程序开始运行随机生成两个一位数，显示题目让学生输入计算结果，程序给出结果是否正确。

可以使用 Math.random()方法产生一个 0.0～1.0 的随机 double 型值，不包括 1.0。要生成一个一位数，使用下面的表达式：

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

程序 3.2　AdditionQuiz.java

```
import java.util.Scanner;
public class AdditionQuiz {
   public static void main(String[]args){
     //随机产生2个一位数
     int number1 = (int)(Math.random()*10);
     int number2 = (int)(Math.random()*10);
     Scanner input = new Scanner(System.in);
     System.out.print(number1 + " + " +number2 + " = ");
     int answer = input.nextInt();            //接收用户输入的答案
     if(answer==(number1+number2)){
          System.out.println("恭喜你，答对了！");
     }else{
          System.out.println("很遗憾，答错了！正确答案是\n");
          System.out.println(number1 + " + " +number2+" = "+(number1+number2));
     }
   }
```

```
}
```

下面是程序的一次运行结果：

```
1 + 6 = 10
很遗憾，答错了！正确答案是
1 + 6 = 7
```

3.1.3 嵌套的 if 语句和多分支的 if-else 语句

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

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

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，这种结构有时称为阶梯式 if-else 结构。

下面程序要求输入学生的百分制成绩，打印输出等级的成绩。等级规定为，90 分（包括）以上为“优秀”，80 分（包括）以上为“良好”，70 分（包括）以上为“中等”，60 分（包括）以上为“及格”，60 分以下为“不及格”。

程序 3.3　ScoreLevel.java

```
import java.util.Scanner;
public class ScoreLevel{
   public static void main(String[] args){
     Scanner sc = new Scanner(System.in);
     System.out.print("请输入成绩：");
     double score = sc.nextDouble() ;
     String level = "";
     if(score >100 || score < 0){
```

```
            System.out.println("输入的成绩不正确。");
            System.exit(0);      //结束程序运行
        }else if(score >=90){
            level = "优秀";
        }else if(score >=80){
            level = "良好";
        }else if(score >=70){
            level = "中等";
        }else if(score >=60){
            level = "及格";
        }else{
            level = "不及格";
        }
        System.out.println("你的成绩等级为：" + level);
    }
}
```

下面是程序的一次运行结果：

```
请输入成绩：78
你的成绩等级为：中等
```

3.1.4 条件运算符

条件运算符（conditional operator）的格式如下：

```
condition ? expression1 : expression2
```

因为有三个操作数，又称为三元运算符。这里 condition 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 expression1 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 expression2 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, b 是 int 型变量，下面结构：

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

用条件运算符表示为：

```
max = (a > b)? a : 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。

3.1.5 switch 语句结构

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

```
switch (expression){
   case value1:
      statements    [break;]
   case value2:
      statements    [break;]
   ⋮
   case valuen:
      statements    [break;]
   [default:
     statements]
}
```

其中 expression 是一个表达式，它的值必须是 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。程序进入 switch 结构，首先计算 expression 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，直到遇到 break 语句为止。

break 语句的功能是退出 switch 结构。如果在某个情况处理结束后就离开 switch 结构，则必须在该 case 结构的后面加上 break 语句。

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。

编写程序，从键盘输入一个年份（如 2000 年）和一个月份（如 2 月），输出该月的天数（29）。

程序 3.4　SwitchDemo.java

```
import java.util.Scanner;
public class SwitchDemo{
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      System.out.print("输入一个年份：");
      int year = input.nextInt();
      System.out.print("输入一个月份：");
      int month = input.nextInt();
      int numDays = 0;
      switch (month) {
         case 1: case 3: case 5:
         case 7: case 8: case 10:
         case 12:
```

```
                numDays = 31;
                break;
            case 4: case 6: case 9: case 11:
                numDays = 30;
                break;
            case 2:     //对2月需要判断是否是闰年
                if (((year % 4 == 0) &&  !(year % 100 == 0))
                    || (year % 400 == 0))
                    numDays = 29;
                else
                    numDays = 28;
                break;
            default:
                System.out.println("月份非法.");
                break;
        }
        System.out.println("该月的天数为："  + numDays);
    }
}
```

下面是程序的一次运行结果：

```
输入一个年份：2016
输入一个月份：2
该月的天数为：29
```

从 Java SE 7 开始，可以在 switch 语句的表达式中使用 String 对象，下面代码根据英文月份的字符串名称输出数字月份。

程序 3.5　StringSwitchDemo.java

```
import java.util.Scanner;
public class StringSwitchDemo {
    public static void main(String[] args) {
        String month = "";
        int monthNumber = 0;
        Scanner input = new Scanner(System.in);
        System.out.println("请输入一个月份的英文名称：");
        month = input.next();
        switch (month.toLowerCase()) {
            case "january": monthNumber = 1;break;
            case "february":monthNumber = 2;break;
            case "march": monthNumber = 3; break;
            case "april": monthNumber = 4; break;
            case "may": monthNumber = 5; break;
            case "june": monthNumber = 6; break;
            case "july": monthNumber = 7; break;
```

```
            case "august": monthNumber = 8; break;
            case "september": monthNumber = 9; break;
            case "october": monthNumber = 10; break;
            case "november": monthNumber = 11; break;
            case "december": monthNumber = 12; break;
            default:
                monthNumber = 0; break;
        }
        if (monthNumber == 0) {
            System.out.println("输入的月份名非法");
        }else {
            System.out.println(month +"是" + monthNumber +"月");
        }
    }
}
```

程序中 month.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。

教学视频

3.2 循　　环

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 4 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）循环体部分：需要重复执行的代码。

3.2.1 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：

```
[initialization]
while (condition){
    //循环体
   [iteration]
}
```

其中，initialization 为初始化部分；condition 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；iteration 为迭代部分。

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 3-3 所示。

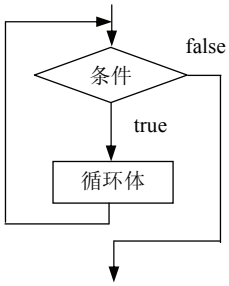

图 3-3　while 循环结构

下面一段代码使用 while 结构求 1～100 之和。

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

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

程序 3.6　GuessNumber.java

```
import java.util.Scanner;
public class GuessNumber{
   public static void main(String[] args){
      int magic = (int)(Math.random()*101)+100;
      Scanner sc = new Scanner(System.in);
      System.out.print("请输入你猜的数：");
      int guess = sc.nextInt();
      while(guess != magic){
         if(guess > magic)
           System.out.print("错误!太大，请重猜：");
         else
           System.out.print("错误!太小，请重猜：");
         //输入下一次猜的数
         guess = sc.nextInt();
      }
      System.out.println("恭喜你，答对了！\n该数是："+magic);
   }
}
```

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 型随机数。程序中该方法乘以 101 再转换为整数，得到 0～100 的整数，再加上

100，则 magic 为 100～200 的整数。

3.2.2 do-while 循环

do-while 循环的一般格式如下：

```
[initialization]
do{
   //循环体
   [iteration]
}while(condition);
```

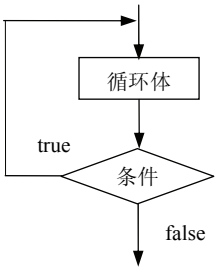

图 3-4　do-while 循环结构

do-while 循环执行过程如图 3-4 所示。

该循环首先执行循环体，然后计算条件表达式。如果表达式的值为 true，则返回到循环的开始继续执行循环体，直到 condition 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。

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

程序 3.7　DoWhileDemo.java

```
import java.util.Scanner;
public class DoWhileDemo {
    public static void main(String[] args) {
       double sum = 0,avg = 0;
       int n = 0;
       double number;
       Scanner input = new Scanner(System.in);
       do{
           System.out.print("请输入一个数（输0结束）:");
           number = input.nextDouble();
           if(number != 0){
              sum = sum + number;
              n = n + 1;
           }
       }while(number!=0);
       avg = sum / n;
       System.out.println("sum = "+ sum);
       System.out.println("avg = "+ avg);
    }
}
```

3.2.3 for 循环

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

```
for (initialization; condition; iteration){
    //循环体
}
```


其中，initialization 为初始化部分，condition 为循环条件，iteration 为迭代部分，三部分用分号隔开。循环开始执行时首先执行初始化部分，该部分在整个循环中只执行一次。在这里可以定义循环变量并赋初值。

接下来判断循环条件，若为 true 则执行循环体部分，否则退出循环。当循环体执行结束后，程序控制返回到迭代部分，执行迭代，然后再次判断循环条件，若为 true 则反复执行循环体。

下面代码使用 for 循环计算 1～100 之和。

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

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

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

for 循环中的一部分或全部可为空，循环体也可为空，但分号不能省略。例如：

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

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 5.1.2 节中讨论。

3.2.4 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。

下面程序打印输出九九乘法表，这里使用了嵌套的 for 循环。

程序 3.8　NineTable.java

```
public class NineTable{
   public static void main(String[] args){
     for(int i = 1; i <=9; i++){
         for(int j = 1; j <= i; j++)
            System.out.print(j +" * " + i +" = " + i*j + "  ");
         System.out.println();    //换行
     }
   }
}
```

程序输出结果为：

```
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
```

3.2.5 break 语句和 continue 语句

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

1. break 语句

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

```
break;
break label;
```

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。下面程序演示了 break 语句的使用。

程序 3.9　BreakDemo.java

```
public class BreakDemo{
   public static void main(String[] args){
     int n = 1;
     int sum = 0;
```

```
        while(n <= 100){
          sum = sum + n;
          if(sum > 100){
             break;          //若条件成立退出循环
          }
          n = n + 2;
        }
        System.out.println("n = " + n);
        System.out.println("sum = " + sum);
    }
}
```

程序输出结果为：

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

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

例如：

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

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

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

2. continue 语句

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

```
continue;
continue  label;
```

以下代码会输出 0～9 之间的数字，但不会输出 5。

```
for(int i =0; i < 10; i++){
   if(i ==5){
```

```
            continue;
        }
        System.out.println(i);
    }
```

当 i=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=6 时。

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

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

这段代码的运行结果如下：

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

注意：

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

（2）标签命名遵循标识符的命名规则，相互包含的块不能用相同的标签名。

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

提示：在 C/C++语言中，可以使用 goto 语句从内层循环跳到外层循环。在 Java 语言中，尽管将 goto 作为关键字，但不能使用，也没有意义。

3.3 示 例 学 习

教学视频

3.3.1 任意抽取一张牌

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、梅花和方块。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 为梅花，39～51 为方块。

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 3.10　PickCards.java

```
public class PickCards {
   public static void main(String[] args){
      int card =(int) (Math.random()*53);
      String suit="",rank="";
      switch(card / 13){                  //确定牌的花色
        case 0: suit="黑桃";break;
        case 1: suit="红桃";break;
        case 2: suit="方块";break;
        case 3: suit="梅花";break;
      }
      switch(card % 13){                  //确定是第几张牌
        case 0: rank="A";break;
        case 10: rank = "J";break;
        case 11: rank = "Q";break;
        case 12: rank = "K";break;
        default:rank = ""+(card %13 +1);
      }
      System.out.println("你抽取的牌是：" + suit + " " + rank);
   }
}
```

下面是程序的一次运行结果：

```
你抽取的牌是：梅花 2
```

3.3.2　求最大公约数

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。

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。

程序 3.11　GCD.java

```
import java.util.Scanner;
public class GCD{
   public static void main(String[] args){
      Scanner input = new Scanner(System.in);
```

```
      System.out.print("Enter first integer:");
      int m = input.nextInt();
      System.out.print("Enter second integer:");
      int n = input.nextInt();
      //求m和n的最大公约数
      int gcd = 1;
      int k = 2;
      while(k <= m && k <= n){
         if(m % k == 0 && n % k == 0) //判断k是否能同时被n1和n2整除
            gcd = k;
         k++;
      }
      System.out.println("The GCD of "+m +" and "+ n +" is "+gcd);
   }
}
```

下面是程序的一次运行结果：

```
Enter first integer:16
Enter second integer:24
The GCD of 16 and 24 is 8
```

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。

3.3.3 打印输出若干素数

素数（prime number）又称质数，有无限个。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。下面的程序计算并输出前 50 个素数，每行输出 10 个。

程序 3.12　PrimeNumber.java

```
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.println("The first 50 primes are:\n");
     while(count < 50){
        isPrime = true;
        for(int divisor = 2; divisor * divisor <= number; divisor ++){
          if(number % divisor ==0){
             isPrime = false;
             break;
          }
```

```
            }
            if(isPrime){
               count ++;
               if(count%10==0)
                  System.out.println(number);
               else
                  System.out.print(number + " ");
            }
            number ++;
         }
      }
   }
```

程序输出结果如下：

```
The first 50 primes are:

2 3 5 7 11 13 17 19 23 29
31 37 41 43 47 53 59 61 67 71
73 79 83 89 97 101 103 107 109 113
127 131 137 139 149 151 157 163 167 173
179 181 191 193 197 199 211 223 227 229
```

3.4 小　　结

（1）Java 的选择结构有以下几种类型：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。

（2）使用 Math 类的 random()方法可以随机生成 0.0～1.0（不包含）的一个 double 型数。在此基础上通过编写简单的表达式，生成任意范围的随机数。

（3）条件运算符（condition?expression1：expression2）是 Java 唯一的三元运算符，可以简化 if-else 语句的编写。

（4）switch 语句根据 char、byte、short、int、String 或 enum 类型的 switch 表达式来进行控制决定。

（5）在 switch 语句中，关键字 break 是可选的，但它通常用在每个分支的结尾，以终止执行 switch 语句的剩余部分。如果没有出现 break 语句，则执行接下来的全部 case 语句。

（6）循环语句有 4 种：while 循环、do-while 循环、for 循环和增强的 for 循环。循环中包含重复执行语句的部分称为循环体。

（7）while 循环首先检查循环继续条件，如果条件为 true，则执行循环体；如果条件为 false，则循环结束。

（8）do-while 循环与 while 循环类似，只是 do-while 循环先执行循环体，然后再检查循环继续条件，以确定是继续循环还是终止循环。

（9）for 循环控制由三部分组成。第一部分是初始操作，通常用于初始化控制变量；第二部分是循环继续条件，决定是否执行循环体；第三部分是每次迭代后执行的操作，用于

调整控制变量。

（10）for 循环一般用在循环体执行次数固定的情况。

（11）在循环体中可以使用 break 和 continue 这两个关键字，break 立即终止包含 break 的最内层循环；continue 只是终止当前迭代。

编 程 练 习

3.1　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。

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

```
请输入年份：2017
2017年不是闰年。
```

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

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

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

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

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

3.5　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。

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

3.7　编写程序，要求用户从键盘输入一个年份和月份，然后显示这个月的天数。例如，如果用户输入的是 2012 年 2 月，那么程序应该显示“2012 年 2 月有 29 天”。如果用户输入的是 2015 年 3 月，那么程序应该显示“2015 年 3 月有 31 天”。

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

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

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

3.10　编写程序，计算并输出 0～1000 含有 7 或者是 7 倍数的整数之和及个数。

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

3.12　编写程序，从键盘输入一个整数，计算并输出该数的各位数字之和。例如：

```
请输入一个整数：8899123
各位数字之和为：40
```

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

3.14　编写程序，计算下面级数之和：

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

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

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

3.17　从键盘输入两个整数，计算这两个数的最小公倍数和最大公约数并输出。

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

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

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

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