项目 2　Java 语法概述

项目 1 中，学习了一些 Java 有关的基本概念，本章将学习 Java 程序语言设计基础语法，主要介绍 Java 语言的语法规则，规则一方面是程序员编写程序的约束；另一方面，规则又可以帮助检查出程序错误的地方，并可让程序员充分应用规则，进行程序代码的各种编写尝试。

2.1 项目概述

在本项目中学习基本符号、变量和常量的概念，Java 基本数据类型、运算符、表达式以及几种常用的程序控制语句，还要学习数组的定义和使用。

2.2 项目目的

- 掌握 Java 语言中的基本符号、数据类型的定义及使用方法。
- 掌握 Java 语言中的运算符和表达式、程序流程控制的使用方法。
- 掌握 Java 语言中数据和字符串的使用方法。

2.3 项目支撑知识

2.3.1　项目开发背景知识 1 Java 的基本符号

一、标识符

标识符是程序语句中使用标志符来命名变量、类、创建的对象及对象方法。标识符由一串字符组成，是程序语句的基本组成部分。在 Java 语言中，标识符的命名规则如下，否则会产生编译错误。

（1）标识符必须以字母、下划线（_ ）或美元符号（ $ ）开头，后面可以是字母、下划线、美元符号、数字（0 ~ 9）。

（2）标识符不能是关键字。关键字是编译程序本身所使用的标识符，如上

面例子中的 int，double 均是关键字。

（3）标识符可有任意长度。标识符不宜过短，过短的标识符会导致程序的可读性差；标识符不宜过长，过长将增加录入工作量和出错的可能性。

（4）标识符区分大小写，如 Mybook 与 mybook 是完全不同的两个标识符。

例如，i1，i2，count，value_ add 等都是合法的标识符，而关键字不能当做标识符使用，所以 2count，high#，null 等都是非法的标识符。Java 语言区分字母大小写，所以 VALUE，Value，value 表示不同的标识符。

二、关键字

关键字具有专门的意义和用途，不能当做一般的标识符使用。Java 语言中的关键字均用小写字母表示。主要可以分为如下几类。

（1）访问控制：private，protected，public。

（2）类，方法和变量修饰符：abstract，class，extends，final，implements，interface，native，new，static，strictfp，synchronized，transient，volatile。

（3）程序控制语句：break，continue，return，do，while，if，else，for，instanceof，switch，case，default。

（4）错误处理：catch，finally，throw，throws，try。

（5）包相关：import，package。

（6）基本类型：boolean，byte，char，double，float，int，long，short。

（7）变量引用：super，this，void。

（8）语法保留字：null，true，false。

注意：关键字 goto 和 const 是 C + + 保留的关键字，在 Java 中不能使用。sizeof，String，大写的 NULL 也不是关键字。

三、注释

Java 允许在源程序文件中添加注释（comment），以增加程序的可读性，系统不会对注释的内容进行编译。Java 有 3 种形式的注释。

（1）单行注释。

单行注释以“//”开头，至该行行尾。其格式如下：

```
//单行注释(comment on one line)
```

（2）多行注释。

多行注释以“/ *”开头，以“ *”结束。其格式如下：

```
/* 单行或多行注释
(comment on more lines) * /
```

（3）文件注释。

文件注释用来产生一个 HTML 文件，从而为程序提供文档说明。文件注释以“/ * *”开头，以“ *”结束。其格式如下：

```
/* *文件注释
(documenting comment) */
```

四、分隔符

Java 使用一些特殊字符作为分隔符（separator），表 2 - 1 列出了 Java 定义的分隔符及功能。

表 2 - 1　Java 中的分隔符

分隔符	名称	功能说明
{ }	大括号（花括号）	用来定义程序块、类、方法以及局部范围，也用来包括自动初始化的数组的值。
[ ]	中括号（中括号）	用来进行数组的声明，也用来表示撤销对数组值的引用。
( )	小括号（圆括号）	在定义和调用方法时用来容纳参数表。在控制语句或强制类型转换组成的表达式中用来表示执行或计算的优先权。
;	分号	用来表示一条语句的结束。在 for 控制语句中，用来将圆括号内的语句连接起来。
,	逗号	在变量声明中，用于分隔变量表中的各个变量。
.	点号	用来将软件包的名字与它的子包或类分隔。也用来将引用变量与变量或方法分隔。

2.3.2　项目开发背景知识 2 Java 数据类型

计算机处理的对象是数据，可以是数值数据，如 34.5，也可以是非数值数据，如字符串“Hello!”等。通常算法语言将数据按其性质进行分类，每一类称为一种数据类型（datatype）。数据类型定义了数据的性质、取值范围、存储方式以及对数据所能进行的运算和操作。

程序中的每一个数据都属于一种类型，决定了数据的类型也就相应决定了数据的性质以及对数据进行的操作，同时数据也受到类型的保护，确保对数据不进行非法操作。

在 Java 语言中，数据类型分为简单类型（primitive type）和引用类型（reference type）。简单类型有整型（integral）、浮点型（floating）、逻辑型（logical）和字符型（textual）。引用类型包括类（class）、数组（array）和接口（interface）。

Java 语言的数据类型实际上都是用类实现的，即引用对象的使用方式，同时 Java 也提供了类似 C 语言中简单类型的使用方式，即声明类型的变量。

一、整型

整数有正整数、零、负整数，含义同数学中一样。Java 的整数有 3 种进制的形式表示。

十进制：用多个 0 ~ 9 之间的数字表示，如 123 和 - 100，其首位不能为 0。

八进制：以 0 打头，后跟多个 0 ~ 7 之间的数字，如 0123。

十六进制：以 0x 或 0X 打头，后跟多个 0 ~ 9 之间的数字或 a ~ f 之间的小写字母或 A ~ F 之间的大写字母，a ~ f 或 A ~ F 分别表示值 10 ~ 15，如 0X123E。

Java 定义了 4 种表示整数的整型：字节型（byte）、短整型（short）、整型（int）、长整型（long）。每种整型的数据都是带符号位的。它们的特性如表 2 - 2 所示。

表 2 - 2　Java 的 4 种整型

类　型	数据位	范　围
字节型 byte	8	- 128 ~ 127，即 $-2^7 \sim 2^7 - 1$
短整型 short	16	- 32768 ~ 32767，即 $-2^{15} \sim 2^{15} - 1$
整型 int	32	- 2147483648 ~ 2147483647 即 $-2^{31} \sim 2^{31} - 1$
长整型 long	64	- 9223372036854775808 ~ 9223372036854775807，即 $-2^{63} \sim 2^{63} - 1$

一个整数隐含为整型。当要将一个整数强制表示为长整数时，需在后面加 L 或 l。

【例 2 - 1】4 种整数类型的常量的最大值。

```
1  public class intmax
2  {
3    public static void main(String args[])
4    {
5      long lmax = Java.lang.Long.MAX_VALUE;
6      int imax = Java.lang.Integer.MAX_VALUE;
7      short smax = Short.MAX_VALUE;       //省略类库 Java.lang
8      byte bmax = Byte.MAX_VALUE;         //省略类库 Java.lang
9      System.out.println("Max value of long  :" + lmax);
10     System.out.println("Max value of int   :" + imax);
11     System.out.println("Max value of short :" + smax);
12     System.out.println("Max value of byte  :" + bmax);
13   }
14 }
```

【运行结果】如图 2 - 1 所示。

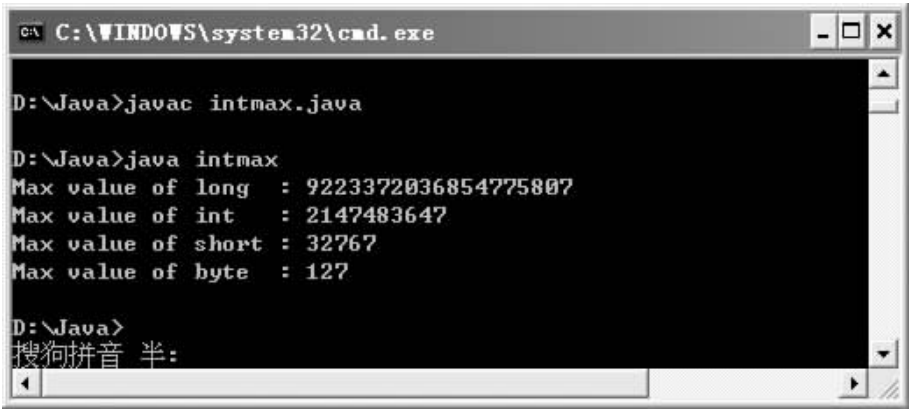

图 2 - 1 【例 2 - 1】程序运行结果

【程序分析】

第 5 ~ 8 句：Java 中每种数据类型都封装为一个类，如 Long，Integer，Short，Byte，Float，Double，通过类型类的 MAX_ VALUE 常量找到各种数值数据类型的取值最大值。对应将 MAX_ VALUE 替换为 MIN_ VALUE，可以输出各种整型类型的最小值。

第 7 ~ 8 句：Java. lang 这个类库实在是太常用了，默认的 Java 程序会将它加载，省略类库 Java. lang，程序仍然可以正确运行。

二、浮点型

Java 用浮点型表示数学中的实数，即整数部分和小数部分。浮点数有两种表示方式。

（1）标准计数法：由整数部分、小数点和小数部分构成，如 1.0，123.45 等。

（2）科学计数法：由十进制整数、小数点、小数和指数部分构成，指数部分由字母 E 或 e 跟上带正负号的整数表示，如 123.45 可表示为 1.2345E + 2。浮点数用于需要小数位精确度的计算。例如，计算平方根或三角函数等，都会产生浮点型的值。Java 的浮点格式完全遵循 IEEE - 754 标准。Java 的浮点型有单精度浮点（float）和双精度浮点（double）两种。它们的宽度和范围如表 2 - 3 所示。

表 2 - 3 Java 的两种浮点型

类 型	数据位	范 围
单精度浮点（float）	32	3.4E - 038 ~ 3.4E + 038
双精度浮点（double）	64	1.7E - 308 ~ 1.7E + 308

一个浮点数默认为 double 型。在一个浮点数后加字母 F 或 f，将其强制转换

为 float 型。例如：float f = 3.0f。

【例 2 - 2】浮点型的常量值。

```
1  public class floatmax
2  {
3    public static void main(String args[])
4    {
5      System.out.println("f_max = " +Java.lang.Float.MAX_VALUE);
6      System.out.println("f_min = " +Java.lang.Float.MIN_VALUE);
7      System.out.println("d_max = " +Java.lang.Double.MAX_VALUE);
8      System.out.println("d_min = " +Java.lang.Double.MIN_VALUE);
9    }
10 }
```

【运行结果】如图 2 - 2 所示。

图 2 - 2　【例 2 - 2】程序运行结果

【程序分析】

第 5 ~ 8 句：Java 中每种数据类型都封装为一个类，如 Long，Integer，Short，Byte，Float，Double，通过类型类的 MAX_ VALUE 常量找到各种浮点数据类型的最大值。也可以省略 Java. lang 类。

三、布尔型

布尔型（Boolean）用来表示逻辑值，也称为逻辑型。它只有真（true）和假（false）两个值。true 和 false 不能转换成数字表示形式。

所有关系运算（如 a < b）的返回值都是布尔型的值。布尔型也用于控制语句中的条件表达式，如 if，while，for 等语句。

四、字符型

字符型（char）用来存储字符。一个字符用一个 16 位的 Unicode 码表示。所有可见的 ASCII 字符都可以用单引号括起来成为字符，如'a','A','#'等。一些控

制字符不能直接显示，可以利用转义序列来表示，如表 2 - 4 所示。

表 2 - 4　转义序列

转义序列	说　明
\ n	换行，将光标移至下一行的开始
\ t	水平制表，将光标移至下个制表符位置
\ r	回车，将光标移至当前行的开始，不移到下一行
\ \	反斜杠，输出一个反斜杠
\ `	单引号，输出一个单引号
\ "	双引号，输出一个双引号

字符串是用双引号括起来的字符序列，如" hello!"。转义序列、八进制、十六进制也可以用在字符串中。字符串只能在同一行，不能换行。

【例 2 - 3】输出转义字符。

```
1  public class app2_3
2  {
3    public static void main(String args[])
4    {
5      char ch = \";  //将 ch 赋值为 \"
6      System.out.println(ch + "Spring is coming!" + ch);
7      System.out.println(" \"Birds are flying! \"");
8    }
9  }
```

【运行结果】

如图 2 - 3 所示。

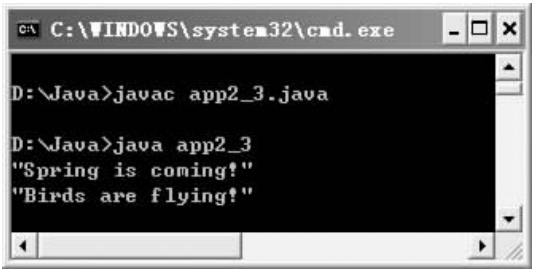

图 2 - 3　【例 2 - 3】程序运行结果

【程序分析】

第 5 句：将一个转义字符 \ " 赋值给一个字符变量。

第 6 句：+ 起到连接字符串的作用。

第 7 句：要区分开转义字符 \ " 和"。

2.3.3　项目开发背景知识 3 常量和变量

一、常量

当一个数值在程序中多处重复出现时，可以考虑使用常量来简化该数值的表示，这对程序的日后维护、修改都是极其有益的。因为当这些程序引用到的数值需要进行修改时，程序员不必逐个数值进行修改，只需要把表示该数值的常量变量的值一次性修改完毕就可以了，如以下语句：

```
final int LEFT_ARROW = 1;
final int RIGHT_ARROW = 0;
final double BASIC_SALARY = 2000.50;
```

以上语句定义了常量 LEFT_ ARROW，RIGHT_ ARROW，BASIC_ SALARY，可以看到，声明一个 Java 常量，必须在其前面使用 final 关键字。这里，常量名均是大写命名的，这是一个编程风格，使得变量与常量很容易被区分。Java 类库中的常量也是大写命名的。

常量是一种特殊的变量，也是占用内存空间的，不同于 C 语言预编译所定义的宏。常变量在声明的同时，就必须被初始化，而且在程序中不能再重新被赋值给常量。

二、变量

变量是在程序运行过程中其值可以被改变的量，通常用来记录运算中间结果或保存数据。变量包括变量名、变量值两部分。变量名就是用户自己为变量定义的标识符，而变量值则是存储在变量名中的数据，修改变量的值仅仅是改变存储单元中存储的数据，而不是改变存储数据的位置，即存储数据的位置没有改变。例如：int a = 10，改为 a = 5；等号左边的标识符 a 是一个 int 型的变量名，标识整数 10 的存储位置，改变的只是 a 的存储的内容，由整数 10 变为整数 5。

变量必须先声明后使用。变量声明是要告诉编译器根据数据类型为变量分配合适的存储空间。变量声明包括为变量命名，指定变量的数据类型，如果需要还可以为变量指定初始数值。声明变量的格式如下：

```
Type var_list[ = value];
```

其中 Type 表示 Java 的某种数据类型，可以是基本数据类型也可以是类类型。var_ list 则为变量的标志符列表，可以同时写上多个变量，变量之间必须以逗号分隔，也可以在定义时为变量赋初值。如：

```
int  k;        //声明一个存放整型且名是 k 的变量
float  x, y;       //声明浮点型变量 x,y
double a = 1.0, b = 2.0;     //声明变量 a,b 并分别赋值 1.0,2.0
```

【例 2 - 4】源程序名“ComputeArea. Java”，计算半径为 10 的圆的面积，并显示结果。通过此案例来学习常量和变量的使用。

```
1  public class ComputeArea
2  {
3    public static void main(String[] args)
4    {
5      final double PI = 3.14159; //定义常量 PI
6      double area; //定义变量 area
7      area = PI * 10 * 10;
8      System.out.println("The area for the circle of 10 is" + area);
9    }
10 }
```

【运行结果】

如图 2 - 4 所示。

图 2 - 4　【例 2 - 4】程序运行结果

【程序分析】

第 5 句：常量在使用前予以说明和初始化。常量 PI 不能改变它的值。

第 6 句：变量的使用，定义变量 area 是双精度类型。变量 area 在计算过程中是可以随半径的改变而改变。

第 7 句：在程序运行时，给变量 area 赋值。

2.3.4　项目开发背景知识 4 运算符和表达式

一、赋值

1. 赋值运算

赋值运算的作用是使变量获得值。赋值的格式如下：

< 变量名 > = < 表达式 >

其中，“ = ”是赋值运算符，< 变量名 > 获得计算出的 < 表达式 > 的值。赋值的运算次序是从右向左的，即先计算 < 表达式 > 的值，再将表达式的结果值赋给 < 变量名 >。

例如，

```
int i,j;
i = 10;              //变量 i 获得值
```

```
i = i +1;            //变量 i 获得原先值再加 1,则 i =11
j = i +10            //变量 j 获得表达式的值,则 j =21
```

2. 赋值运算的语法错误

赋值中的变量名必须已声明，而且表达式必须能计算出确定值，否则将产生编译错误。例如，

```
k =10;
```

当 k 声明时，系统将产生“不能解析符号”的编译错误。再有，

```
int  i, k;
k = i +10;
```

系统将产生“变量 i 可能还未被初始化”的编译错误。改成如下的程序才正确：

```
int i =0, k;
k = i +10;                  //则 k =10
```

3. 赋值相容

如果参加运算的变量类型和表达式的类型是相同的，就可以赋值，称为类型相同。如果两者类型不相同，并且变量类型比表达式类型长时，系统会自动将表达式转化为较长的类型，如 int 转化为 long，这时也可以赋值，称为赋值相容(assignment compatible)。

例如，

```
long bigValue =99L;           //类型相同
long bigval =6;               //6 是 int,自动转化为 long,赋值相容
double z =12.414F             //12.414F 是 float,赋值相容
```

如果变量类型比表达式类型短，则赋值不相容，编译时产生“可能存在的精度损失”的错误。

例如：

```
int smallval =99L;             //99L 是 long,赋值不相容
float z1 =12.414;              //12.414 是 double,赋值不相容
```

赋值不相容时，需要使用强制类型转换，其格式如下：

```
(<目标类型>)   <表达式>
```

例如，

```
i =(int)99L;                  //数值的转换
int j =(int)(bigValue);          //变量的转换
```

将 long 转换为 int 后，可以赋值。

【例 2-5】简单的赋值运算。

```
1  public class app2_5
2  {
3    public static void main(String args[])
```

```
4    {
5      int age =14; //声明整数变量 age,并赋值为 14
6      System.out.println("before compute,age = " +age);//输出 age 的值
7      age =age +1;  //将 age 加 1 后再设置给 age 存放
8      System.out.println("after compute,age = " +age);//输出计算后 age
       的值
9    }
10 }
```

【运行结果】

如图 2 - 5 所示。

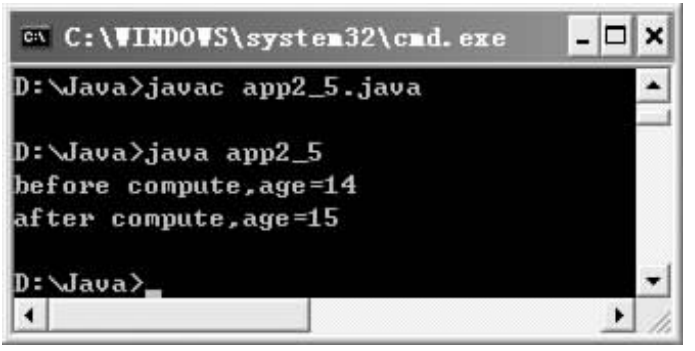

图 2 - 5　【例 2 - 5】程序运行结果

【程序分析】

第 5 句：定义整型变量 age，并赋初值。

第 6 句：输出整型变量 age 的初值。

第 7 句：修改整型变量 age 的初值。

第 8 句：输出修改后整型变量 age 的值。

二、运算符

Java 提供了十分丰富的运算符，Java 的运算符主要分为 4 类：算术运算符、位运算符、关系运算符和布尔运算符。

1. 算术运算符

Java 的算术运算符分为一元运算符和二元运算符。一元运算符只有一个操作数，而二元运算符有两个操作数参加运算。

（1）一元运算符：一元正（ + ），一元负（ - ），加 1（ + + ）和减 1（ - - ）

加 1、减 1 运算符既可放在操作数之前（如 + + i），也可放在操作数之后（如 i + + ），两者的运算方式不同。如果放在操作数之前（如 + + i），操作数先加 1 或减 1，然后将结果用于表达式的运算；如果放在操作数之后（如 i + + ），

则操作数先参加其他的运算，然后再进行加 1 或减 1。

例如：

```
int i =10, j, k, m, n;
j = +i;                    //取原值,则 j =10
k = -i;                    //取相反符号值,则 k = -10
m =i + +;                  //先 m = i,再 i = i +1,则 m =10,i =11
m = + +i;                  //先 i = i +1,再 m = i,则,m =12,i =12
n =i - -;                  //先 n = i,再 i = i -1,则,n =12,i =11
n = - -i;                  //先 i = i -1,再 n = i,则 n =10,i =10
```

一元运算符与操作数之间不允许有空格。加 1 或减 1 运算符不能用于表达式，只能用于简单变量。例如，+ +（x +1）有语法错误。

【例 2 -6】一元运算符的使用。

```
1  public class app2_6
2  {
3    public static void main(String args[])
4    {
5      int a =3,b =3;
6      System.out.print("a =" +a);  //输出 a
7      System.out.println(",a + + =" +(a + +) +",a =" +a);  //输出 a + +
8      System.out.println("a - - =" +(a - -) +",a =" +a);  //输出 a - -
9      System.out.print("b =" +b);  //输出 b
10      System.out.println(", + +b =" +( + +b) +",b =" +b);  //输出 + +b
11      System.out.println(" - -b =" +( - -b) +",b =" +b);  //输出 - -b
12    }
11 }
```

【运行结果】

如图 2 -6 所示。

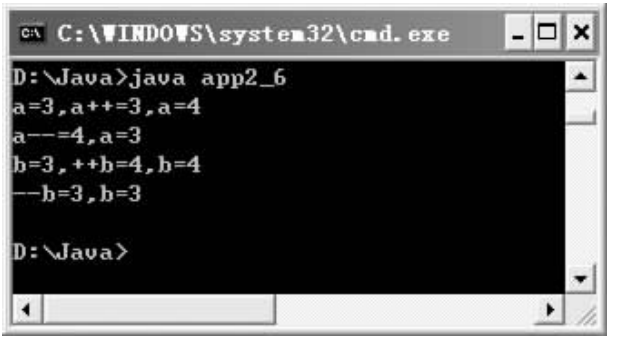

图 2 -6　【例 2 -6】程序运行结果

【程序分析】

第 6 句：先输出变量 a 的值。

第 7 句：先引用变量 a 的值计算表达式 a + +的值并输出，之后变量 a 的值

加 1 再输出变量 a 的值。

第 8 句：先引用变量 a 的值计算表达式 a - - 的值并输出，之后变量 a 的值减 1 再输出变量 a 的值。

第 9 句：先输出变量 b 的值。

第 10 句：先引用变量 b 的值计算表达式 + + b 的值并输出，之后变量 b 的值加 1 再输出变量 b 的值。

第 11 句：先引用变量 b 的值计算表达式 - - b 的值并输出，之后变量 b 的值减 1 再输出变量 b 的值。

注意：从这一个程序中一定要明白，前缀和后缀的区别。

（2）二元运算符：加（+），减（-），乘（*），除（/）和取余（%）。

其中，+，-，*，/完成加、减、乘、除四则运算，% 则是求两个操作数相除的余数。这 5 种运算符均适用于整型和浮点型。当在不同数据类型的操作数之间进行算术运算时，所得结果的类型与精度最高的那种类型一致。

例如：

```
7/2 =3                    //整除
7.0/2 =3.5                //除法
7% 2 =1                   //余数为整数
7.0% 2 =1.0               //余数为浮点数
 -7% 2 = -1               //结果的符号与被除数相同
7% -2 =1
```

例如，用 w 表示今天的星期天，如 w = 1 表示星期一，则明天、昨天分别用下式表示：

```
w =(w +1)% 7              //明天
w =(w -1 +7)% 7           //昨天
```

思考题：怎样用% 表示月份的上一个月和下一个月？

2. 位运算符

位运算是对整数中的位进行测试、置位或移位处理，是对数据进行按位操作的手段，Java 的位操作数只限于整型。Java 的位运算符有：非（~）、与（&）、或（|）、异或（∧）、右移（>>）、左移（<<）、0 填充的右移（>>>）。运算符的真值表如表 2-5 所示。

表 2-5　位运算符的真值表

A	B	A&B	A | B	A ∧ B	~A
0	0	0	0	0	1
1	0	0	1	1	0
0	1	0	1	1	1
1	1	1	1	0	0

例如：

```
~4 = -5        //等价于二进制          ~00000100 =11111011
6 |2 =6        //等价于二进制          0110 |0010 =0110
4&2 =0         //等价于二进制          0100&0010 =0000
6 ∧2 =4        //等价于二进制          0110 ∧0010 =0100
9 > >2 =1      //1001 右移 2 位为 0010
1 < <2 =4      //0001 左移 2 位为 0100
```

3．关系运算符

关系运算符用于比较两个值之间的大小，结果返回布尔值。关系运算符有 6 种：等于（ = = ）、不等于（ ! = ）、大于（ > ）、大于等于（ > = ）、小于（ < ）和小于等于（ < = ）。

例如：

```
3 < =2                    //结果值为 false
'A' <'a'                  //结果值为 true
```

4．布尔运算符

布尔运算符只能处理布尔值，所得结果都是布尔值。Java 的布尔运算符有：逻辑与（&）、逻辑或（ | ）、逻辑非（!）、逻辑异或（ ∧ ）、条件与（&&）和条件或（ || ）。布尔运算符的真值表如表 2 -6 所示。

表 2 -6　布尔运算符的真值表

A	B	A&B	A | B	A ∧ B	! A
false	false	false	false	false	true
true	false	false	true	true	false
false	true	false	true	true	true
true	true	true	true	false	false

例如：

```
(i > =0)&(i < =9)        //判断 i 值在 0 ~9 之间
(ch = ='A') |(ch = ='a')      //判断 ch 是否为字母 A,大小与同义
```

条件与（&&）的运算规则与运算符（&）相同，条件或（ || ）的运算规则与运算符（ | ）相同。区别在于：&& 和 || 具有短路计算功能，而 & 和 | 运算符没有短路计算功能。

例如：

```
(n > =100)&&(n < =999)        //判断 n 是否为一个三位数
```

所谓短路计算（short - circuit）功能，即从左向右依次逐个计算条件是否成立，一旦发现有一个条件不成立（如 n < 100），就立即终止计算并且得到复合条件的结果值为 false，实际上不用再去计算余下的条件（如 n < =999）是否成立。

例如，执行：a = -1;
b = 5;
a + + &&b - -;
结果为：a = 0　b = 5
例如，执行：a = 0;
b = 5;
a + + | | b - -;
结果为：a = 1　b = 5

5. 其他运算符

（1）赋值运算符与其他运算符的简捷使用方式。

赋值运算符可以与二元、布尔和位运算符组合成简捷使用方式，从而简化一些常用的表达式，如表 2 -7 所示。

表 2 -7　赋值运算符与其他运算符的简捷使用方式

运算符	用　法	等价于	说　明
+ =	s + = i	s = s + i	s，i 是数值型
- =	s - = i	s = s - i	s，i 是数值型
* =	s * = i	s = s * i	s，i 是数值型
/ =	s/ = i	s = s/i	s，i 是数值型
% =	s% = i	s = s% i	s，i 是数值型
& =	a& = b	a = a&b	a，b 是布尔型或整型
| =	a| = b	a = a | b	a，b 是布尔型或整型
∧ =	a∧ = b	a = a∧b	a，b 是布尔型或整型
< < =	s < < = i	s = s < < i	s，i 是整型
> > =	s > > = i	s = s > > i	s，i 是整型
> > > =	s > > > = i	s = s > > > i	s，i 是整型

（2）运算符［］和（）。

方括号［］是数组运算符，方括号［］中的数值是数组的下标，整个表达式就代表数组中该下标所在位置的元素值。

括号（）用于改变表达式中运算符的优先级。

（3）字符串合并运算符。

Java 用“ + ”运算符来合并两个字符串。当“ + ”合并一个字符串与一个操作数时，Java 会自动将操作数转化为字符串。

例如：

```
System.out.println("max = " +max);
```

这种转化对于基本数据类型是自动的，而对于引用类型则需通过调用 toS-

tring（）方法转换。

（4）三元条件运算符（?:）。

Java 语言提供了高效简便的三元条件运算符（?:）。该运算符的格式如下：

<表达式 1 >? <表达式 2 >: <表达式 3 >

该运算符的作用是：先计算 <表达式 1 > 的值，当 <表达式 1 > 的值为 true 时，则将 <表达式 2 > 的值作为整个表达式的值；当 <表达式 1 > 的值为 false 时，则将 <表达式 3 > 的值作为整个表达式的值。

例如：

```
int a =1,b =2,max;
max =a >b? a:b;                          //max 获得 a,b 之中的较大值
System.out.println("max =" +max);       //输出结果为 max =2
```

（5）对象运算符 instanceof。

对象运算符 instanceof 用来测试一个指定对象是指定类（或它的子类）的实例，若是则返回 true，否则返回 false.

（6）强制类型转换符。

Java 强制类型转换符能将一个表达式的类型强制转换为某一指定类型，格式如下：

（<类型 >）<表达式 >

（7）点运算符。

点运算符“.”的作用有两个；一是引用类中成员，二是分隔包（package）的各个域。

【例 2 -7】int 类型的溢出。

```
1  public class app2_7
2  {
3    public static void main(String args[])
4    {
5      int i =Java.lang.Integer.MAX_VALUE;
6      System.out.println("i =" +i);
7      System.out.println("i +1 =" +(i +1));
8      System.out.println("i +2 =" +(i +2));
9      System.out.println("i +2 =" +(i +2L));
10      System.out.println("i +3 =" +((long)i +3));
11    }
12 }
```

【运行结果】

如图 2 -7 所示。

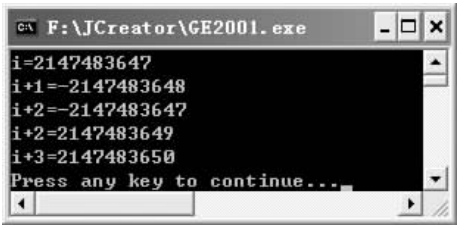

图 2－7　【例 2－7】程序运行结果

【程序分析】

第 5 句：将整形类型的最大值赋值给变量 i。

第 6 句：输出变量 i 的值。

第 7 句：i 的值加 1 变成整数类型表示范围的最小值。

第 8 句：i 的值加 2 变成整数类型表示范围的次小值，这就是数据类型的“溢出”。

第 9 句：为了避免溢出，在表达式 i＋2 的常量部分 2 之后加上 2L，执行结果会自动变成 Long 整数类型。

注意：这里用到了数据类型的自动转换，转换的规则是在数据类型兼容的情况下由表示范围小的向表示范围大的转换。

第 10 句：为了避免溢出，应用强制类型转换，执行结果会强制变成 long 整数类型。

【例 2－8】整数和浮点数的转换。

```
1  public class app2_8
2  {
3    public static void main(String args[])
4    {
5      int a =155;
6      int b =9;
7      float g,h;
8      System.out.println("a =" +a +",b =" +b); //输出 a,b 的值
9      g =a/b;          //将 a 除以 b 的结果放在 g 中
10      System.out.println("a/b =" +g +"\n");  //输出 g 的值
11      System.out.println("a =" +a +",b =" +b); //输出 a,b 的值
12      h =(float)a/b;  //将 a 除以 b 的结果放在 h 中
13      System.out.println("a/b =" +h);       //输出 h 的值
14    }
15 }
```

【运行结果】

如图 2－8 所示。

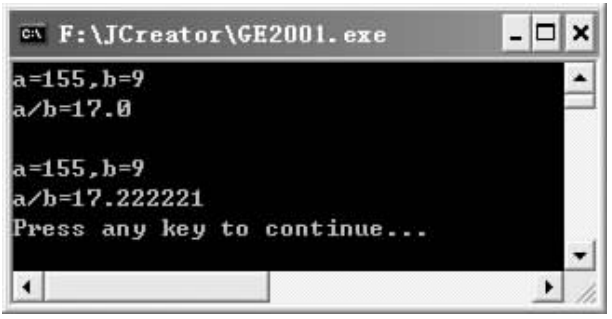

图 2－8　【例 2－8】程序运行结果

【程序分析】

第 9 句：当 2 个整数相除时，小数点以后的数字会被截断，即 155/9 = 17，当把 155/9 的结果赋值给 float 类型的变量 g 时，会发生数据自动类型转换，即 g = 17.0。

第 10 句：这里注意转义字符 \ n，表示换行。

第 12 句：强制数据类型转换。

6．运算符的优先级

表 2－8 按优先级从高到低的次序列出 Java 定义的所有运算符，分隔符的优先级最高，表中“左右”表示从左向右的运算顺序。

表 2－8　运算符的优先级

优先级	运算符	结合性
1	. [] () ; ,	右左
2	++ -- += ~ !+ -（一元）	左右
3	* / %	左右
4	+ -（二元）	左右
5	<< >> >>>	左右
6	< > <= >= instanceof	左右
7	== !=	左右
8	&	左右
9	∧	左右
10	|	左右
11	&&	左右
12	||	左右
13	?:	右左
14	= *= /= %= += -= <<= >>= >>>= &= ∧= |=	右左

三、表达式

表达式是算法语言的基本组成部分，它表示一种求值规则，通常由操作数、运算符和圆括号组成。操作数是参加运算的数据，可以是常数、常量、变量或方法引用。表达式中出现的变量名必须已经被初始化。

表达式按照运算符的优先级进行计算，求得一个表达式的值。运算符中圆括号的优先级最高，运算次序是“先内层后外层”，因此先计算由圆括号括起来的子表达式，圆括号还可以多级嵌套。大多运算符按照从左向右的次序进行计算，少数运算符的运算次序是从右向左的，如赋值运算符、三元条件运算符等。

Java 规定了表达式的运算规则，对操作数类型、运算符性质、运算结果类型及运算次序都做了严格的规定，程序员使用时必须严格遵循系统的规定，不得自定义。

由于操作数和运算符都是有类型的，因而表达式也是有类型的，表达式的类型不一定和操作数相同，它取决于其中的运算结果。

例如：

```
(i+1)*2                          //结果为 int
(i>=0)&(I<=9)                   //结果为 boolean
"Abc"+"xyz"                    //结果为 string
```

Java 表达式既可以单独组成语句，也可出现在循环条件、变量说明、方法的参数调用等场合。

【例 2-9】表达式的综合应用。首先求得这个三位数的个、十、百位上的数字，将各位数字相加就是该三位数的数字之和。

程序如下：

```
1  public class Digsum3
2  {
3    public static void main(String[] args)
4    {
5      int n=123,a=0,b=0,c=0,digsum=0;
6      a=n%10;                          //个位
7      b=(n%100)/10;                     //十位
8      c=n/100;                         //百位
9      digsum=a+b+c;
10      System.out.println("Digsum("+n+")="digsum);
11   }
12 }
```

【运行结果】

如图 2-9 所示。

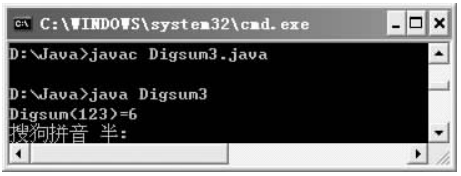

图 2－9　【例 2－9】程序运行结果

【程序分析】

第 6 句：求出这个三位数的个位；

第 7 句：求出这个三位数的十位；

第 8 句：求出这个三位数的百位。

2.3.5　项目开发背景知识 5 由键盘输入数据

在程序语言中，通过用户从键盘输入数据，不仅仅是程序的需求，更可以增加与用户之间的互动。下面的格式即为输入数据时所需要编写的基本架构。

```
import Java.io.*;   //载入 Java.io 类库里的所有类
public class class_name
{
  public static void main(String args[]) throws IOException
  {
     BufferedReader buf;//声明 buf 为 BufferedReader 类的变量
     String str; //声明 str 为 String 类的变量
     …
     buf = new BufferedReader(new InputStreamReader(System.in));//产生 buf 对象
     str = buf.readLine();  //读入字符串至 buf
     …
  }
}
```

由键盘输入的数据，不管是文字还是数字，Java 皆视为字符串，因此若是键盘输入的数字，则必须再经过转化，接下来看几个例子。

【例 2－10】由键盘输入字符串。

```
1  import Java.io.*;   //载入 Java.io 类库里的所有类
2  public class app2_11
3  {
4    public static void main(String args[]) throws IOException
5    {
6      BufferedReader buf;
7      String str;
8      buf = new BufferedReader(new InputStreamReader(System.in));
9      System.out.print("Input a string:");
```

```
10      str = buf.readLine();  //将输入的文字指定给字符串变量 str 存放
11      System.out.println("string = " + str);   //输出字符串
12    }
13  }
```

【运行结果】

如图 2 – 10 所示。

图 2 – 10　【例 2 – 10】程序运行结果

【程序分析】

第 1 句：import 是加载 Java. io. * 类库里的所有的类，供以后的代码程序使用。

第 4 句：throws IOException 属于 Java 异常的内容，将在项目 4 学习。

第 12 句：会等待用户输入数据，输入完毕后按下【Enter】键，所输入的文字都会被设给字符串变量 str，根据用户输入的不同，输出的内容也不同。

【例 2 – 11】由键盘输入整数。

```
1  import Java.io.*;
2  public class app2_12
3  {
4    public static void main(String args[]) throws IOException
5    {
6      int num;
7      String str;
8      BufferedReader buf;
9      buf = new BufferedReader(new InputStreamReader(System.in));
10      System.out.print("Input an integer:");
11      str = buf.readLine();     //将输入的文字指定给字符串变量 str 存放
12      num = Integer.parseInt(str); //将 str 转成 int 类型后指定给 num 存放
13      System.out.println("The integer is " + num);
14    }
15  }
```

【运行结果】

如图 2 – 11 所示。

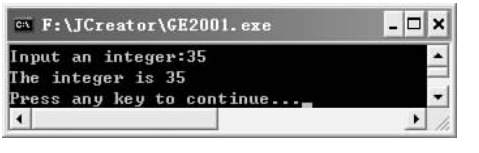

图 2 – 11　【例 2 – 11】程序运行结果

【程序分析】

第 12 句：由于从键盘输入的数据，Java 皆视为字符串，因此想要取得数值，必须做一个转换的操作，这条语句就是将输入的字符串转换成 int 类型的数值。若是想输入其他类型的数值，可以利用表 2 -9 中的转换的方法。

表 2 -9　字符串转换成数值类型的 method

数据类型	转换的 method
Long	Long. parseLong ( )
Int	Int. parseInt ( )
Short	Short. parseShort ( )
Byte	Byte. parseByte ( )
Double	Double. parseDouble ( )
Float	Float. parseFloat ( )

2. 3. 6　项目开发背景知识 6 程序流程控制

按程序的执行流程，程序的控制结构可分为 3 种：顺序结构、分支结构和循环结构。这 3 种结构的流程图如图 2 -12 所示。

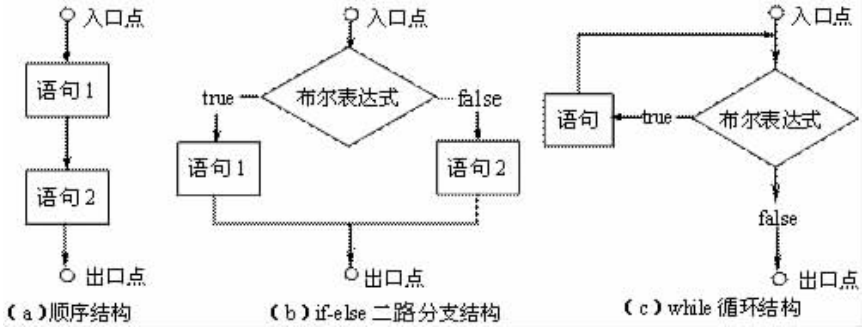

图 2 -12　程序结构流程图

一、顺序结构

一般情况下，程序按语句的书写次序依次顺序执行（sequential execution），如图 2 -12（a）所示。先执行 < 语句 1 > ，再执行 < 语句 2 > 。顺序结构是最简单的一种基本结构。

二、分支结构

程序中有些语句的执行是有条件的。当某种条件成立时，执行一段程序；条件不成立时，执行另一段程序，或不执行，这种情况称为“二路分支结构”，如图 2 -12（b）所示。在此结构中有一个菱形的判断框，它有两个分支，根据条

件 < 布尔表达式 > 是否成立（true/false）而分别执行 < 语句 1 > 或 < 语句 2 >。除此之外还有多路分支结构。

1. 简单的 if 条件语句

简单 if 条件语句只在条件为真时执行，如图 2 – 13，if 语句流程图所示。其语法如下。

```
if(条件)
{
语句(块);
}
```

若布尔表达式的值为真，则执行块内语句。

【例 2 – 12】简单的 if 条件语句。

```
1  public class app2_ 13
2   {
3    public static void main (String args [])
4     {
5     int score = 75;
6     if (score > = 60)
7     {System.out.println (" 你及格了");}
8     if (score < 60)
9      System.out.println (" 你没有及格");
10    }
11  }
```

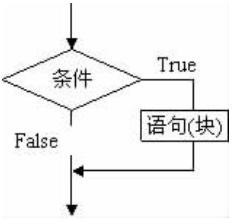

图 2 – 13　if 语句流程图

【运行结果】

如图 2 – 14 所示。

图 2 – 14　【例 2 – 12】程序运行结果

【程序分析】

第 6、8 句：在 if 子句末不能加分号（;），在 if 语句中，布尔表达式总应该用括号括住。

第 10 句：如果块中只有一条语句，花括号可以省略。但建议使用花括号以避免编程错误。

2. 简单 if – else 条件语句

当指定条件为真时简单 if 语句执行一个操作，当条件为假时什么也不做。那么，如果需要在条件为假时选择一个操作，则可以使用 if – else 语句来指定不同

的操作，如图 2 – 15 if – else 语句流程图所示。下面是这种语句的语法。

```
if(布尔表达式)
{
 布尔表达式为真时执行的语句 1（块）;
}
else
{
 布尔表达式为假时执行的语句 2（块）;
}
```

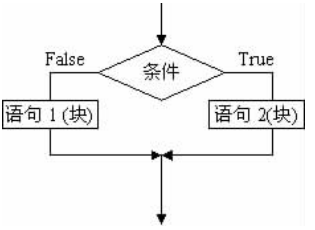

图 2 – 15　if – else 语句流程

若布尔表达式计算为真，执行语句 2（块）(true 时执行)，否则，执行语句 1（块）(false 时执行)。

【例 2 – 13】简单 if – else 条件语句。

```
1  public class app2_14
2  {
3    public static void main(String args[])
4    {
5      int score =75;
6      if(score > =60)
7      {
8        System.out.println("你及格了");
9      }
10      else
11      {
12        System.out.println("你没有及格");
13      }
14    }
15  }
```

【运行结果】如图 2 – 16 所示。

图 2 – 16　【例 2 – 13】程序运行结果

【程序分析】

第 6 句：在 if 语句中，布尔表达式总应该用括号括住；

第 7 ~ 9 句：语句 1（块），如果块中只有一条语句，花括号可以省略。但建议使用花括号以避免编程错误；

第 11 ~ 13 句：语句 2（块）。

3. if 语句的嵌套

if 或 if – else 语句中的语句可以是任意合法的 Java 语句，包括其他 if 或 if – else 语句。内层的 if 语句称为嵌套在外层 if 语句中。内层 if 语句又可以包含另一个 if 语句，事实上，嵌套的深度没有限制。

```
if(Score<60)
System.out.println("不及格");
else
if(Score<80)
  System.out.println("及格");
else
  if(Score<90)
    System.out.println("良好");
  else
    System.out.println("优秀:);
```

这个 if 语句的执行过程如下：测试第一个条件（Score < 60），若真，显示”不及格”；若假，测试第二个条件（Score < 80），若第二个条件为真，显示”及格”；若假，继续测试第三个条件（Score < 90），若第三个条件为真，显示”良好”，否则显示”优秀”。注意，只有在前面的所有条件都为假时才进行测试下一个条件。

前面的 if 语句与下述语句等价：

```
if(score<60)
  System.out.println("不及格");
else if (score<80)
  System.out.println("及格");
else if (score<90)
  System.out.println("良好");
else
  System.out.println("优秀");
```

事实上，这是多重选择 if 语句比较好的书写风格。这个风格可以避免深层缩进，并使程序容易阅读。

注意：else 子句与同一块中离得最近的 if 子句相匹配。

【例 2 – 14】if 语句的嵌套

```
1  public class app2_15
2  {
3    public static void main(String args[])
4    {
5      int score=75;
```

```
6      if(score<60)
7        System.out.println("不及格");
8      else if (score<80)
9        System.out.println("及格");
10      else if (score<90)
11        System.out.println("良好");
12      else
13        System.out.println("优秀");
14    }
15  }
```

【运行结果】

如图 2－17 所示。

图 2－17　【例 2－14】程序运行结果

【程序分析】

第 7～13 句：if 与 else 的嵌套问题

4．switch 语句

在编程的过程中还经常会碰到需要测试指定的变量是否等于某一个值的现象，如果不匹配，则对其他的值进行再匹配，这一点很像 if－else － if－else 的形式。但是当值很多的时候，用这种形式将变得非常麻烦，而且使程序变得很难懂。例如分别判断 5 分制成绩的时候，用 if－else－if－else 的形式会是以下的情况。

```
if(Score=5)
  System.out.println("优秀");
else if(Score=4)
  System.out.println("优良");
else if(Score=3)
  System.out.println("良好");
else if(Score=2)
  System.out.println("及格");
else
  System.out.println("不及格");
```

代码不简练，如果有块语句的时候或者是条件不是 5 个而是 10 个甚至更多个的时候，代码会更加的不清晰。为了避免 if－else 语句嵌套引起阅读和运行上的错误，可以使用 switch 语句，switch 语句根据表达式的结果来执行多个可能操

作中的一个。

语法形式如下：

```
switch（表达式）
{
case 常量1：语句1；[break;]
case 常量2：语句2 ；[break;]
…
case 常量n：语句n ；[break;]
[default：缺省处理语句;break;]
}
```

switch 语句中的每个“case 常量:”称为一个 case 子句，代表一个 case 分支的入口。switch 语句的流程图如图 2－18 所示。

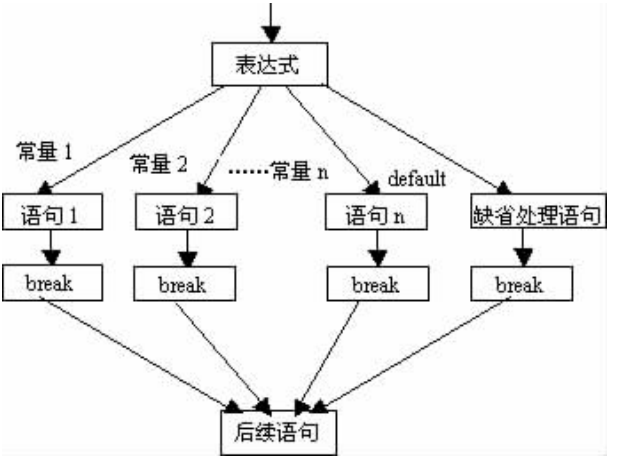

图 2－18　switch　语句流程图

Switch 语句遵从下述规则：

（1）switch 语句先计算括号中表达式的值。

（2）常量 1……常量 *N* 必须与 switch 表达式的值具有相同的数据类型。当表达式的值与 case 语句的值相匹配时，执行该 case 分支语句。直到碰到 break 或}跳出 switch 语句。

（3）关键字 break 是可选的。break 语句终止整个 switch 语句。若 break 语句不存在，下一个 case 语句将被执行。

（4）默认情况（default）是可选的，它用来执行指定情况都不为真时的操作。默认情况总是出现在 switch 语句块的最后。

【例 2－15】switch 语句的使用。

```
1  public class app2_16
2  {
3    public static void main(String args[])
```

```
4    {
5      int score = 75,a = 0;
6      a = score/10;
7      switch(a)
8      {
9       case 10:
10       case 9:System.out.println ("优秀");
11       break;
12       case 8:System.out.println ("优良");
13       break;
14       case 7:System.out.println ("良好");
15       break;
16       case 6:System.out.println ("及格");
17       break;
18       case 5:
19       case 4:
20       case 3:
21       case 2:
22       case 1:
23       case 0:System.out.println ("不及格");
24       break;
25       default:
26       break;
27      }
28   }
29 }
```

【运行结果】

如图 2 – 19 所示。

图 2 – 19　【例 2 – 15】程序运行结果

【程序分析】

第 9 ~ 10 句：当成绩为 90 ~ 100 的时候，成绩的等级均为优秀，也是说 case 10 和 case 9 这 2 个分支具有相同的动作，因此可以只在最后一个分支上写上执行语句。

第 18 ~ 23 句：道理同第 9 ~ 10 句的分析。

第 14 句：因为 a 的值为 7，执行 case 7 之后的分支语句，输出结果为良好，

之后，碰到 break 语句，结束 switch 语句的执行。

第 15 句：如果之后缺少第 15 句的 break 语句，结果会怎样？自己修改程序并运行，领会 break 语句的作用。

三、循环结构

有些程序段在某种条件下需要重复执行多次，从而形成循环结构，如图 2 - 12（c）所示。当条件满足（即 < 布尔表达式 > 为 true）时，反复执行 < 语句 >，一旦条件不满足（即 < 布尔表达式 > 为 false）时就不再执行 < 语句 >，循环结束，执行下一个基本结构。它的特点是“先判断，后执行”。如果在开始时条件就不满足，循环就不执行。

Java 的 3 种循环语句有 while、do - while 和 for 循环语句。

1. while 循环

它的执行过程如图 2 - 20 所示，其语法如下。

```
while(条件)
{
循环体
}
```

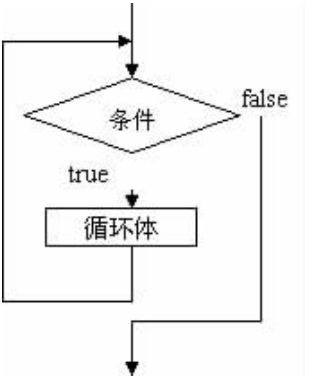

图 2 - 20　while 语句流程

说明：循环条件是一个布尔表达式，它必须放在括号中。在循环体执行前一定先计算循环条件，若条件为真，执行循环体，若条件为假，整个循环中断并且程序控制转移到 while 循环后的语句。

【例 2 - 16】while 循环的使用。

```
1  public class app5_3
2  {
3    public static void main(String args[])
4    {
5      int i =1,sum =0;
6      while(i < =10)
7      {
8        sum + =i;  //累加计算
9        i + +;
10     }
11     System.out.println("1 +2 +... +10 =" +sum);  //输出结果
12   }
13 }
```

【运行结果】

如图 2 - 21 所示。

图 2－21　【例 2－16】程序运行结果

【程序分析】

第 6 行：进入 while 循环的判断条件 i < =10，第一次进入循环时，由于 i 的值为 1，所以判断条件为真，进入循环主体。

第 7～10 行：为循环主体，sum + i 的值再指定给 sum 存放，i 的值加 1，再回到循环起始处，继续判断 i 的值是否仍在限定的范围内，直到 i 大于 10 即会跳出循环，表示累加的操作已经完成，最后将结果 sum 的值输出即可。

2．do－while 循环语句

do 循环其实就是 while 循环的变体。它的执行过程如图 2－22 所示，其的语法如下。

```
do
{
循环体;
}while(条件);
```

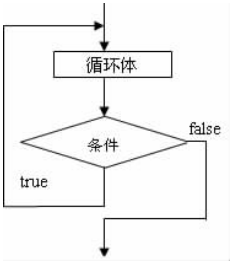

图 2－22　do 语句流程

注意：在 do 循环中 while 条件判断之后需要添加一个分号。

do－while 的循环流程是和 while 循环不一样的，二者的主要差别在于循环条件和循环体的执行顺序不同。另外，当循环初始条件满足时，二者没有区别，功能结果一样，但是当循环初始条件不满足时，while 循环的循环主体一次也不执行，但 do－while 循环的循环主体至少要执行一次。

【例 2－17】do－while 循环语句的使用。

```
1  import Java.io.*;
2  public class app2_18
3  {
4    public static void main(String args[]) throws IOException
5    {
6      int n,i =1,sum =0;
7      String str;
8      BufferedReader buf;
9      buf =new BufferedReader(new InputStreamReader(System.in));
10      do{
11        System.out.print("Input upper limit:");
```

```
12          str = buf.readLine();
13          n = Integer.parseInt(str);
14        }while(n < =0);  //输入 n,n 要大于 0,否则会一直重复输入
15        do
16          sum + = i + +;  //计算
17        while(i < =n);
18        System.out.println("1 +2 +... + " +n + " = " +sum);  //输出结果
19      }
20  }
```

【运行结果】

如图 2 – 23 所示。

图 2 – 23　【例 2 – 1】程序运行结果

【程序分析】

第 10 ~ 14 行：利用 do – while 循环判断所输入的值 n 小于 0 时，会重复输入直到 n 大于 0。

第 15 ~ 17 行：再次利用 do – while 循环计算累加 1 到 n 的结果。do – while 循环语句的判断条件是循环控制变量 i 小于等于 n 时，就执行循环主体——程序的第 16 行。

3. for 循环语句

for 循环的执行过程如图 2 – 24 所示，一般地，它的语法如下。

for（循环变量初始化；循环条件；调整语句）

{

循环体；

}

for 循环语句以关键字 for 开始，然后是由括号括住的三个控制元素，循环体括在大括号内。控制元素由分号分开，控制循环体的执行次数和终止条件。下面的 for 循环可以打印" Welcome!" 100 次。

```
int i;
for (i =0;i <100;i + +)
```

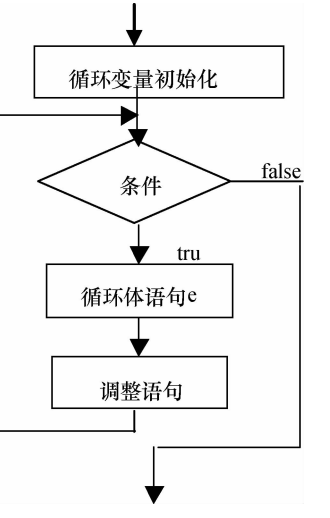

图 2 – 24　for 语句流程

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

第一个元素为 i = 0，初始化循环变量。循环变量跟踪循环体的执行次数，调整语句修改它的值。

第二个元素为 i < 100，是布尔表达式，用作循环条件。

第三个元素是调整控制变量的语句，循环变量的值最终必须使循环条件变为假。

另外，循环变量也可以在 for 循环中进行说明和初始化。上例还可写成下列语句：

```
for (int i = 0; i < 100; i + + )
{
  System.out.println("Welcome!");
}
```

【例 2 - 18】源程序 TestSum. Java，使用 for 循环计算从 1 到 100 的数列的和。

```
1   //本程序利用 for 循环计算 1 到 100 的和
2   public class TestSum
3   {
4     public static void main(String[] args)
5     {
6       int sum = 0;
7       for (int i = 1;i < = 100;i + + )
8       {
9         sum + = i;
10      }
11      System.out.println("The sum is " + sum);
12    }
13  }
```

【运行结果】如图 2 - 25 所示。

图 2 - 25　【例 2 - 18】程序运行结果

【程序分析】

第 7 ~ 10 句组成的 for 循环，变量 i 从 1 开始，每次增加 1，当 i 大于 100 时循环终止。

四、转向语句

break 和 continue 语句用于分支语句和循环语句中，使得程序员更方便地控

制程序执行的流程。

1. break 语句

break 有两种形式，一种是不带语句标号的 break，用于立刻终止包含它的最内层循环。如在 switch 语中，break 语句用来终止 switch 语句的执行。

另一种是带标号的 break，用于多重循环中，跳出它所指定的块（在 Java 中，每个代码块可以加一个括号和语句标号），并从紧跟该块的第一条语句处执行。

例如：下面 break 语句中断内层循环并把控制立即转移到外层循环后的语句。

```
outer:
for(int i =1; i <10; i + +)
{
inner:
for(int j =1; j <10;j + +);
{
if(i * j >50)
break outer;
System.out.println(i * j);
}
}
```

如果把上述语句中的 break outer 换成 break，则 break 语句终止内层循环，仍然留在外层循环中。如果想退出外循环，就要使用带标号的。

【例 2 - 19】源程序 TestBreak. Java，测试 break 语句对程序结果的影响。

```
1   //本程序测试 break 语句
2   public class TestBreak
3   {
4     public static void main(String[] args)
5     {
6       int sum =0;
7       int item =0;
8       while(item <5)
9       {
10        item + +;
11        sum + =item;
12        if(sum > =6) break;
13      }
14      System.out.println("The sum is " +sum);
15    }
16  }
```

【运行结果】

如图 2 – 26 所示。

图 2 – 26　【例 2 – 19】程序运行结果

【程序分析】

第 8 ~ 13 句的 while 循环中，如果不用 12 行的 if 语句，本程序计算从 1 到 5 的和。如果有了 if 语句，总和大于等于 6 时循环终止。

2．continue 语句

continue 语句用来结束本次循环，跳过循环体中下面尚未执行的语句，接着进行终止条件的判断，以决定是否继续循环。

【例 2 – 20】源程序“TestContinue. Java”，测试 continue 语句。

```
1   //本程序测试 continue 语句
2   public class TestContinue
3   {
4     public static void main(String[] args)
5     {
6       int sum = 0;
7       int i = 0;
8       while (i < 5)
9       {
10        i + +;
11        if(i = =2) continue;
12        sum + = i;
13      }
14      System.out.println("The sum is " + sum);
15    }
16  }
```

【运行结果】如图 2 – 27 所示。

图 2 – 27　【例 2 – 20】程序运行结果

【程序分析】

第 8 ~ 13 句的 while 循环中，continue 语句终止当前迭代，当 i 变为 2 时不再执行循环体的剩余语句，即不加到 sum 中。如果没有 if 语句，所有的项都加到 sum 中，包括 i = 2。

3. return 语句

return 语句用来使程序从方法中返回，并为方法返回一个值。return 语句的格式如下。

return 返回值。

如果 return 语句未出现在子方法中，则执行子方法的最后一条语句后自动返回到主方法。

注意：return 语句的用法详见 2. 3. 8 小节。

2.3.7 项目开发背景知识 7 数组和字符串

Java 语言中，数组是一种最简单的复合数据类型。数组是有序数据的集合，要求数组中的每个元素具有相同的数据类型，可以用一个统一的数组名和下标来唯一地确定数组中的元素，下标用 [ ] 封装，数组的元素数目称为数组长度。数组有一维数组和多维数组。

一、一维数组

数组的定义和创建是有区别的，定义只需声明数组类型，没有数组长度的要求。创建是给数组分配空间，可用 new 运算符，也可用枚举初始化来创建。

1. 一维数组的定义

（1）数据类型 [ ] 数组名；

（2）数据类型　数组名 [ ]；

数据类型可以为 Java 中任意的数据类型，包括简单类型和复合类型。例如：

```
int[] intArray;
date[] dateArray;
```

2. 一维数组的创建

当一个数组被定义以后，就可以通过下面的语法用 new 操作符创建它。

数组名 = new 数据类型 [数组大小]；

另外，定义和创建数组可以被合并在一个语句里，如下所示。

（1）数据类型 [ ] 数组名 = new 数据类型 [数组大小]；

（2）数据类型 数组名 [ ] = new 数据类型 [数组大小]；

例如：int [ ] myArray = new int [10]；

这条语句能够创建一个由 10 个 int 型元素构成的数组，为了指定数组中能够储存多少元素，给数组分配内存空间时，数组的大小必须事先给定。当一个数组创建完毕，不能再改变它的大小。

3. 一维数组的初始化

（1）静态初始化。

```
int[]  intArray = {1,2,3,4};
  String  stringArray[] = {"abc", "How", "you"};
```

（2）动态初始化。

- 简单类型的数组。

```
  int intArray[];
  intArray = new int[5];
```

- 复合类型的数组。

```
String stringArray[ ];
String stringArray = new String[3];   /* 为数组中每个元素开辟引用空间(32位) */
stringArray[0] = new String("How");  //为第一个数组元素开辟空间
stringArray[1] = new String("are");   //为第二个数组元素开辟空间
stringArray[2] = new String("you");   //为第三个数组元素开辟空间
```

4. 一维数组元素的引用

数组元素的引用方式为：

数组名［下标］

数组下标，可以为整型常数或表达式，下标从 0 开始。每个数组都有一个属性 length 指明它的长度，数组下标从 0 到 length－1。例如：intArray. length 指明数组 intArray 的长度。数组元素分别是 intArray［0］、intArray［1］、…… intArray［intArray. length－1］。

一维数组长度的获取是 数组名 . length。

【例 2－21】源程序 ArrayDemo. Java，创建一个整型数组。

```
1   //本程序创建一个整型数组
2   class ArrayDemo
3   {
4     public static void main(String[] args)
5     {
6       int[] anArray;              //声明一个整型数组
7       anArray = new int[3];       //分配存储空间
8       anArray[0] = 100;           //初始化第一个元素
9       anArray[1] = 200;           //初始化第二个元素
10      anArray[2] = 300;           //初始化第三个元素
```

```
11      System.out.println("Element at index 0: " + anArray[0]);
12      System.out.println("Element at index 1: " + anArray[1]);
13      System.out.println("Element at index 2: " + anArray[2]);
14    }
15  }
```

【运行结果】

如图 2－28 所示。

图 2－28　【例 2－21】程序运行结果

【程序分析】

在第 6～13 句先创建了一个整型数组，然后输出其中元素的值。

二、二维数组

Java 语言中，多维数组被看作数组的数组。

1. 二维数组的定义

（1）数组类型 数组名 [ ] [ ];

（2）数组类型 [ ] [ ] 数组名;

2. 二维数组的初始化

（1）静态初始化

```
int intArray[ ][ ] = {{1,2},{2,3},{3,4,5}};
```

Java 语言中，由于把二维数组看作是数组的数组，数组空间不是连续分配的，所以不要求二维数组每一维的大小相同。

（2）动态初始化。

- 直接为每一维分配空间，格式如下。

```
arrayName = new type[arrayLength1][arrayLength2];
int a[ ][ ] = new int[2][3];
```

- 从最高维开始，分别为每一维分配空间。

```
arrayName = new type[arrayLength1][ ];
arrayName[0] = new type[arrayLength20];
arrayName[1] = new type[arrayLength21];
…
```

```
arrayName[arrayLength1 - 1] = new type[arrayLength2n];
```

- 例如，二维简单数据类型数组的动态初始化如下。

```
int a[ ][ ] = new int[2][ ];
a[0] = new int[3];
a[1] = new int[5];
```

对二维复合数据类型的数组，必须首先为最高维分配引用空间，然后再顺次为低维分配空间。而且，必须为每个数组元素单独分配空间，例如：

```
String s[ ][ ] = new String[2][ ];
s[0] = new String[2];//为最高维分配引用空间
s[1] = new String[2]; //为最高维分配引用空间
s[0][0] = new String("Good");//为每个数组元素单独分配空间
s[0][1] = new String("Luck");//为每个数组元素单独分配空间
s[1][0] = new String("to");//为每个数组元素单独分配空间
s[1][1] = new String("You");//为每个数组元素单独分配空间
```

3. 二维数组元素的引用

对二维数组中的每个元素，引用方式为：

```
arrayName[index1][index2]
```

例如：num［1］［0］；

【例 2 - 22】从二维不规则数组中查找最大值，并指明最大值所在的行号和列号。

```
1  public class FindMax{
2    public static void main(String[ ] args){
3      int [][]m = {{0,1,2,3},{400,5},{8,9,10}};
4      int max = m[0][0];
5      int row = 0;
6      int column = 0;
7      for(int i = 0;i < m.length;i + + )
8       for(int j = 0;j < m[i].length;j + + ){
9         if(m[i][j] > max){
10         max = m[i][j];
11         row = i;
12         column = j;
13       }
14      }
15      System.out.println( "max = " + max + " locate at row = " + row + "
column = " + column);
16    }
17  }
```

【运行结果】

如图 2 – 29 所示。

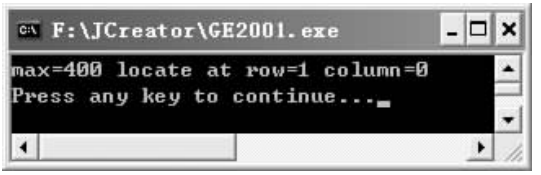

图 2 – 29　【例 2 – 22】程序运行结果

【程序分析】

第 7、8 行：是一个 for 循环的嵌套，循环变量 i、j 分别控制二维数组的行下标，及每一行元素的个数。在二维数组中，可以利用同一维数组相同的方法 . length 来获取整个数组的行数，即数组名 . length，或是某行元素的个数，即数组名［行下标］. length。

第 9、10 行：比较判断 m［i］［j］和 max 的大小。

三、字符串

Java 的字符串用关键字 String 标记，但 String 不是一个简单类型，而是 Java 定义的一个类（class），属于引用类型。在此将 String 当做一个数据类型使用，有关类的概念将在项目 3 中介绍。

1. 声明字符串变量

声明字符串变量的格式与其他变量一样，如下式声明了一个 String 类的变量 str。

```
String str;
```

同样也可以在声明时初始化变量，使变量 str 获得字符串常量值：

```
String str = "abc";
String str = ""; //包含一个空格的字符串
String str = null; //空字符串,与包含空格的字符串不同
```

还可以用另外一种的声明形式：

```
String str = new String("abc");
```

注意：字符串的声明形式还有其他的，由于篇幅限制，不再一一介绍。

字符串是引用类型，其存储方式与简单类型变量不同，两者的存储方式如图 2 – 30 所示。

2. 字符串运算

程序中可以用赋值运算为字符串变量赋值，除此之外，Java 定义“ + ”可用于两个字符串的连接运算。例如。

```
str = "abc" + "xyz"     //str 的值为"abcxyz"
```

如果字符串与其他类型变量进行“ + ”运算，系统自动将其他类型转换为字符串。例如：

```
int i =10;
String str = "i = " + i;          //str 的值为"i =10"
```

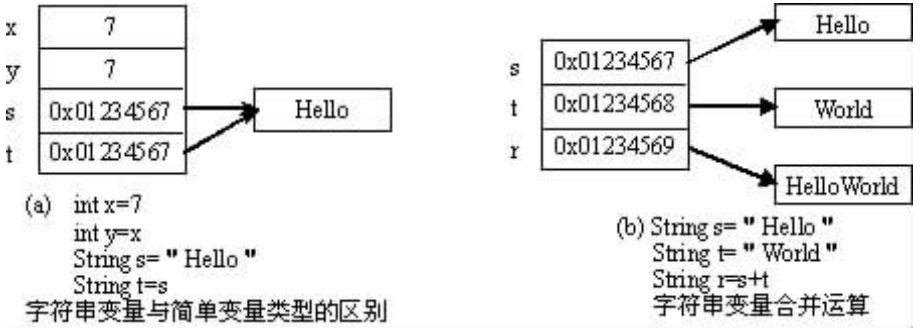

图 2 – 30　字符串变量

3. String 类的常用方法

Java 为 String 类定义了许多方法。表 2 – 9 列出了 String 类的常用方法。

表 2 – 9　String 类的常用方法

方　法	说　明
int length （）	返回字符串的长度
boolean equals （Object obj）	比较字符串是否相等
int compare To （String str）	比较字符串，返回两者之间的差值
String concat （String str）	连接字符串
String substring （int beginIndex）	返回字符串从 beginIndex 开始的子串
String substring （int beginIndex，int endIndex）	返回从 beginIndex 开始至 endIndex 的子串
char charAt （int index）	返回 index 指定位置的字符
int indexOf （String str）	返回 str 在字符串中第一次出现的位置
String replace （char oldc，char newc）	以 newc 字符替换串中所有 oldc 字符

可以通过下述格式调用 Java 定义的方法：

〈字符串变量〉.〈方法名〉

比较两个字符串有两种方法：equals （） 和 compareTo （）。

方法 equals （） 比较两个字符串是否相等，返回 booleam 类型的值。当两串相等时，返回 true，否则返回 false。

方法 compareTo （） 比较两个字符串，返回两者之间的差值 （int 型）。返回值分 3 种情况。

（1）若两个字符串 st1、st2 相等，则 st1. compareTo（st2）返回 0。

（2）若两个字符串 st1、st2 不等，则从头开始依次将两串中的每个字符进行比较，当遇到第 1 个不同字符时，st1. compareTo（st2）返回这两个不同字符的差值，即：

```
st1.charAt(k) - st2.charAt(k)                    //k 为第 1 个不同字符的位置
```

例如，设 st1 = " a1c"，st2 = " a3c"，则 st1. compareTo（st2）返回 st1. charAt（1）－st2. charAt（1）的值为 －2。

（3）若两个字符串 st1、st2 仅长度不等，则 st1. compareTo（st2）返回两者长度的差值，即：

```
st1.length() - st2.length()
```

例如，设 st1 = " abcdef"，st2 = " ab"，则 st1. compareTo（st2）返回值为 4。

注意：String 类的其他常用方法请参见 Java 相关书籍类库附录。

2.3.8 项目开发背景知识 8 函数

函数可以简化程序的结构，精简重复的程序流程，把特定功能的程序代码独立出来，达到程序模块化的目的。在 Java 里，函数称为 method，对于 method 应该不陌生，在每一个类里出现的 main（）即是 method。

method 可用如下的语法来定义：

```
返回值类型 method 名称(类型 参数 1,类型 参数 2,……)
{
程序语句;
return 表达式;
}
```

【例 2 －23】没有参数、没有返回值的 method 的使用。

```
1  public class app2_24
2  {
3    public static void main(String args[])
4    {
5      star();  //调用 star() method
6      System.out.println("I love Java");
7      star();  //调用 star() method
8    }
9    public static void star()  //star() method
10   {
11     for(int i =0;i <14;i + +)
12     System.out.print(" * ");  //输出 14 个星号
13     System.out.print(" \n");    //换行
14   }
15 }
```

【运行结果】

如图 2 - 31 所示。

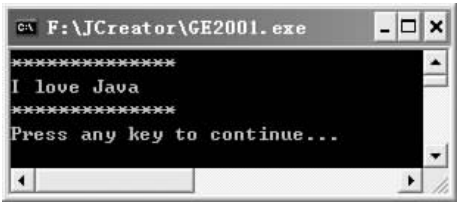

图 2 - 31　【例 2 - 31】程序运行结果

【程序分析】

第 3 ~ 8 行：程序定义 main（），main（）是程序执行的起点。

第 9 ~ 14 行：程序定义 star（）。

第 5 行：main（）调用 star（），此时程序的运行流程便会进到第 9 ~ 14 行的 star（）里执行，执行完毕后，程序返回到 main（），继续运行第 7 行。

第 8 行：main（）再次调用 star（），调用完毕后返回，接下来 main（）已经没有程序代码可供执行，于是结束程序。

注意：star（）并没有返回值，所以 star（）前面加上一个 void 关键字，即使 star（）没有参数传递，star（）后面的（）必须保留。至于 star（）前面的 static 关键字的作用将在 3. 3. 6 小节里详细介绍。

【例 2 - 24】有参数、有返回值的 method 的使用。

```
1  public class app2_25
2  {
3    public static void main(String args[])
4    {
5      int num;
6      num = star(7);  //输入 7 给 star(),并以 num 接收返回的数值 x
7      System.out.println(num + " stars printed");
8    }
9    public static int star(int n)  //star() method
10   {
11     for(int i =1;i < =2 * n;i + + )
12       System.out.print(" * ");  //输出 2 * n 个星号
13     System.out.print("\n");    //换行
14     return 2 * n;  //返回整数 2 * n
15   }
16 }
```

【运行结果】

如图 2 - 32 所示。

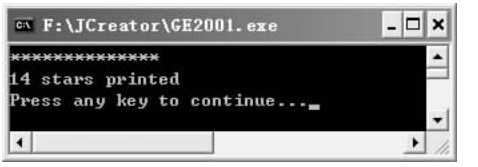

图 2 - 32　【例 2 - 24】程序运行结果

【例 2 - 25】计算长方形的面积。

```
1  public class app2_26
2  {
3    public static void main(String args[])
4    {
5      double area;
6      area = show_area(7,3);  //输入 7 与 3 两个参数到 show_area()里
7      System.out.println("area = " + area);
8    }
9    public static double show_area(int m, int n)
10     {
11       return m * n;  //返回面积 8
12     }
13 }
```

【运行结果】

如图 2 - 33 所示。

图 2 - 33　【例 2 - 25】程序运行结果

【程序分析】

第 6 行：main（）调用 show_ area（7，3），并把 7 与 3 两个参数分别给 show_ area（）里的参数吗 m，n。

第 11 行：将 show_ area（）的计算结果，通过 return 语句带回到 main（）中，并赋值给变量 area。

2.4 项目实施

【项目 1】源程序 TestWhile. Java，读入一系列整数并计算其和，输入 0 则表示输入结束。

```
1  public class TestWhile
2  {
3    public static void main(String[] args)
4    {
5      int data;
6      int sum = 0;
7      System.out.println("Enter an int value");
8      data = readInt();
9      while (data! =0)  //判断输入数据是否为 0,如不为 0,继续输入
10      {
11       sum + = data;
12       System.out.println( "Enter an int value,the program exits if
the input is 0");
13       data = readInt();
14      }
15      System.out.println("The sum is " + sum);
16    }
17  }
```

【运行结果】如图 2 – 34 所示。

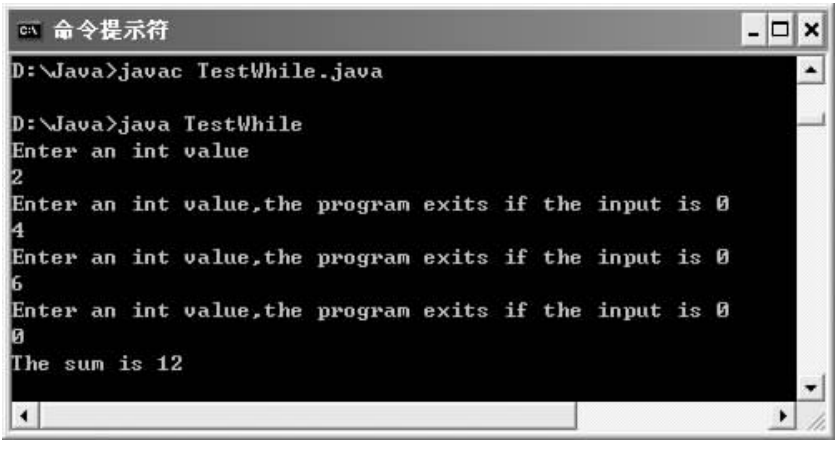

图 2 – 34　【例 2 – 26】程序运行结果

【程序分析】

第 6 句：执行 readInt 时，计算机开始等待键盘输入，直到按下回车键为止。

第 9 ~ 14 句：是 while 语句的应用。其中第 11 ~ 13 句是循环体语句。

第 12 句：while 循环中，若 data 非 0，则将它加到总和上并读取下一个输入数据。若 data 为 0，不执行循环体并且 while 循环终止。特别地，若第一个输入值为 0，则不执行循环体，结果 sum 为 0。

注意：要保证循环条件最终可以变为假，以便程序能够结束。

【项目 2】源程序 TestMulTable. Java，使用嵌套的 for 循环打印九九乘法表。

```
1  //本程序打印九九乘法表
2  public class TestMulTable
3  {
4    public static void main(String[] args)
5    {
6      System.out.print("   ");
7      for (int j =1;j < =9;j + +)
8        System.out.print("   " +j);
9      System.out.println("  ");
10      for (int i =1;i < =9;i + +)
11      {
12        System.out.print(i +"  ");
13        for (int j =1;j < =i;j + +)
14        {
15          if (i * j <10)
16           System.out.print("   " +i * j);
17          else
18           System.out.print("  " +i * j);
19        }
20       System.out.println();
21      }
22    }
23  }
```

【运行结果】

如图 2 – 35 所示。

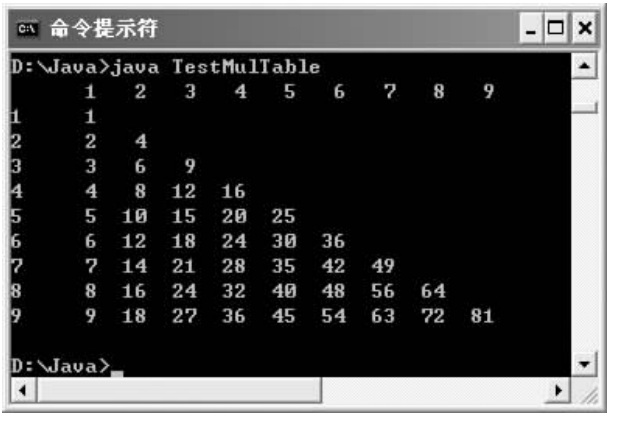

图 2 – 35　【例 2 – 27】程序运行结果

【程序分析】

第 7 ~ 9 句组成的第一个循环显示数 1 到 9；

第 10 ~ 21 句是一个嵌套的 for 循环，对每个外循环的循环变量 i，内循环的循环变量 j 都要逐个取 1，2，...，9，并显示出 i * j 的值。

第 15 ~ 18 句的 if 语句使结果右对齐。

【项目3】一个简单的计算器，完成两个整数的加、减、乘、除运算，参与运算的两个整数及运算符从命令行参数传入。例如，要计算 100 + 200，则在命令行输入：

```
Java SimpleCaulator 100 + 200
```

程序实现如下：

```
1  public class SimpleCalculator{
2    public static void main(String[] args){
3      if(args.length! =3){
4       System.out.println("Usage: Java SimpleCalculator " +
5         "operand1 operator operand2");
6       System.out.println("Example: Java SimpleCalculator 100 + 200");
7       System.exit( -1);
8      }
9      int oprand1 = Integer.parseInt(args[0]);
10      String operator = args[1];
11      int oprand2 = Integer.parseInt(args[2]);
12      int result = Integer.MIN_VALUE;
13      if(operator.equals("/")&&oprand2 = =0){
14        System.out.println("can not divide by 0!");
15        System.exit( -1);
16      }
17      if(operator.equals(" +"))
18        result = oprand1 + oprand2;
19      else if(operator.equals(" -"))
20        result = oprand1 - oprand2;
21      else if(operator.equals(" *"))
22        result = oprand1 * oprand2;
23      else if(operator.equals("/"))
24        result = oprand1 /oprand2;
25      else{
26        System.out.println("Error operator!");
27        System.exit( -1);
28      }
29      System.out.println(args[0] + args[1] + args[2] + " = " + result);
30    }
31  }
```

【运行结果】

如图 2 – 36 所示。

图 2 – 36　【例 2 – 28】程序运行结果

【项目 4】判断回文字符串。

回文是一种“从前向后读”和“从后向前读”都相同的字符串，如“rotor”是一个回文字符串。程序中使用了两种算法来判断回文字符串。

算法一：分别从前向后和从后向前依次获得源串 str 的一个字符 ch1 和 ch2，比较 ch1 和 ch2，如果不相等，则 str 肯定不是回文字符串，yes = false，立即退出循环；否则，继续比较，直到 str 的所有字符全部比较完，yes 的值仍为 true，才能肯定 str 是回文字符串。

算法二：将源串 str 反转成 temp 串，再比较两串，如果相等则是回文字符串。

程序中比较两个字符时，使用关系运算符 = =；而比较两个字符串，则需使用 equals（）方法。程序如下。

```
1  public class Rotor
2  {
3    public static void main(String[] args)
4    {
5      String str = "rotor";
6      int i = 0,n;
7      boolean yes = true;
8      if(args.length > 0)
9        str = args[0];
10     System.out.println("str = " + str);
11     n = str.length();
12     char ch1,ch2;
13     while(yes&&(i < n/2))                    //算法一
14     {
15       ch1 = str.charAt(i);
16       ch2 = str.charAt(n - i - 1);
17       System.out.println("ch1 = " + ch1 + " ch2 = " + ch2);
18       if(ch1 = = ch2)
19         i + +;
20       else
21         yes = false;
```

```
22      }
23      System.out.println("算法一:" +yes);
24      String temp = "";
25      String sub1 = "";
26      for(i =0;i <n;i + +)                    //算法二
27      {
28        sub1 =str.substring(i,i +1);
29        temp =sub1 +temp;
30      }
31      System.out.println("temp = " +temp);
32      System.out.println("算法二:" +str.equals(temp));
33    }
34  }
```

【运行结果】

如图 2 – 37 所示。

图 2 – 37　【例 2 – 29】程序运行结果

【程序分析】

程序中方法 main（String［］args）的参数称为命令行参数，是指运行时跟在文件名后输入的多个字符串，保存在 args［］数组中，其间以空格分隔。本例使用参数 args［］作为输入数据，带参数的运行命令如下：

Java　Rotor　12345

则程序运行结果如下：

str = 12345

ch1 = 1　ch2 = 5

算法一：false

temp = 54321

算法二：false

如果没有参数时，str 的值仍为“rotor”。

为突出重点问题，降低算法复杂度，本章之前的例题淡化了输入方式，多以常数为例运行程序，以后各章仍多采用常数。读者可根据需要使用命令行参数 args［］。

2.5 项目总结

标识符用于命名编程实体，如变量、常量、方法、类和包。变量是表示数据的符号，变量值在赋值语句中可以改变。所有变量在使用前必须用标识符和类型说明，引用变量前必须赋以初值。常量是表示程序中一直不变的量的符号，不能给常量赋一个新值。

Java 提供 4 种整型（byte、short、int、long），表示 4 种不同范围的整数，提供两种浮点型（float、double），表示两种不同范围的浮点数。字符型（char）表示单个字符，布尔型（boolean）有 true 或 false 两个值。

Java 提供数值操作的运算符：+（加法）、-（减法）、*（乘法）、/（除法）和%（求余）。整数除法（/）得整数解，求余运算（%）得除法的余数。

增量运算符（++）和减量运算符（--）给变量加 1 或减 1。若它们前置于变量，变量先加 1 或减 1，再用于表达式运算，若后置于变量，则变量先参与表达式运算，再加 1 或减 1。

程序控制指定了程序中语句执行的顺序。条件语句用于建立程序中的选择步骤。三种循环语句：while 循环、for 循环和 do 循环。

while 循环先检查循环条件。若条件为 true，执行循环体，若为 false，循环结束。Do 循环与 while 循环类似，只是 do 循环先执行循环体，后检查循环条件，以确定继续还是终止。由于 while 循环和 do 循环包含依赖循环体的循环条件，所以重复的次数由循环体决定。因此，while 循环和 do 循环常用于不确定循环次数的情况。

for 循环一般用于预知执行次数的循环，执行次数不是由循环体确定的。循环控制由带初始值的控制变量、循环条件和调整语句组成。

数组是最简单的复合数据类型，数组中的每个元素具有相同的数据类型，可以用一个统一的数组名和下标来唯一地确定数组中的元素。Java 中，对数组定义时并不为数组元素分配内存，只有初始化后，才为数组中的每一个元素分配空间。已定义的数组必须经过初始化后，才可以引用。数组的初始化分为静态初始化和动态初始化两种。

2.6 扩展演练

1. 判断某一年份是否是闰年（如果这个年份能被 4 整除，但不能被 100 整除；或者，如果这个年份能被 4 整除，又能被 400 整除；满足以上两个条件中的一个的年份就是闰年）。

2. 输入学生的学习成绩的等级，给出相应的成绩范围。程序要点：设 A 级为 85 分以上（包括 85 分）；B 级为 70 分以上（包括 70 分）；C 级为 60 分以上

(包括 60 分)；D 级为 60 分以下。

3. 编程实现求 Fibonacci 数列的前 10 个数字。Fibonacci 数列的定义为：

```
F1 = 1,
F2 = 1,
…
Fn = Fn - 1 + Fn - 2        (n > = 3)
```

4. 实现数组元素从小到大排序。