第3章 Java语言基础（下）

名家观点

算法＋数据结构＝程序。

——尼克劳斯·威茨(Niklaus Wirth,Pascal语言之父,1984年图灵奖获得者)

理论永远是灰色的,而实践之树常青。

——恩格斯(德国思想家,马克思主义创始人之一)

本章学习目标

- 熟悉Java运算符的应用：算术运算符、关系运算符、逻辑运算符、位运算符、赋值运算符、条件运算符、其他运算符。
- 掌握Java表达式的计算。
- 熟练掌握Java流程控制语句及其应用：if…else…、switch…case…default…finally、for、while、do…while、break/continue、return。
- 熟练掌握Java一维数组、二维数组及其应用。
- 掌握数组工具类Arrays的常用方法。
- 掌握常见英文单词。

课程思政——工匠精神

从三峡大坝、港珠澳大桥到中国高速公路网；从“北斗三号”全球卫星导航系统到“辽宁号”航空母舰、“东风快递”(东风-17弹道导弹)和“歼20”；从大疆无人机到C919、C929等国产大飞机；从华为等公司的5G技术、鸿蒙操作系统、HMS服务到量子通信技术(“墨子号”)；以及中国的“新四大发明”(高速铁路、扫码支付、共享单车和网络购物)。“中国制造”正在逐步转型为“中国智造”“中国创造”,从民用产品到国防军工产品,从引进技术到输出技术,从自主创新到制定标准,鼓舞人心的“中国制造”频频刷屏,一张张有底气的大国名片背后,是一位位大国工匠奋斗的身影。

什么是工匠精神？狭义地讲,工匠精神是指匠人在制造产品时追求高品质,一丝不苟,拥有耐心与恒心的态度和精神。广义的工匠精神则是“从业人员的一种价值取向与行为表现,与其人生观和价值观紧密相连,是其从业过程中对职业的态度和精神理念”。由此,可以认为工匠精神是“从业者为追求产品、服务的高品质而具有的高度责任感、专注甚至痴迷、持之以恒、精益求精、勇于创新等精神”。

在各行各业，许多品牌的背后是工匠精神。中药行业著名老字号“同仁堂”在300多年的风雨历程中，历代同仁堂人始终恪守“炮制虽繁必不敢省人工，品味虽贵必不敢减物力”的古训，培养“修合无人见，存心有天知”的自律意识，树立了制药过程中兢兢业业、精益求精的“严细精神”。

“坐而论道，不如起而行之”。应让工匠精神真正扎根在我们的精神价值和理想信念中。

3.1 运算符和表达式

3.1.1 机器数

1. 进位计数制

进位计数制是利用固定的数字符号和统一的规则来计数的方法，被用得最多的是十进制。在十进制中，基数为10，有0～9共计10个数码，采用“逢十进一”的规则。

二进制、八进制、十进制、十六进制的基数、数码和位权如表3-1所示。

表3-1 常用进位计数制的基数、数码和位权

进制	基数(逢基数进位)	数 码	位权	备注
二进制	2	0、1	2^n	
八进制	8	0、1、2、3、4、5、6、7	8^n	
十进制	10	0、1、2、3、4、5、6、7、8、9	10^n	
十六进制	16	0、1、2、3、4、5、6、7、8、9 A、B、C、D、E、F	16^n	

所有进制数可以采用每位上的数码乘以该位的位权再求和的方法转换为对应的十进制数。举例如下：

$$(01111000)_2 = 0\times2^7+1\times2^6+1\times2^5+1\times2^4+1\times2^3+0\times2^2+0\times2^1+0\times2^0$$
$$= (120)_{10}$$

$$(143.6)_8 = 1\times8^2+4\times8^1+3\times8^0+6\times8^{-1} = (99.75)_{10}$$

$$(63.C)_{16} = 6\times16^1+3\times16^0+C\times16^{-1} = (99.75)_{10}$$

2. 十进制数与二、八、十六进制数之间的转换

十进制数须先转换为二进制，然后将三位二进制合为一位以转换为八进制，或将四位二进制合一位以转换为十六进制。举例如表3-2所示。

表3-2 十、二、八、十六进制数的转换

数 值	位 权											
	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	128	64	32	16	8	4	2	1	0.5	0.25	0.125	
十进制99.75	0	1	1	0	0	0	1	1	1	1	0	0
十六进制63.C	6				3				C			
八进制143.6	1			4			3			6		

注意：二～十六进制数转换，按箭头方向取四位合一位，不满四位时整数最高位前补0(小数最低位后补0)。二～八进制数转换，从个位开始三位合一位，不满三位时整数最高位前补0(小数最低位后补0)。

3. 机器数的原码、反码和补码

(1) 正数的原码、反码和补码相同。

(2) 负数的反码符号位不变，数值部分按位取反。

(3) 负数的补码符号位不变，数值部分在反码最低位加1。

举例如下：－99的原码、反码和补码如表3-3所示。为简单起见，字长暂定为8位，最高位为符号位(实际上可能是32位或64位)。

表3-3 －99的原码、反码和补码示例

机器数	符号位 0正1负	数值部分7Bits						
		64	32	16	8	4	2	1
－99的原码	1	1	1	0	0	0	1	1
－99的反码	1	0	0	1	1	1	0	0
－99的补码	1	0	0	1	1	1	0	1

3-1 运算符和表达式

运算符之于程序员犹如菜刀之于厨师、宝剑之于侠客。程序员利用各种各样的运算符完成不同的运算。按运算功能划分，运算符可分为7类，如表3-4所示。

表3-4 运算符分类(按运算功能)

分　类	运　算　符
算术运算符	＋、－、＊、/、＋＋、－－、％
关系运算符	＜、＜＝、＞、＞＝、＝＝、!＝
逻辑运算符	&、&&、||、|、!、^
位运算符	&、|、～、^、>>、>>>、≪
赋值运算符	＝、＋＝、－＝、＊＝、/＝、&＝、|＝、％＝、<<＝、>>＝、>>>＝
条件运算符	?：
其他运算符	(类型)、.、[]、()、instanceof、new

【拓展知识3-1】 我眼中的程序员。程序员就像一个厨师，数据就是需要加工的食材，运算符就像厨师手中的菜刀，处理过程(算法)就是厨师的煎炒烹炸，最后输出的是一盘色香味俱全的菜品。

3.1.2 算术运算符

算术运算符包括＋、－、＊、/、＋＋、－－、％。操作数类型要求是除逻辑类型之外的基本数据类型。需要单独说明的运算符如下，其余不再赘述。

(1) /：当操作数均为整数类型时，商自动取整。

(2) ＋＋和－－：前置运算先进行自增或自减运算，再使用操作数变量的值；后置运算先使用操作数变量的值，再进行自增或自减运算。

【拓展知识3-2】 符号“＋”的三种应用。

（1）正负号中的正号。

（2）算术运算中的加法符号，如 20＋10。

（3）字符串表达式的连接操作，只要＋两侧的操作数中有一个为字符串类型，则先将另一个操作数转换为字符串类型的数据，然后进行连接操作，相当于 String 类的 concat()方法。

【示例程序 3-1】 算术运算符演示程序（ArithOprTest.java）。

功能描述： 本程序演示了算术运算符中重要而且容易出错的操作符%、/、＋。

```
01  public class ArithOprTest {
02      public static void main(String[] args) {
03          System.out.println(10 % 3);                  //取余,控制台输出 1
04          System.out.println(10/3);                    //整除,控制台输出 3
05          System.out.println(3/6 * 12);                //优先级自左向右,控制台输出 0
06          System.out.println(3.0/6 * 12);              //取余,控制台输出 6
07          //System.out.println(10/0);                  //被 0 除,出错
08          System.out.println(10 + " * " + 20);         //控制台输出 10 * 20
09          System.out.println(10 + 20 + " * ");         //控制台输出 30 *
10          System.out.println(10 + 20 + ' * ');         //控制台输出 72
11          System.out.println(" * " + 10 + 20);         //控制台输出 * 1020
12      }
13  }
```

注意：

（1）' * '的 ASCII 码为 42。

（2）第 7 行如果去掉单行注释，将抛出异常 java.lang.ArithmeticException: / by zero，程序中止运行。

3.1.3 关系运算符

关系运算符包括：＜、＜＝、＞、＞＝、＝＝、!＝，主要用于比较两个操作数的大小。

注意：

（1）当操作数是基本数据类型时直接比较它们的值。

（1）当操作数是引用类型时，用＝＝比较的是对象的地址，如果要想比较对象的内容，则使用 equals()方法。

3.1.4 逻辑运算符

逻辑运算符包括：与运算符 &、或运算符|、短路与 &&、短路或||、取反运算符!、异或运算符^。

计算机采用二进制，实现了算术运算和逻辑运算的统一。逻辑与（&）相当于算术乘（*），逻辑或（|）相当于算术加（＋），异或运算（^）是不同为 true，相同为 false。逻辑运算符如表 3-5 所示。

表 3-5 逻辑运算符

a	b	a&b 算术乘(a*b)	a|b 算术加(a+b)	!a	a^b
false 0	false 0	false 0	false 0	true 1	false 0
false 0	true 1	false 0	true 1	true 1	true 1
true 1	false 0	false 0	true 1	false 0	true 1
true 1	true 1	true 1	true 1	false 0	false 0

注意：true 相当于 1，false 相当于 0。

(1) &(|)用在整型(byte、short、char、int、long)之间时是位运算符，用在逻辑数据之间是逻辑运算符。

(2) 与 & 和短路 &&、或|和短路||的两侧要求必须是布尔表达式，区别在于：短路与判断第一个条件为 false 时，第二个条件不再计算和判断。短路或判断第一个条件为 true 时，第二个条件不再计算和判断。与和短路与、或和短路或的最大区别是运算效率不同。

【示例程序 3-2】 逻辑运算符演示程序(LogicOprTest.java)。

功能描述：本程序演示了逻辑运算符中重要而且容易出错的操作符%、/、+。

```
01  public class LogicOprTest {
02      public static void main(String[] args) {
03          boolean a = true;
04          boolean b = false;
05          System.out.println(a&b);                          //控制台输出: false
06          System.out.println(a|b);                          //控制台输出: true
07          System.out.println(!a);                           //控制台输出: false
08          System.out.println(a^b);                          //控制台输出: true
09          boolean b1 = (2 == 1)|(2 == 2)|(2 == 3)|(2 == 4);
10          System.out.println(b1);                           //控制台输出: true
11          boolean b2 = (2 == 1)||(2 == 2)||(2 == 3)||(2 == 4);
12          System.out.println(b2);                           //控制台输出: true
13          boolean b3 = (2 > 1)&(2 > 3)&(4 > 3)&(5 > 4);
14          System.out.println(b3);                           //控制台输出: false
15          boolean b4 = (2 > 1)&&(2 > 3)&&(4 > 3)&&(5 > 4);
16          System.out.println(b4);                           //控制台输出: false
17      }
18  }
```

注意：

(1) 第 9 行全部运算完才能赋值，第 11 行运算到 2==2 时结果已出运算中止、赋值，第 11 行效率更高。

(2) 第 13 行全部运算完才能赋值，第 15 行运算到 2>3 时结果已出运算中止、赋值，第

15 行效率更高。

3.1.5 位运算符

位运算只能对 byte、short、char、int、long 类型的数据进行，低于 int 型的操作数自动转换为 int(32 位)。

位运算符的学习要求包含进位计算制、进制转换、原码、反码和补码等相关知识，详细请参考 3.1.1 节。在 Java 中采用补码形式进行机器数的存储，最高位为符号位(0 代表＋，1 代表－)。Java 位运算符主要包括：&(按位与)、|(按位或)、^(按位异或)、～(按位取反)、<<(左移位)、>>(带符号位右移位)、>>>(不带符号右移位，补 0)。

a&b 运算过程如表 3-6 所示，a|b 运算过程如表 3-7 所示，a^b 运算过程如表 3-8 所示，～a 运算过程如表 3-9 所示，a≪2 运算过程如表 3-10 所示。为简单起见，字长暂定为 8 位，最高位为符号位。

```
int a = 0b0010_0111;        //39
int b = 0b0111_1101;        //125
```

表 3-6 按位与

与运算	符号位	数值部分							运算结果
a	0	0	1	0	0	1	1	1	39
b	0	1	1	1	1	1	0	1	125
a&b	0	0	1	0	0	1	0	1	37

表 3-7 按位或

或运算	符号位	数值部分							运算结果
a	0	0	1	0	0	1	1	1	39
b	0	1	1	1	1	1	0	1	125
a|b	0	1	1	1	1	1	1	1	127

表 3-8 按位异或

异或运算	符号位	数值部分							运算结果
a	0	0	1	0	0	1	1	1	39
b	0	1	1	1	1	1	0	1	125
a^b	0	1	0	1	1	0	1	0	90
a^b^b	0	0	1	0	0	1	1	1	39

表 3-9 按位取反

非运算	符号位	数值部分							运算结果
a	0	0	1	0	0	1	1	1	39
～a	1	1	0	1	1	0	0	0	－40

表 3-10 左移位

左移位	符号位	数值部分							运算结果
a	0	0	1	0	0	1	1	1	39
a≪1	0	1	0	0	1	1	1	0	78
		数值部分左移 1 位,低位补 0							

【示例程序 3-3】 移位运算符演示(ShiftTest.java)。

功能描述:本程序演示了 a&b,a|b,a^b,a^b^b,～a,a≪2 等移位运算的结果。

```
01  public class ShiftTest {
02     public static void main(String[ ] args) {
03        int a = 39;
04        int b = 125;
05        int c = - 27;
06     System.out.println("(39)10: " + Integer.toBinaryString(39));
07        //控制台输出: (39)10: 100111
08     System.out.println("(125)10: " + Integer.toBinaryString(125));
09        //控制台输出: (125)10: 1111101
10     System.out.println("( - 27)10: " + Integer.toBinaryString( - 27));
11        //控制台输出 32 位二进制,最高位为符号位: ( - 27)10: 11111111111111111111111111
12  00101
13        System.out.println(a&b);                    //与运算
14        //控制台输出: 37
15        System.out.println("a&b: " + Integer.toBinaryString(a&b));
16        //控制台输出: a&b: 100101
17        System.out.println(a|b);                    //或运算
18        //控制台输出: 127
19        System.out.println("a|b: " + Integer.toBinaryString(a|b));
20        //控制台输出: a|b: 1111111
21        System.out.println(～a);                    //按位取反,包括符号位
22        //控制台输出: - 40
23        System.out.println("～a: " + Integer.toBinaryString(～a));
24        //输出～a: 11111111111111111111111111011000
25        System.out.println(a^b);                    //异或运算: 不同 true,相同 false
26        //控制台输出: 90
27        System.out.println(a^b^b);                  //a 和 b 进行两次异或,得到 a 的原值
28        //控制台输出: 39
29        System.out.println("a << 2: " + (a << 2));  //左移位
30        //扩大 4 倍,控制台输出: 156
31     }
32  }
```

【拓展知识 3-3】 计算机为什么采用二进制?物理电路采用二进制设计原理简单,性能稳定,抗干扰能力强;二进制实现了算术运算和逻辑运算的统一;二进制运算规则简单,减法可以转换为加上负数的补码,同理,乘除运算也可以转换为加法,最后的运算归结为加法和移位。二进制编码规则简单,易于加密压缩、差错控制、存储和传输等。

3.1.6 条件运算符

语法格式如下：

```
逻辑表达式 1?表达式 2：表达式 3
```

功能：先判断逻辑表达式 1 的值，若为 true，则结果为表达式 2 的值，否则为表达式 3 的值。条件运算符相当于一个简单的 if 语句。举例如下：

```
01  int i = 10;
02  System.out.println(i < 5?"左岸": "右岸");          //控制台输出：右岸
03  System.out.println(i % 2 == 1?"奇数": "偶数");     //控制台输出：偶数
```

3.1.7 表达式

表达式是用运算符将操作数（常量、变量和方法等）连接起来，有确定值，符合 Java 语法规则的式子。

3.1.8 语句

Java 语句是构成程序的基本单元，可以对计算机发出操作指令。一般情况下，Java 语句要写到方法中，以“；”结束。

常见的 Java 语句有以下几类。

（1）package、import 语句。

（2）赋值语句。

（3）复合语句：{ …；…；}。

（4）流程控制语句：if…else，switch…case…defaut，for，while，do…while，break，continue。

（5）方法调用语句：例如 System.out.println()。

3.2 Java 流程控制语句

3-2 Java 流程控制语句

3.2.1 顺序结构

顺序结构是 3 种流程控制语句结构中最简单的一种，即语句按照书写的顺序依次执行。顺序结构流程图 3-1 所示。

3.2.2 分支结构

分支结构又称为选择结构，它根据计算所得的判断条件表达式的值来判断应选择执行哪一个流程的分支。分支结构流程图 3-2 所示。

图 3-1　顺序结构

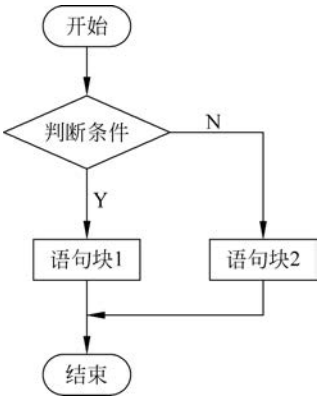

图 3-2　选择结构流程图

Java 中提供的分支语句有 if 语句和 switch 语句。if 语句可以判断布尔表达式，而 switch 只能判断表达式的内容。

1. if 语句

if 语句能根据条件从两个分支中选择一个执行。利用 if 语句的嵌套可以实现从多个分支中选择一个执行。

if 语句的语法格式如下：

```
if(条件表达式){
   语句 1;
   …
}[else{

   语句 2;
   …
}]
```

注意：

(1) 条件表达式的值应为布尔类型。

(2) 建议语句块 1、语句块 2 即使只有一句也用{}括起。

(3) 为防止出现错误，不推荐 if 嵌套。

(4) else 子句为可选内容。

【示例程序 3-4】 计算 BMI 程序(BMI.java)。

功能描述：从键盘输入身高 h(m)和体重 w(kg)，计算体质指数 BMI(Body Mass Index)。根据 BMI 输出体重状态(BMI＜18.5：偏瘦；18.5≤BMI＜24：正常；24≤BMI＜27：偏胖；27≤BMI＜30：肥胖；BMI≥30：重度肥胖)。

```
01  public class BMI {
02      public static void main(String[] args) {
03          //从键盘输入体重(kg)和身高(m)
```

```
        Scanner sc = new Scanner(System.in);
        double w = sc.nextDouble();
        double h = sc.nextDouble();
        //计算其 BMI 指数,根据 BMI 得出体重状态
        double bmi = w/(h * h);
        String result = "";
        if(bmi < 18.5) {
           result = "偏瘦";
        }
        if(bmi >= 18.5 & bmi < 24) {
           result = "正常";
        }
        if(bmi >= 24 & bmi < 27) {
           result = "偏胖";
        }
        if(bmi >= 27 & bmi < 30) {
           result = "肥胖";
        }
        if(bmi >= 30) {
           result = "重度肥胖";
        }
        //输出 BMI 和体重状态.
        System.out.printf("BMI: %.2f,status: %s",bmi,result);
    }
}
```

【拓展知识 3-4】 体重管理。1917 年，毛泽东主席在《新青年》上发表《体育之研究》中提出：欲文明其精神，先自野蛮其体魄。让我们每个人每天锻炼 1 小时，健康工作 50 年，幸福生活一辈子，进行有效的体重管理，建立一个健康的生活方式。

2. switch 语句

switch 语句用于多分支选择结构。

switch 语句的语法格式如下：

```
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: 其他语句; break; ]
}
```

switch 语句流程如图 3-3 所示。

注意：

（1）表达式必须是下述类型之一：int、byte、char、short、enum。JDK 1.7 增加了 String 类型。

（2）在 case 子句中给出的值必须是一个常量，且各 case 子句中的常量值各不相同。

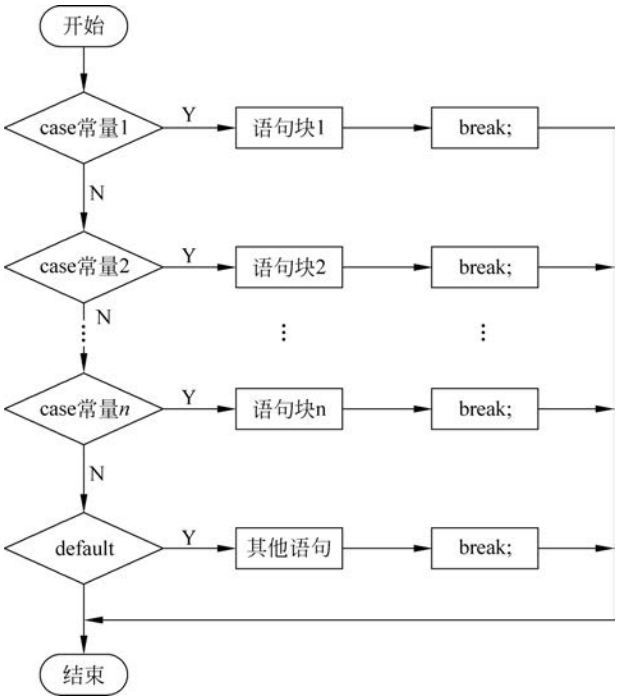

图 3-3　switch 语句流程图

(3) 如果 break 子句丢失,就会出现"穿越现象"(下面 case 条件不再判断,直接执行 case 后的语句)。

【示例程序 3-5】 根据年份和月份,输出该月的天数(Days.java)。

功能描述:本程序演示了利用 switch 语句实现根据年份和月份,输出该月天数的功能。根据规定 1、3、5、7、8、10、12 月有 31 天,4、6、9、11 月有 30 天,闰年的 2 月有 29 天,其他年份的 2 月只有 28 天。闰年指能被 4 整除不能被 100 整除;或者能被 400 整除的年份。

```
01  public class Days {
02      public static void main(String[ ] args) {
03          int year = 2000;
04          int month = 2;
05          int days = 0;
06          switch (month) {
07          case 1:
08          case 3:
09          case 5:
10          case 7:
11          case 8:
12          case 10:
13          case 12:
14              days = 31;
15              break;
16          case 4:
```

```
        case 6:
        case 9:
        case 11:
            days = 30;
            break;
        case 2:
            if ((year % 4 == 0&&year % 100!= 0)||year % 400 == 0){
                days = 29;
            } else {
                days = 28;
            }
            break;
        }
        System.out.printf("%d年%d月有%d天!",year,month,days);
    }
}
```

【拓展知识 3-5】 公历(阳历)是以地球绕太阳公转的运动周期为基础而制定的历法。地球自转一圈为一天(24小时),地球绕太阳一圈定为一年365天5小时48分46秒。一年12个月,其中1、3、5、7、8、10、12月为31天,4、6、9、11月为30天,闰年的2月为29天,其他年份为28天。平年365天,闰年366天。每四年一闰,每满百年少闰一次,到第400年再闰,即每400年中有97个闰年。

【拓展知识 3-6】 汉历(农历、阴历)是以月亮围绕地球转动的规律来制定的历法。汉历以月亮圆缺一次的时间定做一个月,共29.5天。为了计算方便,大月被定做30天,小月29天,一年12个月,大月小月交替排列,共354天。每过2～3年加一个闰月,使汉历与地球绕太阳公转相一致。

【示例程序 3-6】 根据年份和月份,输出该月的天数(Days2.java)。

功能描述：本程序演示了利用switch最新语法对示例程序3-5进行的改写,语句相对简洁。

```
public class Days2 {
    public static void main(String[] args) {
        int year = 2000;
        int month = 2;
        int days = 0;
        days = switch(month){
              case 1,3,5,7,8,10,12 -> 31;
              case 4,6,9,11 -> 30;
              case 2 -> ((year % 4 == 0&&year % 100!= 0)||year % 400 ==
0)?29: 28;
              default -> 0;
        };
        System.out.printf("%d年%d月有%s天!",year,month,days);
    }
}
```

3.2.3 循环结构

循环结构是指在一定条件下反复执行同一段语句的流程结构。

1. for 语句

一般用于已知循环次数的情况下。for 语句是当型循环,特点:先判断,后执行;循环体执行次数≥0;当循环条件为真时执行。

for 语句的语法格式如下:

```
for(设定循环变量初值; 循环条件; 修改循环变量表达式){
    循环体代码
}
```

for 语句示意图如图 3-4 所示。

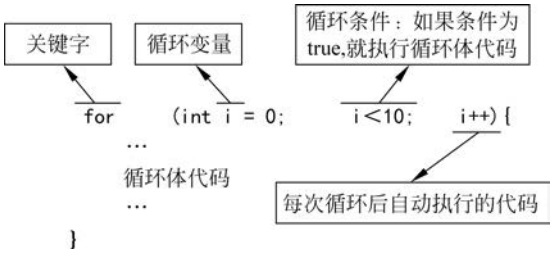

图 3-4 for 语句示意图

2. while 语句

while 语句用于已知循环条件的情况。while 语句也是当型循环,特点:先判断,后执行;循环体执行次数≥0;当循环条件为真时执行。while 语句的语法格式如下:

```
[循环前初始化语句]
while(循环条件){
循环体
    [修改循环变量的表达式]
}
```

用 for 语句和 while 语句实现无限循环示例代码如下:

```
01  //用 for 语句实现无限循环
02  for(; ; ){
03  }
04  //用 while 语句实现无限循环
05  while(true){
06  }
```

3. do while 语句

do…while 语句是直到型循环,do…while 语句的特点:先执行,后判断;循环体执行次数>=1;当循环条件为真时执行。do…while 语句的语法格式如下:

```
[循环前初始化语句]
do{
循环体
    [修改循环变量表达式]
} while(循环条件)
```

当型循环流程图如图 3-5 所示，直到型循环流程图如图 3-6 所示。

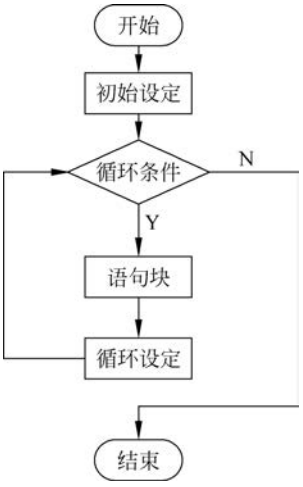

图 3-5 当型循环流程图

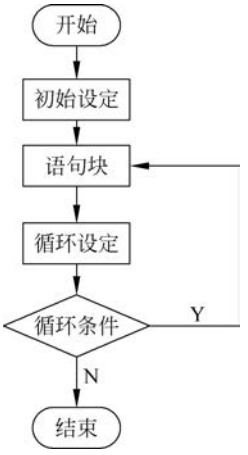

图 3-6 直到型循环流程图

【示例程序 3-7】 3 种方式求 1＋2＋…＋100 的和(ForWhileTest.java)。

功能描述：本程序用 for、while 和 do…while 3 种方式来求 1＋2＋…＋100 的和(累加求和)。

```
01  public class ForWhileTest {
02      public static void main(String[] args) {
03          //用 for 语句求 1 + 2 + … + 100 的和
04          int sum = 0;
05          for(int i = 1; i <= 100; i++){
06              sum += i;
07          }
08          //用 while 语句求 1 + 2 + … + 100 的和
09          sum = 0;
10          int i = 1;
11          while(i <= 100){
12              sum += i;
13              i++;
14          }
15          //用 do…while 语句求 1 + 2 + … + 100 的和
16          sum = 0;
17          i = 0;
18          do{
```

```
            sum += i;
            i++;
        } while(i <= 100);
    }
}
```

【拓展知识 3-7】 求和求积。在计算机程序中累加求和,累加器(变量)先清 0;累乘求积,累乘器(变量)先置 1。

3-3 九九乘法表和卡拉 OK

【示例程序 3-8】 在控制台输出九九乘法表(Table99.java)。

功能描述:本程序在控制台利用双重循环输出九九乘法表,其中外层变量 i 从 1 循环到 9,内层变量 j 从 1 循环到 i。

```
public class Table99{
    public static void main(String[] args) {
        //九九乘法表
        for(int i = 1; i <= 9; i++){
            for(int j = 1; j <= i; j++){
                //System.out.printf("%d * %d = %2d\t",i,j,i * j);
                System.out.print(i + " * " + j + " = " + i * j + "\t");
            }
            System.out.println();
        }
    }
}
```

【示例程序 3-9】 卡拉 OK 程序(OK.java)。

功能描述:卡拉 OK 比赛,有 N 个评委为歌手打分,评分规则:去掉一个最高分和一个最低分,然后取平均值作为歌手的最终得分。

编程提示:从键盘输入第一个评委的分数 score,既作为最高分 max,又作为最低分 min,sum 保存总分。循环输入其他评委的打分:和 max 比较,如果比 max 大,则赋值给 max;和 min 比较,如果比 min 小,则赋值给 min;累加求和存入 sum。

```
public class OK {
    public static void main(String[] args) {
        final int N = 5;
        Scanner sc = new Scanner(System.in);
        System.out.print("请输入第 1 个选手的成绩: ");
        double score = sc.nextDouble();
        double max = score;
        double min = score;
        double sum = score;
        for(int i = 2; i <= N; i++) {
            System.out.print("请输入第" + i + "个选手的成绩: ");
            score = sc.nextDouble();
            sum = sum + score;
```

```
            if(score > max) {
                max = score;
            }
            if(score < min) {
                min = score;
            }
        }
        double f = (sum - max - min)/(N - 2);
        System.out.printf("该选手的最高分: %.2f,最低分是: %.2f,最终
得分是: %.2f,",max,min,f);
    }
}
```

3.2.4 break 和 continue 语句

1. break 语句

语法格式：

```
break;
```

break 语句用于终止某个语句块的执行，使程序跳转到该语句块后的第一个语句开始执行。break 可用在 switch 语句中。

2. continue 语句

语法格式：

```
continue;
```

continue 语句用于跳过某个循环语句块的剩余部分，使程序直接执行下一次循环。

3.2.5 算法

算法(Algorithm)是指完成特定计算的一组有序操作。算法一词虽然广泛应用在计算机领域，但却完全源自数学。一个算法的优劣可以用空间复杂度和时间复杂度来衡量。

算法必须具备如下 3 个重要特性。

(1) 有穷性，执行有限步骤后，算法必须中止。

(2) 确切性，算法的每个步骤都必须确切定义。

(3) 可行性，特定算法须可以在特定的时间内解决特定问题。

主宰世界的十大算法如下。

(1) 排序算法：最基本的算法之一，最为经典、效率最高的三大排序算法：归并排序、快速排序和堆积排序。

(2) 傅里叶变换和快速傅里叶变换：用于实现时间域函数与频率域函数之间的相互转换。

(3) Dijkstra 算法：解决最短路径问题，其稳定性至今无法取代。

(4) RSA 非对称加密算法：用以解决在保证安全的情况下，如何在独立平台和用户之间分享密钥的问题。

(5) 哈希安全算法：一组加密哈希函数主要适用于数字签名标准里面定义的数字签名算法。

(6) 整数质因子分解算法：几百年来，许多数学家致力于寻找更快的整数因子分解算法。其复杂性成为现代密码学的重要理论基础。

(7) 链接分析算法：用于求解本征值问题。

(8) 比例微积分算法：通过“控制回路反馈机制”，减小预设输出信号与真实输出信号间的误差。

(9) 数据压缩算法：使数据的传输和存储更轻松，效率更高。

(10) 随机数生成算法：目前为止，计算机还没有办法生成“真正的”随机数，但伪随机数生成算法足够使用。

【示例程序 3-10】 红包算法程序(RedEnvelope.java)。

功能描述：从键盘输入红包的金额 money(元)和个数 n，输出 n 个红包(元，用数组表示)。算法要求：红包最小是 0.01 元；每个红包是[0.01，money－0.01＊(n－1)]之间的随机金额；n 个红包加起来的总和等于红包金额 money。

```
01  public class RedEnvelope {
02     //根据红包的金额(分)和红包个数生成红包的方法
03     public static int[] createRedEnvelope(int money,int num) {
04        int[] da = new int[num];
05        int j = 0;
06        if(num == 1){
07           da[0] = money;
08           return da;
09        }
10        int n = num;
11        for(int i = 0; i < money; i++){
12           long max = (long)(Math.random() * Integer.MAX_VALUE)
13  + Integer.MAX_VALUE;
14           if(max % (money - i)< n){
15              da[j++] = i + 1;
16              n -- ;
17           }
18        }
19        //分区间处理
20        for(int i = num - 1; i > 0; i -- ){
21           if(i == (num - 1)){
22              da[i] = money - da[i - 1];
23           }else{
24              da[i] = da[i] - da[i - 1];
25           }
26        }
27        return da;
28     }
```

```
29      public static void main(String[] args) {
30          Scanner sc = new Scanner(System.in);
31          System.out.print("请输入红包的金额(元)和个数,用空格分开: ");
32          double money = sc.nextDouble();
33          int num = sc.nextInt();
34          int[] ia = createRedEnvelope((int)money * 100,num);
35          double[] da = new double[num];
36          for(int i = 0; i < ia.length; i++){
37              da[i] = (ia[i]/100.0);
38          }
39          System.out.println(Arrays.toString(da));
40      }
41  }
```

【拓展知识 3-8】 微信之父张小龙。微信(WeChat)是腾讯公司于 2011 年 1 月 21 日推出的一个为智能终端提供即时通信服务的免费应用程序,由张小龙带领团队打造。张小龙由此被誉为“微信之父”。2021 年微信小程序日活超过 4.5 亿。移动互联网时代,微信取代了百度成为移动互联网的入口。商业社交互联网时代,微信将从一个熟人社交平台成为下一代移动端的操作系统。

3-4 Java 数组

3.3 Java 数组

一个变量只能存储一个数据。为方便存储一组相同数据类型的多个数据,Java 提供了数组这一数据结构。数组主要分为一维数组和二维数组。一维数组相当于高中数学中的数列,二维数组相当于线性代数中的矩阵。灵活使用数组和循环可以解决许多复杂的问题。

3.3.1 一维数组

1. 一维数组的定义(动态初始化)

语法格式如下:

```
数据类型[]数组名 = new 数组元素类型[元素个数];
```

说明:创建数组时,JVM 会根据数组元素类型自动为其赋初值:数值型默认赋值为 0,布尔型默认赋值为 false,引用类型默认赋值为 null。长度为 n 的数组合法的下标取值范围为 0～n－1。通过“数组名[下标]”的方式可访问一维数组的元素。数组下标在程序运行过程中如果超过范围系统,则会抛出下标越界异常信息:IndexOutOfBoundsException。

2. 一维数组的定义(静态初始化)

在声明一个数组时,对数组的每个元素进行赋值。

语法格式如下:

```
数据类型[]数组名 = {初值表};
```

【示例程序 3-11】 一维数组的基本应用技巧示例(ArrayTest.java)。

功能描述：本程序演示了 Java 一维数组的基本应用技巧，如数组的定义、数组动态初始化和静态初始化、数组遍历等。

```
01  package chap03;
02  public class ArrayTest{
03      public static void main(String[] args) {
04          //1.一维数组的定义(动态初始化)
05          int[] ia = new int[3];      //创建一个包含 3 个 int 元素的一维数组
06          //2.对数组元素进行赋值
07          ia[0] = 1; ia[1] = 2; ia[2] = 3;
08          //3.数组长度可以通过 ia.length 来读取
09          System.out.println(ia.length);
10          //4.一维数组的定义(静态初始化)适用于数组元素个数较少且各元素的值已知
11          double[] da = {3.14,2.728,3.45,1.2,9.8,9.1};
12          //5.用 for 实现数组的遍历
13          for(int i = 0; i < da.length; i++){
14              System.out.println(da[i]);
15          }
16          //6.用 for 语句增强实现数组的遍历,可以有效防止下标越界
17          for(double d: da){
18              System.out.println(d);
19          }
20          //7.用 Arrays.toString()输出元素较少的数组;
21          System.out.println(Arrays.toString(ia));
22      }
23  }
```

3.3.2 二维数组

1. 二维数组的定义(动态初始化)

语法格式如下：

```
数据类型[][]数组名 = new 数组元素类型[元素个数][元素个数];
```

说明：数组创建时，JVM 会根据数组元素类型自动为其赋初值：数值型默认赋值为 0，布尔型默认赋值为 false，引用类型默认赋值为 null。

例如：

(1) `int[][] a = new int[3][];` // 在声明锯齿数组时，至少要给出第一维的长度，即确定行数，每行元素的个数还不确定。

(2) `int[][] a = new int[3][4];` //创建了一个三行四列的二维数组。

2. 二维数组的定义(静态初始化)

在声明一个数组的同时对数组的每个元素进行赋值。

语法格式如下：

```
数据类型[][]数组名 = {{初值表},{初值表},{初值表},…};
```

下面举例以数组 a 为例，数组 a 定义如下：

```
int[][] a = {{1,2,3,4},{5,6},{7,8,9}};
```

（1）通过“数组名[行下标][列下标]”的方式访问二维数组的元素。

（2）二维数组的长度以及每一行元素的个数：a.length 代表二维数组的行数，a[0].length、a[1].length、a[2].length 分别代表二维数组 a 各行元素的个数，如图 3-7 所示。

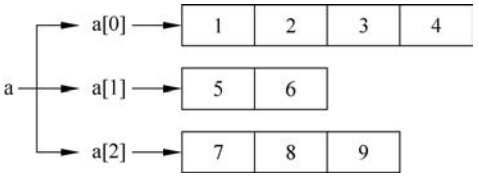

图 3-7 Java 二维锯齿型数组示意图

【示例程序 3-12】 二维数组的基本应用技巧示例(Array2Test.java)。

功能描述：本程序演示了 Java 二维数组的基本应用技巧，如数组的定义、数组动态初始化和静态初始化、二维数组的遍历等。

```
01  public class Array2Test {
02      public static void main(String[] args) {
03          // 1.二维数组的定义(动态初始化)
04          int[][] ia = new int[2][2]; // 创建一个 2 行 2 列的二维数组
05          // 2.对数组元素进行赋值,下标从 0 开始
06          ia[0][0] = 1;
07          ia[0][1] = 1;
08          ia[1][0] = 1;
09          ia[1][1] = 1;
10          // 3.二维数组的行数可以通过 ia.length 来读取
11          System.out.println(ia.length);
12          // 4.二维数组第 2 行元素的个数
13          System.out.println(ia[1].length);
14          // 5.二维数组的定义(静态初始化)适用于数组元素个数较少且各元素的值已知
15          int[][] a = {{1,2,3,4},{5,6},{7,8,9}};
16          // 6.用 for 实现二维数组的遍历
17          for (int i = 0; i < a.length; i++){
18              for (int j = 0; j < a[i].length; j++){
19                  System.out.print(a[i][j] + "\t");
20              }
21              System.out.println();
22          }
23      }
24  }
```

多重循环与数组结合,可以解决许多复杂的问题。

【示例程序 3-13】 二维矩阵乘法(MatrixMultiply.java)。

功能描述:本程序用二维数组实现矩阵乘法:C=A×B。

$$A=\begin{bmatrix}3 & 0 & 0 & 7\\ 0 & 0 & 0 & -1\\ 0 & 2 & 0 & 0\end{bmatrix},\quad B=\begin{bmatrix}4 & 1\\ 0 & 0\\ 1 & -1\\ 0 & 2\end{bmatrix},\quad C=A\times B=\begin{bmatrix}12 & 17\\ 0 & -2\\ 0 & 0\end{bmatrix}$$

```
01  public class MatrixMultiply{
02     //printMatrix 负责在控制台输出二维矩阵
03     public static void printMatrix(int[][] a){
04        for(int i = 0; i < a.length; i++){
05           for(int j = 0; j < a[i].length; j++){
06              System.out.print(a[i][j] + "\t");
07           }
08           System.out.println();
09        }
10     }
11     public static void main(String[] args){
12        //二维数组的静态初始化
13        int[][] a = {{3,0,0,7},{0,0,0, - 1},{0,2,0,0}};
14        int[][] b = {{4,1},{0,0},{1, - 1},{0,2}};
15        int[][] c = new int[3][2];
16        //计算矩阵 A 和矩阵 B 相乘
17        for(int i = 0; i < a.length; i++) {
18           for(int j = 0; j < b[j].length; j++) {
19              c[i][j] = 0;
20              for(int k = 0; k < b.length; k++) {//a[0].length
21                 c[i][j] = c[i][j] + a[i][k] * b[k][j];
22              }
23           }
24        }
25        //直接调用定义好的静态方法
26        printMatrix(a);
27        printMatrix(b);
28        printMatrix(c);
29     }
30  }
```

3.3.3 数组工具类

java.util.Arrays 类提供了一些用以操作数组的实用方法。主要掌握以下方法即可,其他方法请查阅 JDK API 文档。

(1) public static void fill(double[] a,double val):用指定值 val 去填充数组的每一个元素。

(2) public static void sort(double[] a):对指定的 double 数组按升序进行排序。

(3) public static int binarySearch(double[] a,double key):在升序排序的 double 数组中查找指定的 key。

(4) public static String toString(double[] a):将数组直接转换成一个字符串,这样开

发者就无须使用 for 语句以遍历数组。

【示例程序 3-14】 Arrays 类应用示例(ArraysTest.java)。

功能描述：本程序演示了利用数组工具类 Arrays 提供的方法实现一维数组的填充、排序等功能。

```
01  import java.util.Arrays;
02  import java.util.Collections;
03  public class ArraysTest {
04      public static void main(String args[]) {
05          int ia[] = new int[10];
06          Arrays.fill(ia,1);
07          //用 toString 方法将数组转换为字符串输出
08          System.out.println(Arrays.toString(ia));
09          int ib[] = {13,2,30,5,34,908, - 5,1200,234,9};
10          //用 sort 方法实现一维数组的升序排序
11          Arrays.sort(ib);
12          System.out.println(Arrays.toString(ib));
13          //数组排序后就可以用 binarySearch()方法进行高效的二分法查找了
14          int n = 908;
15          int pos = Arrays.binarySearch(ib, n); //返回数组元素下标
16          System.out.println(pos);
17          //将数组 ic 降序排序(ic 必须是对象数组)
18          Integer ic[] = {13,2,30,5,34,908, - 5,1200,234,9};
19          Arrays.sort(ic, Collections.reverseOrder());
20          System.out.println(Arrays.toString(ic));
21      }
22  }
```

3.3.4 在 Eclipse 中调试程序

1. 从 Java 透视图切换到 Debug 透视图

Eclipse 的 Debug 透视图如图 3-8 所示。

2. 调试程序

单击 Run→Debug 命令，启动调试。

3. 添加/删除断点

在 Editor 编辑窗口左侧栏双击空白处可添加断点，再次双击可删除断点。

4. 调试工具栏

Eclipse 的 Debug 工具栏及其注释如图 3-9 所示。

(1) Resume：停止调试，恢复正常执行，快捷键：F8。

(2) Suspend：暂停调试。

(3) Terminate：中止调试进程，快捷键：Ctrl+F2。

(4) Step Into：单步进入逐层调用的方法观察执行效果，快捷键：F5。

(5) Step Over：在当前程序的表面单步执行，快捷键：F6。

(6) Step Return：退出当前方法，返回调用层，快捷键：F7。

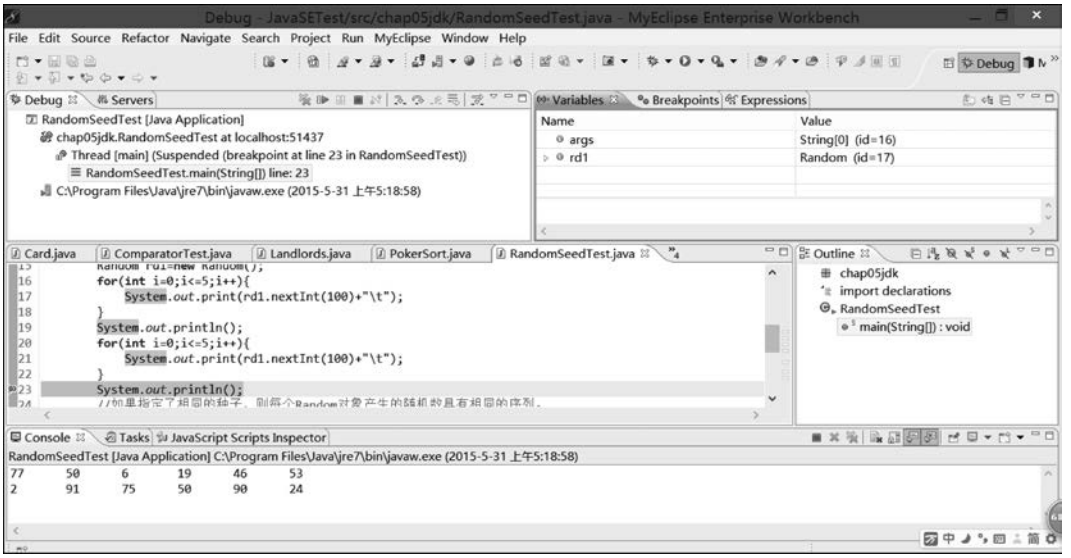

图 3-8　Debug 透视图

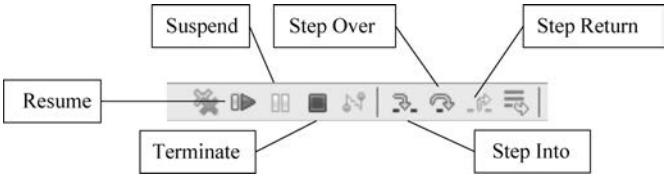

图 3-9　Debug 工具栏及其注释

5. 查看变量或表达式的值

在 Eclipse 中查看变量、表达式和断点的窗口如图 3-10 所示。

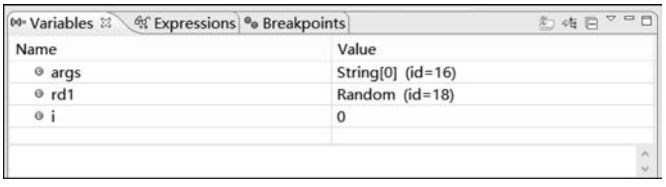

图 3-10　查看变量、表达式的值和断点的窗口

3.4　综合实例：计算两点间距离，了解北斗卫星导航系统

3.4.1　案例背景

1993 年，我国“银河号”远洋货轮在波斯湾被某大国以“莫须有”的罪名关闭其导航服务，在大海上被困 33 天，这就是震惊中外的“银河号”事件。这一事件说明现在的世界依然遵循“丛林法则”，只有国家强大起来，才能维护国家的尊严。

北斗卫星导航系统(BeiDou Navigation Satellite System，BDS)是中国自行研制的全球

卫星导航系统，也是继美国的全球定位系统(GPS)、俄罗斯的格洛纳斯(GLONASS)之后的世界上第三个成熟的卫星导航系统。

中国高度重视北斗系统的建设和发展，自20世纪80年代开始探索适合国情的卫星导航系统发展道路，形成了“三步走”发展战略：2000年年底，建成“北斗一号”系统，向中国提供服务；2012年年底，建成“北斗二号”系统，向亚太地区提供服务；2020年，建成“北斗三号”系统(55颗卫星)，向全球提供服务。在30年时间里，中国“北斗”人从无到有地建立起一套全面自主可控、技术一流、服务全球的卫星导航系统。

中国人终于可以在导航系统方面不受制于人，终于可以挺直腰杆、扬眉吐气。

3.4.2 知识准备

首先通过北斗导航系统得到地球上两个点的经纬度，然后根据半正矢公式(Haversin公式)计算这两点间的距离(精确到米)。

1. 求两个已知经纬度的点之间的距离要用到半正矢公式

$$d = 2r\arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

其中，r为地球半径，φ_1 和 φ_2 表示两点的纬度；λ_1 和 λ_2 表示两点的经度。

2. 编程中涉及Math类的相关方法

(1) public static double asin(double a)：反正弦。

(2) public static double sin(double a)：正弦。

(3) public static double sqrt(double a)：开平方。

(4) public static double cos(double a)：余弦。

(5) public static double pow(double a,double b)：a^b。

(6) public static double toRadians(double angdeg)：角度转换为弧度。

3.4.3 编程实践

【示例程序 3-15】 计算邯郸到北京的距离(DistanceCal.java)。

功能描述：北京市的经纬度分别为：116.41667°E,39.91667°N。邯郸市的经纬度分别为：114.490686°E,36.612273°N。根据两点经纬度，计算两点之间距离。

```
01  public class DistanceCal{
02      //地球半径,单位为 km
03      private static final double EARTH_RADIUS = 6378.137;
04      private double longitude;      //经度
05      private double latitude;       //纬度
06      public Point() {
07          super();
08      }
09      public Point(double longitude,double latitude) {
10          super();
11          this.longitude = longitude;
```

```
        this.latitude = latitude;
    }
    public static double getDistance(Point p1,Point p2) {
        // 纬度
        double lat1 = Math.toRadians(p1.getLatitude());
        double lat2 = Math.toRadians(p2.getLatitude());
        // 经度
        double lng1 = Math.toRadians(p1.getLongitude());
        double lng2 = Math.toRadians(p2.getLongitude());
        // 纬度之差
        double a = lat1 - lat2;
        // 经度之差
        double b = lng1 - lng2;
        // 计算两点距离的公式
        double s = 2 * Math.asin(Math.sqrt(Math.pow(Math.sin(a/2),2)
+ Math.cos(lat1) * Math.cos(lat2) * Math.pow(Math.sin(b/2),2)));
        // 弧长乘地球半径, 返回值单位为 km
        s = s * EARTH_RADIUS * 1000;
        return s;
    }
    public double getLongitude() {
        return longitude;
    }
    public void setLongitude(double longitude) {
        this.longitude = longitude;
    }
    public double getLatitude() {
        return latitude;
    }
    public void setLatitude(double latitude) {
        this.latitude = latitude;
    }
    public static void main(String[] args) {
        //北京市经纬度(116.41667,39.91667)
        //邯郸市经纬度(114.490686,36.612273)
        Point p1 = new Point(116.41667,39.91667);
        Point p2 = new Point(114.490686,36.612273);
        System.out.printf("邯郸 - 北京: %.4f",getDistance(p1, p2));
    }
}
```

3.5 本章小结

本章介绍了 Java 运算符和表达式、Java 流程控制语句及其应用、Java 数组及其应用、调试程序的常用方法等内容。

读者要仔细阅读并理解本书内容,认真理解每个示例程序,认真观看教学视频和编程微视频,并在 Eclipse 环境中进行程序编写、程序调试、程序运行,仔细辨析每个容易混淆的概

念，理解每行代码，将理论和实践相结合，从而迅速掌握 Java 语言的编程技巧。

3.6 自测题

一、填空题

1. 写出下面语句的控制台输出结果：

   System.out.println(456/100)；//控制台输出：_____

   System.out.println(123/10%10)；//控制台输出：_____

   System.out.println(789%10)；//控制台输出：_____

   System.out.println(3.0/6*12)；//控制台输出：_____

2. 异或运算程序段：

   a=01001110，b=01111101。a^b=_____，a^b^b=_____。

3. 条件运算程序段：

   int i=9；

   System.out.println(i%2==1?"奇数":"偶数")；//控制台输出：_____。

4. for 语句的无限循环语句：_____；while 语句的无限循环语句：_____。

5. 补齐代码以生成 1～10 随机的一个数字：

   int n=(int)(_____+1)；

6. 补齐代码以生成'A'～'Z'随机的一个字符：

   char c =(char)(65+_____)；

7. 补齐下面代码：

   int[] ia={3,1,7,5,2}；

   Arrays._____(ia)；//对数组 ia 进行排序(升序)

   System.out.println(Arrays._____(ia))；//输出该数组所有元素

二、选择题(SCJP 考试真题)

1. 以下语句的输出结果是(　　)。

```
01  public class SwitchTest {
02      public static int switchIt(int x) {
03          int j = 1;
04          switch(x){
05              case 1: j++;
06              case 2: j++;
07              case 3: j++;
08              case 4: j++;
09              case 5: j++;
10              default: j++;
11          }
12          return j + x;
13      }
```

```
14      public static void main(String[ ] args) {
15          System.out.println("Vaule = " + switchIt(4));
16      }
17  }
```

A. value=3　　B. value=4　　C. value=5　　D. value=6

E. value=7　　F. value=8

2. 以下语句的输出结果是(　　)。

```
01  for (int i = 0; i < 3; i++) {
02      switch(i) {
03          case 0: break;
04          case 1: System.out.print("one ");
05          case 2: System.out.print("two ");
06          case 3: System.out.print("three ");
07      }
08  }
09  System.out.println("done");
```

A. done　　B. one two done

C. one two three done　　D. one two three two three done

E. Compilation fails.

3. 以下语句的输出结果是(　　)。

```
01  for(int i = 0; i < 4; i += 2) {
02      System.out.print(i + " ");
03  }
04  System.out.println(i);
```

A. 0 2 4　　B. 0 2 4 5　　C. 0 1 2 3 4　　D. 编译失败

E. 程序运行时抛出一个异常

4. 下列哪两个语句能实例化一个数组变量?(　　)

A. int[] ia = new int[15];　　B. float fa = new float[20];

C. char[] ca = "Some String";　　D. Object oa = new float[20];

E. int ia[][] = {4,5,6}, {1,2,3};

三、编程实践

1. 斐波那契数列(Fibonacci.java)。

3-5 斐波那契数列

编程要求:斐波那契数列 c1=1,c2=1,从第 3 项开始每一项等于前两项之和。例如:1,1,2,3,5,8…要求输出斐波那契数列的前 100 项,每行输出 10 个。

编程提示:用循环实现,如图 3-11 所示。

扩展要求:输出斐波那契数列,当数列项达到 1000 时停止。

2. 求一张纸对折多少次后,其厚度能够超过珠穆朗玛峰的高度(PaperFolding.java)。

c1	c2	c3
1	1	2
1	2	3
2	3	5
…	…	…

图 3-11 用 3 个变量循环生成斐波那契数列

编程提示：珠穆朗玛峰的高度为 8848.86m，一张纸的厚度为 0.065mm。

3. 输入一个数字，打印其所有因子(Factor.java)。

3-6 打印整数的所有因子

编程要求：

从键盘上输入一个整数 n，输出其所有因子(能被其整数的数)，例如：

16＝1×2×2×2×2；

15＝1×3×5；

73＝1×73；

编程提示：

(1) 用 2 去除 n，能整除则输出×2，接着再用 2 去除 n/2，直到不能被 2 整除；

(2) 用 3 去除 n，如果能整除，则输出×3，然后再用 3 去除 n/3，直到不能被 3 整除；

(3) 依次类推，直到 n。

4. 52 周存钱法(Weeks52.java)。

编程要求："你不理财，财不理你"。假设第 1 周存 10 元钱，第 2 周 20 元，第 3 周 30 元，…，第 52 周 520 元(每周增加 10 元)，计算到第 52 周时存款共多少元。

5. 利用异或运算实现字符串的加密和解密(Encryption.java)。

3-7 利用异或实现加密解密

编程要求：从键盘上输入一个字符串(明文)，然后用一个字符(密钥，如'a')对其加密，输出密文，然后再进行解密，输出明文。

编程提示：

(1) 利用^的运算特性。

(2) 用 charAt(int n)方法依次取明文各位上的字符，循环和密钥字符进行异或运算加密，然后再进行解密。

(3) 字符数组和字符串之间的相互转换。

扩展要求：如果密钥由一个字符改为一个 String，怎么编程实现？

6. 随机生成 100 个六位随机密码(RandomKey.java)。

3-8 随机生成 100 个密码

编程要求：随机生成 100 个六位随机密码(要求第 1、3、5 位为数字，第 2、4、6 位为大写字母)存储到数组并输出。

编程提示：

(1) 随机生成一个数字字符：

```
(char)('0' + Math.random() * 10)
```

(2) 随机生成一个字母字符：

```
(char)('A' + Math.random() * 26)
```

(3) 如何将 char[]转换或 String：

```
String s = new String(ca);
```

(4) 如果要求随机密码不能重复呢？

7. 理性消费，远离套路贷(Loan.java)。

编程要求：A 同学为了买最新款的手机，在网上从小额贷款公司借了 10 000 元(本金)，日息 0.1%。A 同学涉世不深，以为一天 10 元钱，忽略了合同中隐藏的复利条款和计息周期，便签下了借款合同。问在单利和复利计息方式下，A 同学 1 年后、2 年后、3 年后、4 年后、5 年后分别需要还多少钱？

编程提示：

(1) 单利：是指按照固定的本金计算利息。单利的计算公式是：收益＝本金 *(1＋利率 * n)，n 为一个计息周期。

(2) 复利：是指第一期产生利息后，第二次的本金包括本金和第一次产生的利息。复利又叫利滚利，驴打滚。复利的计算公式是：收益＝本金 *(1＋利率)^n，n 为一个计息周期。

3-9 骰子概率统计

8. 骰子概率统计(Dice.java)。

编程要求：掷骰子 6000 次，输出每面出现的次数。

编程提示：

(1) 骰子有 6 面，需要有 6 个计数器分别计算 6 个面出现的次数。

(2) 如果采用 6 个元素的整型数组做计数器，会怎么样呢？

3-10 杨辉三角形

9. 杨辉三角形的输出(YangHui.java)。

编程要求：

(1) 用二维数组实现，杨辉三角形的分析示例如图 3-12 所示。

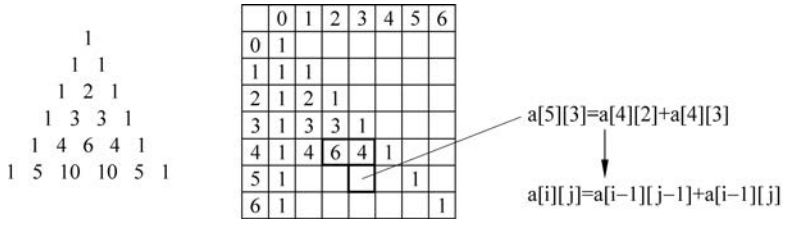

图 3-12 杨辉三角形的分析示意图

(2) 用以下组合公式实现。

$$C_m^n = \frac{m!}{n!(m-n)!}$$

```
C(0,0)
C(1,0), C(1,1)
C(2,0), C(2,1),C(2,2)
```

…

10. 输入一个小数，输出其金额大写形式(Money.java)。

编程要求：

输入金额最多两位小数，否则四舍五入。例如：

(1) 输入 123.445，输出壹佰贰拾叁元肆角伍分。

(2) 输入 123.45，输出壹佰贰拾叁元肆角伍分。

(3) 输入 123.4，输出壹佰贰拾叁元肆角零分。

(4) 输入 123，输出壹佰贰拾叁元零角零分。

3-11 输出金额大写形式

编程提示：

(1) 通过乘以 100，然后取整，以便统一处理。

(2) 各位上的数字与汉字大写数组的下标是一致的。

(3) `char[] ca = {'零','壹','贰','叁','肆','伍','陆','柒','捌','玖'};`

(4) 货币单位数组从 0 开始依次取即可。

```
String[]ma = {"分","角","元","拾","佰","仟","万","拾万","百万","千万","亿"};
```