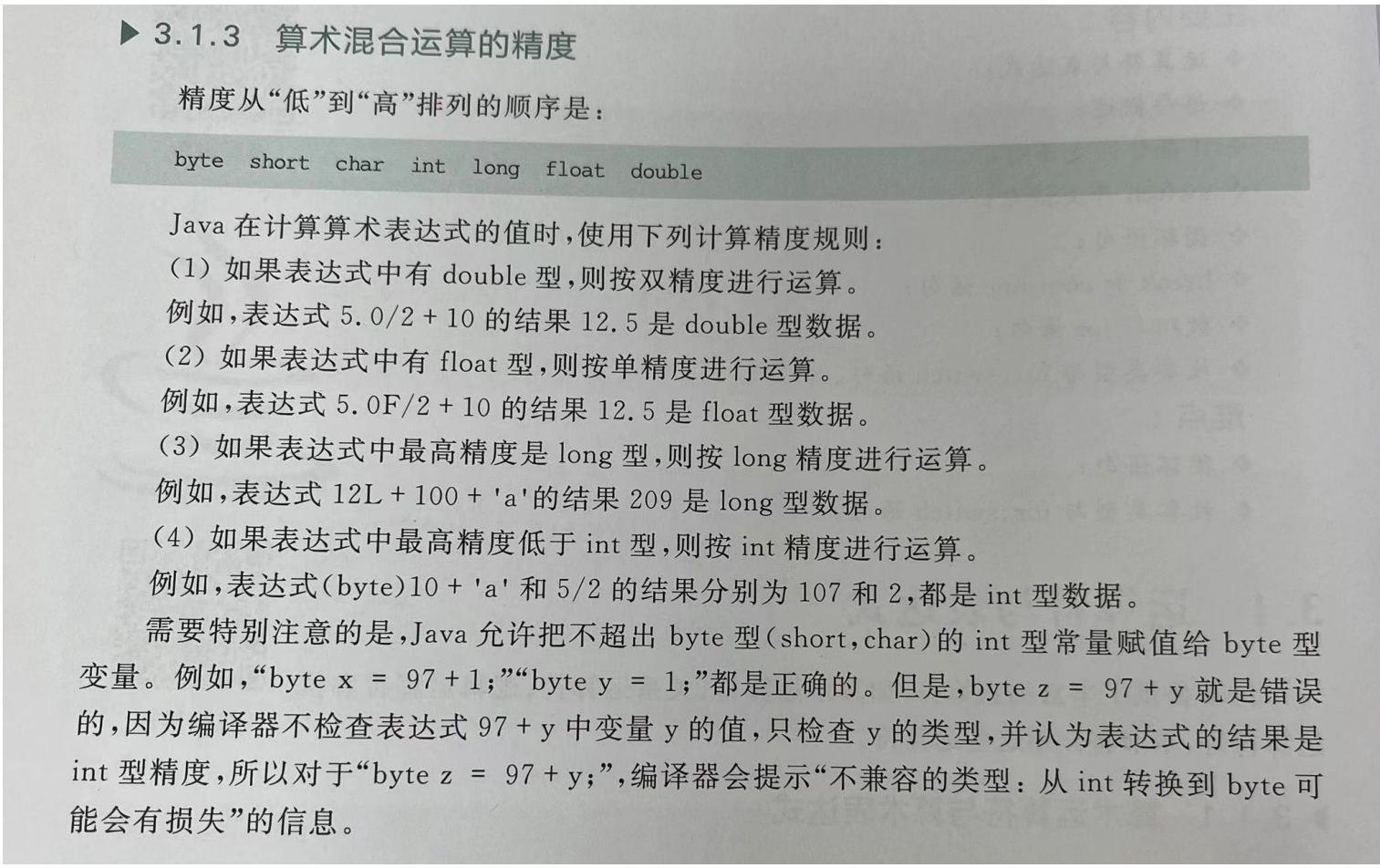
Chapter3 运算符、表达式和语句

本小节仅做对C++的补充

1.运算符和表达式

算术混合运算的精度



▶ 3.1.3 算术混合运算的精度

精度从“低”到“高”排列的顺序是：

```
byte  short  char  int  long  float  double
```

Java 在计算算术表达式的值时，使用下列计算精度规则：

(1) 如果表达式中有 double 型，则按双精度进行运算。

例如，表达式 5.0/2 + 10 的结果 12.5 是 double 型数据。

(2) 如果表达式中有 float 型，则按单精度进行运算。

例如，表达式 5.0F/2 + 10 的结果 12.5 是 float 型数据。

(3) 如果表达式中最高精度是 long 型，则按 long 精度进行运算。

例如，表达式 12L + 100 + 'a'的结果 209 是 long 型数据。

(4) 如果表达式中最高精度低于 int 型，则按 int 精度进行运算。

例如，表达式(byte)10 + 'a' 和 5/2 的结果分别为 107 和 2，都是 int 型数据。

需要特别注意的是，Java 允许把不超出 byte 型(short,char)的 int 型常量赋值给 byte 型变量。例如，“byte x = 97 + 1;”“byte y = 1;”都是正确的。但是，byte z = 97 + y 就是错误的，因为编译器不检查表达式 97 + y 中变量 y 的值，只检查 y 的类型，并认为表达式的结果是 int 型精度，所以对于“byte z = 97 + y;”，编译器会提示“不兼容的类型：从 int 转换到 byte 可能会有损失”的信息。

位运算符(数据都是补码表示的呢！)

位。int 型数据 7 的二进制表示为：

```
00000000 00000000 00000000 00000111
```

左边最高位是符号位，最高位是 0 表示正数，是 1 表示负数。负数采用补码表示，例如 -8 的二进制表示为：

```
11111111 11111111 1111111 11111000
```

这样就可以对两个整型数据实施位运算，即对两个整型数据对应的位进行运算得到一个新的整型数据。

❶ 按位与运算符

按位与运算符“&”是双目运算符，用于对两个整型数据 a、b 按位进行运算，运算结果是一个整型数据 c。运算法则是，如果 a、b 两个数据的对应位都是 1，则 c 的该位是 1，否则是 0。如果 b 的精度高于 a，那么结果 c 的精度和 b 相同。

例如：

```
      a:  00000000  00000000  00000000  00000111
&     b:  10000001  10100101  11110011  10101011
      c:  00000000  00000000  00000000  00000011
```

❷ 按位或运算符

按位或运算符“|”是二目运算符，用于对两个整型数据 a、b 按位进行运算，运算结果是一个整型数据 c。运算法则是，如果 a、b 两个数据的对应位都是 0，则 c 的该位是 0，否则是 1。如果 b 的精度高于 a，那么结果 c 的精度和 b 相同。

❸ 按位非运算符

按位非运算符“~”是单目运算符，用于对一个整型数据 a 按位进行运算，运算结果是一个整型数据 c。运算法则是，如果 a 的对应位是 0，则 c 的该位是 1，否则是 0。

❹ 按位异或运算符

按位异或运算符“^”是二目运算符，用于对两个整型数据 a、b 按位进行运算，运算结果是一个整型数据 c。运算法则是，如果 a、b 两个数据的对应位相同，则 c 的该位是 0，否则是 1。如果 b 的精度高于 a，那么结果 c 的精度和 b 相同。

由异或运算法则可知：

```
a^a=0
a^0=a
```

因此，如果 c = a ^ b，那么 a = c ^ b，也就是说，“^”的逆运算仍然是“^”，即 a ^ b ^ b 等于 a。

使用位运算符也可以操作逻辑型数据，法则如下：

(1) 当 a、b 都是 true 时，a&b 是 true，否则 a&b 是 false。

(2) 当 a、b 都是 false 时，a|b 是 false，否则 a|b 是 true。

(3) 当 a 是 true 时，~a 是 false；当 a 是 false 时，~a 是 true。

位运算符在操作逻辑型数据时，与逻辑运算符“&&”“||”“!”不同的是，位运算符要在计算完 a 和 b 之后再给出运算的结果。例如，x 的初值是 1，那么经过下列逻辑比较：

```
((y=1)==0)&&((x=6)==6));
```

运算后，x 的值仍然是 1。但是，如果经过下列位运算：

```
((y = 1) == 0))&((x = 6) == 6));
```

运算后，x 的值将是 6。

instanceof运算符（非常有用呢！）

```
a isinstanceof A
```

二目运算符，左边的操作员是一个对象，右边是一个类。

```
double d=0.0001
if(d isinstanof double)
```

2.语句

Java中的语句分为以下6种

Java 中的语句可分为以下 6 类。

视频讲解

（1）方法调用语句。例如：

```
System.out.println(" Hello");
```

（2）表达式语句。表达式语句指由一个表达式构成一个语句，即在表达式尾加上分号。例如赋值语句：

```
x = 23;
```

（3）复合语句。在 Java 中，可以用{ }把一些语句括起来构成复合语句，例如：

```
{   z = 123 + x;
    System.out.println("How are you");
}
```

（4）空语句。一个分号也是一条语句，称为空语句。

（5）控制语句。控制语句分为条件分支语句、开关语句和循环语句 3 种类型，将在后面的 3.3 节、3.4 节和 3.5 节进行介绍。

（6）package 语句和 import 语句。package 语句和 import 语句和类、对象有关，将在第 4 章讲解。

开关语句switch

```
switch(表达式)
{
   case 常量值1:
                若干语句
                break;
    ...
    default:
                若干语句
}
```

- 表达式中的值可以是byte\short\char\int\枚举类型和String类型（此处就是字符常量喽）！！！【不允许double和long的形式！！！】
  其中常量和常量的匹配只能是以“enum中定义的形式”，而不是他所对应的整数！

do...while...

```
do{
    若干语句
} while;//此处需要加引号哦！
```

3.数组与for语句（JDK中新定义的）

```
for(声明循环变量：数组的名字)
{
    ...
}
tip：不可以把声明放在for的外边呢！
```

其中，声明的循环变量的类型必须和数组相同！！！
理解：对于循环变量依次取数组的每一个元素的值！！！，此时i就代表了a[i]了呢！（是取的数组中元素的值）

```
public class test{
    public static void main(String args[]){
        int a[]={1,2,3,4};
        for(int i:a){
            System.out.println(i);
        }//循环变量i，依次取数组a的每一个元素的值！！！
    }
}
```

4.枚举类型和for、switch语句

枚举类型的名字.values()

返回一个数组，该数组元素的值和枚举类型中的常量一一对应!

ex:

weekday a[]=WeekDay.values();

此时a[0]-a[6]分别对应着sum\mon\tue...sat

例 3.9

Example3_9.java

```
enum Color {
   红,蓝,绿,黄,黑
}
public class Example3_9 {
   public static void main(String args[]) {
      for(Color a:Color.values()) {
         for(Color b:Color.values()) {
            for(Color c:Color.values()) {
               if(a!= b&&a!= c&&b!= c) {
                  System.out.print(a+","+b+","+c+" |");
               }
            }
         }
      }
   }
}
```

JDK 1.5 以后的版本允许 switch 语句中表达式的值是枚举类型(见 3.4 节)。下面的例 3.10 结合 for 语句和 switch 语句显示了 5 种水果中部分水果的价格,其中,for 语句和 switch 语句都使用了枚举类型,运行效果如图 3.11 所示。

```
C:\chapter3>java Example3_10
苹果500克的价格：1.5元
香蕉500克的价格：2.8元
芒果500克的价格：6.8元
```

图 3.11 显示部分水果的价格

例 3.10

Example3_10.java

```
enum Fruit {
  苹果,梨,香蕉,西瓜,芒果
}
public class Example3_10 {
  public static void main(String args[]) {
     double price = 0;
     boolean show = false;
     for(Fruit fruit:Fruit.values()) {
        switch(fruit) {
```

```
                case 苹果: price = 1.5;
                           show = true;
                           break;
                case 芒果: price = 6.8;
                           show = true;
                           break;
                case 香蕉: price = 2.8;
                           show = true;
                           break;
                default: show = false;
            }
            if(show) {
                System.out.println(fruit + "500 克的价格: " + price + "元");
            }
        }
    }
}
```