Java 基础（4）：Java 运算顺序的深入解析

与 C/C++不同的是，在 Java 中，表达式的计算与结果是确定的，不受硬件与环境的影响。如：

```
int i = 5;
int j = (i++) + (i++) +(i++);
```

在 C/C++中，这个例子的运算结果将会根据不同的开发环境而不同。Turbo C 下，j 的值是 15；在 VC 下，j 的值是 18。

在 Java 中，表达式的计算顺序是从左往右的，也就是先计算左侧的结果，再计算右侧的结果。上面的例子计算结果就一定是 18。也就是说，右侧(i++)表达式使用的 i 的值 就是左侧(i++)表达式计算完后 i 的值，即左侧比右侧先进行运算。

- **Example 1**

```
 int a[] = new int[]{0,0,0,0};
   int i = 1;
   a[i++] = i++;
   System.out.println("i="+i);
   System.out.println(Arrays.toString(a));
```

运行结果：

```
i=3
[0, 2, 0, 0]
```

- **Example 2**

```
   int a[] = new int[]{0,0,0,0};
   int i = 1;
   a[i++] = i = 4;
   System.out.println("i="+i);
   System.out.println(Arrays.toString(a));
```

运行结果：

```
i=4
[0, 4, 0, 0]
```

- **Example 3**

```
   int a[] = new int[]{0,0,0,0};
   int b[] = new int[]{1,2,3,4,5};
   int cc[] = a;
   int i = 1;
   a[++i] = (a=b)[i];
```

```
    System.out.println("i="+i);
    System.out.println("数组 a[]： "+Arrays.toString(a));
    System.out.println("数组 c[]： "+Arrays.toString(cc));
```

运行结果：

```
i=2
数组 a[]： [1, 2, 3, 4, 5]
数组 c[]： [0, 0, 3, 0]
```

复合运算符有一个特点：**可以自动将右侧的运算结果类型转换成左侧操作数的类型。**如：

```
byte b += 1;    //正确
     b = b+1;  //错误，1 是整形 int，所以右侧的 b+1 的结果是 int 类型。需强制转换
```

所以，复合类型的表达式，如 b += 1; 是相当于：

```
byte b = (int)(b + 1);
```

除此之外， **复合运算符也是遵守操作数从右往左计算的原则。** 也就是说，在执行赋值操作之前，首先会确定左侧的操作数。

```
int a = 1;
a += ++a;
System.out.println(a);
```

运行结果：

```
3
```

根据上面的所说的，这个程序就不难理解了，先计算左侧 a 的值是 1，然后再计算出右侧++a 表达式的值是 2 ，最后便是计算 1+2 的值为 3，赋值给 a。

如果还是觉得难理解的，可以写成等价的普通式子，然后从左往右计算右侧的表达式。这可能比较容易理解：

```
a = a + ++a;
```

再看一个例子，加深理解：

```
int a = 5;
a *= a=2;
System.out.println(a);
```

运行结果：

```
10
```