1 算术运算符

在 Java 中，使用算术运算符`+`、`-`、`*`、`/`表示加、减、乘、除运算。

当参与`/`运算的两个操作数都是整数时，表示整数乘法，结果舍去小数部分取整；否则，表示浮点除法，结果为浮点数。

`%`是整数的求余运算，两个操作数必须都是整数。

整数被 0 除会产生异常，浮点数被 0 除会得到无穷大或`NaN`结果。

在默认情况下，虚拟机允许对浮点计算的中间结果采用扩展精度，以避免溢出和提高精度，但损失了可再生性。如果想要保证可再生性，可以使用`strictfp`关键字，限制方法或类必须使用严格的浮点计算来生成可再生的结果。

2 数学函数与常量

在`Math`类中，包含了各种各样的数学函数和常量，例如：

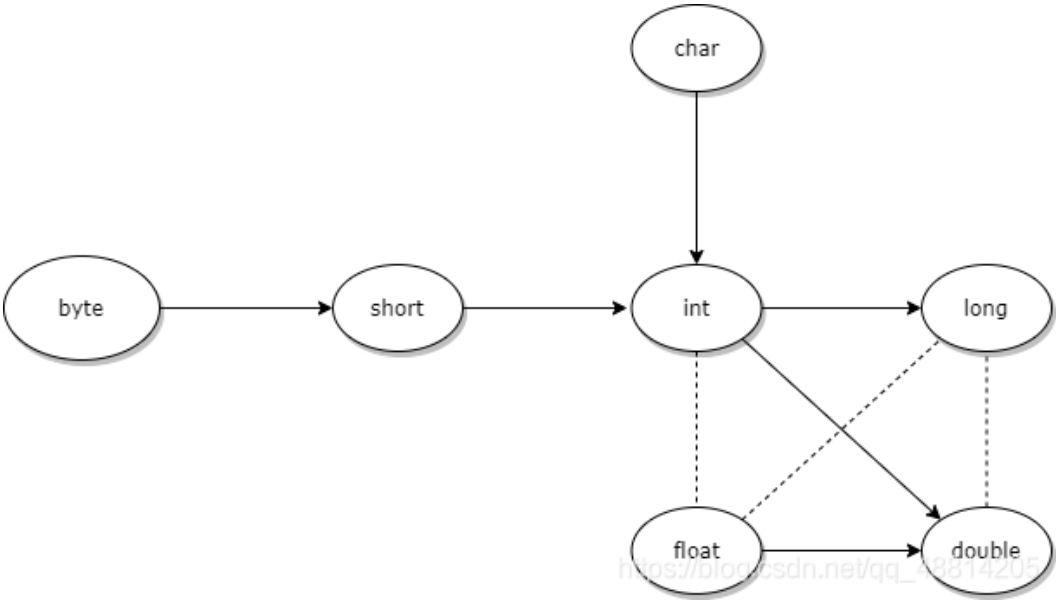
```
Math.sqrt(x); // 平方根
Math.pow(x, a); // x 的 a 次幂，参数和返回值都是 double 类型
Math.floorMod(x, n); // 计算和 x 模 n 同余的数，当 x 为正数时，范围为 0~n-1
Math.sin(x); // 正弦函数
Math.cos(x); // 余弦函数
Math.tan(x); // 正切函数
Math.atan(x); // 反正切函数，参数和返回值为 double 类型
Math.atan2(x, y); // 计算直角坐标系下点 (x, y) 在极坐标系下的极角，返回值为弧度，范围从 -π~π
Math.exp(x); // e 的 x 次幂
Math.log(x); // x 的自然对数
Math.log10(x); // 以 10 为底的对数

Math.PI // 圆周率 π
Math.E // 自然对数的底 e
```

只要在源文件顶部加上下面这条代码，就可以省略类名`Math`：

```
import static java.lang.Math.*;
```

3 数值类型之间的类型转换

下图给出了数值类型之间的合法转换。



图中 6 条实线箭头表示无信息丢失的转换，3 条虚线（方向从上到下）表示可能有精度损失的转换。

当用二元运算符连接两个值时，先要将两个操作数按上述规则转换为同一种类型，再进行计算。

- 如果两个操作数中有一个是`double`类型，另一个操作数就会转换为`double`类型。
- 否则，如果其中一个操作数是`float`类型，另一个操作数就会转换为`float`类型。
- 否则，如果其中一个操作数为`long`类型，另一个操作数就会转换为`long`类型。
- 否则，两个操作数都会转换为`int`类型。

4 强制类型转换

上面给出的 9 种类型转换都可以自动进行，除此之外的其他类型转换都必须使用强制类型转换显式指定。强制类型转换的语法格式是在圆括号中给出想要转换的目标类型，后面紧跟待转换的变量名。例如：

```
double x = 9.97;
int nx = (int) x;
```

将浮点类型强制转换为整型时，会截断小数部分，因此上例中`nx`的值为 9。

如果想对浮点数进行舍入运算，可以使用`Math.round`方法。例如：

```
double x = 9.97;
int nx = (int) Math.round(x);
```

`round`方法的返回值为`long`类型，因此这里还需要强制类型转换。现在，`nx`的值为 10。

5 自增与自减运算符

自增与自减运算符有前缀、后缀两种形式，这两种形式都可以使变量的值加 1 或减 1。区别在于，当用在表达式中时，前缀形式会用加 1 后的值参与运算，而后缀形式使用加 1 之前的值参与运算。例如：

```
int m = 7;
int n = 7;
int a = 2 * ++m; // a 为 16, m 为 8
int b = 2 * n++; // b 为 14, n 为 8
```

最好不要在表达式中使用自增、自减运算符，因为这样的代码很容易让人困惑。

6 关系和 `boolean` 运算符

Java 包含丰富的关系运算符，关系运算符的运算结果是`boolean`类型值。关系运算符有：

- `==`（判等）、`!=`（判断不相等）
- `>`（大于）、`<`（小于）、`>=`（大于等于）、`<=`（小于等于）

`boolean`运算符有：

- `&&`（逻辑与）
- `||`（逻辑或）
- `!`（逻辑非）

`&&`和`||`是按照“短路”方式来求值的：如果第一个操作数已经能够确定表达式的值，第二个操作数就不必计算了。

Java 支持三元运算符`?:`，格式如下：

```
条件 ? 表达式1 : 表达式2
```

当条件为`true`时，表达式 1 的值就作为整个表达式的值，否则计算为表达式 2 的值。

7 位运算符

位运算符有：

- `&`（按位与）
- `|`（按位或）
- `^`（按位异或）
- `~`（按位取反）
- `>>`（右移）、`<<`（左移）、`>>>`（右移补零）

8 运算符优先级

下表从上到下，优先级递减：

运算符	结合性
`[]`，`()`（方法调用），`.`	从左向右
`!`（逻辑非），`~`（按位取反），`++`（自增），`--`（自减），`new`，`()`（强制类型转换）	从右向左
`*`，`/`，`%`	从左向右
`+`，`-`	从左向右
`<<`（左移），`>>`（右移），`>>>`（右移补零）	从左向右
`<`，`<=`，`>`，`>=`，`instanceof`	从左向右
`==`（判等），`!=`（判断不等）	从左向右
`&`	从左向右
`^`	从左向右
`|`	从左向右
`&&`	从左向右
`||`	从左向右
`?:`	从右向左
`=`，`+=`，`-=`，`*=`，`/=`，`%=`，`&=`，`|=`，`^=`，`<<=`，`>>=`，`>>>=`	从右向左