# Operations on int variables

## 1. Evaluating integer expressions

The right side of an *assignment operator* (equal sign) can be any *expression* — any combination of numbers, variables, and mathematical operators (`+`, `-`, `*`, `/`).

You can also use parentheses `()`. In Java, as in mathematics, expressions inside parentheses are evaluated first, and then what is outside the parentheses.

Multiplication and division have equal precedence and are higher than addition and subtraction.

Examples:

| Statement | Note |
|---|---|
| `int a = (2 + 2) * 2;` | The value of the variable will be `8` |
| `int b = (6 - 3) / (9 - 6);` | The value of the variable will be `1` |
| `int c = (-2) * (-3);` | The value of the variable will be `6` |
| `int d = 3 / 0;` | Executing this statement will produce a "division by zero" error, and the program will terminate. |

An expression can also include variables:

| Statement | Note |
|---|---|
| `int a = 1;`<br>`int b = 2;`<br>`int c = a * b + 2;` | The value of the variable `a` will be `1`<br>The value of the variable `b` will be `2`<br>The value of the variable `c` will be `4` |

What's more, the same variable can be on both the left and the right of the *assignment operator*:

| Statement | Note |
|---|---|
| `int x = 5;`<br>`x = x + 1;`<br>`x = x + 1;`<br>`x = x + 1;`<br>`x = x + 1;`<br>`x = x + 1;` | The value of the variable `x` will be `5`<br>The value of the variable `x` will be `6`<br>The value of the variable `x` will be `7`<br>The value of the variable `x` will be `8`<br>The value of the variable `x` will be `9`<br>The value of the variable `x` will be `10` |

The point here is that in Java the = symbol does not mean equality. Instead, it is an operator that assigns to the *variable* on the left of the = sign the calculated value of the *expression to the right* of the = sign.

---

## 2. Division of integers

In Java, dividing an **integer** by an **integer** always results in an **integer**. The remainder of the division operation is discarded. Or, you could say that the result of division is always rounded down to the nearest integer.

Examples:

| Statement | Result of division | Note |
|---|---|---|
| `int a = 5 / 2;` | `2.5` | The value of the variable `a` will be `2` |
| `int b = 20 / 3;` | `6.3333(3)` | The value of the variable `b` will be `6` |
| `int c = 6 / 5;` | `1.2` | The value of the variable `c` will be `1` |
| `int d = 1 / 2;` | `0.5` | The value of the variable `d` will be `0` |

## 3. Remainder of division of integers

Besides addition, subtraction, multiplication, and division of integers, Java also has the **modulo** operator. It is the percent symbol (%). This operator returns the whole number remainder of dividing an integer by an integer (not the fractional part).

Examples:

| Statement | Result of division | Note |
|---|---|---|
| `int a = 5 % 2;` | `2` with a remainder of `1` | The value of the variable `a` will be `1` |
| `int b = 20 % 4;` | `5` with a remainder of `0` | The value of the variable `b` will be `0` |
| `int c = 9 % 5;` | `1` with a remainder of `4` | The value of the variable `c` will be `4` |
| `int d = 1 % 2;` | `0` with a remainder of `1` | The value of the variable `d` will be `1` |

This is a very useful operator. It is used a lot. For example, to find out whether a number is **even or odd**, just divide it by 2 and compare the

remainder with zero. If the remainder is zero, then the number is even; if it is equal to one, then the number is odd.

Here's what this check looks like:

```
(a % 2) == 0
```

where, you guessed it, `a % 2` is the remainder of division by `2` (i.e. `0` or `1`), and `==` is used to compare with zero.

---

## 4. Increment and decrement

In programming, increasing or decreasing a variable by one are very common operations. There are special commands for these actions in Java:

The *increment* (increment by one) operator looks like this:

```
a++;
```

*Increment*

This statement is exactly the same as `a = a + 1;` It increases the variable `a` by one.

The *decrement* (decrement by one) operator looks like this:

```
a--;
```

*Decrement*

This statement is exactly the same as `a = a - 1;` It decreases the variable `a` by one.

Examples

| Statement | Note |
| --- | --- |
| `int x = 5;`<br>`x++;`<br>`x++;`<br>`x++;`<br>`x++;`<br>`x++;` | The value of the variable `x` will be `5`<br>The value of the variable `x` will be `6`<br>The value of the variable `x` will be `7`<br>The value of the variable `x` will be `8`<br>The value of the variable `x` will be `9`<br>The value of the variable `x` will be `10` |
| `int x = 5;`<br>`x--;`<br>`x--;`<br>`x--;`<br>`x--;`<br>`x--;`<br>`x--;` | The value of the variable `x` will be `5`<br>The value of the variable `x` will be `4`<br>The value of the variable `x` will be `3`<br>The value of the variable `x` will be `2`<br>The value of the variable `x` will be `1`<br>The value of the variable `x` will be `0`<br>The value of the variable `x` will be `-1` |