**Синтаксис Java навеен С++.**

Все переменные объявляются явно, с указанием типа.

**Примитивные типы:**

* Boolean = true ur false

Boolean не конвертируется в другие примитивные типы

* char
* byte, short, int, long
* float, double

**Ссылочные типы:**

String s = “Hello world”;

String t = s;

String u = null;

Есть 4 boolean операции

! - отрицание

&& - и (Конюнкция)

II – или (Дизюнкция)

^ - исключающее или

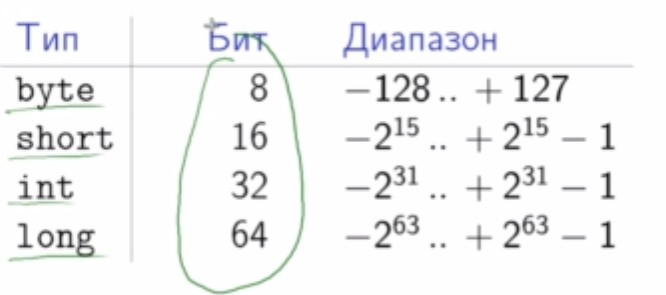
Конъюнкция и дизъюнкция реализуют вычисления по сокращенной схеме, это означает что правый операнд вычисляется только в том случае, когда он действительно нужен.

Например сравниваем: myTime && hisTime – если первый операнд будет false, то второй уже сравнивать, нет смысла и при любом раскладе, результатом спавнения будет false.

Сравнение по полной схемен происходит с помощбю следубщих операторов & и I – одиночные.

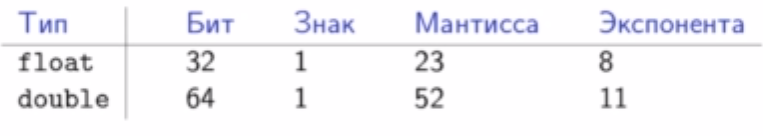


|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | b | | (или) | & (и) | ^ (скл. или) | ! (отрицание) |
| 1 | 1 | 1 | 1 | 0 | a=0; !a=true |
| 1 | 0 | 1 | 0 | 1 | a=1; !a=false |
| 0 | 1 | 1 | 0 | 1 |  |
| 0 | 0 | 0 | 0 | 0 |  |



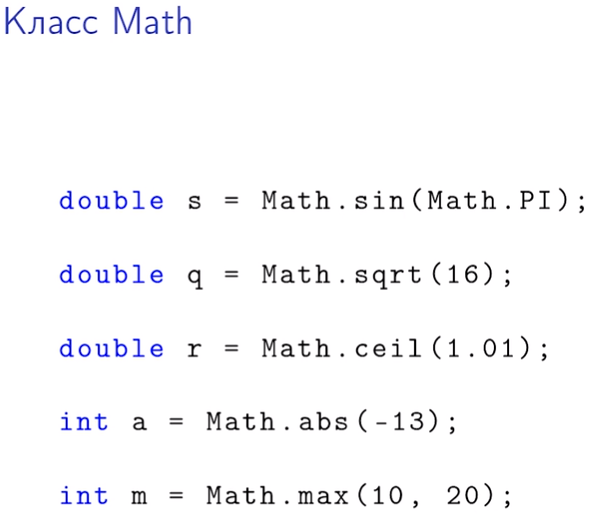
**Char** хранит не просто число, а код символа в кодировке unicode.





Nan = получается при деление 0 на 0 или при сложение + и – бесконечностей

Nan не равен самому себе



**Побитовые логические операции**

int neg = ~a; //операция отрицания к биту

int and = a & b;

int or = a | b;

int xor a ^ b;

int arithmeticShiftRight = a >> b; // Арифметический сдвиг вправо(младшие биты теряются, а освободившиеся старшие позиции заполняются знаковым битом)

int logicalShiftRight = a >>> b; // Логический сдвиг вправо (Освободившиеся позиции заполняются нулями)

int shiftLeft = a << b; // Сдвиг влево (Теряются старшие биты, а младшие заполняются нулями)

Праввым операндом может быть любое число, но java возьмет остаток от деления этого числа на размер целочисленного типа(например для int будет взят остаток от деления на 32(%32))