

# Язык программирования Java



# Урок № 3

## Операторы

### Содержание

<b>1. Операции.....</b>	<b>3</b>
Побитовые операции.....	5
Операции сравнения.....	7
Операция присваивания.....	9
Тернарный оператор.....	10
<b>2. Управляющие конструкции.....</b>	<b>11</b>
Условные операторы.....	11

## 1. Операции

Операции в языке Java – это специальные символы, которые показывают какое действие необходимо выполнить для заданных операндов.

**Операнд** – может быть литералом, переменной или выражением, над которыми производится операция.

Операции могут применяться к одному (унарная операция), двум (бинарная операция) или трем (тернарная операция) операндам.



Операции могут быть объединены в выражения.

В отличие от C++ в Java отсутствует перегрузка операций, так как создатели языка решили, что перегрузка операций существенно запутывает код и усложняет его понимание.

Есть несколько операций, перегруженных по умолчанию:

- + – используется для сложения чисел и конкатенации строк;
- & – используется для побитовой операции с числами и для логического И;
- | – используется для побитовой операции с числами и для логического ИЛИ;
- ^ – используется для побитовой операции с числами и для логического исключающего ИЛИ;
- == – сравнивает любые типы;
- != – сравнивает любые типы.

## Арифметические операции

Обозначение	Описание	Пример
+	Унарный плюс, не меняет значение операнда.	<pre>int i = -1; i = +i; System.out.println(i);</pre>
	Бинарный плюс, суммирует операнды.	<pre>int i = -1; i = i + 3; System.out.println(i);</pre>
-	Унарный минус – меняет знак операнда на противоположный.	<pre>int i = 1; i = -i; System.out.println(i);</pre>
	Бинарный минус – вычитает правый операнд из левого.	<pre>int i = 3; i = i - 2; System.out.println(i);</pre>
*	Умножение операндов	<pre>int i = 2; i = i * 2; System.out.println(i);</pre>
/	Деление левого операнда на правый.	<pre>int i = 4; i = i / 2; System.out.println(i);</pre>
%	Вычисление остатка от деления левого операнда на правый.	<pre>int i = 3; i = i % 2; System.out.println(i);</pre>
++	Бинарный оператор увеличения значения переменной на 1. Инкремент.	i++; заменяет выражение i = i + 1;
--	Бинарный оператор уменьшения значения переменной на 1. Декремент.	i--; заменяет выражение i = i - 1;

## Побитовые операции

Побитовые операции производят вычисления используя в качестве операндов побитовое представление числа (в двоичной системе счисления).

**ВАЖНО!!!** В качестве операндов в побитовых операциях могут быть использованы только целые примитивные типы или их классы-обертки.

Операция	Описание	Пример	Результат
&	И (AND)	<p>В каждом разряде результата получаем значащий бит, в случае если присутствует значащий бит в левом и правом операнде.</p> <pre>int i = 123 &amp; 456;</pre> <p>Битовое представление</p> <pre>000000000000000000000000000000001111011 &amp; 00000000000000000000000000000000111001000 = 0000000000000000000000000000000001001000</pre>	72
	ИЛИ (OR)	<p>В каждом разряде результата получаем значащий бит, в случае если присутствует значащий бит в левом или правом операнде.</p> <pre>final int one = 123; final int two = 456; int i = one   two;</pre> <p>Битовое представление</p> <pre>000000000000000000000000000000001111011   00000000000000000000000000000000111001000 = 00000000000000000000000000000000111111011</pre>	507

Операция	Описание	Пример	Результат
^	исключающее ИЛИ (XOR)	В каждом разряде результата получаем значащий бит, в случае если присутствует значащий бит в левом или правом операнде. Если бит есть в разряде обоих операндов, в результате получаем 0. <pre>int i = 123 ^ 456;</pre> Битовое представление $\begin{array}{r} \text{0000000000000000000000000000000111011} \\ \text{0000000000000000000000000000000111001000} \\ \hline \text{0000000000000000000000000000000110110011} \end{array}$	435
~	поразрядное отрицание (NOT)	Инвертирует все биты числа на противоположные (включая знаковый бит) <pre>int i = ~123;</pre> Битовое представление: $\sim \begin{array}{r} \text{00000000000000000000000000000001111011} \\ \hline \text{11111111111111111111111111111000100} \end{array}$	-124
<<	сдвиг влево	Смещает биты левого операнда влево на количество бит указанном в правом операнде, заполняя справа нулем.	
>>	сдвиг вправо	Смещает биты левого операнда вправо на количество бит указанном в правом операнде, заполняя разряды справа нулем.	
>>>	без знаковый сдвиг вправо	Смещает биты левого операнда вправо на количество бит указанном в правом операнде, заполняя слева нулем игнорируя битовый сдвиг.	

`Integer.toBinaryString(value)` – преобразует целое число `value` в строку в побитовом представлении.

Пример метода, выводящего число в побитовом представлении с дополнением незначимыми нулями:

```
private static void printToBinary(int value)
{
    System.out.println(String.format("%32s", Integer.
        toBinaryString(value)).replace(' ', '0'));
}
```

## Операции сравнения

Результат операций сравнения всегда имеет тип **boolean**.

Обозначение	Условие	Пример
>	операнд слева больше, чем операнд справа	<code>int a = 7;</code> <code>boolean r = a &gt; 5;</code>
<	операнд слева меньше, чем операнд справа	<code>int a = 7;</code> <code>boolean r = a &lt; 5;</code>
>=	операнд слева больше или равен операнду справа	<code>int a = 7;</code> <code>boolean r = a &gt;= 5;</code>
<=	операнд слева меньше или равен операнду справа	<code>int a = 7;</code> <code>boolean r = a &lt;= 5;</code>
==	операнд слева в точности равен операнду справа	<code>int a = 7;</code> <code>boolean r = a == 5;</code>
!=	операнд слева не равен операнду справа	<code>int a = 7;</code> <code>boolean r = a != 5;</code>

## Логические операции

Логические операции применяются только к операндам типа **boolean**. Результат логических операций имеет тип **boolean**.

Обозначение операции	Описание
&	Логическое И
	Логическое ИЛИ
^	Логическое исключающее ИЛИ
	Оператор быстрой оценки ИЛИ*
&&	Оператор быстрой оценки И*
!	Отрицание

\* Операторы быстрой оценки не вычисляют результат выражения в правом операнде, если результат операции можно определить по значению первого операнда.

Пример для И:

```
boolean t = 5 < 3 && 5 > 3
```

В данном примере в результате вычисления выражения левого операнда получим `false`, поэтому для вычисления результата операции нет необходимости вычислять правый операнд.

Пример для ИЛИ:

```
boolean t = (2 == 2) || 3 != 2
```

В данном примере в результате вычисления выражения левого операнда получим `true`, поэтому для вычисления результата операции ИЛИ нет необходимости вычислять правый операнд.



## Таблица истинности логических операций

Операнд 1	Операнд 2	Операция			
			&	^	!Операнд1
false	false	false	false	false	true
true	false	true	false	true	false
false	true	true	false	true	true
true	true	true	true	false	false

### Операция присваивания

Для сохранения в переменной значения литерала, переменной или значения выражения используется операция присваивания. Оператор присваивания имеет самый низкий приоритет, поэтому в начале выполняются все операции в выражении и только потом вычисленное значение передается в переменную.

Синтаксис:

переменная = выражение;

*Пример:*

```
int x = 3; // переменная будет инициализирована
           // значением 3
x = 6; // значение переменной будет перезаписано,
        // старое значение будет потеряно
```

Java позволяет присваивать значения одновременно нескольким переменным.

*Пример:*

```
int x;
int y;
int z;
```

```
x = y = z = 3;
System.out.println(x + y + z);
```

Результат: 9

## Тернарный оператор

Тернарный оператор содержит три операнда. Результатом операции может быть второй или третий операнд. Первый операнд должен иметь тип `boolean`.

Синтаксис:

```
выражение1 ? выражение2 : выражение3;
```

Операция вернет значение «выражение2», если результат вычисления «выражения1» примет значение истина, иначе вернет значение «выражения3».

Пример:

```
int x = 1;
int y = 2;
int a = (x != y) ? x : y; // для лучшего чтения кода
    // рекомендуется брать
    // условие в скобки.
System.out.println(a);
```

Результат: 1

## Приоритеты операций

Операции	Описание
() []	Круглые и квадратный скобки
++ -- + - ~ !	Декремент, инкремент, унарный плюс, унарный минус, поразрядное отрицание, логическое отрицание

Операции	Описание
* / %	Умножение, деление, остаток от деления
-	Сложение вычитание
>> >>> <<	Побитовые сдвиги: вправо, беззнаковый вправо, побитовый сдвиг влево
> >= < <=	Сравнение на: больше, больше или равно, меньше, меньше или равно
== !=	Сравнение на равенство, сравнение на неравенство
& ^   &&	Поразрядное и логическое: И, исключающее ИЛИ, ИЛИ, быстрой оценки И, ИЛИ
?:	Тернарная условная операция
операция=	Комбинированные операторы

Операции расположены в таблице от высокого приоритета к более низкому, сверху вниз и слева направо.

## 2. Управляющие конструкции

Для реализации алгоритма ветвления в коде в Java присутствует два условных оператора, это оператор **if ... else** и оператор **switch**. Для реализации циклических алгоритмов есть три оператора, это **for**, **while**, **do..while**. Синтаксис и принцип работы похож на синтаксис и работу в языках C++ и C#.

### Условные операторы

Синтаксис:

```
if (выражение) операция;
```

В круглых скобках может быть указано любое выражение, результатом которого должен быть тип **boolean**. В случае если в скобках выражение примет значение **true**, то будет выполнена «операция» следующая за круглыми

скобками, иначе управление программы перейдет к следующей строке кода.

*Пример:*

```
int a = 3;
if (a == 3) a = a + 2;
System.out.println(a);
```

*Результат: 5*

Если в результате выполнения условия в операторе **if** необходимо выполнить несколько операций, необходимо использовать «блочный разделитель» (фигурные скобки) для обозначения блока операций, которые будут выполнены.

*Пример:*

```
int a = 3;
if (a == 3) { // начало блока
    a = a + 2;
    a--;
    // конец блока }
System.out.println(a);
```

*Результат: 4*

Совместно с оператором **if** возможно использоваться ключевого слова **else**.

*Синтаксис:*

```
if (выражение) операция1;
else операция2;
```

В случае если в скобках «выражение» примет значение **false**, то будет выполнена «операция 2», «операция 1» выполнена не будет.

*Пример:*

```
int x = 3;
if (x > 3) System.out.println(++x);
else System.out.println(--x);
```

*Результат: 2*

Для **else**, в случае если есть необходимость выполнить несколько команд, так же может быть использован блочный разделитель.

*Пример:*

```
int a = 3;
if (a != 3) {
    System.out.println(a - 3);
}
else
{
    System.out.println(a + 3);
}
```

Оператор **if** может использоваться совместно с **else**, для создания логических цепочек.

*Пример:*

```
int age = 18;
if (age < 1) System.out.println("Грудничок");
else if (age >= 1 && age < 16) System.out.
    println("Ребенок");
else if (age >= 16 && age < 19)
    System.out.println("Подросток");
else if (age >= 19) System.out.println("Взрослый");
```

Для упрощения реализации алгоритма множественного выбора в Java добавлен оператор **switch**. Оператор

анализирует выражение в скобках и передает управление одному из сценариев (**case**). Далее программа выполнит весь код, находящийся в сценарии до конца оператора **switch**. Значения вариантов в **case** не должны повторяться.

Синтаксис:

```
switch (выражение)
{
    case вариант1:
        операция1;
    case вариант2:
        операция2;
    case вариант3:
        операция3;
    default:
        операция;
}
```

В скобках допустимо наличие значений примитивных типов `byte`, `short`, `char`, `int*`.

(\* Начиная с версии JSE 5.0, в качестве выражения можно использовать перечисления. Начиная с версии JSE 7.0, в качестве выражения можно использовать строки (`String`).).

Пример:

```
int x = 2;
switch(x)
{
    case 1:
        System.out.println(1);
    case 2: // будет выбран этот сценарий и выполнены
           //все команды до конца блока switch
        System.out.println(2);
    case 3:
        System.out.println(3);
}
```

Результат: 2, 3

Для прерывания сценария в блоке `switch` можно использовать ключевое слово `break`.

*Пример:*

```
int x = 2;
switch(x)
{
    case 1:
        System.out.println(1);
        break; // прерывание выполнение сценария,
              // управление передается в конец за блок switch
    case 2:
        System.out.println(2);
        break;
    case 3:
        System.out.println(3);
        break;
}
```

*Результат: 1*

Ключевое слово `default` может использоваться в операторе **`switch`** в случае, если ни один сценариев не был выполнен. В блоке `switch` может присутствовать только одно ключевое слово `default`. Порядок расположения блоков **`case`** и **`default`** не имеет значения, но принято располагать блоки `case` по возрастанию значения, а `default` – в самом конце.

*Пример:*

```
int x = 5;
switch(x)
{
    case 3:
        System.out.println(3);
}
```

```
    break;
case 1:
    System.out.println(1);
    break;
default:
    System.out.println("default");
case 2:
    System.out.println(2);
    break;
}
```

*Результат: default*







## Урок № 3 Операторы

© Виталий Унгурян

© Компьютерная Академия «Шаг»

[www.itstep.org](http://www.itstep.org)

Все права на охраняемые авторским правом фото-, аудио- и видеопроизведения, фрагменты которых использованы в материале, принадлежат их законным владельцам. Фрагменты произведений используются в иллюстративных целях в объёме, оправданном поставленной задачей, в рамках учебного процесса и в учебных целях, в соответствии со ст. 1274 ч. 4 ГК РФ и ст. 21 и 23 Закона Украины «Про авторське право і суміжні права». Объём и способ цитируемых произведений соответствует принятым нормам, не наносит ущерба нормальному использованию объектов авторского права и не ущемляет законные интересы автора и правообладателей. Цитируемые фрагменты произведений на момент использования не могут быть заменены альтернативными, не охраняемыми авторским правом аналогами, и как таковые соответствуют критериям добросовестного использования и честного использования.

Все права защищены. Полное или частичное копирование материалов запрещено. Согласование использования произведений или их фрагментов производится с авторами и правообладателями. Согласованное использование материалов возможно только при указании источника.

Ответственность за несанкционированное копирование и коммерческое использование материалов определяется действующим законодательством Украины.