**Java Operators**

In this tutorial, you'll learn about different types of operators in Java, their syntax and how to use them with the help of examples.

Operators are symbols that perform operations on variables and values. For example, + is an operator used for addition, while \* is also an operator used for multiplication.

Operators in Java can be classified into 5 types:

1. Arithmetic Operators
2. Assignment Operators
3. Relational Operators
4. Logical Operators
5. Unary Operators
6. Bitwise Operators

**1. Java Arithmetic Operators**

Arithmetic operators are used to perform arithmetic operations on variables and data. For example,

a + b;

Here, the + operator is used to add two variables a and b. Similarly, there are various other arithmetic operators in Java.

|  |  |
| --- | --- |
| Operator | Operation |
| + | Addition |
| - | Subtraction |
| \* | Multiplication |
| / | Division |
| % | Modulo Operation (Remainder after division) |

**Example 1: Arithmetic Operators**

class Main {

public static void main(String[] args) {

// declare variables

int a = 12, b = 5;

// addition operator

System.out.println("a + b = " + (a + b));

// subtraction operator

System.out.println("a - b = " + (a - b));

// multiplication operator

System.out.println("a \* b = " + (a \* b));

// division operator

System.out.println("a / b = " + (a / b));

// modulo operator

System.out.println("a % b = " + (a % b));

}

}

[Run Code](https://www.programiz.com/java-programming/online-compiler)

**Output**

a + b = 17

a - b = 7

a \* b = 60

a / b = 2

a % b = 2

In the above example, we have used +, -, and \* operators to compute addition, subtraction, and multiplication operations.

**/ Division Operator**

Note the operation, a / b in our program. The / operator is the division operator.

If we use the division operator with two integers, then the resulting quotient will also be an integer. And, if one of the operands is a floating-point number, we will get the result will also be in floating-point.

In Java,

(9 / 2) is 4

(9.0 / 2) is 4.5

(9 / 2.0) is 4.5

(9.0 / 2.0) is 4.5

**% Modulo Operator**

The modulo operator % computes the remainder. When a = 7 is divided by b = 4, the remainder is **3**.

**Note**: The % operator is mainly used with integers.

**2. Java Assignment Operators**

Assignment operators are used in Java to assign values to variables. For example,

int age;

age = 5;

Here, = is the assignment operator. It assigns the value on its right to the variable on its left. That is, **5** is assigned to the variable age.

Let's see some more assignment operators available in Java.

|  |  |  |
| --- | --- | --- |
| Operator | Example | Equivalent to |
| = | a = b; | a = b; |
| += | a += b; | a = a + b; |
| -= | a -= b; | a = a - b; |
| \*= | a \*= b; | a = a \* b; |
| /= | a /= b; | a = a / b; |
| %= | a %= b; | a = a % b; |

**Example 2: Assignment Operators**

class Main {

public static void main(String[] args) {

// create variables

int a = 4;

int var;

// assign value using =

var = a;

System.out.println("var using =: " + var);

// assign value using =+

var += a;

System.out.println("var using +=: " + var);

// assign value using =\*

var \*= a;

System.out.println("var using \*=: " + var);

}

}

[Run Code](https://www.programiz.com/java-programming/online-compiler)

**Output**

var using =: 4

var using +=: 8

var using \*=: 32

**3. Java Relational Operators**

Relational operators are used to check the relationship between two operands. For example,

// check if a is less than b

a < b;

Here, < operator is the relational operator. It checks if a is less than b or not.

It returns either true or false.

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| == | Is Equal To | 3 == 5 returns **false** |
| != | Not Equal To | 3 != 5 returns **true** |
| > | Greater Than | 3 > 5 returns **false** |
| < | Less Than | 3 < 5 returns **true** |
| >= | Greater Than or Equal To | 3 >= 5 returns **false** |
| <= | Less Than or Equal To | 3 <= 5 returns **true** |

**Example 3: Relational Operators**

class Main {

public static void main(String[] args) {

// create variables

int a = 7, b = 11;

// value of a and b

System.out.println("a is " + a + " and b is " + b);

// == operator

System.out.println(a == b); // false

// != operator

System.out.println(a != b); // true

// > operator

System.out.println(a > b); // false

// < operator

System.out.println(a < b); // true

// >= operator

System.out.println(a >= b); // false

// <= operator

System.out.println(a <= b); // true

}

}

[Run Code](https://www.programiz.com/java-programming/online-compiler)

**Note**: Relational operators are used in decision making and loops.

**4. Java Logical Operators**

Logical operators are used to check whether an expression is true or false. They are used in decision making.

|  |  |  |
| --- | --- | --- |
| Operator | Example | Meaning |
| && (Logical AND) | expression1 **&&** expression2 | true only if both expression1 and expression2 are true |
| || (Logical OR) | expression1 **||** expression2 | true if either expression1 or expression2 is true |
| ! (Logical NOT) | **!**expression | true if expression is false and vice versa |

**Example 4: Logical Operators**

class Main {

public static void main(String[] args) {

// && operator

System.out.println((5 > 3) && (8 > 5)); // true

System.out.println((5 > 3) && (8 < 5)); // false

// || operator

System.out.println((5 < 3) || (8 > 5)); // true

System.out.println((5 > 3) || (8 < 5)); // true

System.out.println((5 < 3) || (8 < 5)); // false

// ! operator

System.out.println(!(5 == 3)); // true

System.out.println(!(5 > 3)); // false

}

}

[Run Code](https://www.programiz.com/java-programming/online-compiler)

**Working of Program**

* (5 > 3) && (8 > 5) returns true because both (5 > 3) and (8 > 5) are true.
* (5 > 3) && (8 < 5) returns false because the expression (8 < 5) is false.
* (5 < 3) || (8 > 5) returns true because the expression (8 > 5) is true.
* (5 > 3) || (8 < 5) returns true because the expression (5 > 3) is true.
* (5 < 3) || (8 < 5) returns false because both (5 < 3) and (8 < 5) are false.
* !(5 == 3) returns true because 5 == 3 is false.
* !(5 > 3) returns false because 5 > 3 is true.

**5. Java Unary Operators**

Unary operators are used with only one operand. For example, ++ is a unary operator that increases the value of a variable by **1**. That is, ++5 will return **6**.

Different types of unary operators are:

|  |  |
| --- | --- |
| Operator | Meaning |
| + | **Unary plus**: not necessary to use since numbers are positive without using it |
| - | **Unary minus**: inverts the sign of an expression |
| ++ | **Increment operator**: increments value by 1 |
| -- | **Decrement operator**: decrements value by 1 |
| ! | **Logical complement operator**: inverts the value of a boolean |

**Increment and Decrement Operators**

Java also provides increment and decrement operators: ++ and -- respectively. ++ increases the value of the operand by **1**, while -- decrease it by **1**. For example,

int num = 5;

// increase num by 1

++num;

Here, the value of num gets increased to **6** from its initial value of **5**.

**Example 5: Increment and Decrement Operators**

class Main {

public static void main(String[] args) {

// declare variables

int a = 12, b = 12;

int result1, result2;

// original value

System.out.println("Value of a: " + a);

// increment operator

result1 = ++a;

System.out.println("After increment: " + result1);

System.out.println("Value of b: " + b);

// decrement operator

result2 = --b;

System.out.println("After decrement: " + result2);

}

}

[Run Code](https://www.programiz.com/java-programming/online-compiler)

**Output**

Value of a: 12

After increment: 13

Value of b: 12

After decrement: 11

In the above program, we have used the ++ and -- operator as **prefixes (++a, --b)**. We can also use these operators as **postfix (a++, b++)**.

There is a slight difference when these operators are used as prefix versus when they are used as a postfix.

To learn more about these operators, visit [increment and decrement operators](https://www.programiz.com/article/increment-decrement-operator-difference-prefix-postfix).

**6. Java Bitwise Operators**

Bitwise operators in Java are used to perform operations on individual bits. For example,

Bitwise complement Operation of 35

35 = 00100011 (In Binary)

~ 00100011

\_\_\_\_\_\_\_\_

11011100 = 220 (In decimal)

Here, ~ is a bitwise operator. It inverts the value of each bit (**0** to **1** and **1** to **0**).

The various bitwise operators present in Java are:

|  |  |
| --- | --- |
| Operator | Description |
| ~ | Bitwise Complement |
| << | Left Shift |
| >> | Right Shift |
| >>> | Unsigned Right Shift |
| & | Bitwise AND |
| ^ | Bitwise exclusive OR |

These operators are not generally used in Java. To learn more, visit [Java Bitwise and Bit Shift Operators](https://www.programiz.com/java-programming/bitwise-operators).

**Other operators**

Besides these operators, there are other additional operators in Java.

**Java instanceof Operator**

The instanceof operator checks whether an object is an instanceof a particular class. For example,

class Main {

public static void main(String[] args) {

String str = "Programiz";

boolean result;

// checks if str is an instance of

// the String class

result = str instanceof String;

System.out.println("Is str an object of String? " + result);

}

}

[Run Code](https://www.programiz.com/java-programming/online-compiler)

**Output**

Is str an object of String? true

Here, str is an instance of the String class. Hence, the instanceof operator returns true. To learn more, visit [Java instanceof](https://www.programiz.com/java-programming/instanceof).

**Java Ternary Operator**

The ternary operator (conditional operator) is shorthand for the if-then-else statement. For example,

variable = Expression ? expression1 : expression2

Here's how it works.

* If the Expression is true, expression1 is assigned to the variable.
* If the Expression is false, expression2 is assigned to the variable.

Let's see an example of a ternary operator.

class Java {

public static void main(String[] args) {

int februaryDays = 29;

String result;

// ternary operator

result = (februaryDays == 28) ? "Not a leap year" : "Leap year";

System.out.println(result);

}

}

[Run Code](https://www.programiz.com/java-programming/online-compiler)

**Output**

Leap year

In the above example, we have used the ternary operator to check if the year is a leap year or not.