Operators in Java

**Operator** in [Java](https://www.javatpoint.com/java-tutorial) is a symbol that is used to perform operations. For example: +, -, \*, / etc.

There are many types of operators in Java which are given below:

* Unary Operator,
* Arithmetic Operator,
* Shift Operator,
* Relational Operator,
* Bitwise Operator,
* Logical Operator,
* Ternary Operator and
* Assignment Operator.

Java Operator Precedence

|  |  |  |
| --- | --- | --- |
| **Operator Type** | **Category** | **Precedence** |
| Unary | postfix | *expr*++ *expr*-- |
| prefix | ++*expr* --*expr* +*expr* -*expr* ~ ! |
| Arithmetic | multiplicative | \* / % |
| additive | + - |
| Shift | Shift | << >> >>> |
| Relational | comparison | < > <= >= instanceof |
| equality | == != |
| Bitwise | bitwise AND | & |
| bitwise exclusive OR | ^ |
| bitwise inclusive OR | | |
| Logical | logical AND | && |
| logical OR | || |
| Ternary | ternary | ? : |
| Assignment | assignment | = += -= \*= /= %= &= ^= |= <<= >>= >>>= |

### **Java Unary Operator Example: ++ and --**

1. **public** **class** OperatorExample{
2. **public** **static** **void** main(String args[]){
3. **int** x=10;
4. System.out.println(x++);//10 (11)
5. System.out.println(++x);//12
6. System.out.println(x--);//12 (11)
7. System.out.println(--x);//10
8. }}

**Output:**

10

12

12

10

### **Java Unary Operator Example: ~ and !**

1. **public** **class** OperatorExample{
2. **public** **static** **void** main(String args[]){
3. **int** a=10;
4. **int** b=-10;
5. **boolean** c=**true**;
6. **boolean** d=**false**;
7. System.out.println(~a);//-11 (minus of total positive value which starts from 0)
8. System.out.println(~b);//9 (positive of total minus, positive starts from 0)
9. System.out.println(!c);//false (opposite of boolean value)
10. System.out.println(!d);//true
11. }}

**Output:**

-11

9

false

true

### **Java Arithmetic Operators**

Java arithmetic operators are used to perform addition, subtraction, multiplication, and division. They act as basic mathematical operations.

### **Java Arithmetic Operator Example**

1. **public** **class** OperatorExample{
2. **public** **static** **void** main(String args[]){
3. **int** a=10;
4. **int** b=5;
5. System.out.println(a+b);//15
6. System.out.println(a-b);//5
7. System.out.println(a\*b);//50
8. System.out.println(a/b);//2
9. System.out.println(a%b);//0
10. }}

**Output:**

15

5

50

2

0

### **Java Left Shift Operator**

The Java left shift operator << is used to shift all of the bits in a value to the left side of a specified number of times.

### **Java Left Shift Operator Example**

1. **public** **class** OperatorExample{
2. **public** **static** **void** main(String args[]){
3. System.out.println(10<<2);//10\*2^2=10\*4=40
4. System.out.println(10<<3);//10\*2^3=10\*8=80
5. System.out.println(20<<2);//20\*2^2=20\*4=80
6. System.out.println(15<<4);//15\*2^4=15\*16=240
7. }}

### **ava AND Operator Example: Logical && and Bitwise &**

The logical && operator doesn't check the second condition if the first condition is false. It checks the second condition only if the first one is true.

The bitwise & operator always checks both conditions whether first condition is true or false.

1. **public** **class** OperatorExample{
2. **public** **static** **void** main(String args[]){
3. **int** a=10;
4. **int** b=5;
5. **int** c=20;
6. System.out.println(a<b&&a<c);//false && true = false
7. System.out.println(a<b&a<c);//false & true = false
8. }}

**Output:**

false

false

### **Java OR Operator Example: Logical || and Bitwise |**

The logical || operator doesn't check the second condition if the first condition is true. It checks the second condition only if the first one is false.

The bitwise | operator always checks both conditions whether first condition is true or false.

1. **public** **class** OperatorExample{
2. **public** **static** **void** main(String args[]){
3. **int** a=10;
4. **int** b=5;
5. **int** c=20;
6. System.out.println(a>b||a<c);//true || true = true
7. System.out.println(a>b|a<c);//true | true = true
8. //|| vs |
9. System.out.println(a>b||a++<c);//true || true = true
10. System.out.println(a);//10 because second condition is not checked
11. System.out.println(a>b|a++<c);//true | true = true
12. System.out.println(a);//11 because second condition is checked
13. }}

**Output:**

true

true

true

10

true

11

### **Java Ternary Operator**

Java Ternary operator is used as one line replacement for if-then-else statement and used a lot in Java programming. It is the only conditional operator which takes three operands.

### **Java Ternary Operator Example**

1. **public** **class** OperatorExample{
2. **public** **static** **void** main(String args[]){
3. **int** a=2;
4. **int** b=5;
5. **int** min=(a<b)?a:b;
6. System.out.println(min);
7. }}

**Output:**

2

### **Java Assignment Operator**

Java assignment operator is one of the most common operators. It is used to assign the value on its right to the operand on its left.

### **Java Assignment Operator Example**

1. **public** **class** OperatorExample{
2. **public** **static** **void** main(String args[]){
3. **int** a=10;
4. **int** b=20;
5. a+=4;//a=a+4 (a=10+4)
6. b-=4;//b=b-4 (b=20-4)
7. System.out.println(a);
8. System.out.println(b);
9. }}

**Output:**

14

16

After type cast:

1. **public** **class** OperatorExample{
2. **public** **static** **void** main(String args[]){
3. **short** a=10;
4. **short** b=10;
5. a=(**short**)(a+b);//20 which is int now converted to short
6. System.out.println(a);
7. }}

**Output:**

20

Java Keywords

Java keywords are also known as reserved words. Keywords are particular words that act as a key to a code. These are predefined words by Java so they cannot be used as a variable or object name or class name.

List of Java Keywords

A list of Java keywords or reserved words are given below:

1. **[abstract](https://www.javatpoint.com/abstract-keyword-in-java)**

**:** Java abstract keyword is used to declare an abstract class. An abstract class can provide the implementation of the interface. It can have abstract and non-abstract methods.

1. **[boolean:](https://www.javatpoint.com/boolean-keyword-in-java)**

Java boolean keyword is used to declare a variable as a boolean type. It can hold True and False values only.

1. **[break](https://www.javatpoint.com/java-break)**

**:** Java break keyword is used to break the loop or switch statement. It breaks the current flow of the program at specified conditions.

1. **[byte](https://www.javatpoint.com/byte-keyword-in-java)**

**:** Java byte keyword is used to declare a variable that can hold 8-bit data values.

1. **[case](https://www.javatpoint.com/case-keyword-in-java)**

**:** Java case keyword is used with the switch statements to mark blocks of text.

1. **[catch](https://www.javatpoint.com/try-catch-block)**

**:** Java catch keyword is used to catch the exceptions generated by try statements. It must be used after the try block only.

1. **[char](https://www.javatpoint.com/char-keyword-in-java)**

**:** Java char keyword is used to declare a variable that can hold unsigned 16-bit Unicode characters

1. **[class](https://www.javatpoint.com/class-keyword-in-java)**

**:** Java class keyword is used to declare a class.

1. **[continue](https://www.javatpoint.com/java-continue)**

**:** Java continue keyword is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition.

1. **[default](https://www.javatpoint.com/default-keyword-in-java)**

**:** Java default keyword is used to specify the default block of code in a switch statement.

1. **[do](https://www.javatpoint.com/java-do-while-loop)**

**:** Java do keyword is used in the control statement to declare a loop. It can iterate a part of the program several times.

1. **[double](https://www.javatpoint.com/double-keyword-in-java)**

**:** Java double keyword is used to declare a variable that can hold 64-bit floating-point number.

1. **[else](https://www.javatpoint.com/java-if-else)**

**:** Java else keyword is used to indicate the alternative branches in an if statement.

1. **[enum](https://www.javatpoint.com/enum-in-java)**

**:** Java enum keyword is used to define a fixed set of constants. Enum constructors are always private or default.

1. **[extends](https://www.javatpoint.com/inheritance-in-java)**

**:** Java extends keyword is used to indicate that a class is derived from another class or interface.

1. **[final](https://www.javatpoint.com/final-keyword)**

**:** Java final keyword is used to indicate that a variable holds a constant value. It is used with a variable. It is used to restrict the user from updating the value of the variable.

1. **[finally](https://www.javatpoint.com/finally-block-in-exception-handling)**

**:** Java finally keyword indicates a block of code in a try-catch structure. This block is always executed whether an exception is handled or not.

1. **[float](https://www.javatpoint.com/float-keyword-in-java)**

**:** Java float keyword is used to declare a variable that can hold a 32-bit floating-point number.

1. **[for](https://www.javatpoint.com/java-for-loop)**

**:** Java for keyword is used to start a for loop. It is used to execute a set of instructions/functions repeatedly when some condition becomes true. If the number of iteration is fixed, it is recommended to use for loop.

1. **[if](https://www.javatpoint.com/java-if-else)**

**:** Java if keyword tests the condition. It executes the if block if the condition is true.

1. **[implements](https://www.javatpoint.com/interface-in-java)**

**:** Java implements keyword is used to implement an interface.

1. **[import](https://www.javatpoint.com/package)**

**:** Java import keyword makes classes and interfaces available and accessible to the current source code.

1. **[instanceof](https://www.javatpoint.com/downcasting-with-instanceof-operator)**

**:** Java instanceof keyword is used to test whether the object is an instance of the specified class or implements an interface.

1. **[int](https://www.javatpoint.com/int-keyword-in-java)**

**:** Java int keyword is used to declare a variable that can hold a 32-bit signed integer.

1. **[interface](https://www.javatpoint.com/interface-in-java)**

**:** Java interface keyword is used to declare an interface. It can have only abstract methods.

1. **[long](https://www.javatpoint.com/long-keyword-in-java)**

**:** Java long keyword is used to declare a variable that can hold a 64-bit integer.

1. **native:** Java native keyword is used to specify that a method is implemented in native code using JNI (Java Native Interface).
2. **[new](https://www.javatpoint.com/new-keyword-in-java)**

**:** Java new keyword is used to create new objects.

1. **[null](https://www.javatpoint.com/null-keyword-in-java)**

**:** Java null keyword is used to indicate that a reference does not refer to anything. It removes the garbage value.

1. **[package](https://www.javatpoint.com/package)**

**:** Java package keyword is used to declare a Java package that includes the classes.

1. **[private](https://www.javatpoint.com/private-keyword-in-java)**

**:** Java private keyword is an access modifier. It is used to indicate that a method or variable may be accessed only in the class in which it is declared.

1. **[protected](https://www.javatpoint.com/protected-keyword-in-java)**

**:** Java protected keyword is an access modifier. It can be accessible within the package and outside the package but through inheritance only. It can't be applied with the class.

1. **[public](https://www.javatpoint.com/public-keyword-in-java)**

**:** Java public keyword is an access modifier. It is used to indicate that an item is accessible anywhere. It has the widest scope among all other modifiers.

1. **[return](https://www.javatpoint.com/return-keyword-in-java)**

**:** Java return keyword is used to return from a method when its execution is complete.

1. **[short](https://www.javatpoint.com/short-keyword-in-java)**

**:** Java short keyword is used to declare a variable that can hold a 16-bit integer.

1. **[static](https://www.javatpoint.com/static-keyword-in-java)**

**:** Java static keyword is used to indicate that a variable or method is a class method. The static keyword in Java is mainly used for memory management.

1. **[strictfp](https://www.javatpoint.com/strictfp-keyword)**

**:** Java strictfp is used to restrict the floating-point calculations to ensure portability.

1. **[super](https://www.javatpoint.com/super-keyword)**

**:** Java super keyword is a reference variable that is used to refer to parent class objects. It can be used to invoke the immediate parent class method.

1. **[switch](https://www.javatpoint.com/java-switch)**

**:** The Java switch keyword contains a switch statement that executes code based on test value. The switch statement tests the equality of a variable against multiple values.

1. **[synchronized](https://www.javatpoint.com/synchronization-in-java)**

**:** Java synchronized keyword is used to specify the critical sections or methods in multithreaded code.

1. **[this](https://www.javatpoint.com/this-keyword)**

**:** Java this keyword can be used to refer the current object in a method or constructor.

1. **[throw](https://www.javatpoint.com/throw-keyword)**

**:** The Java throw keyword is used to explicitly throw an exception. The throw keyword is mainly used to throw custom exceptions. It is followed by an instance.

1. **[throws](https://www.javatpoint.com/throws-keyword-and-difference-between-throw-and-throws)**

**:** The Java throws keyword is used to declare an exception. Checked exceptions can be propagated with throws.

1. **[transient](https://www.javatpoint.com/transient-keyword)**

**:** Java transient keyword is used in serialization. If you define any data member as transient, it will not be serialized.

1. **[try](https://www.javatpoint.com/try-catch-block)**

**:** Java try keyword is used to start a block of code that will be tested for exceptions. The try block must be followed by either catch or finally block.

1. **void:** Java void keyword is used to specify that a method does not have a return value.
2. **[volatile](https://www.javatpoint.com/volatile-keyword-in-java)**

**:** Java volatile keyword is used to indicate that a variable may change asynchronously.

1. **[while](https://www.javatpoint.com/java-while-loop)**

**:** Java while keyword is used to start a while loop. This loop iterates a part of the program several times. If the number of iteration is not fixed, it is recommended to use the while loop.