**Java Operators**

Operators are used to perform operations on variables and values.

In the example below, we use the + **operator** to add together two values.

public class Main {

public static void main(String[] args) {

int x = 100 + 50;

System.out.println(x);

}

}

Although the + operator is often used to add together two values, like in the example above, it can also be used to add together a variable and a value, or a variable and another variable.

public class Main {

public static void main(String[] args) {

int sum1 = 100 + 50;

int sum2 = sum1 + 250;

int sum3 = sum2 + sum2;

System.out.println(sum1);

System.out.println(sum2);

System.out.println(sum3);

}

}

Java divides the operators into the following groups:

* Arithmetic operators
* Assignment operators
* Comparison operators
* Logical operators
* Bitwise operators

Arithmetic Operators

* Arithmetic operators are used to perform common mathematical operations.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operator** | **Name** | **Description** | **Example** | **Try it** |
| + | Addition | Adds together two values | x + y | Write a simple program for the example |
| - | Subtraction | Subtracts one value from another | x - y | [Write a simple program for the example](https://www.w3schools.com/java/tryjava.asp?filename=demo_oper_sub) |
| \* | Multiplication | Multiplies two values | x \* y | Write a simple program for the example |
| / | Division | Divides one value by another | x / y | Write a simple program for the example |
| % | Modulus | Returns the division remainder | x % y | Write a simple program for the example |
| ++ | Increment | Increases the value of a variable by 1 | ++x | Write a simple program for the example |
| -- | Decrement | Decreases the value of a variable by 1 | --x | Write a simple program for the example |

**Java Assignment Operators**

Assignment operators are used to assign values to variables.

In the example below, we use the **assignment** operator (=) to assign the value **10** to a variable called **x**.

public class Main {

public static void main(String[] args) {

int x = 10;

System.out.println(x);

}

}

The **addition assignment** operator (+=) adds a value to a variable…

public class Main {

public static void main(String[] args) {

int x = 10;

x += 5;

System.out.println(x);

}

}

A list of all assignment operators:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Example** | **Same As** | **Try it** |
| = | x = 5 | x = 5 | Write a simple program for the example |
| += | x += 3 | x = x + 3 | Write a simple program for the example |
| -= | x -= 3 | x = x - 3 | Write a simple program for the example |
| \*= | x \*= 3 | x = x \* 3 | Write a simple program for the example |
| /= | x /= 3 | x = x / 3 | Write a simple program for the example |
| %= | x %= 3 | x = x % 3 | Write a simple program for the example |
| &= | x &= 3 | x = x & 3 | Write a simple program for the example |
| |= | x |= 3 | x = x | 3 | Write a simple program for the example |
| ^= | x ^= 3 | x = x ^ 3 | Write a simple program for the example |
| >>= | x >>= 3 | x = x >> 3 | Write a simple program for the example |
| <<= | x <<= 3 | x = x << 3 | Write a simple program for the example |

Java Comparison Operators

Comparison operators are used to compare two values:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Example** | **Try it** |
| == | Equal to | x == y | public class Main {  public static void main(String[] args) {  int x = 5;  int y = 3;  System.out.println(x == y); // returns false because 5 is not equal to 3  }  } |
| != | Not equal | x != y | public class Main {  public static void main(String[] args) {  int x = 5;  int y = 3;  System.out.println(x != y); // returns true because 5 is not equal to 3  }  } |
| > | Greater than | x > y | public class Main {  public static void main(String[] args) {  int x = 5;  int y = 3;  System.out.println(x > y); // returns true because 5 is greater than 3  }  } |
| < | Less than | x < y | public class Main {  public static void main(String[] args) {  int x = 5;  int y = 3;  System.out.println(x < y); // returns false because 5 is not less than 3  }  } |
| >= | Greater than or equal to | x >= y | public class Main {  public static void main(String[] args) {  int x = 5;  int y = 3;  System.out.println(x >= y); // returns true because 5 is greater, or equal, to 3  }  } |
| <= | Less than or equal to | x <= y | public class Main {  public static void main(String[] args) {  int x = 5;  int y = 3;  System.out.println(x <= y); // returns false because 5 is neither less than or equal to 3  }  } |

Java Logical Operators

Logical operators are used to determine the logic between variables or values:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operator** | **Name** | **Description** | **Example** | **Try it** |
| && | Logical and | Returns true if both statements are true | x < 5 &&  x < 10 | public class Main {  public static void main(String[] args) {  int x = 5;  System.out.println(x > 3 && x < 10); // returns true because 5 is greater than 3 AND 5 is less than 10  }  } |
| || | Logical or | Returns true if one of the statements is true | x < 5 || x < 4 | public class Main {  public static void main(String[] args) {  int x = 5;  System.out.println(x > 3 || x < 4); // returns true because one of the conditions are true (5 is greater than 3, but 5 is not less than 4)  }  } |
| ! | Logical not | Reverse the result, returns false if the result is true | !(x < 5 && x < 10) | public class Main {  public static void main(String[] args) {  int x = 5;  System.out.println(!(x > 3 && x < 10)); // returns false because ! (not) is used to reverse the result  }} |

More on Java Strings

String Length

A String in Java is actually an object, which contain methods that can perform certain operations on strings. For example, the length of a string can be found with the length() method.

public class Main {

public static void main(String[] args) {

String txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

System.out.println("The length of the txt string is: " + txt.length());

}

}

There are many string methods available, for example toUpperCase() and toLowerCase().

public class Main {

public static void main(String[] args) {

String txt = "Hello World";

System.out.println(txt.toUpperCase());

System.out.println(txt.toLowerCase());

}

}

Finding a Character in a String

The indexOf() method returns the **index** (the position) of the first occurrence of a specified text in a string (including whitespace).

public class Main {

public static void main(String[] args) {

String txt = "Please locate where 'locate' occurs!";

System.out.println(txt.indexOf("locate"));

}

}

Java counts positions from zero.  
0 is the first position in a string, 1 is the second, 2 is the third ...

**String Concatenation**

The + operator can be used between strings to combine them. This is called **concatenation**…

public class Main {

public static void main(String args[]) {

String firstName = "John";

String lastName = "Doe";

System.out.println(firstName + " " + lastName);

}

}

Note that we have added an empty text (" ") to create a space between firstName and lastName on print.

You can also use the concat() method to concatenate two strings…

public class Main {

public static void main(String[] args) {

String firstName = "John ";

String lastName = "Doe";

System.out.println(firstName.concat(lastName));

}

}

Adding Numbers and Strings

WARNING!

Java uses the + operator for both addition and concatenation.

Numbers are added. Strings are concatenated.

If you add two numbers, the result will be a number:

public class Main {

public static void main(String[] args) {

int x = 10;

int y = 20;

int z = x + y;

System.out.println(z);

}

}

If you add two strings, the result will be a string concatenation:

public class Main {

public static void main(String[] args) {

String x = "10";

int y = 20;

String z = x + y;

System.out.println(z);

}

}

**Java Booleans**

Very often, in programming, you will need a data type that can only have one of two values, like:

* YES / NO
* ON / OFF
* TRUE / FALSE

For this, Java has a boolean data type, which can take the values true or false…

Boolean Values

A boolean type is declared with the boolean keyword and can only take the values true or false.

public class Main {

public static void main(String[] args) {

boolean isJavaFun = true;

boolean isFishTasty = false;

System.out.println(isJavaFun);

System.out.println(isFishTasty);

}

}

**Boolean Expression**

A **Boolean expression** is a Java expression that returns a Boolean value: true or false.

You can use a comparison operator, such as the **greater than** (>) operator to find out if an expression (or a variable) is true:

public class Main {

public static void main(String[] args) {

int x = 10;

int y = 9;

System.out.println(x > y); // returns true, because 10 is higher than 9

}

}

Or

public class Main {

public static void main(String[] args) {

System.out.println(10 > 9); // returns true, because 10 is higher than 9

}

}

In the examples below, we use the **equal to** (==) operator to evaluate an expression…

public class Main {

public static void main(String[] args) {

int x = 10;

System.out.println(x == 10); // returns true, because the value of x is equal to 10

}

}

**Java Conditions and If Statements**

Java supports the usual logical conditions from mathematics:

* Less than: a < b
* Less than or equal to: a <= b
* Greater than: a > b
* Greater than or equal to: a >= b
* Equal to a == b
* Not Equal to: a != b

You can use these conditions to perform different actions for different decisions.

Java has the following conditional statements:

* Use if to specify a block of code to be executed, if a specified condition is true
* Use else to specify a block of code to be executed, if the same condition is false
* Use else if to specify a new condition to test, if the first condition is false
* Use switch to specify many alternative blocks of code to be executed

**The if Statement**

Use the if statement to specify a block of Java code to be executed if a condition is true.

Syntax: if (condition) {

// block of code to be executed if the condition is true

}

Note that if is in lowercase letters. Uppercase letters (If or IF) will generate an error.

In the example below, we test two values to find out if 20 is greater than 18. If the condition is true, print some text…

public class Main {

public static void main(String[] args) {

if (20 > 18) {

System.out.println("20 is greater than 18"); // obviously

}

}

}

We can also test variables:

public class Main {

public static void main(String[] args) {

int x = 20;

int y = 18;

if (x > y) {

System.out.println("x is greater than y");

}

}

}

**Example explained**

In the example above we use two variables, **x** and **y**, to test whether x is greater than y (using the > operator). As x is 20, and y is 18, and we know that 20 is greater than 18, we print to the screen that "x is greater than y".

**The else Statement**

Use the else statement to specify a block of code to be executed if the condition is false.

Syntax

if (condition) {

// block of code to be executed if the condition is true

} else {

// block of code to be executed if the condition is false

}

public class Main {

public static void main(String[] args) {

int time = 20;

if (time < 18) {

System.out.println("Good afternoon.");

} else {

System.out.println("Good evening.");

}

}

}

**Example explained**

In the example above, time (20) is greater than 18, so the condition is false. Because of this, we move on to the else condition and print to the screen "Good evening". If the time was less than 18, the program would print "Good day".

**The else if Statement**

Use the else if statement to specify a new condition if the first condition is false.

Syntax

if (condition1) {

// block of code to be executed if condition1 is true

} else if (condition2) {

// block of code to be executed if the condition1 is false and condition2 is true

} else {

// block of code to be executed if the condition1 is false and condition2 is false

}

public class Main {

public static void main(String[] args) {

int time = 22;

if (time < 10) {

System.out.println("Good morning.");

} else if (time < 20) {

System.out.println("Good afternoon.");

} else {

System.out.println("Good evening.");

}

}

}

Example explained

In the example above, time (22) is greater than 10, so the **first condition** is false. The next condition, in the else if statement, is also false, so we move on to the else condition since **condition1** and **condition2** is both false - and print to the screen "Good evening".

However, if the time was 14, our program would print "Good day."