What is Java?

Java is a popular programming language, created in 1995.

It is owned by Oracle, and more than **3 billion** devices run Java.

It is used for:

* Mobile applications (specially Android apps)
* Desktop applications
* Web applications
* Web servers and application servers
* Games
* Database connection
* And much, much more!

Why Use Java?

* Java works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.)
* It is one of the most popular programming language in the world
* It is easy to learn and simple to use
* It is open-source and free
* It is secure, fast and powerful
* It has a huge community support (tens of millions of developers)
* Java is an object oriented language which gives a clear structure to programs and allows code to be reused, lowering development costs
* As Java is close to [C++](https://www.w3schools.com/cpp/default.asp) and [C#](https://www.w3schools.com/cs/default.asp), it makes it easy for programmers to switch to Java or vice versa

--------------------------------------------------------------------------------------

## Java Comments

Comments can be used to explain Java code, and to make it more readable. It can also be used to prevent execution when testing alternative code.

## Single-line Comments

Single-line comments start with two forward slashes (//).

Any text between // and the end of the line is ignored by Java (will not be executed).

## Java Multi-line Comments

Multi-line comments start with /\* and ends with \*/.

Any text between /\* and \*/ will be ignored by Java.

--------------------------------------------------------------------------------------

## Java Data Types

As explained in the previous chapter, a [variable](https://www.w3schools.com/java/java_variables.asp) in Java must be a specified data type:

Data types are divided into two groups:

* Primitive data types - includes byte, short, int, long, float, double, boolean and char
* Non-primitive data types - such as [String](https://www.w3schools.com/java/java_strings.asp), [Arrays](https://www.w3schools.com/java/java_arrays.asp) and [Classes](https://www.w3schools.com/java/java_classes.asp) (you will learn more about these in a later chapter)

## Primitive Data Types

A primitive data type specifies the size and type of variable values, and it has no additional methods.

There are eight primitive data types in Java:





Non-Primitive Data Types

Non-primitive data types are called **reference types** because they refer to objects.

The main difference between **primitive** and **non-primitive** data types are:

* Primitive types are predefined (already defined) in Java. Non-primitive types are created by the programmer and is not defined by Java (except for String).
* Non-primitive types can be used to call methods to perform certain operations, while primitive types cannot.
* A primitive type has always a value, while non-primitive types can be null.
* A primitive type starts with a lowercase letter, while non-primitive types starts with an uppercase letter.
* The size of a primitive type depends on the data type, while non-primitive types have all the same size.

Examples of non-primitive types are [Strings](https://www.w3schools.com/java/java_strings.asp), [Arrays](https://www.w3schools.com/java/java_arrays.asp), [Classes,](https://www.w3schools.com/java/java_classes.asp)[Interface](https://www.w3schools.com/java/java_interface.asp), etc. You will learn more about these in a later chapter.

Java Type Casting

Type casting is when you assign a value of one primitive data type to another type.

In Java, there are two types of casting:

* **Widening Casting** (automatically) - converting a smaller type to a larger type size  
  byte -> short -> char -> int -> long -> float -> double
* **Narrowing Casting** (manually) - converting a larger type to a smaller size type

## Java Operators

Operators are used to perform operations on variables and values.

Java divides the operators into the following groups:

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

## Arithmetic Operators

Arithmetic operators are used to perform common mathematical operations.



## Java Assignment Operators

Assignment operators are used to assign values to variables.

A list of all assignment operators:



## Java Comparison Operators

Comparison operators are used to compare two values:



## Java Logical Operators

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



## Java Strings

Strings are used for storing text.

A String variable contains a collection of characters surrounded by double quotes

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

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

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

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

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

Because strings must be written within quotes, Java will misunderstand this string, and generate an error:

String txt = "We are the so-called "Vikings" from the north.";

The solution to avoid this problem, is to use the **backslash escape character**.

The backslash (\) escape character turns special characters into string characters:

The sequence \"  inserts a double quote in a string



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

## Short Hand If...Else

There is also a short-hand [if else](https://www.w3schools.com/java/java_conditions.asp), which is known as the **ternary operator** because it consists of three operands.

It can be used to replace multiple lines of code with a single line, and is most often used to replace simple if else statements:

### **Syntax**

variable *= (*condition*) ?* expressionTrue *:*  expressionFalse*;*

## Java Arrays

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

To declare an array, define the variable type with **square brackets**:

String[] cars;

We have now declared a variable that holds an array of strings. To insert values to it, we can use an array literal - place the values in a comma-separated list, inside curly braces:

String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

## Access the Elements of an Array

You access an array element by referring to the index number.

This statement accesses the value of the first element in cars:

String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

System.out.println(cars[0]);

// Outputs Volvo

## Change an Array Element

To change the value of a specific element, refer to the index number:

cars[0] = "Opel";

**Array Length**

To find out how many elements an array has, use the length property:

String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

System.out.println(cars.length);

// Outputs 4

## Loop Through an Array

You can loop through the array elements with the for loop, and use the length property to specify how many times the loop should run.

## Multidimensional Arrays

A multidimensional array is an array of arrays.

To create a two-dimensional array, add each array within its own set of **curly braces**:

### **Example**

int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };