Meaning

Java is a popular programming language, created in 1995. It is owned by Oracle, and more than 3 billion devices run Java. 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

Syntax

Java Syntax

In the previous chapter, we created a Java file called Main.java, and we used the following code to print "Hello World" to the screen:

Main.java

public class Main {

public static void main(String[] args) {

System.out.println(“Hello World”);

}

}

Example explained

Every line of code that runs in Java must be inside a class. In our example, we named the class Main. A class should always start with an uppercase first letter.

Note: Java is case-sensitive: "MyClass" and "myclass" has different meaning.

The name of the java file must match the class name. When saving the file, save it using the class name and add ".java" to the end of the filename.

The main Method

The main() method is required and you will see it in every Java program:

public static void main(String[] args)

Any code inside the main() method will be executed. Every Java program has a class name which must match the filename, and that every program must contain the main() method.

System.out.println()

Inside the main() method, we can use the println() method to print a line of text to the screen:

public static void main(String[] args) {

System.out.println(“Hello World”);

}

Note: The curly braces {} marks the beginning and the end of a block of code.

Note: Each code statement must end with a semicolon.

Conditional Statement

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.

Java Switch Statements

Use the switch statement to select one of many code blocks to be executed.

Syntax

switch(expression) {  
  case x:  
    // code block  
    break;  
  case y:  
    // code block  
    break;  
  default:  
    // code block  
}

This is how it works:

The switch expression is evaluated once.

The value of the expression is compared with the values of each case.

If there is a match, the associated block of code is executed.

The break and default keywords are optional, and will be described later in this chapter

Loops

Loops can execute a block of code as long as a specified condition is reached.

Loops are handy because they save time, reduce errors, and they make code more readabl

Java While Loop

The while loop loops through a block of code as long as a specified condition is true:

Syntax

while (condition) {  
// code block to be executed  
}

Note: Do not forget to increase the variable used in the condition, otherwise the loop will never end!

## The Do/While Loop

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax

do {  
// code block to be executed}  
while (condition);

## Java For Loop

When you know exactly how many times you want to loop through a block of code, use the for loop instead of a while loop:

Syntax

for (*statement 1*;*statement 2*;*statement 3*) {  
  // code block to be executed  
}

code block to be executed

}

**Statement 1** is executed (one time) before the execution of the code block.

**Statement 2** defines the condition for executing the code block.

**Statement 3** is executed (every time) after the code block has been executed.