

# Welcome to the Java Course!

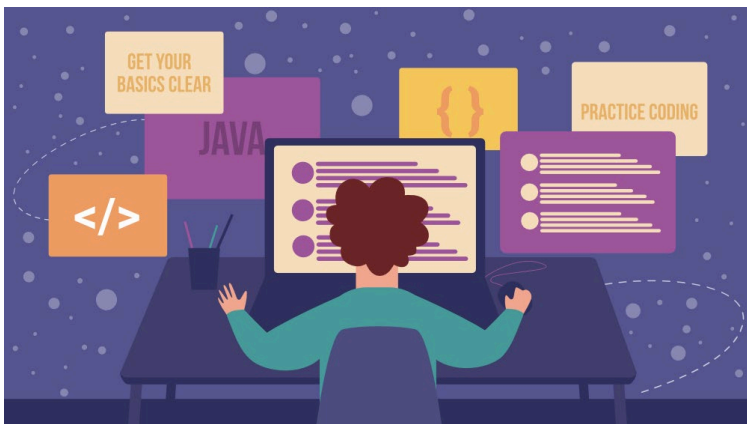
## What is Java & why use it?

Java, developed in the '90s, was originally used for Web-based applications but is an easy-to-use general-purpose language.

It can be used to develop apps for the web, servers, phones & tablets, android apps and software tools.

Java is faster and more efficient than Python as it is a 'compiled language' (meaning the program must be 'built' converting the code into machine code), however, it's more difficult to learn.

Java is excellent for beginners as the '**syntax**' (**we will look at this later in the lesson**) is alike other programming languages, meaning it will be easier to pick up other languages. It's also very exciting as it is used to create games!



What we will look at to begin learning about python:

## Syntax

In programming, code cannot be understood by the computer without the correct syntax – it will return a **syntax error** resulting in a program that will not run!

Of course, that is not what we want so we will start with the basics...

### Java is case sensitive

Let's take the word 'String':

'String' is not the same as 'string'

The computer will understand them as 2 different words.

'String' will be interpreted as a **data type (will explore later)** string.

'string' will be interpreted as generic text depending on where it is placed.

*A string is **an array** of characters.*

### Java apps begin with a class - file names must match the class name

Here is the default Hello World application that you are presented with upon opening and creating your first Java file 'Main.java'. This can look very confusing, but we will look at each line separately.

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("Hello World");  
    }  
}
```

The application above prints the output:

**Hello World**

## Comments

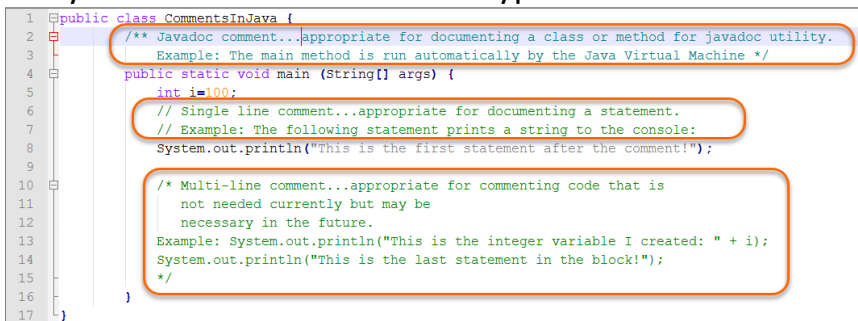
Comments have a couple of uses in programming. The first is obvious, they are used to explain (chunks of) code. Breaking code up with explanations makes it easier to read and understand at a later point or for another user.

The second use is to have the computer 'ignore' lines of code. This is particularly useful for testing new code.

### Java consists of 3 types of comments Syntax

1. Single line comments `//`
2. Multi-line comments `/* */`
3. Documentation comments `/** Comment begins`
  - `* defines parameter`
  - `* or heading`
  - `*/ End`

We only need to look at the first 2 types at the moment.



```
1 public class CommentsInJava {
2     /** Javadoc comment...appropriate for documenting a class or method for javadoc utility.
3     Example: The main method is run automatically by the Java Virtual Machine */
4     public static void main (String[] args) {
5         int i=100;
6         // Single line comment...appropriate for documenting a statement.
7         // Example: The following statement prints a string to the console:
8         System.out.println("This is the first statement after the comment!");
9
10        /* Multi-line comment...appropriate for commenting code that is
11        not needed currently but may be
12        necessary in the future.
13        Example: System.out.println("This is the integer variable I created: " + i);
14        System.out.println("This is the last statement in the block!");
15        */
16    }
17 }
```

The screenshot shows a code editor with line numbers 1 to 17. The code is a Java class named `CommentsInJava`. It contains a Javadoc comment (lines 2-3) highlighted in blue, a single-line comment (lines 6-7) highlighted in orange, and a multi-line comment (lines 10-15) highlighted in green. The code includes a `main` method that prints a string and uses an integer variable `i`.

As you can see comments are also represented in green within code, making them even easier to identify and differentiate from actual code.

## Variables

Variables are used to contain data values such as numbers or text etc...

There are 5 types of variables within Java:

- String – stores **text**, usually all characters. Must be surrounded by “ ” or ‘ ’  
`String colour = 'Green';`  
`System.out.println(colour);`
- int – stores integers which are **whole numbers** (no decimal point)  
`int counter = 10;`  
`System.out.println(counter);`
- float – Floating point numbers are numbers with decimals (**2.2, 4.1**)  
`float distance = 10.3;`  
`System.out.println(distance);`
- char – stores a single character using “ ” (single quotes) e.g. ‘g’ or ‘G’  
`char intial = 'A';`  
`System.out.println(initial);`
- Boolean – Stores 2 types of values : **True / False**  
`bool passed = True;`  
`System.out.println(passed);`

```
1 package demo;
2
3 public class Demo {
4
5     public static void main(String[] args) {
6
7         int result = 100 ;
8     }
9 }
10
11
```

Diagram illustrating the components of a variable declaration in Java:

- datatype**: Points to the word `int`.
- variable\_name**: Points to the word `result`.
- variable\_value**: Points to the value `100`.

## Rules of Declaring variables in Java

- A variable name can consist of Capital letters A-Z, lowercase letters a-z, digits 0-9, and two special characters such as underscore and dollar sign.
- The first character must be a letter.
- Blank spaces cannot be used in variable names.
- Java keywords cannot be used as variable names.
- Variable names are case-sensitive.

## Scope of Variables in Java:

Variable Scope means - The limit, as far as the variable can be used.

In Java there are various types of variable scope:

- Local variables
- Instance variables
- Class/Static variables

## Local variables

A variable that is declared within the method is called local variables. It is defined in method or other statements, such as defined and used within the cache block, and outside the block or method, the variable cannot be used.

## Instance variables

A non-static variable that is declared within the class but not in the method is called instance variable. Instance variables are related to a specific object; they can access class variables.

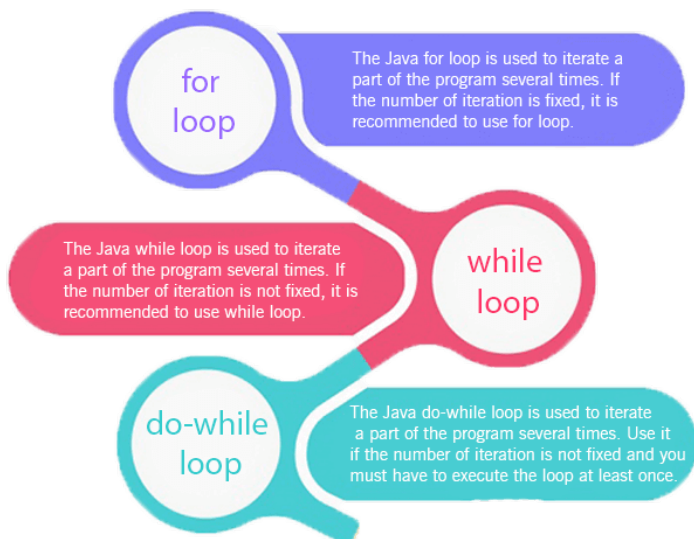
## Class/Static variables

A variable that is declared with static keyword in a class but not in the method is called static or class variable.

Example:

## Loops

There are three types of for loops in Java.



## Now lets compare the three types of loops!

Comparison	for loop	while loop	do-while loop
Introduction	The Java for loop is a control flow statement that iterates a part of the <b>programs</b> multiple times.	The Java while loop is a control flow statement that executes a part of the programs repeatedly on the basis of given boolean condition.	The Java do while loop is a control flow statement that executes a part of the programs at least once and the further execution depends upon the given boolean condition.
When to use	If the number of iteration is fixed, it is recommended to use for loop.	If the number of iteration is not fixed, it is recommended to use while loop.	If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use the do-while loop.
Syntax	<pre>for(init;condition;incr/decr){   // code to be executed }</pre>	<pre>while(condition){   //code to be executed }</pre>	<pre>do{   //code to be executed }while(condition);</pre>
Example	<pre>//for loop for(int i=1;i&lt;=10;i++){   System.out.println(i); }</pre>	<pre>//while loop int i=1; while(i&lt;=10){   System.out.println(i);   i++; }</pre>	<pre>//do-while loop int i=1; do{   System.out.println(i);   i++; }while(i&lt;=10);</pre>
Syntax for infinitive loop	<pre>for(;;){   //code to be executed }</pre>	<pre>while(true){   //code to be executed }</pre>	<pre>do{   //code to be executed }while(true);</pre>