**JAVA: Zero to Mastery**

# **Java Introduction**

In this documentation, I will not talk about how user friendly java is or how important to learn java is or etc. In this specific documentation, I will talk about my own journey of learning java. I will talk about my understanding of java.

# **Getting Started**

To get started with java, at first you need two things.

1. IDE: IDE stands for Integrated Development Environment which is basically a text editor with some cool features that helps in acceleration as a developer. Throughout the entire documentation, we will be using Visual Studio Code (VS Code).
2. JDK: JDK stands for Java Development Kit which is a kit that is not actually required but important for Java. Why is that? Well, we will talk about it in a moment.

So, go through google, download an IDE as your preference or you can download VS Code and download JDK and install both.

Note: While installing JDK, try to install the LTS (Long Term Support) version rather than the latest version. Just because the version is the latest, that doesn’t mean it will support you in the long run. In my case, the LTS version is version 21.

# **JDK, JRE, JVM**

To understand JDK, JRE and JVM better, let’s think of a situation.

*Let’s say, a farmer wants to cook rice from paddy (the grain that rice comes from). Will he directly wash the paddy and transfer that into a pot to cook rice? No. Then, what are the procedures to have rice from paddy? Well, let’s discuss below:*

1. *Firstly, the farmer has to collect rice grain from the paddy with the help of some sort of machine.That means the* ***machine*** *takes paddy as input and rice grain as output.*
2. *That rice grain then goes into a* ***pot*** *with water in which the rice will be cooked.*
3. *The pot (with rice grain and water) goes on top of some* ***flame source*** *(like a stove) that actually cooks the grain and provides us with edible rice (which is our main destination).*
4. *(Optional but useful) The stove might be placed in the kitchen though, without the* ***kitchen*** *the stove can also be placed.*

I hope the procedure is clear. Now, how is that relevant to Java? Let’s see now how a java program works:

You are a programmer who writes readable programs in English (maintaining some programming language syntax). For example, if you write

**System.out.println(“Cooking rice”);**

It will print the text “Cooking rice” and the whole line is written in Java syntax. Which means it is maintaining a language syntax. But your writing language is English right? This is what i meant earlier by “... in English (maintaining some programming language syntax)”,

So, let’s get back in. What will be your Java code running procedures? They will be:

1. Firstly, you write an understandable Java code and pass through some sort of software that will convert your understandable Java code into bytecode. This **software** that converts the whole thing is called a **compiler**. It will take understandable code as input and output as bytecode.
2. Now this bytecode is transferred into **JVM (Java Virtual Machine)** which will give you your desired output.
3. But in order to run the JVM, you need **JRE (Java Runtime Environment)** which will actually run the JVM to give the output
4. (Optional but useful) The JRE might be placed into **JDK (Java Development Kit)** though without the JDK, JRE can exist.

Maybe you get my point about how the farmer story is relevant to Java. for better understanding, follow the following table:

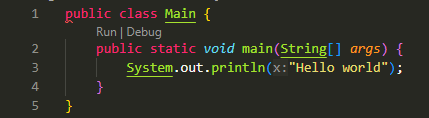
|  |  |  |
| --- | --- | --- |
| **Farmer Story** | *Compared to* | **Java** |
| Paddy | Your code (Java Code) |
| Machine that outputs rice | Compiler that outputs bytecode (JavaC) |
| Pot | JVM |
| Flame source | JRE |
| Kitchen | JDK |

I hope now it’s clear.

# **Syntax**

As for your knowledge, syntax is something that **must be maintained** while writing a code. Every programming language has different syntaxes. So, When writing a code in a specific language, you have to follow the syntax of that corresponding language as well.

## **Basic Syntax (Structure) of Java:**



1. **Main** is a class (in line 1). What is a class?

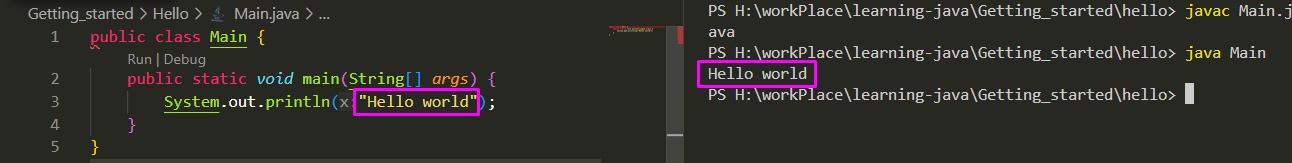
To know the class, you have to have the knowledge of OOP (Object Oriented Programming). Just click on [***OOP***](https://tinyurl.com/2zttas7b)and you will be redirected to our OOP documentation. We will discuss OOP later on but to what extent? In this Java documentation, we will discuss OOP as far as we need. But for core OOP concepts, you should switch to the OOP documentation. But for now, don’t bother with class, just remember it.

1. **main** is a method (in line 2) under which our entire code will take place.

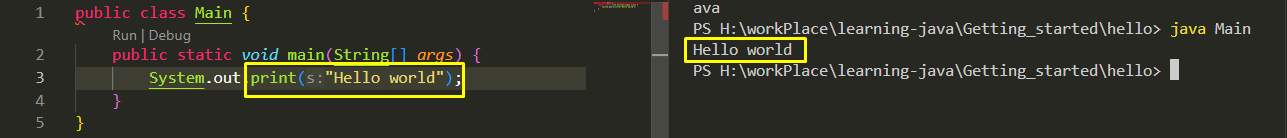
As for the method, method is a way you execute the program/code. By saying “main method”, we meant “This is our main part where we will write the code, the main skeleton code which will run our program”. Now, are there any other methods? a big YES. without methods, we won’t be able to walk through Java. We will discuss those later on. But for now, just remember the syntax and walk with me.

1. (In line 3) **System.out.println(“Hello world”); .** With this line, we print or show outputs in Java. What we basically say to our system is “Hey System, now is your turn to show output. To do that, print the line "Hello world””. ***Here is an interesting fact. We can also do the same thing without the “ln” after the print. What I mean is, if I write System.out.print(“Hello world”); it will provide me the same output.***

With **println,**

****

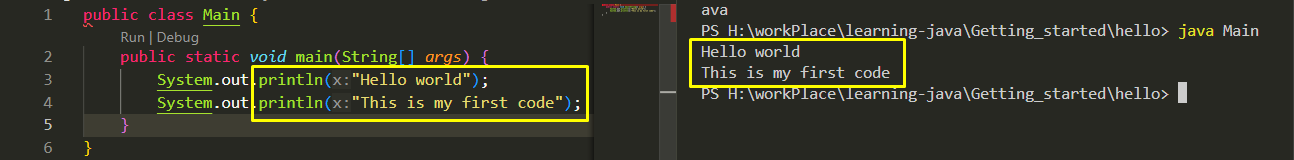
With **print,**

****

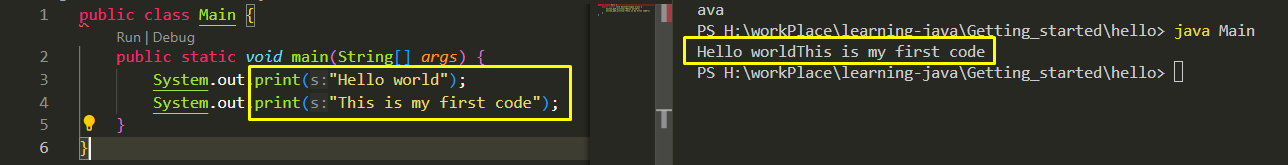
**Then what is the difference?**

The difference appears when you try to print more than one line. For example:

With **prinln,**

****

With **print,**

****

Got the difference? Whenever you are working with **“println”**, it creates an additional line. But when using only **“print”**, it doesn’t create any additional lines. So for better practice, it is good to use **print with a ln.**

## **2. Variable declaration:**

1. Variable: Variable is something in which you store data or values. As you need a pot to store water, the same way you need a variable to store data or values into your system memory.
2. Declaration: in terms of variable declaration, it means creating a variable where the value is to be stored. When you create a variable, it will be called a declaration.

So, how does this whole **“Variable Declaration”** work? The basic syntax to declare a variable is:

**data\_type variable\_name;**

NOW! I understand what a variable\_name is. It is the name I am giving to the variable. BUT what is this **“data\_type”** is? Let’s hope into data types!

### **Data Types:**

Data types are the types of data that can be used in a programming language. For example, numbers are a type of data; characters are a type of data. In this way, the types of data are referred to as data types. But there are some more deep classifications. What are those? Those are:

Now what?༼ つ ◕\_◕ ༽つ

Let’s learn them one by one.

**Primitive data** are the data types that are built-in in Java. Which means, when developers created Java, they also created these. If we (as user) create a data type, then it will be called non-primitive i.e. class. REMEMBER? We took the name of the class in the very beginning. Follow the table below for understanding, but only on primitive now. We’ll talk about non-primitives later.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Primitive type |  | keywords | Size (byte/s) | Description |
| Number | Integer | byte | 1 | Stores’ whole numbers from -128 to 127 |
| short | 2 | Stores’ whole numbers from -32,768 to 32,767 |
| int | 4 | Stores whole numbers from -2,147,483,648 to 2,147,483,647 |
| long | 8 | Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |
| Non integer | float | 4 | Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits |
| double | 8 | Stores fractional numbers. Sufficient for storing 15 to 16 decimal digits |
| Boolean |  | boolean | 1 bit | Stores true or false values |
| Character |  | char | 2 | Stores a single character/letter or ASCII values |

**SOOOOOO!!!!!  
WHERE WE WERE?**

We were in “variable declaration” under syntax, and we have got some idea on variables now. So if we recall the basic structure of the declaration of variable, it was,

**data\_type variable\_name;**

That means if we want to store some regular integer numbers in a variable named “number”, then we will declare that as:

**int number;**

If we want to store a character in a variable named “firstLetter”, then we will declare that as:

**char firstLetter;**

*Smooth like Butter,*right? **¯\\_(ツ)\_/¯**

**Practices With What We Learnt**