Introduction to Java: A Beginner's Guide

After more than two decades, Java is still a popular choice when developing applications.

In this tutorial, we'll explore Java's fundamentals, history, key features, and role in software development.

Java belongs to object-oriented programming languages. Object-oriented programming (or OOP for short) is a paradigm that organizes software around classes and objects.Unlike the other concepts, like procedural or functional programming, in OOP, classes represent the main building blocks. Moreover, we write our code inside some classes.

Additionally, we can make every class "alive" by instantiating it (just a fancy word for creating an object). Objects communicate with each other, sharing information, changing states, and performing actions.

Usually, when creating classes, we bring them close to real life and the way we think.

For instance, let us imagine we want to create a calculator. Instead of a physical calculator, we would have the Calculator class. Additionally, each action (addition, subtraction, multiplication, division, etc.) would be represented by methods.

1. The Interview with Java

Q: Hey, Java, I've heard many stories about you, but I thought they were just rumors. Since you're here at last, I'd like to know you better. Firstly, how old are you?

A: Hi, mate. Yep, I'm pretty much real, and this isn't a question you should ask a lady. Watch your manners! But, anyway, I'll answer. I'm almost 30 years old.

Q: Wow, that's old. I mean… nice. Tell me more about yourself.

A: Well, people tend to choose me when developing applications. I'd say I'm quite a catch. I belong to object-oriented programming languages.

Q: What are those?

A: Ah, I see. We're dealing with a newbie here. Object-oriented programming (or OOP for short) is a paradigm that organizes software around classes and objects. In OOP-based languages, classes represent the main building blocks. To put it differently, you're writing your entire codebase inside classes.

Q: I don't want to sound ridiculous, but what are classes and objects?

A: Let me rephrase. You create classes in your code. Then, you can make every class alive by instantiating it. It's a magical process. Now, objects communicate with each other, share various information, change states, and perform some actions. Usually, when creating classes, we bring them closer to real life and how we think. For instance, suppose you want to create a calculator. Instead of a physical calculator, you would create the Calculator class. Additionally, each action (addition, subtraction, multiplication, division, etc.) would be represented by a method. Simple, right?

Q: And how did you come to this world?

A: Great, I like telling this in the form of a story.

Once upon a time, there was a company called Sun Microsystems. In 1991, the company decided to be different than all the other IT companies. Thus, they started a brand new, fresh-from-the-oven project – The Green Project. Unlike one would think, no, it was not about the environment and saving Mother Earth. The project's main goal was to create a programming language that could run on different operating systems.

There was a brave man, let us call him James Gosling, who worked hard in his office on this super cool magic programming language. Like any other programmer, he faced difficulties finding a name fitting the new language well.

And then, all of a sudden, it came up to him. The little angel whispered in his ear – "Look at this beautiful tree in your yard.". He looked at this window and only saw the big oak. Therefore, he decided to name the language Oak.

You know that feeling when you are so excited about something, and one person tells you, "NOPE.". The same thing happened to James. The marketing team from Sun noticed there was already an IT company

named Oak.

They were back at square one. However, James continued the name discussion over a nice cup of coffee. During this coffee, they agreed that the name of the new language would be Java.

The first version of Java was published in 1995. Finally, after 15 years, Oracle decided to purchase Sun Microsystems with all its products (Java included).

Oracle and Java lived happily ever after.

THE END.

2. How Can I Work with Java?

We, as developers, represent only a tiny piece of a vast and very complex puzzle. We create software solutions by writing commands that perform specific actions. And this is where our job ends. After, Java says: “I’m the captain now.” and takes the control. In addition, Java has a best (although very strict) friend called the compiler. Simply put, the compiler translates your code into the commands the Java Virtual Machine (JVM) understands. After all, the JVM is responsible for running your application.

2.1. Step 1:

You create the source code, which consists of files with .java extensions and some code in them. You can write code in your chosen text editor or use an IDE (Integrated Development Environment), such as IntelliJ IDEA or Eclipse.

2.2. Step 2:

After you’re satisfied with your code, you pass the baton to the compiler. You can think of the compiler as a Grammar Nazi. The main task of the compiler is to check whether you wrote the correct Java syntax. If you haven’t, it’ll scream, “You shall not pass!”. You’ll need to go back to step 1 and check what’s wrong with your code.

2.3. Step 3:

Once your code is error-free, the compiler transforms it into code that the JVM understands. The JVM does not understand anything unless it’s written in pure bytecode. Moreover, the compiler creates a new file with a .class extension and puts the bytecode in it.

Lastly, the JVM will run our application.

3. Key Java Features

Let’s look at some core features that make Java the perfect choice when developing applications.

3.1. Object-Oriented

First and foremost, Java is an object-oriented programming language. It emphasizes using classes and objects.

Thanks to concepts such as inheritance, encapsulation, polymorphism, and abstraction, it encourages

code reusability.

3.2. Platform Independence

You may have heard the saying, “Write once, run everywhere.”. This means that Java was designed to run independently of the operating system it’s running on. Platform independence is possible thanks to the JVM. Our code is translated into bytecode, which can run on any system with a compatible JVM.

As a result, it simplifies cross-platform development.

3.3. Picking up Trash

Do you know those people who don’t know how to pick up a plate after they finish a meal and put it in a dishwasher? Well, Java isn’t one of them. It likes to keep its house clean – it manages its memory.

When objects are created, they take up a certain amount of memory space. Java uses Garbage Collection (or GC for short) to remove unused objects from memory when they are no longer needed.

It helps prevent memory leaks, and it helps prevent us from running off the memory space.

3.4. Pointers no More

Unlike C programming language, Java doesn’t use pointers. Regarding objects, Java becomes very protective (like any other parent would) and doesn’t allow us anywhere near them. In other words, we can’t access objects directly or get the memory’s address. The only way to access them is by using references.

4. Hello World Example

Now that we covered some basics, it’s time to make your first cup of Java. I’ll guide you through the steps required to start.

4.1. Installing Java

1. The first ingredient you’ll need (besides the PC) is the Java Development Kit (JDK). It contains everything you need to develop and run applications written in Java programming language. You can use the official Oracle website or OpenJDK.
2. Once you’ve downloaded the JDK, you need to install it. You can double-click on the downloaded file and follow the instructions.
3. Next, you need to add the JAVA_HOME environment variable. The value should contain the path to your Java installation directory. For instance, on Windows OS, the value could look something like C:\Program Files\Java\jdk21.

4.2. Setting up Your Java Playground

Before programming gained popularity, developers used a text editor like Notepad to develop applications. Nowadays, we use an Integrated Development Environment (IDE) such as IntelliJ IDEA, Eclipse, or Visual Studio Code. These IDEs can help you speed up the development process. In addition, they come with an integrated Java compiler. Thanks to this compiler, you can see the compiler errors at runtime.

I prefer using IntelliJ IDEA. It’s simple, intuitive, and intelligent, saving a lot of time during the development

process.

4.3. Making Your First Java Cup

Now that you’re all set, let’s write your first Java program. Before continuing, let me say it’s perfectly fine if you don’t understand much (especially if you have no prior coding experience). This example is here to familiarize you with the way Java works. Everything will be explained in the later chapters.

In the following steps, we’ll use the IntelliJ IDEA environment. However, you’re free to use any IDE of your choice.

Lastly, let us write a simple “Hello World” program.

Firstly, we need to create a file named Demo.java. Next, let us put the code that prints “Hello World” in the console:

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

Now, we can use the javac compiler to compile Java code into bytecode:

```
javac Demo.java
```

If we do not get any error, we can run our code with java command:

```
java Demo
```

And this is it. It should print “Hello World” into the console.

5. Key Takeaways

In this tutorial, we got an introduction to Java programming language.

Java has a large and active developer community, which means there are ample resources, libraries, and frameworks available for various applications, such as web development (e.g., Spring Framework), mobile app development (e.g., Android), and more.

Read More