Learn Java

Github repo | https://github.com/Thesnowmanndev/Learn-to-Program-Software

Learn the Foundations of the java programming language

Kyle Martin

2020

Contents

[Section 0 – Key Skills & Concepts 3](#_Toc56360300)

[Java Overview 3](#_Toc56360301)

[What is java? 3](#_Toc56360302)

[Origins and Philosophy of Java 3](#_Toc56360303)

[Javas impact on the internet 3](#_Toc56360304)

[Java Bytecode 3](#_Toc56360305)

[Object-oriented Programming 3](#_Toc56360306)

[**Absolutely important for you to understand** 3](#_Toc56360307)

[Section 1 - Getting started in Java 5](#_Toc56360308)

[The Java Development Kit 5](#_Toc56360309)

[Installing an IDE (Integrated Developer Environment) 5](#_Toc56360310)

[First Simple Program 6](#_Toc56360311)

About Github Repo Author

 Hello, everyone who is looking at this Github repository. My name is Kyle and I am here to help others learn how to program. Computer programming skills do not come easy for most people. Especially me. I am not some child prodigy who learned how to program over night. Hell, still to this day do I think there is so much more I can learn, and I know I will be learning topics related to computers, programming, and technology till the day I die. So how did I learn to program? Well it all started back in 2006 or 2007 (my memory is hazy on when I became interested) when I was hard core into MMORPGs. My friends and I would always browse MMORPG.com or other MMORPG list websites for new games. One of my friends came across a website dedicated to private servers of MMOs we have played. This was big, not only for my future hobbies, but just because we had more options to create memories on MMOs. A game we really liked playing was RuneScape. It wasn’t long until we got the idea to create our own private servers for our friends to play on. This is what got me interested in learning how to program computer games and applications. I started to learn the beginning concepts of Java way back then. However, it wasn’t until 2018 when I got a job as an instructor for Airframe, Powerplant, and General Aircraft Maintenance courses that I got back into computer programming. See becoming a teacher encourages you to not only change your view on education but it gets you into wanting to learn things that interest you. Before I joined the US Air Force, I wanted to become a Game Developer or a Programmer, but I did not have money for a college education. After 8 years of service I decided to pursue those original goals of mine and finish out my Bachelor of Science with a Major in Computer Programming. This repo is my way of helping new learners learn fast and establish good skills to take forth with them in their computer programming journey. I hope this helps.

Github Repo - <https://github.com/Thesnowmanndev/Learn-to-Program-Software>

Key skills & concepts

# Section 0 – Key Skills & Concepts

## Java Overview

* Philosophy of Java
* Java’s contribution to the Internet
* Importance of bytecode
* Foundational principles of object-oriented programming

## What is java?

The Java programming language is powerful, packed full of features, and a ‘[general-purpose](https://en.wikipedia.org/wiki/General-purpose_language)’ programming language. Java is an excellent programming language to learn for those who are new to computer programming. Java is a ‘[class-based](https://en.wikipedia.org/wiki/Class-based_programming)’ and ‘[object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming)’ programming language. It was developed to let application developers *write once, run anywhere (WORA)*. Java can run on any platform that support Java even without the need for recompilation.

## Origins and Philosophy of Java

Section needs to be written.

## Javas impact on the internet

Section needs to be written.

## Java Bytecode

Section needs to be written.

## Object-oriented Programming

Section needs to be written.

## **Absolutely important for you to understand**

Something I struggled with when I first got into programming was the question “What can I do with it?”. It seems like a lot of courses, even college courses, forget to show you a practical use for the language. It seems like an easy question to answer but you must remember that it is very easy to lose sight of your end goal or purpose. So, I encourage you to revisit the purpose of why you want to learn Java or programming in general in the first place once a month.

It is also easy to get “lost in the noise” as I would say when learning a programming language. Odds are if you are learning a programming language, no matter what it is, you have done a bit of research and seen videos on YouTube related to the language. Hell, you most likely even have a few subs to programming channels on YouTube. I must warn you that every year YouTube gets flooded with videos like “Languages to learn in 202X” or “Best programming language to learn” or even “Why X language sucks!”. You need to develop the discipline to critically analyze these videos and either disregard them entirely or filter out the noise. They will sway you into learning other languages, or discourage your progress, and even make you lose sight of why you started learning Java in the first place as X language may seem more enticing. A big thing you need to learn is that computer programming languages aren’t entirely like learning a language such as Japanese, Spanish, or French. You are learning how to give a computer instructions to do a process. If you do not understand that then these videos will make you lose sight as they will imply you are wasting your time. You absolutely are not wasting your time, regardless if you are learning Java, C, C++, C#, Python, Ruby, JavaScript, or any other programming languages. Just stick with one and master the fundamentals. You will be able to easily transition between languages once you do.

For me I started with Java because the private servers I liked to run for my friends used Java. I barely skimmed the surface. A decade later I revisited learning how to program and then I spent 2 years stuck in the “Why you should be learning X language and not X language” loop. I then went from Java to C# to JavaScript to Python and then back to Java without learning more than a few fundamental concepts. If I can, I would like to save you from getting stuck in that loop as well. In the computer programming world, you will learn and utilize programming languages that suit your needs best for whatever you are developing at the time. So, for now, stick with one and master the fundamentals.

Getting Started

# Section 1 - Getting started in Java

## The Java Development Kit

If you are just getting started out with programming and Java, then you will need to do this sub section in order to compile and run future programs you develop. Odds are on your PC right now you have Java installed. Just not the JDK also known as the Java Development Kit. Installing this is super simple, but once again mandatory to proceed.

If you will be utilizing my suggested IDE in the subsection below then you can skip these steps as it has a feature to download and install JDKs for you (thanks for that JetBrains!)

1. Open a browser tab and search for “Java Development Kit” on your chosen search engine. One of the first links that pops up should be Oracle.com Java SE Development Kit.
2. Scroll down on the website and find your applicable JDK for your operating system. Mine will be Windows x64 Installer (yours may be different like Linux or MacOS). Download it.
3. ~~~ To be Continued ~~~

## Installing an IDE (Integrated Developer Environment)

I was conflicted with including an IDE in this lesson. I personally feel an awesome way to learn to program is using a text editor such as Notepad ++ with the autocomplete feature disabled. The reason being is that Notepad++ will still stylize your code for you with different colors letting you recognize certain features like keywords but not have you get reliant on Smart Code Completion features commonly found in IDEs. However, Java is quite complex, and I do not want to overload you with so many things at once like writing code, formatting, compiling, terminal use or jars, etc. Plus, a lot of people on dev.to were quite abrasive when I suggested newcomers shouldn’t get reliant on IDEs and that using Notepad++ is more than enough when first learning to program.

**The IDE I will suggest you use while following this lesson/repo will be IntelliJ IDEA**. They have a free version that is open-source, robust, and will help you greatly with programming. They also make it for Windows, MacOS, and Linux. Honestly you can use whatever Java IDE you prefer but IntelliJ is my go-to. If you do not use IntelliJ, you may have to change instructions in subsections to fit your IDE. I will not be writing this guide for all IDEs. I will focus on IntelliJ IDEA.

1. Open a browser tab, navigate to your chosen search engine, and type in IntelliJ IDEA
2. The first, maybe second depending on your search engine and location, link should be jetbrains.com click that
3. Feel free to browse this page and familiarize yourself with the product but don’t get stressed out by all the features. When you are ready select Download
4. On the downloads page you will see a section for Windows, MacOS, and Linux. You will also see a section for Ultimate and Community. I will suggest you use the Community version but if you are currently attending a college you can create a Jetbrains account and redeem the student pack so you can get the Ultimate version for free!
5. Select download for your appropriate operating system. Then install the program. When installing it, if you are asked an option to check a box that says “Add to PATH” make sure you check that box.
6. Once installed open it up and you will be greeted with the Welcome Page of the program.

## First Simple Program

Its tradition in the computer programming world to create a program that simply prints out a sentence or “string” into the terminal. If you have dabbled with other languages odds are you have created this program. If you haven’t and this is your first-time programming than here, we go!

1. Select “+ New Project” on the IntelliJ IDEA Welcome Page.
2. In the Project SDK drop down select Download JDK. It should download the latest version of JDK for you. Once done it should be set as your Project SDK now.
3. Ensure in the top left Java is highlighted and click the next button.
4. You will now be prompted to Create a project from template. Disregard this and hit Next.
5. Choose your project location (Where you want your project to be located on your computer). Hit “…” and navigate to whatever HDD/SDD or folder you want to create the directory for future lessons in this repo. Then hit the folder icon that has a + and name it whatever you would like. I named it “Learn to Program”. Once again in this new folder hit the folder icon with a + and name this one “Java”. One more time make a new folder for “Fundamentals”.
6. Your Project location should look like “C:\Learn to Program\Java\Fundamentals”
7. Now in Project name: name it “HelloWorld”.

If you have followed that correctly you should see something like this:

Text

Description automatically generated

Welcome to the IntelliJ IDEA environment! It will feel overwhelming at the beginning but don’t fret you will get used to this program fast. It has a lot of features, so it is easy to get lost. If you get stuck or lost do some Google searching to figure out how to get where you need to be. After all a big part of computer programming is learning how to utilize search engines to help you out when you get stuck.

Let’s break down some important parts of IntelliJ before we press on. We will be utilizing these components often so its good to learn about them now.

**Menu Bar:**

****

The menu bar is simple as it is in most computer programs. You have various options to help you out with whatever you may need. IntelliJ has a few more options that we will get used to using. For example, you can see a green hammer icon. That represents the *Build* button. It will compile our programs for us using the Java compiler (also known as javac). Compiled programs are no longer in code format. Instead they are a compilation of our code in machine language to allow a computer program to run. Similar to .exe’s on Windows.

**Project Structure / Inspector**

A picture containing text

Description automatically generated

The project inspector or project structure window is a very important aspect of your IDE. It allows you to expand or minimize folders in your program. You can also view External Libraries that you can import into your projects or ones you have imported. We wont worry about External Libraries at this stage of fundamentals.

**Main window/view**

Text

Description automatically generated

This is the window that our code will show up if we were to have a .class open. This is where you will be able to edit your code in the program or read the code.

**Footer/Terminal:**

****

This is IntelliJ IDEA’s footer. This is where we can read problems that arise in our program. Check out our git structure if we were working on a repo. Look at our TODO list in bigger programs. Or most used will be our terminal. When we first start, we will be writing a lot of terminal applications. Most of the fundamentals is learned while programming terminal applications instead of GUI applications. Don’t worry. Everything we learn in the fundamentals will progress over to writing bigger applications like websites, GUI applications, and much more.

Now that we are familiar with our IDE we can press on.

1. Expand your HelloWorld folder in the Project Structure pane. You should see the .idea and src folders.
2. Right click on the src folder and click New > Java Class
   1. A java class in IntelliJ is simply a file with the file format .java. It is what is fed to the javac and then compiled into an application. Most of the applications we create in the fundamentals section will have the file format .class once it is compiled. Any device that has Java installed can run .class files. That’s where Javas philosophy “Write once, run anywhere” comes into play.
3. Name this new file HelloWorld
4. Now it should automatically open and you should see something like this:

Graphical user interface

Description automatically generated

1. The very first thing we are going to do is give this file some room. You should see a open curly brace and a closed curly brace. Everything between those braces will belong to this class named HelloWorld. In future projects you will have to import code from other classes that way it can be read or manipulated. Go ahead and click after the open curly brace, the one that looks like “{“, and hit enter. Now you should have some room to write your first application.
2. In Java programs you need a method called “main”. Whenever you create a program and it is ran by the Java VM it will look for the main method. Think about it like a starting line.
   1. In your newly created line start to type “main” without quotes and give it a second. Code complete will pop up with a suggestion for the main method. Hit the TAB button on your keyboard and the IDE should automatically create the boilerplate for the main method. It should look something like this:

public static void main(String[] args) {  
   
}

* 1. Don’t focus too much on the syntax of this method too much right now. I will explain the keywords public, static, void as well as String[], and args in future sections of this repo. All you need to know right now that this is a method in Java, and this is typically what main methods look like. Here is an example of another method (Do not write this code):

void anotherMethod() {  
   
}

* 1. Notice how in that custom method I wrote it starts with the keyword “void” and has a name (I use camelCase when writing names for my methods). The name is any name you want your method to be called but typically you want it to be descriptive and too the point. More about clean code in future sections. And then finally it has () followed by an open and closed curly brace. Inside the ()s I would be able to pass it arguments or variables from other code blocks. This method. I also left off keywords or modifiers like public and static because they are not always mandatory for every method.

1. Now it is tradition when learning a computer programming language to first learn about printing a single sentence to the terminal. In java we will write this code inside our main method:

System.*out*.println("Hello, World!");

* 1. Let’s analyze this line. First, we are calling the System class. It is a built-in class to the Java language package. Second, we are calling the out variable which is in the Java IO package. Third, we are calling the method println which is a method in the Java IO package. Finally, inside the () of the println method we are passing an argument “Hello, World!”. This is also known as a String. So, in short, we are importing code from another class that was already written by another developer to make our job a little bit easier and we are passing that method an argument. In this case a String.

1. If you followed along your code should look a little something like this:

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

1. Now let’s build our program! Hit the green hammer icon on our menu bar. This will take a second and if you did everything correctly your project should now be compiled. Next to the numbers in your code window 1 and 2 should have a green play button. If you click on either one of those you should be presented with an option to run the main method. Click that.
2. Next your terminal should now appear in the bottom and look something like this:

A picture containing text

Description automatically generated

Awesome! You just wrote your first Java application. You can see in the terminal (which is built into the IntelliJ IDEA) our string “Hello, World!”. If you look over in your Project Structure you will now see a folder named out. This is where our compiled HelloWorld.java file got turned into a HelloWorld.class. Referring to the note in step 2 you will see what a class file is and how it is used.

If you encountered an error. Try reading what the terminal is trying to tell you. This is a simple program so really the only things that could have gone wrong is either you didn’t install the JDK (if so refer to steps above), created a syntax error, or misspelled something (yes I know technically also a syntax error). Reading errors on the terminal can be intimidating and frustrating. Do not give up. Error notices are there to help you see what you messed up. If it is confusing, try Google searching the error. Odds are you will find a website where someone also has a similar issue to yours. Once you figure out what you did wrong, ask yourself questions why that happened in the first place. That way you can start to learn and internalize how the Java language is formatted or how the Java Virtual Machine or Java Compiler needs code to be written in order to properly function.

**Program Complete! Good work!**