What is Kotlin?

Kotlin is a statically typed programming language running on Java Virtual Machine, Android, web browser, or even native as a binary.

Kotlin is **concise:** drastically reduces source code boilerplate. It is **safe:** (almost) impossible to get NullPointerException (NPE). We will learn about NPEs in upcoming sections of this book. Kotlin is **versatile:** it is a general-purpose programming language. Finally, it is **interoperable**. That means that we can use existing JVM libs and frameworks in Kotlin, or use Kotlin developed libraries in other JVM languages.

A team at JetBrains (creators of IntelliJ Idea IDE) developed Kotlin, an open-source language with an army of external contributors.

Kotlin is licensed under Apache 2 Open-Source License. The current version of Kotlin is **1.7.10**.

Some basic concepts explained

Before we start with Kotlin let's explain a couple of terms that we have mentioned in the previous section just in case that you are not yet familiar with them.

What is a statically typed programming language?

We said that Kotlin is a statically typed language. We will explain the meaning of statically typed programming languages in comparison between statically typed and dynamically typed programming languages. Dynamically typed programming languages perform type checking on runtime. On the other hand, statically typed languages such as Java and Kotlin perform type checking at compile time. In Kotlin as an example, variables must be declared before values are assigned to them.

What is Java Virtual Machine or JVM?

If you are not experienced with Java, let's take a moment to explain what JVM (Java Virtual Machine) is. JVM is software running on your system (engine) that is responsible for running all Java software. JVM converts Java bytecode (we will talk more about it soon) into the machine language. Thanks to this JVM applications are the same on all platforms.

Software developers write JVM applications once for all platforms. That code is executable on every operating system. All specificity for the particular operating system is handled by the version of JVM installed for that system. So, we have installations of Java for Linux, macOS, Microsoft Windows, and so on. It is also worth mentioning that JVM is part of JRE (Java Runtime Environment).

Kotlin fits into this as JVM language, which means, all code that we write in Kotlin is compiled into Java bytecode and executed on JVM (Java Virtual Machine).

Java bytecode

We will spend one brief moment explaining the meaning of Java bytecode. Java bytecode represents the instruction set for JVM. Our source code of the program is translated into Java bytecode during compilation by the Java

compiler (Kotlin has its compiler for this purpose). Bytecode is then loaded into JVM and its instructions are executed.

The lifecycle of a typical JVM program looks like this:

- source code for the program
- compiler creates Java bytecode from program source code
- JVM loads bytecode that has been created by the compiler
- JVM converts bytecode into machine code (language)
- instructions are executed by computer hardware.

Note:

If you are not familiar with Java at all we will explain some of its concepts in the "Legacy of Java" section of this book.

Basic characteristics of Kotlin

Kotlin has many powerful characteristics. We will highlight some of the most important ones.

As we previously mentioned **Kotlin runs on JVM** (Java Virtual Machine) meaning that it is cross-platform compatible. For you who don't know, Kotlin is also available in its **Kotlin native** and **JavaScript** flavors. That practically means that we can write the code that will be compiled directly for the execution on computer hardware or we can write Kotlin programs for the web.

Kotlin is **statically typed**, the type of a variable is known at compile time.

Kotlin is **functional** too. With first-class functions, you can store functions as variables or pass as parameters to other functions as parameters.

Immutability: It is guaranteed that the state of an immutable object can't change over time.

Kotlin has **coroutines**. Thanks to coroutines support Kotlin makes your life easier when comes to asynchronous or non-blocking programming. We will cover coroutines in a separate section of this book. Other previously mentioned features will be covered in separate sections as well.

And last but not least, Kotlin is an **object-oriented programming language** that is free and open source. If you are interested to contribute to Kotlin development you can check out (or fork) GitHub repository:

https://github.com/JetBrains/kotlin.

Where is it used?

As a general-purpose programming language Kotlin can satisfy the needs of every professional software developer. We will cover some use-cases of Kotlin.

Server-side development

Kotlin is great for developing server-side applications. Kotlin allows you to write concise and expressive code while maintaining full compatibility with existing Java-based technology stacks and a smooth learning curve.

So, what are the benefits?

- Expressiveness
- Scalability
- Interoperability
- Easy migration
- Great tooling
- Easy learning Curve.

This is the list of some web framework that you can try for web development:

Ktor, https://ktor.io
Spring, https://spring.io
Vert.x, http://vertx.io
Kotlin DSL for HTML, https://github.com/kotlin/kotlinx.html
Wasabi, https://github.com/hhariri/wasabi
Hexagon, https://github.com/jaguililla/hexagon

and many others.

Android mobile development

Kotlin is the main programming language for writing Android applications. It is unimaginable by many Android developers to do their regular programming in anything different than Kotlin. To find out more about Android you can visit the official Android developer's page:

https://developer.android.com.

On market, some tools go beyond the standard language features. One of them is Android KTX, the set of Kotlin extensions for Android development:

https://developer.android.com/kotlin/ktx.

JavaScript development

Kotlin provides us with the ability to produce JavaScript builds. It does so by translating Kotlin source code to JavaScript. The current implementation targets ECMAScript 5.1 but there are plans to eventually target ECMAScript 2015 too.

When you choose the JavaScript target, any Kotlin code that is part of the project as well as the standard library that ships with Kotlin are translated to JavaScript. This excludes the JDK and any JVM or Java framework or library used. Any file that is not Kotlin will be ignored during compilation.

Native development

Kotlin can be compiled to run directly on computer hardware. For this purpose team at JetBrains developed Kotlin/Native technology that is developed side by side with the main Kotlin language. It is really easy to write Kotlin applications that will be delivered as native platform binaries. We will talk more about Kotlin/Native in upcoming sections.

Kotlin for data science

Kotlin found its place even in data science. Kotlin supports integration with some very popular platforms used by data scientists. We will mention some of the most popular.

Apache Zeppelin (comes shipped with Kotlin interpreter):

https://zeppelin.apache.org/

Project Jupyter:

https://jupyter.org/
https://github.com/Kotlin/kotlin-jupyter.

It is also worth mentioning that because of Kotlin's huge popularity developers community created a significant number of great libraries for data-related tasks. We will list some of them.

JetBrains's Lets Plot:

https://github.com/JetBrains/lets-plot, an open-source plotting library for statistical data.

Kmath:

https://github.com/mipt-npm/kmath, a Kotlin-based analog to Python's "numpy" library. In contrast to "numpy" and "scipy" it is modular and has a lightweight core.

Kotlin Statistics:

https://github.com/thomasnield/kotlin-statistics, collection of helpful extension functions to perform exploratory and production statistics in a Kotlin-idiomatic way.

Krangl:

https://github.com/holgerbrandl/krangl, Kotlin library for data wrangling.

Kravis:

https://github.com/holgerbrandl/kravis, a Kotlin grammar for data visualization.

okAlgo:

https://github.com/optimatika/okAlgo, Idiomatic Kotlin extensions for ojAlgo (https://www.ojalgo.org/), with some inspirations from PuLP (https://github.com/coin-or/pulp).

Data2viz:

https://data2viz.io/, data visualization.

Sparklin:

https://github.com/khud/sparklin, Kotlin language support for Apache Spark.

Koma:

https://github.com/kyonifer/koma, a scientific computing environment for Kotlin.

Komputation:

https://github.com/sekwiatkowski/komputation, a neural network framework for the JVM written in Kotlin and CUDA C.

KotlinNLP:

https://github.com/KotlinNLP, natural language processing in Kotlin.

Jinx:

https://exceljava.com/docs/tutorials/kotlin.html, an Excel Add-In that enables developers to extend Excel's capabilities.

Kotlin Algorithm:

https://github.com/gazolla/Kotlin-Algorithm, implementations of popular algorithms and data structures including machine learning.