Krzysztof Rabka, Adam Schaible

CS 3210

April 05, 2019

Introduction to Kotlin

There are many programming languages and Kotlin is one of them. Kotlin expands vast number of implementations that are taken from Java. Here we will focus on brief beginnings of Kotlin, who are the people involved, what were the primary goals for development, what are some of the key paradigms and finally what are some key strengths and weaknesses of the language. Kotlin started as an idea to improve on what Java has to offer (Jemerov xviii). It all started in St. Petersburg, Russia where couple of software developers decided to design a brand-new language (xviii).

The three main developers are Svetlana Isakova, Andrey Breslav and Dimitry Jemerov. Svetlana Isakova works at JetBrains as a developer advocate, she is also co-author of the book “Kotlin in Action” and “Atomic Kotlin” (xvii). From the very early stage, she was involved in the development of Kotlin (xvii). Her main area of focus was overload-resolution of subsystems of the compiler (xxiv). Once Kotlin emerged as production quality product, she began to teach Kotlin and give speeches worldwide during conferences (Svetlana Isakova, Instructor).

Another person, Andrey Breslav is the creator of the Kotlin idea (Andrey Breslav, Instructor). He used to work mainly in academia (Andrey Breslav, Instructor). In 2010 during a visit to JetBrains he was part of a discussion that touched on current production issues with JVM (Andrey Breslav, Instructor). After an hour into the discussion he came up with the first ideas for Kotlin and sketched his thoughts on the whiteboard (Andrey Breslav, Instructor). Shortly after, he decided to join JetBrains to lead the design of the new programming language and work on the compiler (Jemerov xv). Currently, similarly to Svetlana, he also presents as a speaker at conferences and on top of that he maintains Kotlin blog (Andrey Breslav, Instructor).

The third main person involved from the start of Kotlin is Dimitry Jemerov (Dmitry Jemerov). He worked at JetBrains as IntelliJ IDEA product manager (Dmitry Jemerov). He is co-author of “Kotlin in Action” together with Svetlana (Dmitry Jemerov). He began his career in St. Petersburg, however, he moved to Munich, Germany to elevate his career (Dmitry Jemerov). Dimitry is the creator and lead developer of PyCharm, which is one of the successful products created at JetBrains (Dmitry Jemerov). During the early stage of the development, he worker on Kotlin at Google (Dmitry Jemerov). He was also working on the IntelliJ IDEA plugin for Kotlin and leading the Kotlin developers’ team (Dmitry Jemerov).

Kotlin was developed with many crucial goals in mind. These goals include to be a programming language that is based on Java and targets the Java platform (Jemerov 3). Since it is a new language, it is also modern, which makes it to be concise, safe, pragmatic, and focused on interoperability with Java code (3). Also, Kotlin by design can help developers complete their goals with fewer obstacles while writing the code (4). Kotlin is statically typed language and it can be used everywhere where Java is currently used (4). Some examples include server-side development, which are usually backend of web applications (4). We can use it also for Android development, or for mobile applications (4). Kotlin works perfectly with all Java libraries and frameworks, and it performs as effectively as Java (3). Since Java is popular and is used in many environments, such as Google, Twitter, LinkedIn and for other internet-based companies, Kotlin can be used interchangeably with Java (4). With Kotlin for iOS devices we can use the Intel Multi-OS Engine to run the code (5). For desktop applications we can implement Kotlin together with Tornado FX and Java FX (5). We can also use Kotlin as a scripting language, and even compile it to JavaScript, which lets us to run the Kotlin code in web browser (5). In general, Kotlin gives us many productivity improvements, and even though it is very new language it is already widely used by major companies like Lyft, Twitter, Expedia, American Express, Adobe Acrobat Reader, WordPress, NY Times, Amazon Kindle, Netflix and many others (Kotlin and Android).

Kotlin is a hybrid programming language, that combines functional programming, object oriented, imperative, declarative and procedural programming (Learn Kotlin). In functional programming the desired result is declared as the value of a series of function applications (Learn Kotlin). In object oriented it groups instructions together with the part of the state they operate on (Learn Kotlin). In imperative the programmer instructs the machine on how to change its state (Learn Kotlin). In declarative the programmer only declares properties of the desired result, but not how to compute them (Learn Kotlin). In procedural the instructions are in a sequence of steps that are executed in a specific order (Learn Kotlin). This also includes logic, in which the desired result is declared as the answer to a question about a system of facts and rules; also mathematical in which the desired result is declared as the solution of an optimization problem (Learn Kotlin).

Kotlin have some interesting features that help with its classification. It can be written like Python or JavaScript in the top level, but it can be also organized into function blocks (Learn Kotlin). In Kotlin instead of getters and setters, like in Java, it has simple properties (Learn Kotlin). For example, where in Java we would have “javaConstructor.getSomething,” in Kotlin we would simply have “kotlinConstructor.something” (Learn Kotlin). In Kotlin we can, but don’t have to specify types, it is also statically typed, which means that the type of every expression in the program is known at compile time, and the compiler can validate that the methods and fields we are trying to access exists for the objects we are using (Learn Kotlin). Kotlin also uses type inference and overload-resolution (Learn Kotlin). In type inference the ability of the compiler to determines the type from the context (Learn Kotlin). For example, a declaration “val x = 5,” is initialized with an integer value, and Kotlin automatically will determine its type as “Int” (Learn Kotlin). For overload-resolution the compiler can look up an argument of multiple functions with multiple types and determine which type to use, like Java (Jemerov 3).

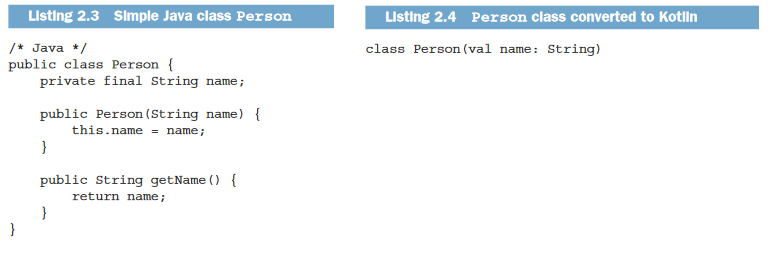
Syntax can have an important role to improve on the simplicity of development. Because Kotlin code is more concise when compared to Java, it is more precise and it is much shorter, that contributes to the development flow (Learn Kotlin). For loops Kotlin uses this same logic throughout, which helps transition between back-end and web development, without the need to learn extra libraries or separate syntax rules (Learn Kotlin). With all those differences, Kotlin also allows to implement the same basic rules as Java, which make it easy to transition from developing in Java (Learn Kotlin).

Kotlin has more advantages when compared to Java. It also addresses some issues that are still a problem in Java. In Kotlin null references are controlled by the type system, there are no raw types, arrays are invariant, it has proper function types, where java has SAM-conversions (Learn Kotlin). Kotlin uses variance without wildcards and doesn’t have checked exceptions (Learn Kotlin).

There are many other features that Kotlin has that other programming languages don’t have, especially when compared with the widely used Java. Kotlin has lambda expressions and inline functions that gives performant custom control structures, it has null-safety, smart casts, primary constructor and one or more secondary constructors, singletons and operator overloading, to name a few (Learn Kotlin).

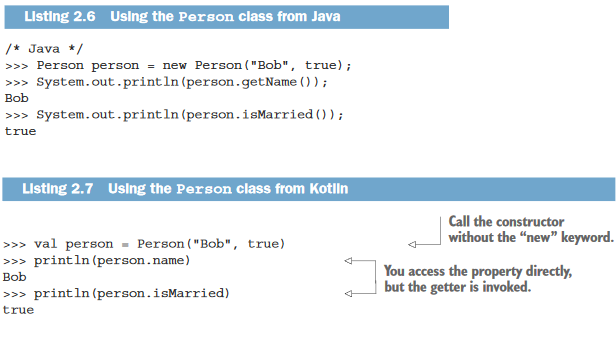
On top of general differences, Kotlin support functional programming from the very beginning (Learn Kotlin). It has function types, that handle other functions and allow to receive them as parameters or can return other functions (Learn Kotlin). Also, it allows us to easily implement lambda expressions, to help with passing blocks of code, and data classes help apply immutable value objects (Learn Kotlin). Overall, we can program in functional style, but Kotlin doesn’t force us to do that (Jemerov 7).

Kotlin addresses some key concepts for the language evaluation criteria, like simplicity, type checking and exception handling (Learn Kotlin). Overall simplicity greatly reduces the amount of code that is used in Java for getters and setters, as well as for writing classes themselves, which makes them easier to write, read in a reliable way (Jemerov 23).



(Figure 1. Class Person, (Jemerov 23)

For getters and setters, we reference the property directly making the code more concise and in this same time maintaining the logic (23).



(Figure 2) (Jemerov 23)

For type checking Kotlin can reliably check types before performing an operation and refers to its version of type-checking as “smart-cast” (Jemerov 23). Also, with exception handling it uses same Try/Catch for catching exceptions as Java and it is reliable (32).

In summary, there are many advantages that make Kotlin unique, such as it can be used much more efficiently and as a result the code is more precise and shorter. Interpretability allows Kotlin to be more easily implemented anywhere where Java can be implemented. In addition, the option for using Kotlin as a scripting language also gives us more flexibility. Also, the hybrid concepts allow Kotlin to be used for many different types of application, since it combines functional, object-oriented, imperative, decelerative and procedural features.

References:

“Andrey Breslav, Instructor.” *Coursera*, www.coursera.org/ instructor/ abreslav.

“Dmitry Jemerov.” *Dmitry Jemerov*, yole.ru/.

Jemerov, Dmitry, and Svetlana Isakova. *Kotlin in Action*. Manning Publications Co., 2017.

“Kotlin and Android.” *Android Developers*, developer.android.com/kotlin.

“Learn Kotlin.” *Kotlin*, kotlinlang.org/docs/reference/.

“Svetlana Isakova, Instructor.” *Coursera*, www.coursera.org/ instructor/ svetlana- isakova.