HASH:1143

Llenguatjes de Programació

Universitat Politecnica de Catalunya

Facultat d'Informàtica de Barcelona

`

Índice

Introducción

Kotlin es un lenguaje de programación de código abierto que se puede ejecutar en distintos dispositivos y sistemas operativos, es decir, multiplataforma. Fue diseñado para ser interoperable con Java, lo que permite a los desarrolladores aprovechar su código existente en Java y combinarlo con Kotlin. Desarrollado por JetBrains en 2011, y se lanzó públicamente en 2016, aunque sigue en constante evolución para mejorar y añadir características nuevas.

Es un lenguaje moderno y conciso que se enfoca en mejorar la legibilidad del código y reducir la cantidad de errores comunes que se pueden encontrar en otros lenguajes como Java.

Kotlin fue nombrado en honor a la isla de Kotlin, cerca de San Petersburgo, donde se encuentra la sede de JetBrains.

Aun que es un lenguaje multiplataforma diseñado para usarse en iOS, Android, macOS, Windows, Linux, watchOS, y otros, es especialmente usado para programar aplicaciones en Android.

Para entender Kotlin, el porqué de su creación y su utilidad, es importante entonces entender un poco de Android.

Android es el sistema operativo móvil más usado del mundo con una cuota del mercado del 90% (en 2018). Está basado en el núcleo de Linux, es decir, programado en C, pero su interfaz está completamente programada en Java. Por lo tanto, cualquier aplicación para Android tendrá que interactuar con lenguaje Java de una forma u otra, ya sea porque la aplicación está diseñada en Java directamente o en algún lenguaje interoperable. Y es aquí donde Kotlin toma importancia y de ahí que este fuese diseñado para ser interoperable con Java.

Desde 2017 Google (compañía propietaria de Android) anunció que daría soporte a Kotlin y desde entonces empezó a ganar popularidad entre los desarrolladores de Android, hasta tal punto, que hoy en dia el 80% del top 1000 aplicaciones de la Play Store contienen Kotlin de alguna forma.

Características del lenguaje

Paradigmas de Programación

Aunque Kotlin está enfocado principalmente al desarrollo de aplicaciones móviles y lo habitual en estas es un paradigma orientado a objetos, Kotlin es un lenguaje multi-paradigma, en específico admite:

Programación orientada a objetos, lo que significa que se basa en la idea de objetos que encapsulan datos y comportamientos. Kotlin admite todas las características de la programación orientada a objetos, como la herencia, el polimorfismo y la encapsulación.

Programación funcional, que se basa en el uso de funciones puras y la eliminación de efectos secundarios. Kotlin admite las funciones de orden superior, la composición de funciones y la inmutabilidad de datos. En este lenguaje, las funciones son ciudadanos de primera clase, lo que significa que se pueden asignar a variables, pasar como argumentos y devolver como resultados. Por ejemplo, el uso de la función map.

```kotlin
fun duplicar(numero: Int): Int {
    return numero * 2
}

fun main() {
    val numeros = listOf(1, 2, 3, 4, 5)

    val numerosDuplicados = numeros.map { duplicar(it) }

    println(numerosDuplicados) // Imprime: [2, 4, 6, 8, 10]
}
```

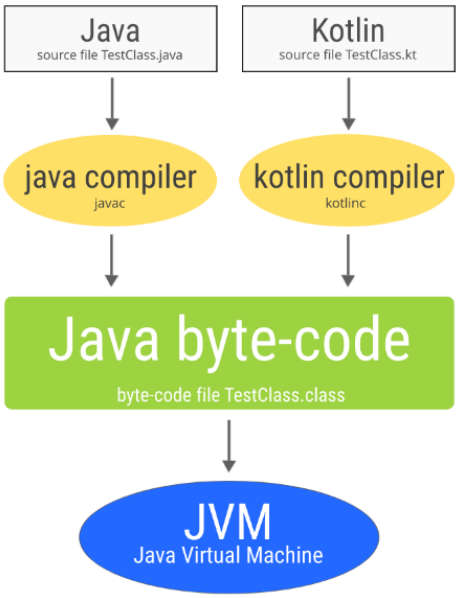
Programación reactiva, que se basa en el uso de flujos de datos asíncronos y eventos para manejar la concurrencia y la comunicación entre componentes. Este tipo de programación es especialmente útil para el desarrollo de aplicaciones móviles ya que estos dependen fuertemente de los eventos del usuario o repuestas de APIs.

Programación estructurada, que se basa en el uso de estructuras de control de flujo, como bucles y condicionales, para controlar el flujo de ejecución del programa.

En general, Kotlin es un lenguaje muy versátil que admite una amplia gama de paradigmas de programación y permite a los desarrolladores elegir el estilo de programación que mejor se adapte a sus necesidades.

Sistema de ejecución

Kotlin es un lenguaje de programación compilado e interpretado a la vez. Esto significa que el código fuente de alto nivel en Kotlin se compila a Bytecode de Java y este se ejecuta (es decir se interpreta) en una máquina virtual de Java (JVM). Por lo que es compatible con todas las bibliotecas y frameworks de Java existentes. Kotlin fue diseñado específicamente para mejorar y solucionar algunas de las limitaciones de Java, pero se integra perfectamente con el ecosistema de Java.



Aun así, Kotlin tiene soporte para compilar código nativo para plataformas específicas como Windows, macOS, Linux, iOS entre otras, lo que significa que Kotlin (desde su versión Native) también puede producir código maquina especifico a la plataforma de ejecución. Esto puede permitir una mayor eficiencia y velocidad ya que no saltamos la capa de interpretación, aunque perdemos la portabilidad del programa (entre otras ventajas del uso de interprete).

Por lo tanto, Kotlin tiene la ventaja de tener dos tipos de sistemas de ejecución en un mismo lenguaje y deja a elección del programador que tipo de ejecución usar según sus necesidades.

Sistema de tipos

El sistema de tipos de Kotlin es estático y fuertemente tipado. Estático significa que todos los tipos de datos deben ser declarados explícitamente y verificados en tiempo de compilación (como por ejemplo en C++). Además, una vez que se asigna un tipo a una variable, no se puede cambiar a un tipo diferente. Esto ayuda a prevenir errores de tipo en tiempo de ejecución y mejora la seguridad del código.

Al ser un lenguaje fuertemente tipado impone restricciones estrictas sobre la manipulación de los tipos de datos, y no permite que se realicen operaciones entre tipos de datos que no son compatibles. En un lenguaje fuertemente tipado, cada valor o variable tiene un tipo de datos definido y explícito, y las operaciones entre tipos incompatibles no están permitidas, esto se verifica como hemos dicho en tiempo de compilación.

Kotlin tiene una amplia gama de tipos de datos, incluyendo tipos primitivos como Int, Double y Boolean, así como tipos de objetos como String, List y Map. También admite la definición de clases y estructuras de datos personalizadas.

Una característica interesante del sistema de tipos de Kotlin es que admite la inferencia de tipos, lo que significa que el compilador puede inferir el tipo de una variable o expresión en función del contexto en el que se utiliza, es decir, cabe la posibilidad de declarar una variable sin tener que especificar su tipo, el compilador entonces le asignara un tipo en función del valor que se le asigne a la variable. Esto puede ayudar a reducir la cantidad de código necesario y mejorar la legibilidad. Este sería un ejemplo en Kotlin:

```
val edad = 25 // El compilador infiere que la variable "edad" es de tipo Int
```

Es importante entender la diferencia que sigue teniendo esto con un lenguaje dinámico como seria Python en el cual la inferencia del tipo de la variable se hace en tiempo de ejecución y no en compilación como seria en el caso de Kotlin. Por lo tanto, lenguaje dinámico no es lo mismo que lenguaje estático con inferencia de tipos.

Además, Kotlin tiene soporte para tipos nulos, lo que significa que una variable puede tener un valor nulo. Esto se controla mediante el uso del operador "?" en la declaración de tipos. Para evitar errores de NullPointerException en tiempo de ejecución, Kotlin obliga a los programadores a manejar explícitamente los valores nulos, utilizando operadores seguros como "?." y "?:", o forzando la verificación de nulabilidad a través del operador "!!".

Características particulares

Kotlin vs Java

Como hemos hablado antes, Kotlin está basado en Java y es interoperable con este pero es importante definir las diferencias entre ellos para entender el uso de Kotlin en diferentes ámbitos.

Uno de los puntos clave de Kotlin y que de distingue con Java es la sintaxis. Kotlin tiene una sintaxis más clara y concisa, lo que significa que es más fácil de leer y escribir. Kotlin también es más expresivo, lo que significa que puede hacer más con menos código.

ejemplokot.kt

```kotlin
fun main() {
    val list = listOf("apple", "banana", "orange")
    val result = list
        .filter { it.startsWith("a") }
        .map { it.toUpperCase() }
    println(result)
}

```

ejemplojav.java > ...

```java
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        List<String> list = Arrays.asList(...a:"apple", "banana", "orange");
        List<String> result = new ArrayList<>();
        for (String s : list) {
            if (s.startsWith(prefix:"a")) {
                result.add(s.toUpperCase());
            }
        }
        System.out.println(result);
    }
}

```

Estas lineas de codigo hacen exactamente lo mismo, filtran los elementos de una lista que comienzan con la letra "a" y los convierte a mayúsculas. Sin embargo, el código en Kotlin es más compacto y legible que el código en Java.

Aunque ya lo hemos comentado antes en Sistema de tiposSistema de tipos, otro punto importante es la nulabilidad. Kotlin ofrece soporte nativo para la nulabilidad, lo que significa que los desarrolladores pueden especificar si una variable puede o no ser nula. En Java, todas las variables pueden ser nulas, lo que puede llevar a errores en tiempo de ejecución.

Ligado también al Sistema de tipos la inmutabilidad de las variables en Kotlin se definen por defecto como inmutables (val), lo que significa que no se pueden cambiar una vez que se han inicializado. En Java, todas las variables son mutables por defecto (a menos que se especifique lo contrario con la palabra clave final).

Kotlin permite a los desarrolladores agregar funciones de extensión, es decir, añadir funciones a clases existentes sin modificarlas directamente. En otras palabras, una función de extensión es una función que puede ser llamada como miembro de una clase, pero está definida por fuera de ella. Java no tiene esta característica.

```kotlin
fun String.addHello(): String {
    return "Hello $this"
}

val greeting = "John".addHello()
println(greeting) // Imprime "Hello John"

```

En este ejemplo, addHello() es una función de extensión para la clase String. La función toma el valor actual de this (que en este caso es el objeto de tipo String al que se le aplica la función de extensión) y devuelve una nueva cadena que agrega el texto "Hello" al principio. Una vez creada esta función podemos usarla en variables del tipo String como es el caso de la variable greeting que devolverá “Hello Jhon”.

Por último, otro cambio importante al programar en Kotlin respecto a Java es el manejo de excepciones: Kotlin elimina la necesidad de especificar las excepciones que pueden ser lanzadas por un método, lo que hace que el código sea más limpio y menos propenso a errores.

(Añadir diferencia en gestión de excepciones)

Principales aplicaciones

Como hemos dicho Kotlin es un lenguaje de programación que se ejecuta en la JVM. Kotlin nace con la idea de que las empresas que usan Java puedan migrar progresivamente a un lenguaje mas moderno, compacto y fácil de aprender sin perder su desarrollo en Java.

Es por esto que Kotlin puede usarse en casi cualquier tipo de entornos, ya que Java es de los lenguajes más versátiles que existen.

Además del uso en Android que ya hemos ido nombrando, las otras principales aplicaciones son muchas.

Kotlin se utiliza en el desarrollo de aplicaciones web sobre todo en el lado del servidor, a través de frameworks como Ktor, Spring, o Spark.

Kotlin también se utiliza para el desarrollo de aplicaciones de escritorio multiplataforma. Por ejemplo, JetBrains desarrolló la herramienta de desarrollo de aplicaciones IntelliJ IDEA utilizando Kotlin.

Se utiliza en el desarrollo de juegos para Android y otras plataformas. También es compatible con las bibliotecas de juegos multiplataforma, como LibGDX.

Kotlin se utiliza en el desarrollo de sistemas embebidos y dispositivos IoT. Los sistemas embebidos son un tipo de sistemas informáticos diseñados para realizar tareas específicas que suelen estar diseñados para funcionar en dispositivos más grandes, como lavadoras, coches, drones, etc… Esto es posible gracias a Kotlin Native.

Por lo tanto, como hemos comentado antes, Kotlin ya sea en su formato de ejecución por JVM o de forma Native puede usarse para una gran variedad de plataformas.

Analisis personal del llenguatje

Tras toda la investigación hay varios puntos que me han gustado especialmente de Kotlin.

Lo primero es que Kotlin es un lenguaje muy intuitivo y fácil de aprender, especialmente teniendo experiencia previa en Java. Además, al ser un lenguaje apoyado directamente por Google cuenta con una documentación muy completa y una amplia comunidad.

Como ya hemos hablado, es un lenguaje muy conciso, lo que no solo significa que se puede escribir menos código para lograr lo mismo que con otros lenguajes, sino que, además, leer y entender código ajeno es mucho más fácil. Esto último me parece realmente interesante ya que considero que a veces puede ser realmente difícil entender lo que hace un código ajeno. Un dato que me pareció que ejemplifica muy bien esto es que el equipo de Google Home, es que la migración a Kotlin consigo una reducción del 33% del tamaño del código.

También me parece realmente interesante el hecho de que Kotlin sea interoperable con Java. Me parece una gran decisión por parte de JetBrains el no intentar sustituir por completo un lenguaje tan utilizado como es Java, ya que es realmente difícil, pero si hacer un nuevo lenguaje compatible con los sistemas ya implementado con Java para que la migración pueda ser progresiva y sin problemas de compatibilidad.

Aun asi, para mi hay dos desventajas o inconvenientes de Kotlin. Primero de todo, aunque es un lenguaje como he dicho compacto, quizás tiene una sintaxis un poco

compleja para alguien que nunca ha programado. Esto no es malo como tal, pero si lo sumamos a que para usar Kotlin realmente deberías saber Java también la curva de aprendizaje para alguien que por ejemplo quiere desarrollar una aplicación en Android y que nunca ha programado antes no es de las mejores en todos los lenguajes de programación. Si que es importante recalcar que para alguien con cierta experiencia en otros lenguajes sí que Kotlin es fácil de entender y aprender ya que es muy intuitivo si por ejemplo ya sabes Java.

Otro inconveniente que le veo a Kotlin es que, aun siendo un lenguaje oficial para el desarrollo de aplicaciones en Android, es un lenguaje muy nuevo y esto dificulta su aprendizaje en el sentido que hay mucho menos material (cursos, ejemplos, problemas resueltos…) que por ejemplo en Java, que ya lleva muchos más años en uso, por lo que hay mucha más documentación. Aun así, la comunidad de Kotlin ha crecido tan rápido y se le ha dado tanto soporte que si espero que llega un gran nivel de contenido de calidad muy rápidamente.

Uso solvente de los recursos de información

Wikipedia

https://es.wikipedia.org/wiki/Kotlin_(lenguaje_de_programaci%C3%B3n)#Filosof%C3%ADa

La página de Wikipedia es buena para hacerse una primera idea del lenguaje, su historia, características principales, su filosofía, etc… Aun así, es una página puramente informativa sin ningún contenido educacional como tal más allá de un pequeño código de ejemplo.

Android Developers

https://developer.android.com/kotlin/learn?hl=es-419

La página oficial de Android Developers es la más completa que he podido encontrar sobre Kotlin. Desde que Google adopto Kotlin en 2017 como lenguaje oficial han hecho un gran trabajo en facilitar la migración y el uso de este. Esta página es un gran ejemplo de esto. Con explicaciones sobre el porque de uso, se da un contexto de lo grande que es la comunidad de Kotlin con ejemplos de grandes aplicaciones desarrolladas en este lenguaje. De ahí se enseña Kotlin desde 0 con ejemplo de código desde el primer momento para irse familiarizando. Veo realmente útil como se plantea las principales características del lenguaje (explicadas anteriormente) con ejemplos en código.

También hay un apartado sobre la guía de estilo de Koltin que define los estándares de codificación (de Android) creados por Google. Es la guía más útil que he encontrado sobre los estándares en Kotlin ya que como en toda la página, se ejemplifica todo con código.

Cuenta también con un apartado para desarrolladores de Java que quieran empezar a usar Kotlin. Desde como integrarlo a tu aplicación de Java ya existente a como aprenderlo (con un enfoque diferente) si ya eres programador de Java.

Kotlin Lang

https://kotlinlang.org/docs/home.html

Aunque la web oficial de Kotlin también tiene una guia sobre como empezar en Kotlin al igual que la pagina de Android Developers me parece mejor estructurada y ejemplificada la pagina de Android Developers. Aun asi, como hemos comentado, aunque el uso que se le da principalmente a Kotlin es en Android, es un lenguaje multiplataforma y obviamente en la página de Android se le da un enfoque a Android. Para entender Kotlin en su plenitud como lenguaje de programación la página oficial es la mejor. Tiene una gran cantidad de información muy bien organizada. En esta se habla de la parte de Kotlin

de la que hay menos información: Kotlin Native, Kotlin para el lado del servidor y para data science y Kotlin Multiplatform como tal.

Esta página es esencial para entender la cantidad de entornos donde se puede trabajar con Kotlin ya que como he comentado el enfoque principal que se le da es puramente en Android.

Hyperskill

https://hyperskill.org/tracks?category=4&_ga=2.267927816.784266275.1683288380-1106570686.1683288380

Hyperskill es una plataforma de aprendizaje sobre diferentes aspectos de la informática (sobretodo lenguajes de programación). Tienen 4 cursos de Kotlin con casi 400 horas en total. Estos cursos han sido creados por JetBrains por lo cual son los mejores cursos para aprender Kotlin de la forma mas correcta posible.

Head First Kotlin

Dawn Griffiths, David Griffiths (Febrero,2019). *Head First Kotlin, A Brain-Friendly Guide.* O'Reilly Media

Como he comentado antes, haciendo este trabajo pienso que Kotlin puede ser difícil de aprender para alguien que no ha programado nunca antes. Así que buscando la mejor documentación para aprender encontré este libro que va más allá de toda la información sobre sintaxis y otras guías que he visto en internet, sino que, se fija en hacerte pensar como un buen programador de Kotlin, además es la guía con más énfasis al apartado funcional que he encontrado sobre Kotlin, cosa que es una pena ya que muchos programadores omiten este tipo de programación y esta es realmente útil.