Kotlin y Java

Historia de Java:

Java nace en 1991, creado por Ariel Gosling junto al equipo de Sun Microsystems.
Nace con el nombre "OAK", pero posteriormente cambiado por Green por problemas legales, y finalmente con la denominación actual JAVA.
El objetivo de java era crear un lenguaje de programación parecido a C++ en estructura y sintaxis, con una máquina virtual propia, y fuertemente orientado al paradigma de objetos, (fuertemente implicando que hay otros paradigmas contemplados, no es el único pero es el principal)

Java fue muy importante ya que muchos lenguajes de programación y aplicaciones estaban diseñados para funcionar en plataformas específicas, como sistemas operativos particulares o tipos de hardware. Esto significaba que un programa escrito para una plataforma no siempre funcionaba en otras sin modificaciones significativas.

En 1996 es lanzado el primer JDK (Paquete de desarrollo en Java), y hasta el día de hoy se siguen creando paquetes y librerías para Java.

Historia de Kotlin:

Kotlin es un lenguaje de programación de código abierto creado por JetBrains. JetBrains es una compañía que se dedica a crear entornos de desarrollo (IDEs). El más famoso es IntelliJ IDEA, el entorno en el que está basado Android Studio.

En 2011 se da a conocer que llevaban un año trabajando en un lenguaje que suplante las carencias que estaban encontrando con Java.
Necesitaban un lenguaje que pudiera usarse con toda la base de código que ya tenían (Java), pero que a su vez les diera toda la potencia de un lenguaje moderno.
El único lenguaje que encontraron con estas características era Scala, pero se habían encontrado con muchos problemas con los tiempos de compilación: Scala era muy lento.
Kotlin ofrece una sintaxis más concisa en comparación con Java. Esto reduce la cantidad de código necesario para realizar las mismas tareas, lo que a menudo resulta en un código más limpio y fácil de leer.

En 2012 deciden liberar el código fuente para convertirlo en código abierto y que cualquier pudiera participar en su creación.

En 2017, recibe un importante impulso al ser nombrado por Google como lenguaje oficial para Android al mismo nivel que Java.

Usos de Java:

Videojuegos: Aunque no tan popular, Java, en combinación con motores de juegos y frameworks, debido a su facilidad de uso y portabilidad es usado para el desarrollo de juegos independientes. Como ejemplos, Minecraft, Runescape, y Project Zomboid.

Programación en la nube: La programación en la nube se refiere al desarrollo y despliegue de aplicaciones y servicios utilizando recursos y servicios proporcionados a través de plataformas en la nube, en lugar de depender de infraestructura física local, como servidores y hardware.
Oracle proporciona servicios y herramientas para el desarrollo, despliegue y gestión de aplicaciones en la nube utilizando Java.

Inteligencia artificial: Java no es el lenguaje más común para la inteligencia artificial, pero se utiliza en aplicaciones de IA mediante el uso de librerías y frameworks

Aplicaciones Web/Aplicaciones de escritorio/Servicios RESTful: Spring es un framework muy popular para el desarrollo de aplicaciones web en Java. Proporciona un conjunto de herramientas para crear aplicaciones empresariales y web robustas y seguras.

Aplicaciones moviles: Java es ampliamente utilizado y compatible en el ecosistema Android.

Usos de Kotlin:

Aplicaciones móviles: Kotlin ha sido ampliamente adoptado y recomendado por Google para el desarrollo de Android, integrándose con Android Studio al igual que Java. Muy eficiente con su sintaxis concisa, manejo seguro de nulos, soporte para corutinas y interoperabilidad con Java.

Backend: Kotlin puede ser utilizado para el desarrollo de aplicaciones web backend mediante frameworks como Ktor, que es un framework web creado por JetBrains específicamente para Kotlin.

Frontend: Kotlin es también totalmente compatible con Spring, puede ser usado también para desarrollar todo la parte visible al usuario de una aplicación.

Desarrollo multiplataforma: Kotlin ofrece soporte para el desarrollo multiplataforma, lo que permite compartir código entre diferentes plataformas, como Android, iOS, y aplicaciones de backend. Esto facilita la creación de aplicaciones con una base de código compartida.

Características de Java:

Estructurado: Es un paradigma de programación basado en la idea de que el código debe dividirse en estructuras claras y lógicas, módulos y funciones, y usar el control de flujo para hacer el código más legible y mantenible.

Imperativo: La programación imperativa es un paradigma que se basa en dar instrucciones paso a paso sobre cómo se debe realizar una tarea. Se enfoca en cómo se deben ejecutar las operaciones para lograr un resultado, utilizando un flujo de control explícito.

Orientado a objetos: Fuertemente orientado a objetos, Java incorpora sus conceptos utilizando objetos, o sea, campos de data con comportamiento y características, asimilando un poco con objetos de la vida real. Entre otras características del paradigma, podemos mencionar el polimorfismo, la herencia, la composición, etc.

Fuertemente tipado: Realiza una verificación estricta de los tipos de datos y no permite conversiones implícitas o errores relacionados con tipos sin intervención explícita del programador.

Compilado e interpretado: Los lenguajes compilados se traducen completamente a código máquina antes de la ejecución, mientras que los lenguajes interpretados se traducen a código máquina en tiempo de ejecución, línea por línea.

Más adelante se explicará cómo es este proceso para Java y Kotlin.

Características de Kotlin:

Tipado estático: El tipado estático significa que el tipo de las variables se conoce en tiempo de compilación, en lugar de en tiempo de ejecución. Los errores de tipo se

detectan durante la compilación, lo que permite mayor seguridad y corrección de errores antes de que el programa se ejecute.

Null safety: La seguridad de nulidad (null safety) se refiere a la capacidad de un lenguaje de programación para manejar valores nulos de manera segura, evitando errores de puntero nulo, que son una causa común de fallos en el tiempo de ejecución, como por ejemplo en Java.

Interoperabilidad: La capacidad de un lenguaje de programación para trabajar con otros lenguajes o sistemas sin problemas, utilizando sus componentes y bibliotecas.
Kotlin está diseñado para ser completamente interoperable con Java. Esto significa que puedes usar bibliotecas Java en proyectos Kotlin y viceversa.

Funcional: Kotlin admite características de la programación funcional, es decir, un paradigma que trata los cálculos como la evaluación de funciones matemáticas y evita cambiar el estado y los datos mutables.
Kotlin cuenta con funciones de orden superior, lambdas, y colecciones inmutables.

BNFs:

La notación de Backus-Naur, es un metalenguaje usado para expresar gramáticas libres de contexto: es decir, una manera formal de describir lenguajes formales.

Algoritmos de ordenamiento:

Bubble sort: Funciona revisando cada elemento de la lista que va a ser ordenada con el siguiente, intercambiándolos de posición si están en el orden equivocado
Es simple, pero no muy eficiente en listas grandes debido a su alta complejidad.

Comienza comparando los dos primeros elementos de la lista, si el primer elemento es mayor que el segundo, los intercambiamos, si no, se quedan como están. Luego pasamos al siguiente par de elementos, los comparamos e intercambiamos si fuera necesario.

Quick sort: Muy eficiente por su alta velocidad de ordenamiento. Funciona eligiendo un elemento del array llamado **“Pivote”**, y los elementos de la lista se reubican a cada lado del pivote, los menores quedando a un lado y los mayores del otro, y los elementos iguales se colocan según la implementación elegida.

La lista queda dividida en dos sublistas, y este proceso se repite de forma recursiva para cada sublista mientras están tengan más de un elemento, hasta que finalmente toda la lista queda ordenada.

Conclusiones:

En ambos algoritmos, Java y Kotlin tuvieron performances similares.
Indagando un poco más sobre el proceso de compilación de ambos:

Java y Kotlin se compilan a bytecode. Bytecode es un formato intermedio de código que es independiente de plataforma, es decir, no es código máquina específico para una arquitectura de software, sino que se ejecuta por una máquina virtual.

El código fuente de ambos lenguajes se transforma en formato bytecode para ser ejecutado en cualquier sistema que tenga una **Java Virtual Machine (JVM)**

Finalmente un **Compilador JIT (Just-In-Time)**, que es parte de la **JVM,** mejora el rendimiento del programa, optimizando el bytecode. El Compilador JIT compila el código antes de la ejecución.
La compilación JIT, encargada de traducir el bytecode a código máquina nativo en tiempo de ejecución, reduce la sobrecarga de interpretar el bytecode, mejorando la velocidad de ejecución general.
Permite optimizaciones basadas en el comportamiento real del programa durante su ejecución, tales como eliminar código redundante y ajustar el código para aprovechar las características del hardware.