

Ingeniería en sistemas de información.

Sintaxis y semántica de los lenguajes.

TRABAJO PRÁCTICO TEÓRICO.

ÁREA TEMÁTICA: Backus-Naur Form

GRUPO N° 22

| Apellido y Nombre | N° de Legajo |
| --- | --- |
| Gaetan, María Luz | 163627-3 |
| Laino, Ramiro Angel | 175883-4 |
| Lamothe, Genty Clarke |  |
| Mendiolar Colombo, Nahuel Nehuen | 169188-0 |
| Rolando, Sebastian |  |

CURSO: K2055

DOCENTE A CARGO: Roxana Leituz

FECHA DE VENCIMIENTO: 24/08/2023

FECHA DE PRESENTACIÓN: 24/08/2023

FECHA DE DEVOLUCIÓN: \_\_/\_\_/\_\_\_\_

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ FIRMA DOCENTE: \_\_\_\_\_\_\_\_\_\_\_\_\_

**Índice**

[Java 2](#_heading=h.j53s52piyg83)

[Kotlin 2](#_heading=h.xt7mymcjfx8f)

[Benchmark 3](#_heading=h.jdhdt02hde8j)

[BFN Java 4](#_heading=h.116sdk8zx6g1)

[BFN Kotlin 4](#_heading=h.ckaxjqbash0e)

[Conclusión 4](#_heading=h.y3ftcomp8pea)

## 

## 

## Java

Es un lenguaje de programación creado por Sun Microsystems a finales de la década de los 80, el cual inicia con un proyecto de investigación liderado por James Gosling con el propósito de desarrollar un software para dispositivos electrónicos.

Java nace en 1991 con el nombre "OAK", posteriormente cambiado por Green por problemas legales, y finalmente por JAVA. El objetivo de java era crear un lenguaje de programación parecido a C++ en estructura y sintaxis, fuertemente orientado a objetos, pero con una máquina virtual propia. Esto se hizo bajo el principio de poder ser usado bajo cualquier arquitectura denominada "Write Once, Run Anywhere”.

En 1992 se presenta el proyecto Green, con los prototipos a bajo nivel. Entre 1993 y 1994 se trabaja para poder presentar un prototipo funcional (hotJava).

En 1995, es presentada la versión alpha de java, y un año después en 1996 es lanzado el primer JDK (JDK 1.0).

## Kotlin

Es un lenguaje de programación de código abierto creado por JetBrains, quien lanzó su primera versión en 2011. Este lenguaje es de tipado estático, ya que se puede desarrollar sobre JVM o JavaScript e incluso sin necesidad de ninguna de ellas ya que en paralelo también se está desarrollando en nativo con LLMV.

El objetivo de su creación fue abordar las limitaciones a las que se enfrentan los programadores con Java en el desarrollo de aplicaciones Android, como la sobrecarga de código. Pero a su vez se buscó su interoperabilidad para que puedan convivir ambos lenguajes dentro de un mismo proyecto y así conseguir una adopción gradual.

En 2016 se lanzó la versión 1.0. A partir de este momento se consideró estable y adecuado para el desarrollo de aplicaciones en producción. Sin embargo su popularidad comenzó en 2017 cuando Google anunció que le daría soporte como lenguaje de programación oficial para el desarrollo de aplicaciones Android.

## 

## 

## Benchmark

Comparativa de un algoritmo que ordena una colección de 1000 elemento:

| JAVA | KOTLIN |
| --- | --- |
| import java.util.ArrayList;  import java.util.Collections;  import java.util.List;  public class Main {  public static void main(String[] args) {  // Crear una lista de ejemplo con 1000 elementos no ordenados  List<Integer> lista = new ArrayList<>();  for (int i = 0; i < 1000; i++) {  lista.add((int) (Math.random() \* 1000)); // Agregar números aleatorios  }  // Ordenar la lista  Collections.sort(lista);  // Imprimir la lista ordenada  System.out.println("Lista ordenada:");  for (Integer elemento : lista) {  System.out.print(elemento + " ");  }  }  } | fun main() {  // Crear una lista de ejemplo con 1000 elementos no ordenados  val lista = mutableListOf<Int>()  repeat(1000) {  lista.add((0..999).random()) // Agregar números aleatorios  }  // Ordenar la lista  lista.sort()  // Imprimir la lista ordenada  println("Lista ordenada:")  lista.forEach { elemento ->  print("$elemento ")  }  } |
| Ejecución : 0.669 seconds | Ejecución : 5.059 seconds |

Comparativa de un algoritmo que calcula dígitos de Pi:

| JAVA | Kotlin |
| --- | --- |
| piDigits.java | piDigits.kt |
| Ejecución: 0.665 seconds | Ejecución: 5.559 seconds |

|  | JAVA | KOTLIN |
| --- | --- | --- |
| Uso de JVM | Si | Si |
| bytecode | archivo.class | archivo.class |
| Orientado | Objetos | Funcional y objetos |
| Seguridad | No acepta valores nulos | No pointer exception |
| Comando  precompilación | java Jasmin miarchivo.j | - |
| Comando compilación | java -jar jasmin.jar miarchivo.j | kotlinc hello.kt -include-runtime -d hello.jar |

## BFN Java

[JAVA BNF](https://docs.google.com/document/d/1nKd_rTXEjtZndgY0-VpbQbWbUNy5RZy3KKPKDDW3IBI/edit?usp=drive_link)

## BFN Kotlin

[Kotlin BFN.pdf](https://drive.google.com/file/d/1FYCe24PbgeOa6OFnTGomOATEwBaI-H-S/view?usp=drive_link)

## Conclusión

Java resultó ser más rápido a la hora de su ejecución y composición del desarrollo. Sin embargo en el contexto del desarrollo de aplicaciones Android, Kotlin suele ser recomendado ya que surgió para facilitar la escritura, siendo más conciso y seguro en términos de manejo de nulos. Al ser diseñado bajo este objetivo, puede consumir los recursos, librerías y hasta la máquina virtual de Java. Por lo que ambos lenguajes pueden convivir en un mismo proyecto.

Sin embargo, para otros tipos de desarrollo, la elección entre Java y Kotlin puede depender de varios factores, como las necesidades específicas del proyecto y las preferencias de los desarrolladores.