**Trabajo práctico N° 1**

***SSL Lenguajes***

| **Nombre y Apellido** | **Legajo** | **Lenguaje** |
| --- | --- | --- |
| Stefano Alejandro Gassmann | 208.380-2 | Java |
| Santiago Rodriguez | 209.171-9 | Python |

***Índice:***

* ***Java***
  + *Historia.*
  + *BNF.*
* ***Python***
  + *Historia.*
  + *BNF.*
* ***Benchmark***
  + ***Búsqueda binaria***
    - *Código Java.*
    - *Código Python.*
    - *Comparación de rendimiento.*
  + ***Fibonacci recursivo***
    - *Código Java.*
    - *Código Python.*
    - *Comparación de rendimiento.*

**Java**

***História***

**E**n 1994 una empresa llamada **Sun Microsystems** (hoy parte de *Oracle*) comienza el desarrollo de un lenguaje pensado para utilizarse en pequeños dispositivos electrónicos. El lenguaje, se denominó inicialmente *Oak* (por un roble que había fuera de la oficina de Gosling, uno de los creadores de este lenguaje), luego pasó a llamarse *red* tras descubrir que *Oak* era ya una marca comercial registrada y finalmente se le renombró ***Java***. Existen numerosas teorías sobre el origen de este nombre, la más aceptada. indica que ***Java*** proviene de una cafetería cercana a las oficinas de los creadores de este lenguaje.

**L**a idea de ***Java*** es que sea capaz de crear programas que posteriormente pudieran ser **ejecutados en múltiples entornos virtuales** sin la necesidad realizar cambios en el código.

**D**esde 1994 hasta la actualidad, ***Java*** ha sufrido una gran cantidad de cambios, comenzando con su versión inicial JDK (Java Development Kit) 1.0 anunciada oficialmente en 1996, hasta la última versión estable, ***Java 20*** lanzada en abril del 2023. Se estima que más de 3.000 millones de dispositivos ejecutan alguna versión de Java y hay más de 7 millones de profesionales que poseen conocimientos en dicho lenguaje, convirtiendo a ***Java*** en uno de los 3 lenguajes más utilizados a nivel mundial, según el índice [Tiobe](https://www.tiobe.com/tiobe-index/).

***Java*** ha atravesado numerosas versiones con el paso del tiempo, pero es importante destacar algunas de ellas.

**Java 1.1 (1997):**

* + Biblioteca de AWT (Abstract Windows Toolkit) mejorada con nuevas clases y funcionalidades.
  + Introducción de clases internas anidadas.
  + Mejoras en el manejo de excepciones.

**Java 1.2 (también conocido como Java 2 o JDK 1.2, 1998):**

* + Introducción de la máquina virtual Java HotSpot para mejorar el rendimiento.
  + Introducción de Swing para la creación de interfaces gráficas de usuario más avanzadas.
  + Colecciones: nuevas clases como List, Set y Map, que reemplazaron las antiguas estructuras de datos.

**Java 1.4 (2002):**

* + Introducción de la máquina virtual Java HotSpot Server.
  + Se incorporan las expresiones regulares.
  + Introducción de las clases ByteBuffer y IO para operaciones de E/S no bloqueantes.

**Java 7 (2011):**

* + Introducción de la notación de tipo de diamante (diamond operator) para inferir tipos genéricos.
  + Mejoras en la biblioteca de concurrencia.
  + Introducción de try-with-resources para manejar recursos automáticamente en bloques try-catch.
  + Introducción de las expresiones lambda y el paquete java.util.function.

**Java 8 (2014):**

* + Expresiones lambda para programación funcional.
  + Interfaz de fecha y hora (java.time) mejorada.
  + Métodos por defecto y estáticos en interfaces.

**Java 11 (2018):**

* + A partir de esta versión, el JDK ya no es gratuito y se debe pagar una licencia mensual por usuario, en su contraparte surge Open JDK, una versión libre de la plataforma de desarrollo Java.
  + Comienza un ciclo de **actualizaciones** **cada 6 meses**.

Para cerrar, ***Java*** es un lenguaje de programación increíblemente potente y muy utilizado en el desarrollo de aplicaciones de escritorio, aplicaciones web, aplicaciones para teléfonos, videojuegos, aplicaciones para dispositivos **embebidos** y dispositivos electrónicos, pero sobre todo, destaca su gran participación en aplicaciones del entorno empresarial.

**BNF de Java:**

El BNF de Java cumple con la normativa **ISO-14977**, este estándar, asegura que Java posea un BNF muy expresivo y fácil de utilizar.

**N**ota: adjuntamos un enlace para su visualización [aquí](https://cs.au.dk/~amoeller/RegAut/JavaBNF.html).  
https://cs.au.dk/~amoeller/RegAut/JavaBNF.html

**Python**

***História***

La historia de Python comienza en 1989, siendo desarrollado por el informático holandés Guido Van Rossum, basándose en la iniciativa "ABC", llevada a cabo en el Centrum Wiskunde & Informatica (CWI), espacio en el que se desempeñaba en aquel momento. El objetivo de Van Rossum era crear un lenguaje fácil de escribir y por ende, de aprender.

El lenguaje comienza a utilizarse en 1989, hasta 1991 donde es lanzada la primera versión pública y accesible para todos: la 0.9.0. La misma ya incluía clases con herencias, funciones, manejo de excepciones y una de las características principales del lenguaje, **funcionamiento modular**. Lo cual lo hacía un lenguaje más 'limpio' y accesible para personas con poco conocimiento de programación.

El historial de lanzamientos de Python, luego de la antes mencionada 0.9.0, continua con la **versión 1.0** lanzada en 1994 con nuevas funciones para procesar fácilmente una lista de datos, la asignación, filtrado y reducción. Seis años más tarde, en el 2000, el equipo empieza a formar parte de BeOpen Python Labs.

Esta primera versión triunfó entre el público. Tanto es así, que se creó un foro en el que se conversaba sobre la herramienta. Un hecho que contribuyó a seguir popularizando Python entre los profesionales del sector.

Se continúa con **Python 2.0** el 16 de octubre del año 2000, incorporando características que resultaban útiles para los programadores como la compatibilidad con los caracteres Unicode y una forma más eficiente de recorrer una lista.

Además, se incluyen las referencias cíclicas, el colector de basura y nuevos sistemas de cadenas. En el 2001, aparece la Python Software Foundation, entidad dirigida por Guido Van Rossum y que posee la propiedad de todas las especificaciones del lenguaje.

Finalmente, el 3 de diciembre de 2008 es lanzado Python 3.0, la cual tuvo un coste bastante elevado, dado a los importantes cambios introducidos en el lenguaje. Unas modificaciones que hicieron necesario reescribir los programas. Esta última versión permitió solventar los errores más comunes e incluir la función de impresión y más soporte para la división de números.

En el año 2018, el creador de Python se despide y el desarrollo del lenguaje sigue bajo la dirección de un consejo conformado por 5 desarrolladores. Un equipo directivo que se renueva cada año mediante elecciones.

***Benchmark***

Entre los lenguajes **J**ava y **P**ython, decidimos confeccionar un algoritmo de **búsqueda binaria** aplicado a un Array, nuestro objetivo es comparar el tiempo que tarda cada lenguaje en encontrar el elemento que necesita la mayor cantidad de iteraciones para ver cual de ellos es más eficiente.

*Código Java :* [*Java*](https://github.com/stefanoGassmann/SSL/blob/main/01/Java/busquedaBinaria.java)

[*https://github.com/stefanoGassmann/SSL/blob/main/01/Java/busquedaBinaria.java*](https://github.com/stefanoGassmann/SSL/blob/main/01/Java/busquedaBinaria.java)

*Código Python:* [*Python*](https://github.com/stefanoGassmann/SSL/blob/main/01/Python/BusquedaBinaria.py)[*https://github.com/stefanoGassmann/SSL/blob/main/01/Python/BusquedaBinaria.py*](https://github.com/stefanoGassmann/SSL/blob/main/01/Python/BusquedaBinaria.py)

*Comparación de rendimiento:*

| *Cant elem* | *Java* | *Python* |
| --- | --- | --- |
| *1* | *13571 Nanoseg* | *9059.90 Nanoseg* |
| *10* | *15488 Nanoseg* | *11682.51 Nanoseg* |
| *100* | *18774 Nanoseg* | *57458.87 Nanoseg* |
| *1k* | *26253 Nanoseg* | *454187.39 Nanoseg* |
| *10k* | *382739 Nanoseg*  *0.000382 Seg* | *65132856.36 Nanoseg*  *0.065 Seg* |
| *100k* | *2431181 Nanoseg*  *0.00243 Seg* | *180355787.27 Nanoseg*  *0.18 Seg* |
| *1M* | *85546787 Nanoseg*  *0.0855 Seg* | *593830823.89 Nanoseg*  *0.59 Seg* |
| *10M* | *399142354 Nanoseg*  *0.399 Seg* | *9901055812.83 Nanoseg*  *9.901 Seg* |
| *100M* | *Se ejecuta correctamente* | *El contenedor se cae* |

*Logramos deducir que,* ***J****ava administra mejor los recursos disponibles (Memoria CPU) cuando la cantidad de datos en mayor a 100.*

Entre los lenguajes **J**ava y **P**ython, decidimos confeccionar el algoritmo de **Fibonacci recursivo**, nuestro objetivo es comparar el tiempo que tarda cada conjunto de iteraciones en completar una cantidad de elementos de la serie.

*Código Java :* [*Java*](https://github.com/stefanoGassmann/SSL/blob/main/01/Java/Fibonacci.java)

[*https://github.com/stefanoGassmann/SSL/blob/main/01/Java/Fibonacci.java*](https://github.com/stefanoGassmann/SSL/blob/main/01/Java/Fibonacci.java)

*Código Python:* [*Python*](https://github.com/stefanoGassmann/SSL/blob/main/01/Python/BusquedaBinaria.py)[*https://github.com/stefanoGassmann/SSL/blob/main/01/Python/BusquedaBinaria.py*](https://github.com/stefanoGassmann/SSL/blob/main/01/Python/BusquedaBinaria.py)

*Comparación de rendimiento:*

| *Cant elem* | *Java* | *Python* |
| --- | --- | --- |
| *1* | *2093517 Nanoseg* | *19073.48 Nanoseg* |
| *2* | *579629 Nanoseg* | *33378.60 Nanoseg* |
| *4* | *962046 Nanoseg* | *93460.08 Nanoseg* |
| *8* | *563629 Nanoseg* | *112295.15 Nanoseg* |
| *16* | *9414003 Nanoseg*  *0.00941 Seg* | *1818656.92 Nanoseg*  *0.0018 Seg* |
| *32* | *315397323 Nanoseg*  *0.315 Seg* | *11625167846.67 Nanoseg*  *11.62 Seg* |
| *35* | *618232404 Nanoseg*  *0.618232404 Seg* | *37324287176.13Nanoseg*  *37.32 Seg* |
| *64* | *Sigue ejecutando.* | *El contenedor no puede procesar el cálculo y se cae.* |

*Observamos que nuevamente, Java tiende a tener un mejor desempeño en el tiempo, mientras que Python parece tener dificultades a la hora de enfrentar la serie 33, creemos que se debe a las llamadas al sistema operativo.*