* ***Python***

Python tiene ya funciones propias del lenguaje para la creación, lectura y escritura de archivos. Ahora para acceder a archivos de texto, ya que python soporta archivos binarios y de texto, tenemos varios métodos para hacerlo. Esto hace referencia a cómo queremos utilizar el archivo cuando se deba abrir.

Existen 6 formas para hacer esto:

1. Read Only: (“r”): Esto abre el archivo para solo lectura. Este es el modo por default el cual el archivo se abre. El registro es posicionado en el principio del archivo.
2. Read and Write: (“r+”): Esto abre el archivo para lectura y escritura. El registro es posicionado en el principio del archivo.
3. Write Only (“w”): Esto abre el archivo para sólo escritura. Si el archivo ya existe, cabe recalcar que la información es reescrita.El registro es posicionado en el principio del archivo.
4. Write and Read (“w+”): Esto abre el archivo en forma de lectura y escritura, para un archivo que ya existe la información es reescrita. El registro es posicionado en el principio del archivo.
5. Append Only(“a”): Esto abre el archivo para escritura. El archivo es creado si es que no existe todavía. En este caso el registro se ubica al final del archivo. La información escrita es añadida al final, después de lo que ya se encuentra en el archivo.
6. Append and Read (“a”): Esto abre el archivo para lectura y escritura. El archivo es creado si es que no existe todavía. En este caso el registro se ubica al final del archivo. La información escrita es añadida al final, después de lo que ya se encuentra en el archivo.

**Apertura de un archivo**

En Python esto se realiza utilizando la función: open(), para esto no se necesitan módulos importados.

Ejemplo:

***Archivo = open(r”File\_name”, “Acces\_Mode”)***

El archivo debe existir en el mismo directorio que el archivo de python.

**Cerrar un archivo**

En python esto se realiza utilizando la función close(), esto cierra el archivo y libera espacio el espacio en memoria que este utilizaba. Se utiliza cuando ya no se necesita el archivo o cuando se quiere utilizar el archivo en otro modo.

**Escribiendo en el archivo**

Tenemos 2 maneras de escribir en el archivo:

1. Utilizando la función write(): Añade el String en una sola línea de texto en el archivo.
2. Utilizando la función writelines(): Esto se utiliza cuando tienes una lista de Strings, cada una de las Strings es insertada en el archivo al mismo tiempo.

**Lectura en el archivo**

Tenemos 3 formas de leer información de un archivo.

1. read(): Retorna los bytes leídos en forma de un String, si no se le específica se lee absolutamente todo el archivo.
2. readline(): Lee una línea del archivo y lo retorna en forma de String, en este caso no lee más de una línea.
3. readlines(): Lee todas las líneas y retorna cada una de ellas como un elemento String de una lista.

* ***JavaScript***

Técnicamente en JavaScript no se pueden leer o escribir archivos, sin embargo podemos utilizar el módulo fs de Node.Js, el cual nos permite la lectura y escritura de archivos en el servidor pero javaScript está hecho para navegadores por lo que eso entonces especifica que el usuario final no puede acceder a esos archivos.

Este módulo tiene métodos para leer y escribir archivos de forma asincrónica o sincrónica.

**Lectura del archivo**

El archivo es leído utilizando fs.readFile(), el cual es un método propio del módulo. Este método lee todo el archivo en memoria y lo guarda en un buffer.

Ejemplo:

***fs.readFile(nombreArchivo, codificación, callback\_function )***

Parámetros:

* nombreArchivo: puede ser el nombre del archivo a leer, o la ruta si el archivo está guardado en otra parte.
* Codificación: Guarda la codificación del archivo.
* callback function: Esto es una función la cual debe ser invocada luego de leer el archivo la cual requiere 2 entradas:

1. error: Si hay un error
2. data: El contenido del archivo

**Escritura del archivo**

fs.writeFile() se utiliza para escribir la información de un archivo de manera asincrónica, y si el archivo ya existe este será reemplazado.

Ejemplo:

***fs.writeFile( file\_name, data, options, callback )***

Parámetros:

* Nombre del archivo: Es un String, Buffer o URL e incluso la ruta del archivo
* Información: La información que será enviada al archivo es un String, Buffer, Vector.
* Opciones: Es un String o objeto que se utiliza para darle un output opcional.
* callback: esta función se invoca cuando se ejecuta el código.
* ***java***

Java nos ofrece algunas API (Interfaz de programación de aplicaciones) tales como Java E/S para la lectura y escritura de archivos en versiones anteriores y la mejorada, simplificada y actualizada Java I/O que admite nuevas funciones. En java hay dos tipos de transferencias para interactuar con archivos: Flujos de caracteres y secuencias de bytes. Por ahora, solo trabajamos en flujo de personajes.

**Lectura del archivo**

Para los flujos de caracteres, se utilizan en la lectura varias clases como:

* Reader: Es la clase abstracta para leer el flujo de caracteres. Contiene subclases como:
  + FileInputStreamReader: Clase que convierte el flujo de bytes en flujo de caracteres.
  + FileReader: Clase para leer caracteres de un archivo.
  + BufferedReader: Clase que a diferencia del fileReader nos permite tener una lectura más rápida y completa de una línea del archivo.

Ejemplo:

***FileReader object = new FileReader(archivo)***

***BufferedReader object = new BufferedReader (new FileReader(archivo))***

**Escritura del archivo**

La escritura de archivos de flujo de caracteres se maneja con las siguientes clases:

* Writer: Es la clase abstracta para escribir los flujos de caracteres. Contiene subclases como:
  + OutputStreamWriter: Clase que escribe secuencias de caracteres y a su vez en bytes.
  + FileWriter: Clase para escribir caracteres en un archivo.
  + BufferedWriter: Clase que a diferencia del FileWriter nos permite almacenar datos suficientes hasta hacer la escritura eficiente en el archivo.

Ejemplo:

***FileWriter object = new FileWriter(archivo)***

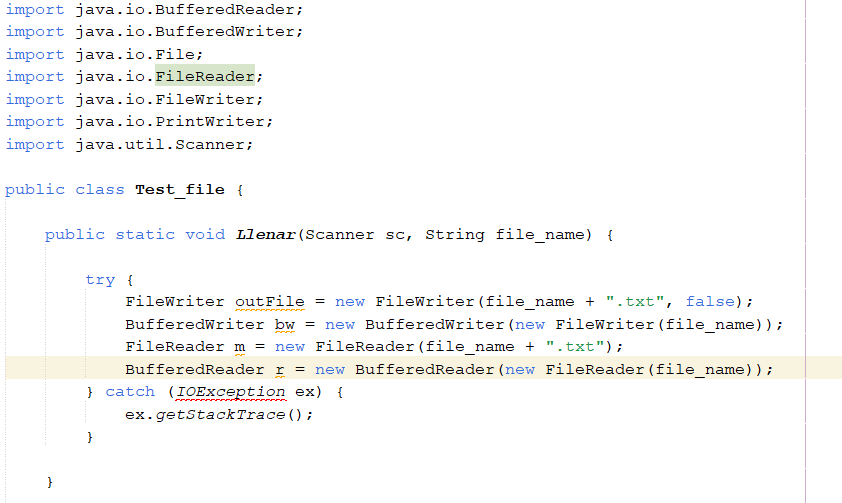
***BufferedWriter object = new BufferedWriter(archivo)***

1. Si bien la mayoría de los lenguajes consultados tienen similitudes a la hora de la escritura y lectura de archivos, algunos suelen ser más flexibles e intuitivos. Por otra parte, probablemente para muchas personas los que suelen ser “difíciles” a ellos se les hace más fácil de comprender ya que llevan más tiempo aprendiendo el lenguaje en el cual hacen la lectura y escritura de archivos por lo que a una persona que lleva trabajando con java por lo menos 5 años le va a parecer más fácil los métodos que utiliza java para la lectura y escritura de archivos. No obstante, para el gusto de nosotros, nos parece más fácil los métodos que utiliza Python en el manejo de archivos.

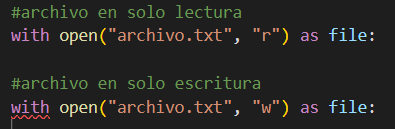
Por ejemplo:

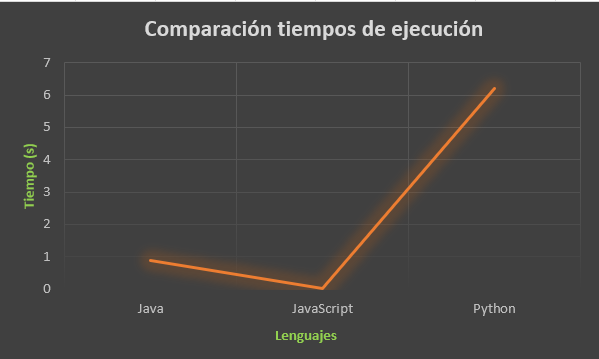
Mientras que en java para la escritura y lectura de archivos hay que instanciar objetos de clases previamente importadas, en Python solamente debes poner w si lo quieres abrir en modo escritura o r en lectura.

Java:



Python:





Las métricas utilizadas para la comparación de los programas fue su tiempo de ejecución. cómo podemos ver en la gráfica, java al hacer la prueba de búsqueda de campos por una palabra clave demoró exactamente 0,879 segundos, mientras que Python tuvo un tiempo de ejecución más elevado, ambos experimentos fueron corridos en el mismo computador. Cabe recordar que estos tiempos pueden variar de acuerdo al equipo, pero es de suma importancia saber que netbeans es un editor de texto para java que compila todo el código antes de correrlo, esto quiere decir que sus tiempos de ejecución serán mucho más rápidos que los de Python ya que este es un intérprete, es decir, el a medida que se va ejecutando va comprobando línea por línea que todo funcione correctamente. Además, es importante mencionar que para JavaScript solo estoy contando la lectura de un archivo .txt de 10000 datos, por eso en la comparación realizada parece el ganador.

En cuanto a líneas de código utilizadas sin contar los espacios innecesarios en java utilizamos 43 líneas de código, mientras que en Python solo se utilizaron 22 líneas de código, casi la mitad de las que utilizamos en java, mientras que en JavaScript para solo la lectura del archivo .txt de 10000 datos se utilizaron 12 líneas de código.

Para finalizar en cuanto a complejidad, como lo mencionamos anteriormente Python es el menos complejo y más intuitivo de los 3, Java se quedaría con el 2 puesto, mientras que JavaScript pensamos que sería el más fácil, pero resultó ser el más complejo (para nosotros) no solo por ser un lenguaje que no tenemos experiencia, sino también por su complejo manejo.

Referencias

1. "Reading and Writing to text files in Python". GeeksForGeeks. <https://www.geeksforgeeks.org/reading-writing-text-files-python/> (accedido el 31 de agosto de 2022).
2. "Read and Write of a file using JavaScript". GeeksForGeeks. <https://www.geeksforgeeks.org/explain-about-read-and-write-of-a-file-using-javascript/> (accedido el 30 de agosto de 2022).
3. "Leer y escribir archivos en Java". Pharos. <https://pharos.sh/leer-y-escribir-archivos-en-java/> (accedido el 1 de septiembre de 2022).