Augusto Alonso Ascencio - 181085
Pablo Alejandro Méndez Morales – 19195
Ángel David Cuellar Bautista - 18382

**Proyecto interprete Lisp**
**Java Collection FrameWork**

Las estructuras del Java Collection FrameWork implementadas fueron:

1. **ArrayList:**
   Esta estructura tuvo como usos principales en nuestro programa guardar los comandos o valores que son propios del lenguaje Lisp. Como por ejemplo un arraylist con los operadores matemáticos.

   También es un arraylist donde guardamos todos los ¨Nodos¨ creados que son leídos y que van a ser después utilizados para leer el código del programa. Se decidió utilizar esta estructura debido que de esta manera vamos recorriendo el arraylist y ejecutando cada nodo del arraylist. Además, es más dinámico en cuanto a la hora de cambiar de tamaño y puede haber una cantidad indeterminada de stacks dentro de un stack. Por ejemplo, en un condicional podés tener una condición o varias condiciones.

   Se utilizo de referencia la siguiente documentación:
   https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/util/ArrayList.html

2. **LinkedList:**
   Este tipo de estructura es el más importante para este programa debido que fue con este que se decidieron trabajar todos los Strings que va leyendo el programa, se van añadiendo o quitando elementos, cambiando de posición entre otras operaciones. Es por eso que decidimos utilizar esta estructura, porque es la más sencilla para acceder a todos los elementos de manera sencilla.

   Un ejemplo es la función remove if donde le pasamos una lamda que cuando se cumple el valor booleano necesario se elimina de nuestra lista.

   Las otras veces donde se utiliza, es también cuando necesitamos el comportamiento de un stack para agregar de último o eliminar de último.

   Se utilizo de referencia la siguiente documentación:
   https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/util/LinkedList.html

3. **HashMap:**

   Esta estructura jugo un papel bastante importante a la hora de crear la recursividad en la clase MainMemory porque es en esta clase tenemos varios HashMap donde se guardan los parámetros, variables y funciones que se mandan a llamar a la hora de ejecutar una función con recursividad. Se decidió usar esto porque es la manera de acceder más rápido a algún elemento.

Augusto Alonso Ascencio - 181085
Pablo Alejandro Méndez Morales – 19195
Ángel David Cuellar Bautista - 18382

Son Hashmaps porque tenemos una key donde se distingue en que casillero debe ir, entonces tenemos el String que representa el nombre de la variable y tenemos su valor dentro de un stack.

En estas variables tenemos variables a nivel global y también guardamos las variables a nivel de función como los parámetros que eso solo existe dentro de una función.

Otro HashMap de funciones tenemos un String donde tenemos el nombre de la función y un stack que representa a la función.

Como implementación curiosa tenemos un HashMap que separa el nombre de la función y un arraylist con el nombre de los parámetros.

Se utilizo de referencia la siguiente documentación:

https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/util/HashMap.html