UNIVERSIDAD DEL VALLE DE GUATEMALA

FACULTAD DE INGENIERÍA

Proyecto: Fase 1

Algoritmos y Estructuras de datos

Diego Fernando Patzán Marroquín / 23525

Osman Emanuel de Leon Garcia / 23428

Ihan Gilberto Alexander Marrroquín Sequén / 23108

Sección 10

LISP

Lisp, que significa "List Processing" en inglés, es uno de los lenguajes de programación más antiguos y emblemáticos. Fue desarrollado por John McCarthy en 1958 en el Instituto Tecnológico de Massachusetts (MIT) como un lenguaje destinado a la inteligencia artificial y la investigación en computación. A lo largo de los años, ha experimentado varias revisiones y versiones, siendo Common Lisp y Scheme las más populares. Desde su concepción en 1955, Lisp fue el primer lenguaje de inteligencia artificial y ha introducido una serie de nuevos conceptos que son de uso común en la actualidad. Además, es el tercer lenguaje de programación de alto nivel más antiguo que aún se utiliza comúnmente, después de Fortran y COBOL. (*Reclu IT*, 2023)

Características de LISP

- *Basado en listas:* Lisp trata los programas y datos como listas estructuradas. Esto permite una manipulación flexible de los programas durante la ejecución.
- *Dinámico y expresivo:* Lisp es conocido por su sintaxis simple pero poderosa que permite una expresión concisa de algoritmos complejos.
- *Programación funcional:* Lisp es un lenguaje de programación funcional, lo que significa que se enfoca en la evaluación de funciones matemáticas y evita el cambio de estado y los efectos secundarios.
- *Interpretado y compilado:* Lisp puede ser interpretado o compilado, lo que brinda flexibilidad en el desarrollo y ejecución de programas.
- *Extensible:* Lisp es altamente extensible debido a su naturaleza simbólica y su capacidad para manipular programas como datos.
  (*LISP - EcuRed*, 2024)

Empleo de LISP

- *Inteligencia Artificial:* Es ampliamente utilizado en la investigación y desarrollo de sistemas de inteligencia artificial y aprendizaje automático.
- *Procesamiento de lenguaje natural:* Lisp ha sido empleado en el desarrollo de sistemas de procesamiento de lenguaje natural debido a su capacidad para manipular estructuras de datos complejas.
- *Simulación y modelado:* Lisp se ha utilizado en aplicaciones de simulación y modelado en campos como la biología, la física y la economía.
- *Desarrollo de software:* Aunque menos común en comparación con otros lenguajes, Lisp todavía se utiliza en el desarrollo de software, especialmente en entornos donde se requiere flexibilidad y manipulación de datos en tiempo de ejecución.

Lisp sigue principalmente el paradigma de programación funcional, mientras que la Programación Orientada a Objetos (POO) se centra en el uso de objetos y clases. En Lisp, los datos y las funciones se manipulan como listas y pueden ser tratados de manera uniforme, mientras que en la POO, los datos y las funciones están encapsulados en objetos con

interfaces específicas. Además, Lisp es altamente flexible y dinámico, lo que permite una manipulación extensa de programas durante la ejecución, mientras que la POO enfatiza la estructura y la encapsulación de datos y comportamientos dentro de objetos definidos previamente. Por último, Lisp se centra en las funciones y la evaluación de expresiones, mientras que la POO se centra en los objetos y sus interacciones (*Preguntas Frecuentes Sobre Lisp - Paul Graham En Español*, 2024).

Java Collection Framework

El Java Collections Framework es una parte esencial de Java que ofrece una estructura unificada para manejar y manipular conjuntos de objetos. Está basado en interfaces y clases concretas que proporcionan una amplia variedad de estructuras de datos y algoritmos para almacenar y manipular conjuntos de elementos. En este marco, las interfaces principales son Collection, List, Set y Map, cada una con sus propias funcionalidades para trabajar con colecciones de objetos de manera eficiente (*Collections Framework Overview*, 2024).

- *Collection:* Es la interfaz raíz de la jerarquía y define métodos comunes para trabajar con colecciones de objetos, como agregar, eliminar, verificar la presencia de elementos, entre otros.
- *List:* Extiende la interfaz Collection y representa una colección ordenada (o secuencia) que permite duplicar elementos duplicados. Proporciona métodos para acceder, agregar, eliminar y modificar elementos por su posición.
- *Set:* También extiende Collection pero representa una colección que no permite elementos duplicados. Proporciona métodos para agregar, eliminar y verificar la presencia de elementos únicos.
- *Map:* Esta interfaz representa una colección de pares clave-valor, donde cada clave está asociada con un único valor. No extiende la interfaz Collection, pero es una parte integral del marco de colecciones de Java.

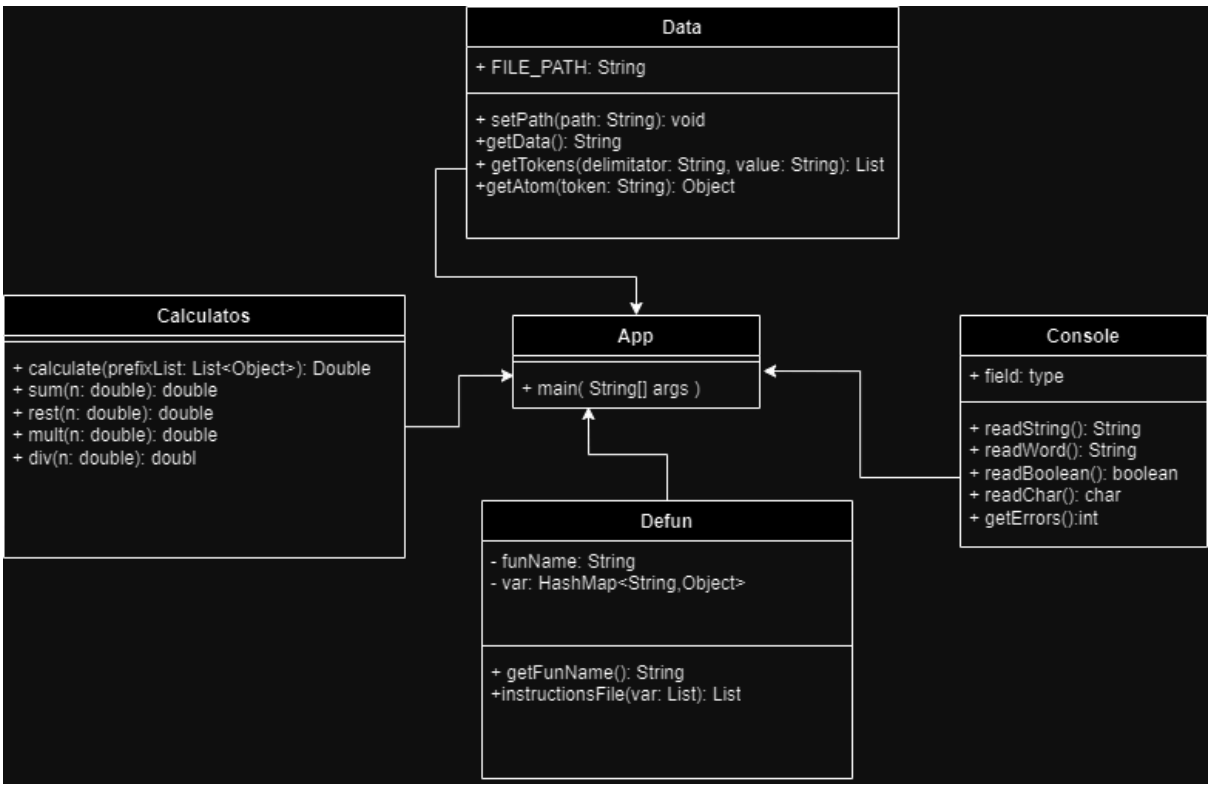
Implementaciones Principales

- *ArrayList:* Implementa la interfaz List y utiliza un arreglo dinámico para almacenar elementos. Proporciona acceso rápido a los elementos por índice, pero es menos eficiente en términos de inserciones y eliminaciones en el medio de la lista.
- *LinkedList:* También implementa la interfaz List y utiliza una estructura de datos de lista doblemente enlazada para almacenar elementos. Ofrece un mejor rendimiento en inserciones y eliminaciones en comparación con ArrayList, pero un acceso menos eficiente por índice.
- *HashSet:* Implementa la interfaz Set y utiliza una tabla hash para almacenar elementos. Garantiza que no haya elementos duplicados y proporciona un rendimiento constante O(1) para operaciones básicas como agregar, eliminar y buscar elementos.
- *TreeSet:* Es una implementación de la interfaz SortedSet que utiliza un árbol binario

de búsqueda para almacenar elementos ordenados. Proporciona un conjunto ordenado de elementos y garantiza un rendimiento logarítmico(log n) para operaciones básicas.

- *HashMap:* Implementa la interfaz Map y utiliza una tabla hash para almacenar pares clave-valor. Proporciona un rendimiento constante O(1) para operaciones básicas como agregar, eliminar y buscar elementos, siempre y cuando la función de dispersión esté bien distribuida.
- *TreeMap:* Es una implementación de la interfaz SortedMap que utiliza un árbol rojo-negro para almacenar pares clave-valor ordenados por clave. Proporciona un mapa ordenado y garantiza un rendimiento logarítmico O(log n) para operaciones básicas.

UML inicial (sujeto a cambios):



Archivos del código del programa experto (repositorio de GitHub):
https://github.com/dpatzan2/Ejemplo-uso-Lisp.git

AMBIENTE DE TRABAJO

REFERENCIAS

Reclu IT. (2021). Recluit.com. https://recluit.com/que-es-lisp/

LISP - EcuRed. (2024). Ecured.cu.

https://www.ecured.cu/LISP#Caracter.C3.ADsticas_de_LISP

Preguntas Frecuentes sobre Lisp - Paul Graham en Español. (2024). Paulgraham.es. https://paulgraham.es/lisp/preguntas-frecuentes-sobre-lisp.html

Collections Framework Overview. (2024). Oracle.com. https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html