UNIVERSIDAD DEL VALLE DE GUATEMALA

Departamento de ingeniería

Proyecto

Algoritmos y estructuras de datos

Instructor: Michaelle Pérez

Gerson Gonzales / 231541

Jose Sánchez/ 231221

Pablo Andrés Cabrera Argüello / 231156

Sección 20

Nueva Guatemala de la Asunción

Lunes, 5 de febrero del año 2024

¿Qué es Lisp?

Lisp, cuyo nombre proviene de "List Processing", es un venerable lenguaje de programación diseñado con el propósito principal de gestionar de manera eficiente las listas de datos. Su origen se remonta a la década de los 50, cuando John McCarthy, un destacado investigador del MIT, se encontraba inmerso en el desarrollo de la inteligencia artificial. McCarthy necesitaba un lenguaje que le permitiera manipular y procesar listas de datos, ya que quería que la inteligencia artificial pudiera utilizar listas para almacenar información y así emular el razonamiento humano.

En ese momento, McCarthy se enfrentó al desafío de la falta de un lenguaje que cumpliera con estas especificaciones, lo que lo llevó a desarrollar Lisp. Este lenguaje revolucionario permitía expresar datos en términos de objetos, que podían ser funciones o diversas operaciones. La capacidad de realizar cálculos con expresiones simbólicas en lugar de simplemente numéricas hizo que Lisp se destacara, especialmente en el campo emergente de la inteligencia artificial.

A lo largo de los años, Lisp ha perdurado como uno de los lenguajes más antiguos aún en uso, y su legado se extiende más allá de su función original en la inteligencia artificial. Su versatilidad en el manejo de listas y la manipulación de datos simbólicos ha encontrado aplicaciones en diversos campos de la informática.

Es fundamental destacar que el enfoque de Lisp en el procesamiento de listas y su capacidad para trabajar con datos simbólicos han sido elementos clave en el desarrollo y evolución de la inteligencia artificial. La historia de Lisp se convierte así en un capítulo fundamental en la narrativa del desarrollo de lenguajes de programación y su impacto en la innovación tecnológica.

¿Qué es Java Collections Framework?

El Java Collections Framework (JCF) es una parte esencial de la plataforma Java, diseñada para facilitar la manipulación de colecciones de objetos. Esta estructura ofrece una serie de interfaces y clases que permiten representar y operar sobre conjuntos de datos de manera eficiente (Caules, 2022).

En la base de esta jerarquía se encuentra la interfaz Collection, que actúa como la raíz de todas las colecciones. A partir de ella se derivan otras interfaces, cada una con un propósito específico. Por ejemplo, la interfaz List representa una secuencia ordenada de elementos donde se puede acceder por su posición. En este grupo se encuentran implementaciones populares como ArrayList, LinkedList y Vector (Caules, 2022).

Otra interfaz importante es Set, que representa una colección de elementos únicos, sin duplicados. Implementaciones comunes incluyen HashSet y TreeSet (Caules, 2022).

Además, existe la interfaz Map, que no extiende de Collection, sino que representa un mapeo de claves a valores únicos. Aquí encontramos implementaciones como HashMap y TreeMap (Caules, 2022).

Cada una de estas interfaces tiene implementaciones específicas que se adaptan a diferentes necesidades. Por ejemplo, ArrayList utiliza un arreglo dinámico para almacenar elementos, mientras que LinkedList utiliza una lista doblemente enlazada (Caules, 2022).

Para el desarrollo de un intérprete LISP, se necesitará hacer uso de estructuras de datos proporcionadas por el Java Collections Framework (JCF) para gestionar eficazmente las expresiones LISP y los entornos de variables durante la evaluación. En este caso, se requerirán principalmente las siguientes interfaces e implementaciones.

List e implementaciones (ArrayList, LinkedList): Estas estructuras son cruciales para almacenar y manipular las expresiones LISP, ya que estas se representan comúnmente como listas enlazadas o arreglos dinámicos (Caules, 2022).

Map e implementaciones (HashMap, TreeMap): Serán útiles para crear un entorno de variables donde se puedan almacenar los valores asociados a las variables definidas en el programa (Caules, 2022).

Set e implementaciones (HashSet, TreeSet): Aunque no son tan esenciales como List y Map, podrían emplearse para ciertas operaciones que requieran verificar la unicidad de los elementos (Caules, 2022).

En el contexto del programa del proyecto, estas estructuras de datos podrían emplearse de la siguiente manera.

ArrayList o LinkedList: Podrían usarse para representar las expresiones LISP y los argumentos de funciones de manera conveniente, permitiendo una manipulación sencilla de los elementos (Caules, 2022).

HashMap o TreeMap: Serían útiles para almacenar las variables y sus valores asociados, lo que sería fundamental para las operaciones de definición de funciones y asignación de variables (por ejemplo, con la instrucción SETQ) (Caules, 2022).

Diagramas UML

Diagrama de conversión de Fahrenheit a centímetros.

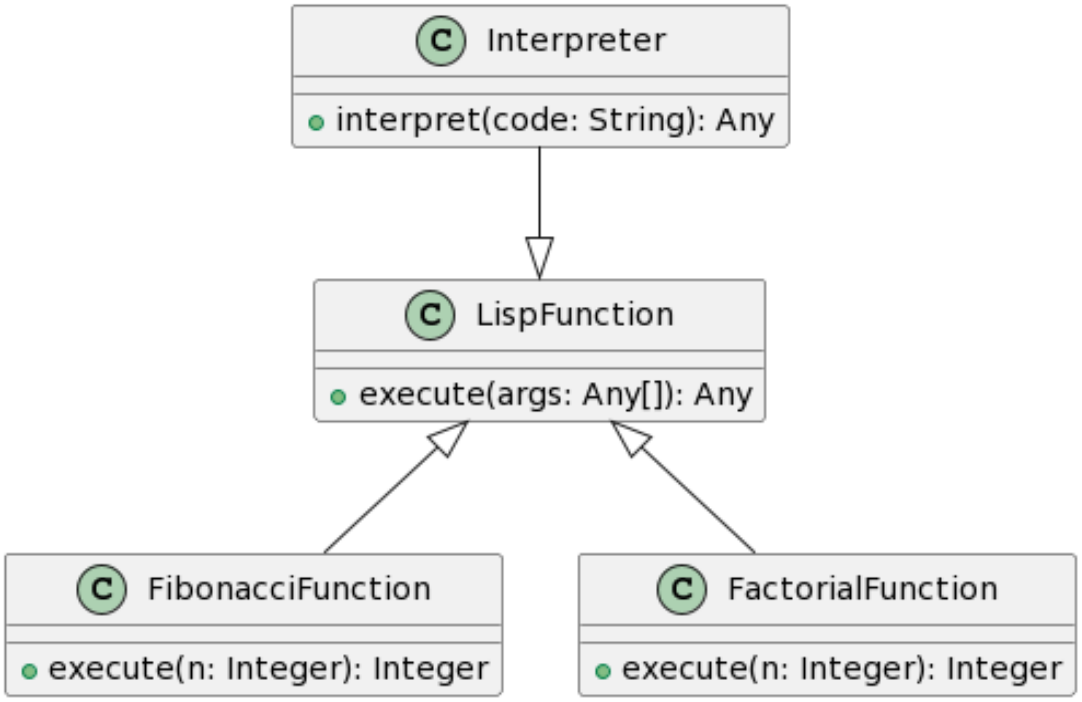
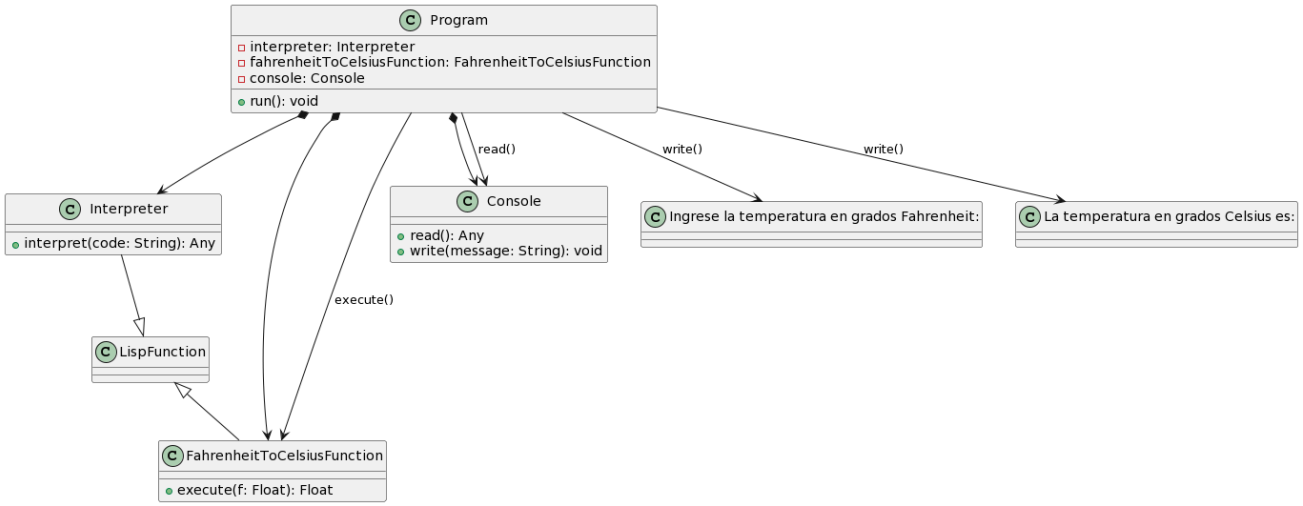


Diagrama de serie de Fibonacci y Factorial de un número



Link de Github: https://github.com/pablouwunya2021/Fase-1-proyecto-

Referencias

Caules, C. Á. (2022, enero 22). *Java Collections Framework y su estructura*. Arquitectura Java. https://www.arquitecturajava.com/java-collections-framework-y-su-estructura/

Reclu IT. (s/f). Recluit.com. Recuperado el 5 de febrero de 2024, de https://recluit.com/que-es-lisp/