Universidad del Valle de Guatemala

Facultad de Ingeniería

Departamento de Ciencias de la Computación

CC2003 - Algoritmos y Estructuras de datos

Proyecto No.1

Fase 1

Jonathan Espinoza – 20022
Juan Galicia – 20298
Elisa Samayoa – 20710

Lisp

Historia

Lisp es un lenguaje de programación que fue diseñado para permitir una manipulación más eficiente de las cadenas de datos. Tiene un mayor soporte en el paradigma de la programación y utiliza una notación matemática que se basa en el cálculo lambda cuya estructura fundamental está basado en las listas enlazadas. Fue inventado a finales de la década de los cincuentas por John McCarthy, un estudiante del Instituto de Tecnología de Massachusetts (MIT), intentando resolver la problemática de que, en ese entonces, no había ningún lenguaje que trabajara de una manera adecuada con las listas. Lisp fue construido a partir de operadores simples y con un sistema de notación para funciones, su nombre es en realidad una abreviatura de "procesamiento de listas". Hasta la fecha, las "s-expression" utilizadas en este lenguaje para describir datos, han influido en otros lenguajes de programación, además de las declaraciones condicionales como if-then-else que fueron creadas por el mismo McCarthy.

Características

- Tiene un buen manejo de memoria automatizado que permite liberar espacio usado por los objetos que ya no se necesitan.
- Usan mecanismos simples para usar evaluaciones de expresiones.
- No tiene un sistema con tipos estáticos, sino que asimila los tipos a los valores en vez que a las variables. Este es un motivo por el cual los errores de mal uso de tipos se detectan en tiempo de ejecución y no de compilación.
- Las implementaciones se programan de manera que los cálculos iterativos se pueden realizar en espacio constante aunque se hayan descrito en la recursividad.

¿Dónde se emplea?

- Su capacidad para calcular las expresiones simbólicas en lugar de los números lo hace muy conveniente para sus usos en el mundo de la inteligencia artificial (IA).
- En el MIT se ha vuelto bastante popular en los proyectos de investigación puesto que su base matemática ayuda a resolver operaciones y ser usado en probar teoremas.

Programación funcional

Idealmente, Lisp se considera como un lenguaje de programación funcional, sin embargo, a medida que han pasado los años, se les han añadido extensiones de programación orientada a objetos.

La programación funcional, como bien su nombre nos puede dar una idea, se centra especialmente en las funciones, donde todos los elementos se pueden entender como funciones y se ejecutan mediante llamadas de función secuenciales. Sus raíces se encuentran alrededor de la década de los treinta, sin embargo, hoy en día, aún tiene una gran popularidad debido a sus usos matemáticos y técnicos, además de ofrecer grandes campos de abstracción.

La principal diferencia entre este tipo de programación con la que está orientada a objetos, es que, en esta se ve todo como una transformación de los datos, es decir que los datos no se almacenan en los objetos,sino que crean nuevas versiones de sí mismos y se

manipulan usando una de las muchas funciones; mientras que en la POO, los datos se almacenan en atributos de objetos y es por medio de funciones donde se pueden manipular.

Java Collections Framework

Una collection es un objeto que representa un grupo de objetos, mientras que un collection framework es una arquitectura unificada para representar y manipular collections independientemente de los detalles de implementación.

Jerarquía de interfaces e implementaciones

Una jerarquía entre interfaces es la que permite llevar a cabo la herencia de tipo simple y múltiple, por lo que, la declaración de una clase y su interfaz pueden incluir la implementación de otras interfaces.

La jerarquía de interfaces permite tres tipos de relaciones, las cuales son:

1) Superclass-Subclass: la subclase se extiende a la superclase y hereda todos los comportamientos de la superclase.
2) Superinterface-Subinterface: la subinterface se extiende a la superinterface y hereda todos los comportamientos de la superinterface.
3) Interface-class: la clase implementa la interfaz y hereda todos los comportamientos de la interfaz.

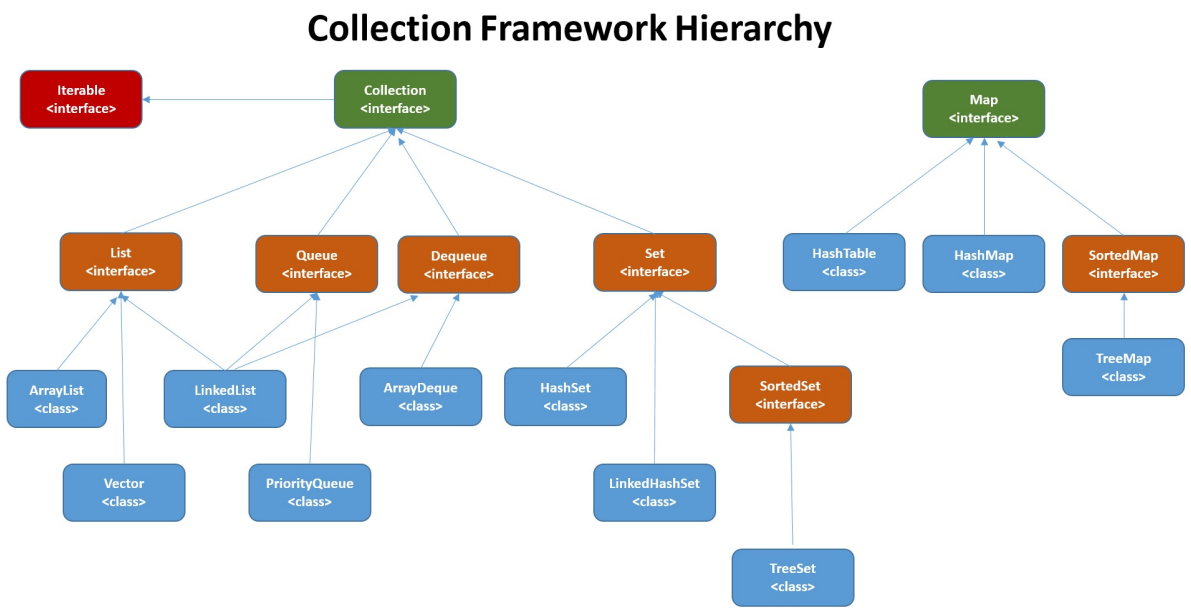
Las implementaciones se utilizan para indicar que una clase pueda especificar o utilizar una o varias interfaces (separadas por coma) mediante el uso de la palabra reservada *implements*. Al implementar la interfaz, es necesario redefinir los métodos con acceso público. Asimismo, todos los métodos de dicha interfaz tendrán el mismo tratamiento que cualquier otro método, sin ninguna otra diferencia.

Las implementaciones más conocidas son:

- HashMap
- HashSet
- TreeSet
- Arraylist
- Vector
- Arraydeque

Como complemento a lo anteriormente mencionado, a continuación se adjunta un esquema para mayor visualización de la jerarquía utilizada en los Java Collection FrameWork.

Esquema #1. Jerarquía de Collection Framework



Gupta, J. 2019. Java : Collection Framework Hierarchy. Facing Issues On IT

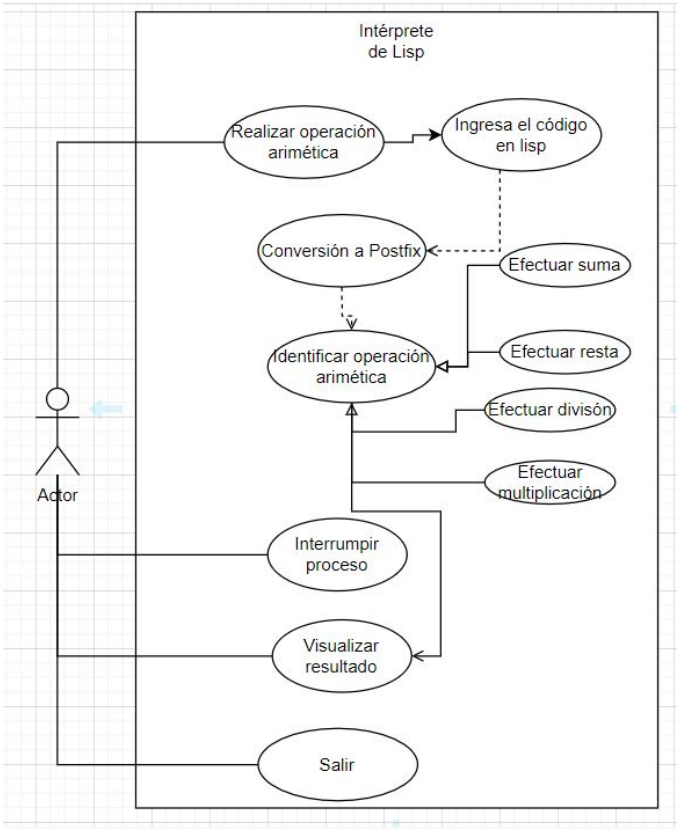
Para el desarrollo del proyecto, se utilizará la relación interface-class, con el uso de stacks, hashmaps y posiblemente treesets.

- Listas: es un subtipo o hijo de la interfaz Collection, que contiene elementos de tipo de lista según la estructura de datos, que permite también valores duplicados. Son principalmente útiles para almacenar y manipular las cantidades de datos y arreglos sin definir el tipo específico. Se utilizarán para guardar el código del intérprete que se vaya a convertir de Lisp a Java, de manera que pueda ser separado previamente y manipulado según sea conveniente.
    - Arraylist: usa un array dinámico para almacenar objetos y elementos, permite duplicados, mantiene el orden de la inserción y no está sincronizada, asimismo, se puede acceder a ellos al azar. Por el momento, se piensa utilizar para ir guardando los resultados de las operaciones realizadas por el código proveído por el usuario al inicializar el intérprete.
    - Vector: la estructura de datos que utiliza es de arrays dinámicos para acceder a los elementos, es sincronizada y tiene muchos métodos que no se incluyen necesariamente en el Collection Framework.
        - Stack: implementación del vector con la estructura de datos “último en entrar es el primero en salir”, contiene todos los métodos del vector, además de proveer métodos booleanos adicionales como “Push(), peek(), push(object o)”. Será especialmente útil para la conversión de postfix cuando se necesite ir separando operando de operador o condicional.

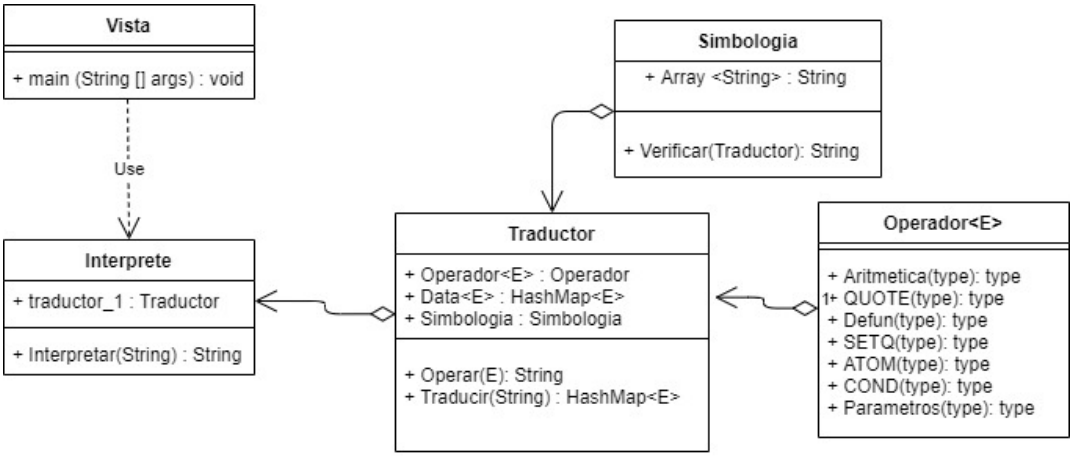
- Los hashmaps: utiliza la estructura de datos de Hash Table. No mantiene orden ni tolera los valores duplicados. Solo permite un valor de llave único y múltiples valores de nulos. No está sincronizada. Se emplea especialmente luego de llevar a cabo la traducción.

UML

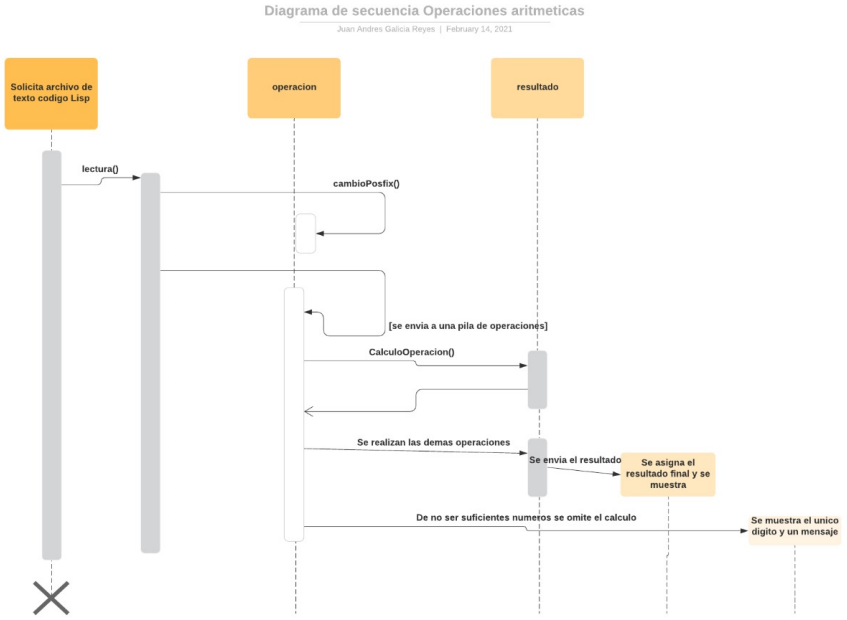
De secuencia



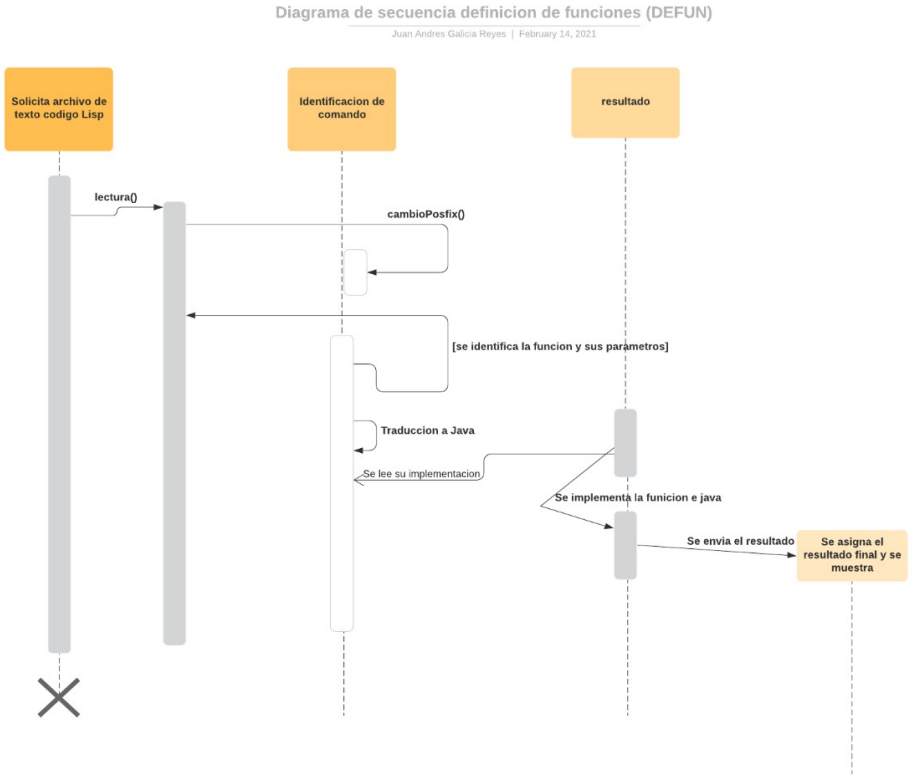
De clases



De secuencia operaciones aritméticas



De secuencia de definición de funciones



Bibliografía

1&1 IONOS España S.L.U. (2020). Programación funcional. IONOS Digitalguide. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/programacion-funcional/

5.3.- Implementación de interfaces. | PROG09.- Utilización avanzada de clases. (s.f.). Implementación de Interfaces. https://ikastaroak.birt.eus/edu/argitalpen/backupa/20200331/1920k/es/DAMDAW/PROG/PROG08/es_DAMDAW_PROG08_Contenidos/website_53_implementacin_de_interfaces.html

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

EcuRed. (2011). LISP - EcuRed. LISP. https://www.ecured.cu/LISP#:%7E:text=LISP%20posee%20un%20manejo%20de,ocurrir%20en%20c%2Fc%2B%2B

Gupta, J. 2019. Java : Collection Framework Hierarchy. Facing Issues On IT https://facingissuesonit.com/2019/10/15/java-collection-framework-hierarchy/

LUCA. (s.f.). *¿Qué es LISP?* ¿Qué es LISP? https://luca-d3.com/es/data-speaks/diccionario-tecnologico/list-procesor#:~:text=Lisp%

Reclu IT. (2020). Qué es Lisp. https://recluit.com/que-es-lisp/#.YBr6B-gzZPY

Sanchez, D. (2017, 21 noviembre). *Cual es la diferencia entre programación funcional y orientada a objetos*. https://yosoydani.com/la-diferencia-programacion-funcional-orientada-objetos/#:%7E:text=En%20la%20programaci%C3%B3n%20orientada%20a%20objetos%2C%20se%20almacenan%20los%20datos,como%20una%20transformaci%C3%B3n%20de%20datos