UNIVERSIDAD DEL VALLE DE GUATEMALA

Departamento de Ingeniería

Proyecto 1

Algoritmos y estructura de datos

Catedrático: Moises Antonio Alonso Gonzales

Milton Giovanni Polanco Serrano / 23471

Esteban Enrique Cárcamo Urízar / 23016

Víctor Samuel Mejía Hernández / 23442

Isabella Recinos Rodríguez / 23003

Sección 20

Guatemala

Miércoles, 14 de febrero del año 2024

Investigación de LISP

Lisp, acrónimo de "LIST Processing" (Procesamiento de Listas), es un lenguaje de programación de propósito general que fue desarrollado en la década de 1950 por John McCarthy. Es uno de los lenguajes de programación más antiguos que aún se utilizan y ha tenido una influencia significativa en el desarrollo de la informática y la inteligencia artificial.

Es reconocido como un lenguaje de programación muy poderoso y versátil en el campo de la inteligencia artificial. Una de las razones es que tiene implementaciones muy sólidas, por ejemplo: tiene SBCL (Steel Bank Common Lisp) que compila expresiones a código de máquina. Lisp también permite escribir código de alto y bajo nivel, haciéndola muy versátil. También cuenta REPL (Bucle de Lectura-Evaluación-Impresión) interactivo que facilita la programación interactiva.

Entre las aplicaciones más destacadas se encuentra Wolfram Alpha, un motor que resuelve consultas matemáticas; Siri, el asistente virtual de Apple y Deep Blue de IBM. Lisp también fue pionero en cuanto a estructuras de datos de árbol, manejo de almacenamiento de automático y el compilador autocontenido.

La programación funcional tiene un énfasis en funciones puras; los datos son inmutables, esto quiere decir que una vez creados los datos no cambian. Está basado en la evaluación de expresiones y la composición de funciones. Algunas de las ventajas que esto presenta es facilitar la depuración al evitar efectos secundarios; también evita problemas de concurrencia gracias a que no modifica datos. Es uno de los lenguajes que ha influido en el desarrollo de la programación funcional. Al tener un énfasis en funciones puras, evaluación de expresiones y manipulación de listas, este se alinea con los principios de la programación funcional.

En POO los programas están organizados como una colección colaborativa de objetos. Cada representa una entidad con datos y su comportamiento (llamado métodos). POO permite usar herencia y composición de código, lo que facilita su uso. Lisp también ha implementado esto y que admite clases, instancias y métodos. A pesar de esto, Lisp no es tan ampliamente utilizado para POO.

Características:

- Sintaxis basada en listas: En Lisp, los programas se escriben en forma de listas anidadas. Esta sintaxis uniforme y sencilla es una de las características más distintivas de Lisp.
- Tratamiento de funciones como datos: En Lisp, las funciones son ciudadanos de primera clase, lo que significa que pueden ser pasadas como argumentos a otras funciones, devueltas como valores de otras funciones y almacenadas en estructuras de datos.
- Evaluación perezosa: Lisp utiliza una estrategia de evaluación perezosa, lo que significa que solo se evalúan las partes de una expresión que son necesarias para obtener el resultado deseado.

- Programación funcional: Lisp es un lenguaje de programación funcional, lo que significa que favorece el uso de funciones puras y evita el uso de efectos secundarios.
- Macros: Lisp permite la definición de macros, que son formas de extender el lenguaje permitiendo al programador definir nuevas formas de expresión que son expandidas en el código antes de ser evaluadas.
- Inteligencia Artificial: Lisp ha sido históricamente utilizado en el campo de la inteligencia artificial. En la década de 1960, el lenguaje se convirtió en uno de los principales lenguajes de programación para la investigación y desarrollo de sistemas de inteligencia artificial.
- Common Lisp: Es una de las implementaciones más populares de Lisp. Common Lisp es un lenguaje de programación de propósito general que extiende las características de Lisp original con una variedad de características modernas.
- Emacs Lisp: Es una variante de Lisp utilizada como lenguaje de extensión para el editor de texto Emacs.

Ventajas

1. Facilidad de Prototipado: Lisp permite desarrollar prototipos rápidos para pruebas e iteraciones. Esto es especialmente útil en el proceso de diseño y experimentación.

2. Soporte de Manejo de Errores: Lisp cuenta con un sólido mecanismo de manejo de errores. Esto facilita la identificación y corrección de problemas en el código.

3. Extensibilidad: Lisp es altamente extensible. Permite a los programadores agregar nuevas funcionalidades o adaptar el lenguaje según sus necesidades específicas.

4. Historia y Contribuciones: Lisp fue pionero en muchas ideas en ciencias de la computación, incluyendo estructuras de datos de árbol, manejo de almacenamiento automático, tipos dinámicos y el concepto de compilador autocontenido.

   A lo largo de los años, ha habido varios dialectos de Lisp, como Scheme, Common Lisp, Emacs Lisp y Clojure.

Conclusiones

Lisp cuenta con numerosas contribuciones a la programación como funciones puras, manipulación de listas y estructuras de datos de árbol, convirtiéndo se en un pionero de la programación funcional. Ha servido como base de proyectos importantes e innovadores en su fecha. También ha sido de influencia en el campo de la inteligencia artificial. Por otro lado, es un lenguaje muy versátil de usar ya que presenta muchas facilidades para trabajar. Tales como los macro, REPL, SBCL y Emacs, facilitando así su prototipado y contando con un soporte de manejo de errores.

Investigación sobre Java Collections Framework

Java Collection Framework fue introducido en la versión JDK 1.2 de Java. Anteriormente, las implementaciones en las versiones pre-JDK 1.2 de la plataforma Java incluían pocas clases de estructuras de datos y no contenían un marco de colecciones. Los métodos estándar para agrupar objetos Java eran a través de las clases de array, Vector y Hashtable, esto dificulta mucho su uso.

Para abordar la necesidad de estructuras de colección reutilizables, se desarrollaron varios marcos independientes. El más utilizado fue el paquete de Colecciones de Doug Lea y la Biblioteca de Colección Genérica (JGL) de ObjectSpace, cuyo objetivo principal era la consistencia con la Biblioteca de Plantillas Estándar (STL) de C + +. Es una arquitectura unificada que se utiliza para representar y manipular colecciones. Esto permite que se manipulen independientemente los detalles de su representación, a su vez, reduce el esfuerzo de programación al proporcionar estructuras de datos y algoritmos. Las interfaces de colección permiten que las colecciones se manipulen independientemente de los detalles de su representación.

Generalmente, en lenguajes con orientación a objetos, las interfaces cuentan con una jerarquía como la que se presenta a continuación:

Collection
- Set
  - SortedSet
  - NavigableSet

List

Queue
- Deque
- BlockingQueue
- TransferQueue
- BlockingDeque

Para el proyecto en específico, se han investigado las siguientes implementaciones útiles a usar que se analizó que pueden llegar a ser usadas durante la elaboración del intérprete. Cada herramienta tiene una serie de métodos que permiten manipular los datos almacenados en otras colecciones, tales como:

HashSet: Es una implementación de la interfaz Set. No permite elementos duplicados y no garantiza ningún orden específico de los elementos. Utiliza una tabla hash para el almacenamiento, lo que proporciona un rendimiento constante para las operaciones básicas como agregar, eliminar y contener. Puede llegar a ser usado para los parámetros de las funciones o en las expresiones regulares para validar frases repetidas.

ArrayList: Permite elementos duplicados y mantiene el orden de inserción. Proporciona operaciones para modificar la lista en cualquier punto y puede crecer dinámicamente. Opción alternativa para los LinkedList, que pueden llegar a tener la misma función.

HashMap: Es una implementación de la interfaz Map. Almacena elementos en pares clave-valor y permite una clave nula y múltiples valores nulos. Ofrece un rendimiento constante para las operaciones básicas cómo obtener e insertar. Esta nos ofrecerá la oportunidad de poder guardar todas las variables que se vayan a realizar o para las funciones. Cada entrada en el mapa almacena los datos en una clave y su valor correspondiente. Un mapa no puede contener claves duplicadas; cada clave puede mapear a lo sumo un valor. Proporciona tres vistas de colección, que permiten que los contenidos de un mapa se vean como un conjunto de claves, una colección de valores, o un conjunto de mapeos clave-valor y algunos de los métodos principales para trabajar con los Mapas son put(K clave, V valor), get(K clave), remove(K clave), containsKey(K clave), containsValue(V valor) y values().

LinkedList: No es tan comúnmente utilizada como las anteriores, LinkedList es una implementación de la interfaz Queue. Es una lista enlazada de doble sentido que puede almacenar elementos duplicados. Proporciona métodos para insertar y eliminar en ambos extremos, lo que la hace adecuada para implementar colas y pilas. Posible opción para poder guardar las expresiones regulares y poner entender token por token que desea realizar el usuario.

Algunas de las características de Java Collections Framework son:

- Reducción del esfuerzo de programación: Proporciona estructuras de datos y algoritmos listos para usar, esto le da la opción al usuario de no tener que escribirlos.
- Alto rendimiento: Ofrece implementaciones de alto rendimiento de estructuras de datos y algoritmos. Además, las diferentes implementaciones de cada interfaz son intercambiables, lo que permite optimizar los programas cambiando las implementaciones.
- Interoperabilidad: Establece un lenguaje común para pasar colecciones entre APIs no relacionadas.
- Facilita el aprendizaje de APIs: No requiere que aprendas múltiples APIs de colección ad hoc.
- Fomenta la reutilización de software: Proporciona una interfaz estándar para colecciones y algoritmos para manipularlas.
- Variedad de implementaciones: Incluye implementaciones de propósito general, implementaciones de propósito especial, implementaciones concurrentes, implementaciones de envoltorio, implementaciones de conveniencia, implementaciones abstractas y algoritmos.
- Métodos de modificación opcionales: Las implementaciones pueden optar por no realizar una o más de estas operaciones, lanzando una excepción en tiempo de ejecución si se intentan.

Ventajas

1. Implementaciones optimizadas de estructuras comunes: Este enfoque ofrece versiones mejoradas de estructuras básicas como listas, conjuntos y mapas. Estas optimizaciones aseguran un manejo de datos más eficaz y un rendimiento superior para diversas aplicaciones.

2. Uniformidad y coherencia en la interfaz: El marco proporciona una interfaz estandarizada para sus colecciones, simplificando así su manejo y entendimiento. Esta característica permite a los usuarios alternar entre distintas implementaciones de forma fluida, sin necesidad de modificar el código existente.

3. Variedad de funcionalidades para mayor versatilidad: Con una amplia gama de herramientas para la organización, búsqueda, filtrado y manipulación de datos, este marco facilita el desarrollo de soluciones complejas de forma más accesible y eficiente.

4. Confiabilidad y seguridad comprobadas: Las colecciones dentro de este marco se han diseñado para ser tanto seguras como fiables. Gracias a un proceso continuo de pruebas y optimizaciones, se han resuelto numerosos problemas de rendimiento y estabilidad, asegurando así un funcionamiento consistente y seguro en variados contextos.

Conclusiones

El Java Collection Framework proporciona una serie de interfaces y clases que permiten a los programadores manejar grupos de objetos de manera eficiente. Las diferentes implementaciones de cada interfaz permiten optimizar los programas cambiando las implementaciones según las necesidades específicas. Al proporcionar una interfaz estándar para las colecciones y los algoritmos para manipularlas, el Java Collection Framework facilita el aprendizaje de las APIs y fomenta la reutilización del software. Esto también permite la interoperabilidad entre APIs no relacionadas. Con una variedad de implementaciones para cada interfaz, el Java Collection Framework es flexible y puede adaptarse a una amplia gama de situaciones.

Diagramas UML para la realización del intérprete de LISP

DIAGRAMA DE CLASES

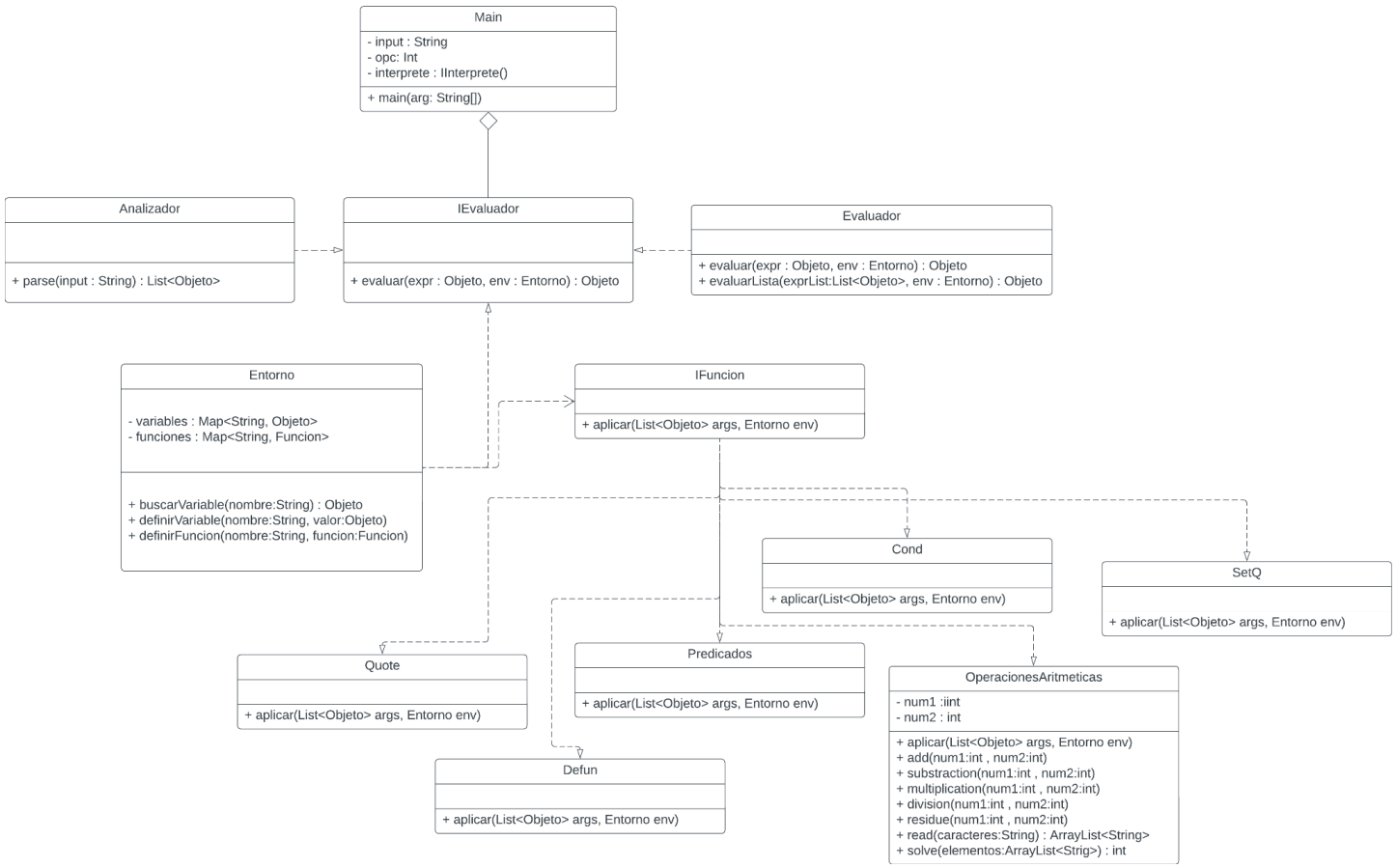


DIAGRAMA DE CASO DE USO

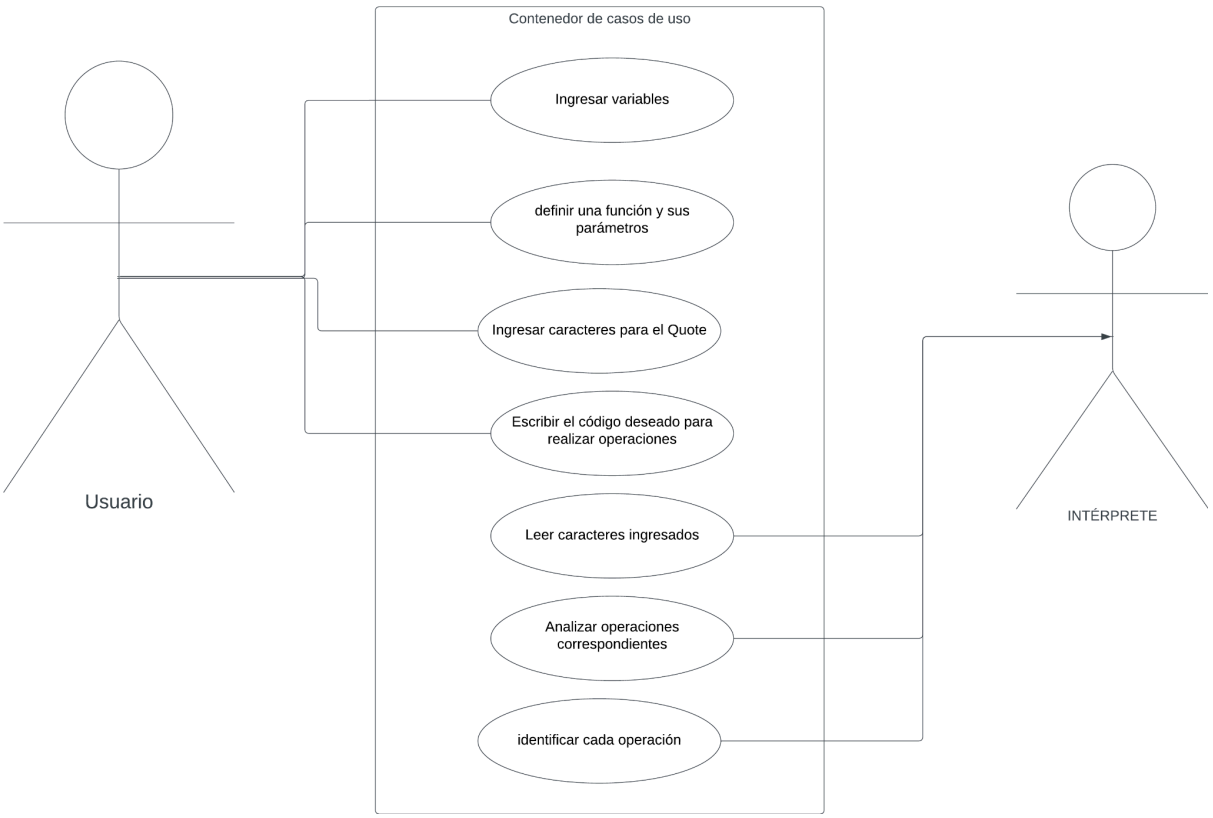
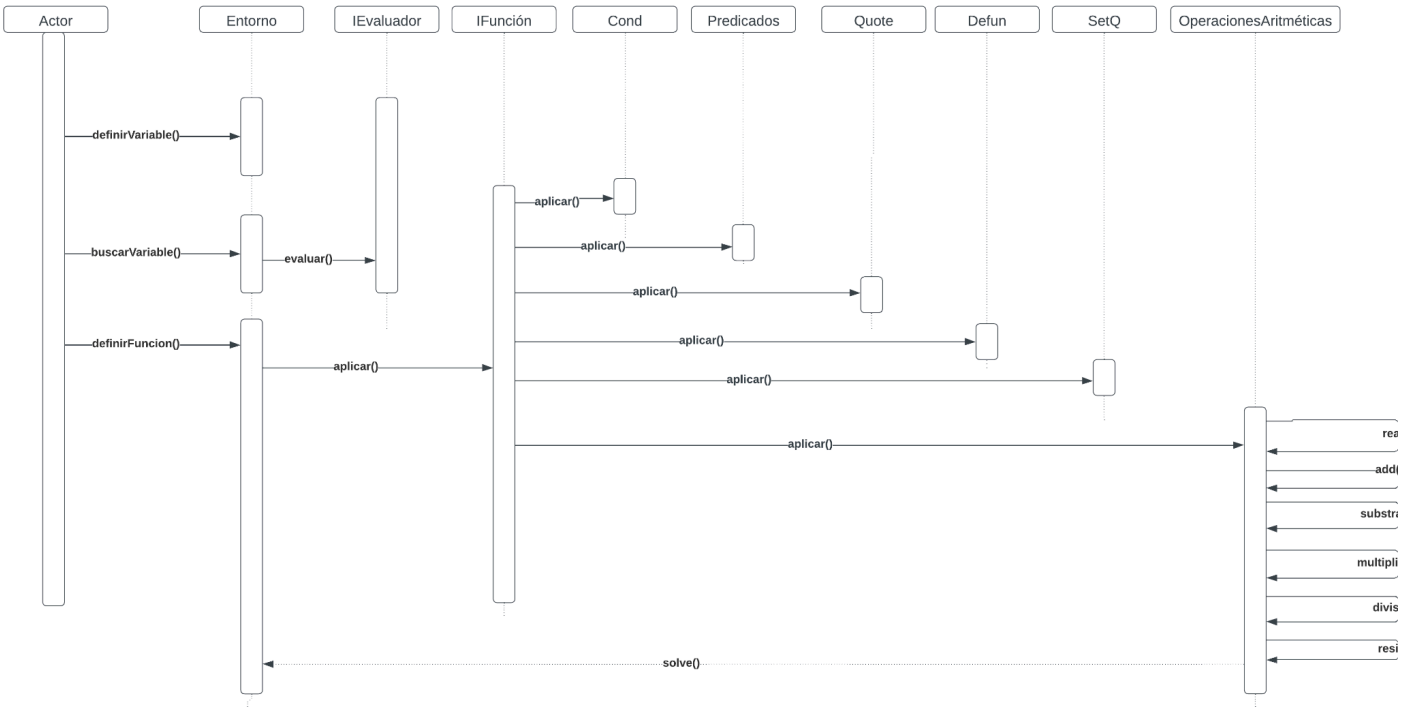


DIAGRAMA DE SECUENCIA



Referencias

(1) ¿Qué es LISP? - Reclu IT. https://recluit.com/que-es-lisp/.
(2) ¿Qué es LISP? - Trucoteca. https://bing.com/search?q=ventajas+de+lisp.
(3) ¿Qué es LISP? - Trucoteca. https://trucoteca.com/que-es-lisp/.
(4) Lisp - Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Lisp.
(5) ¿Cuáles son las ventajas y desventajas de lisp? - Tusclases. https://www.tusclases.mx/questions/programacion/cuales-son--ventajas--desventajas--lisp.
(6) Java Collections Framework in Depth with Examples for Beginners. https://www.javaguides.net/2018/08/collections-framework-in-java.html.
(7) Collections Framework Overview - Oracle. https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html.
(8) Java Collections Framework. https://docs.oracle.com/en/java/javase/21/core/java-collections-framework.html.
(9) Lesson: Implementations (The Java™ Tutorials > Collections) - Oracle. https://docs.oracle.com/javase/tutorial/collections/implementations/.
(10) Java - Collections Framework - Online Tutorials Library. https://www.tutorialspoint.com/java/java_collections.htm.
(11) The Collections Framework - MIT. https://web.mit.edu/java_v1.5.0_22/distrib/share/docs/guide/collections/index.html.
(12) Java Collections Framework: Una introducción. - dCodinGames. https://dcodingames.com/java-collections-framework-una-introduccion/.
(13) Collections Framework Overview - Oracle. https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html.
(14) Map (Java Platform SE 8) - Oracle. https://docs.oracle.com/javase/8/docs/api/java/util/Map.html.
(15) map interface in java - Scaler Topics. https://www.scaler.com/topics/map-interface-in-java/.
(16) Links de los diagramas de clases, secuencia y de uso
https://lucid.app/lucidchart/d961f7ae-0b50-4bec-9ff2-9790371bdf09/edit?viewport_loc=-793%2C792%2C3699%2C1794%2C0_0&invitationId=inv_057bdbb3-d790-417a-a4f2-e7dda6425ea8
(17) Link del repositorio a usar para el proyecto
https://github.com/CarEsteban/proyecto1-Datos