# Proyecto 1, fase 1

Julio Herrera
Oliver de León
Laura Tamath
Algoritmos y estructuras de datos

Universidad del Valle de Guatemala

Facultad de Ingeniería

# LISP

Es una familia de lenguajes de programación, el cual soporta paradigmas de programación y hace uso de una notación matemática práctica que es basada en el "cálculo lambda". Los programas que están desarrollados en Lisp permiten manejar el propio código como si se tratase de una estructura de datos, por lo tanto, permite que macrosistemas desarrollen nuevas sintaxis dentro de este.

En 1958 se desarrolló y en 1960 fue publicado el diseño de Lisp, gracias a John McCarthy. Desde sus inicios estuvo involucrado a los equipos de investigación en el campo de la inteligencia artificial. Pues este fue el primer lenguaje de inteligencia artificial, el cual introdujo una serie de nuevos conceptos, que actualmente son utilizados, como: procesos por listas, estructuras de datos en forma de árbol, manejo de almacenamiento y sentencia if, then y else. La primera persona en implementar la práctica este lenguaje fue Steve Russel, quien implementó algunas funciones mediante el código máquina de un IBM 704 y con el transcurso del tiempo, logró desarrollar un intérprete para Lisp, sin embargo, en 1974, Mike Levin y Tim Hant lograron desarrollar un compilador completo, el cual permitía mezclar libremente funciones interpretadas y compiladas. (Velasco, 2011)

Las **características** que más sobresalen de este lenguaje son:

> Posee un manejo automático en la memoria, el cual libera el espacio utilizado por los objetos que dejan de ser necesarios.
>
> Este no posee un sistema de tipo "static", lo que quiere decir que los tipos no se asocian a las variables, sino que a los valores.
>
> Inclusión de un mecanismo simple para utilizar la evolución perezosa de expresiones.
>
> Los cálculos iterativos se pueden realizar en un espacio constante de memoria, aun así se utiliza recursividad para el cálculo.
>
> La codificación de un programa en Lisp se basa en la descripción del problema, indicando cómo y lo que se quiere conseguir, pero sin la necesidad de indicarle cada paso a la computadora para realizar una tarea específica.
>
> (Ramírez, 2018)

# JAVA COLLECTIONS FRAMEWORK

Una colección es definida como un objeto capaz de almacenar una colección de objetos dentro de sí. Siguiendo este concepto la plataforma Java incluye dentro de su amplia gama de herramientas la Java Collections Framework. Java Collections Framework es un tipo de arquitectura unificada que permite la administración y representación de una colección. Esto permite manipular de forma independiente la colección, inclusive al momento de implementar detalles dentro de la misma.

Las principales ventajas que brinda Java Collection Framework al usuario son:

- Reduce el esfuerzo al momento de programar.
- Incrementa el desempeño.
- Permite la interoperabilidad entre APIs no relacionados.
- Reduce el esfuerzo requerido para aprender, desarrollar, diseñar e implementar API.
- Fomenta la reutilización de software.

Java Collection Framework trabaja con dos colecciones esenciales definidas a continuación.

**Colección de Interfaces (Collection Interfaces)**
La colección de interfaces provistas se dividen en dos grupos fundamentales. La colección de interfaces básicas ***(java.util.Collection)***, con sus respectivos descendientes:

- *java.util.Set*
- *java.util.SortedSet*
- *java.util.NavigableSet*
- *java.util.Queue*
- *java.util.concurrent.BlockingQueue*
- *java.util.concurrent.TransferQueue*
- *java.util.Deque*
- *java.util.concurrent.BlockingDeque*

Por otra parte se encuentra la colección de interfaces que toman como base la extensión **java.util.Map**, las cuales no son colecciones como tal. Estas interfaces contienen operadores *collection-view,* lo que les permite ser manipulados como una.

- *java.util.*SortedMap
- *java.util.*NavigableMap
- *java.util.*concurrent.ConcurrentMap

- *java.util.*concurrent.ConcurrentNavigableMap

**Colección de Implementaciones (Collection Implementations)**
Dentro de los propósitos generales de las implementaciones cabe mencionar que apoyan y dan seguimiento a los operadores opcionales, evitando así las restricciones sobre los elementos que puedan contener.

El listado a continuación presenta las interfaces capaces de actuar como agentes de implementación.

- *AbstractCollection*
- *AbstractSet*
- *AbstractList*
- *AbstractSequentialList*
- *AbstractMap*
- *BlockingQueue*
- *TransferDeque*
- *ConcurrentMap*
- *ConcurrentNavigableMap*

Siguiendo con las concurrentes implementaciones disponibles:

- *LinkedBlockingQueue*
- *ArrayBlockingQueue*
- *DelayQueue*
- *PriorityBlockingQueue*
- *SynchronousQueue*
- *LinkedBlockingDeque*
- *LinkedTransferQueue*
- *CopyOnWriteArrayList*
- *CopyOnWriteArraySet*
- *ConcurrentSkipListSet*
- *ConcurrentHashMap*
- *ConcurrentSkipListMap*

A lo largo del desarrollo del proyecto 1, dado que se trabajará con el lenguaje LISP (basado en su mayoría en la sucesión o implementación de listas) será vital considerar la utilización de implementaciones que incluyan este recurso. Implementaciones como CopyOnWriteArraySet y ArrayList serán fundamentales. De la misma forma interfaces relacionadas con este recurso serán esenciales, interfaces como: java.util.set, AbstractList y AbstractSequentialList serán utilizadas.

Tanto estas implementaciones como interfaces buscan actuar como esqueleto del intérprete que se planea desarrollar, como ya se ha mencionado LISP es un lenguaje cuya estructura se basa en la implementación de listas. Por lo que la implementación de estos recursos buscaría actuar como plano de lo que llegaría a ser a grandes rasgos el intérprete a desarrollar.

(Oracle, 2020)

## REFERENCIAS


Ramírez, F. (2018). *ThinkBig.* Retrieved from Historia de la IA: John McCarthy y el lenguaje de programación LISP: https://empresas.blogthinkbig.com/historia-ia-john-mccarthy-lisp/

Velasco, J. (2011). *hipertextual.* Retrieved from Historia de la tecnología: Lisp: https://hipertextual.com/2011/10/historia-de-la-tecnologia-lisp

Oracle (2020). *Collections Framework Overview.* Retrieved from Oracle: Java Documentation: https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html