Ruth Daniela Ruano – 241098

Giancarlo Sagastume – 24278

Christopher García – 24928

Entrega 1

a. Que es Lisp

Lisp es un lenguaje de programación creado en 1958, por John McCarthy. A pesar de ser muy pero muy antiguo el lenguaje sigue siendo usado y sigue siendo uno de los lenguajes más avanzados de su tipo. Lisp fue diseñado originalmente para la manipulación de datos y se usa mucho para inteligencia artificial y automatización.

Lisp en POO

Lisp y programación orientada a objetos no son lo mismo, pero Lisp tiene un poco de POO y POO tiene un poco de Lisp. Algunos de los sistemas que Lisp integro de POO en algunos de sus dialectos, como “Common Lisp Object System” (CLOS).

Historia

- **1958**: John McCarthy desarrolla Lisp en el MIT como un lenguaje para manipulación simbólica en inteligencia artificial.
- **1960s**: Se convierte en el principal lenguaje de IA, con implementaciones como Maclisp.
- **1970s**: Surgen dialectos importantes como Scheme y Common Lisp.
- **1980s**: Empresas como Symbolics crean hardware especializado para Lisp.
- **1990s – 2025:** Lisp pierde popularidad en la industria, pero sigue en el ámbito académico y en nichos como la automatización y la robótica.

b. Java Collections Framework

El Framework de Colecciones Java (JCF) es un conjunto de clases e interfaces en Java que ofrece estructuras de datos eficaces para el almacenamiento, manejo y procesamiento de colecciones de objetos. Estas estructuras facilitan el manejo de grandes cantidades de datos sin la necesidad de aplicar algoritmos de almacenamiento y búsqueda de manera manual.

JCF es un componente esencial de Java, puesto que maximiza la utilización de la memoria y potencia la eficacia del código.

c. Colocar un ambiente de trabajo Lisp, de cualquiera de los dialectos principales, para poder aprender y practicar el lenguaje.

Vamos a configurar nuestro ambiente de trabajo para aprender y practicar Lisp utilizando **Visual Studio Code** como editor. Optamos por trabajar con el dialecto **Racket (Scheme)**, ya que es amigable para principiantes y cuenta con un entorno robusto. Instalaremos **Racket** desde https://racket-lang.org/, y en Visual Studio Code añadiremos la extensión **Racket** para facilitar la ejecución y depuración de nuestros programas. Así podremos escribir y probar nuestro código Lisp directamente en Visual, aprovechando sus herramientas modernas.

d. Ejecutar un pequeño programa en ese Lisp, como la conversión de grados Farenheit a centígrados.

```
 1 ; SLIME 2.24
 2 CL-USER>
 3 ; No value
 4 CL-USER> (defvar f 10)
 5 F
 6 CL-USER> (setq f 10)
 7 10
 8 CL-USER> f
 9 10
10 CL-USER> (defvar convertir)
11 CONVERTIR
12 CL-USER> (setq convertir (* (/ 5 9) (- f 32)))
13 -110/9
14 CL-USER>
```

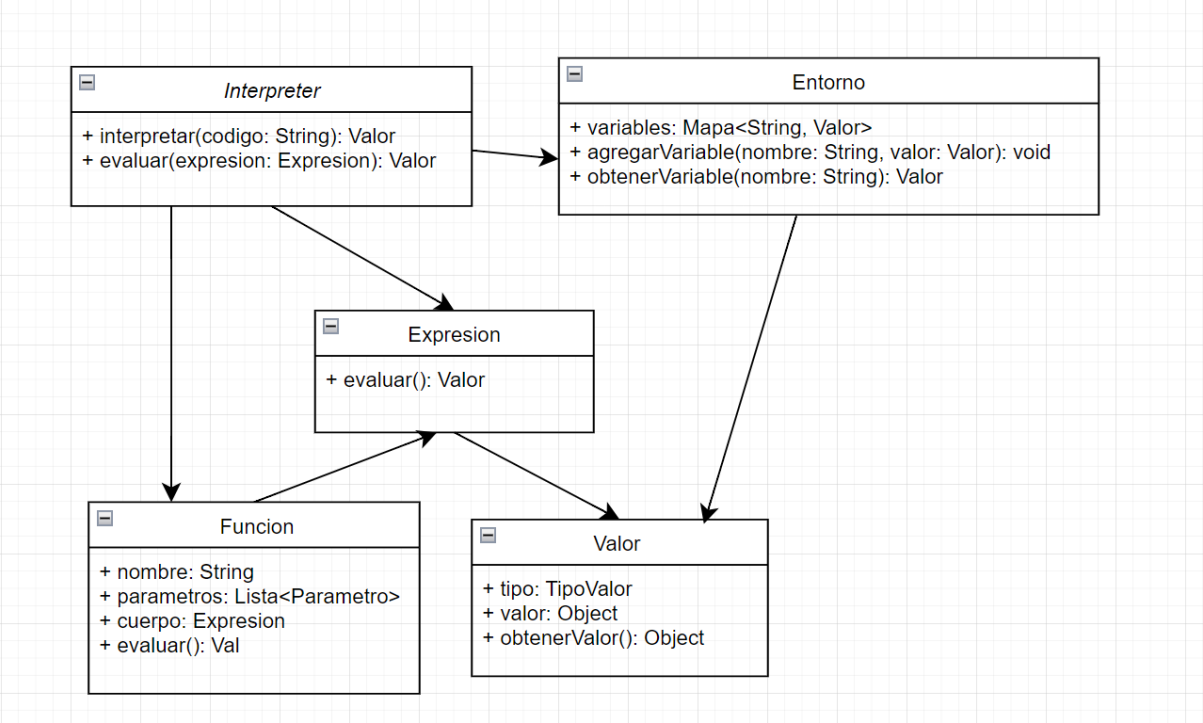
e. Ejecutar un programa en Lisp para la producción del término n de la serie de Fibonacci y Factorial de un número.

```
CL-USER> (defun fibonacci (n)
...   (if (<= n 1)
...       n
...       (+ (fibonacci (- n 1))
...          (fibonacci (- n 2)))))
FIBONACCI
CL-USER> (format t "Fibonacci(10): ~a~%" (fibonacci 10))
Fibonacci(10): 55
NIL
CL-USER>
```

```
CL-USER> (defun factorial (n)
...   (if (= n 0)
...       1
...       (* n (factorial (- n 1)))))
FACTORIAL
CL-USER> (format t "Factorial(5): ~a~%" (factorial 5))
Factorial(5): 120
NIL
CL-USER>
```

f. Inicio del diseño de su propio Lisp: Diagramas UML que permitan conocer la estructura del intérprete (Clases, estados, secuencias, etc.)

UML inicial



Versión Final UML:

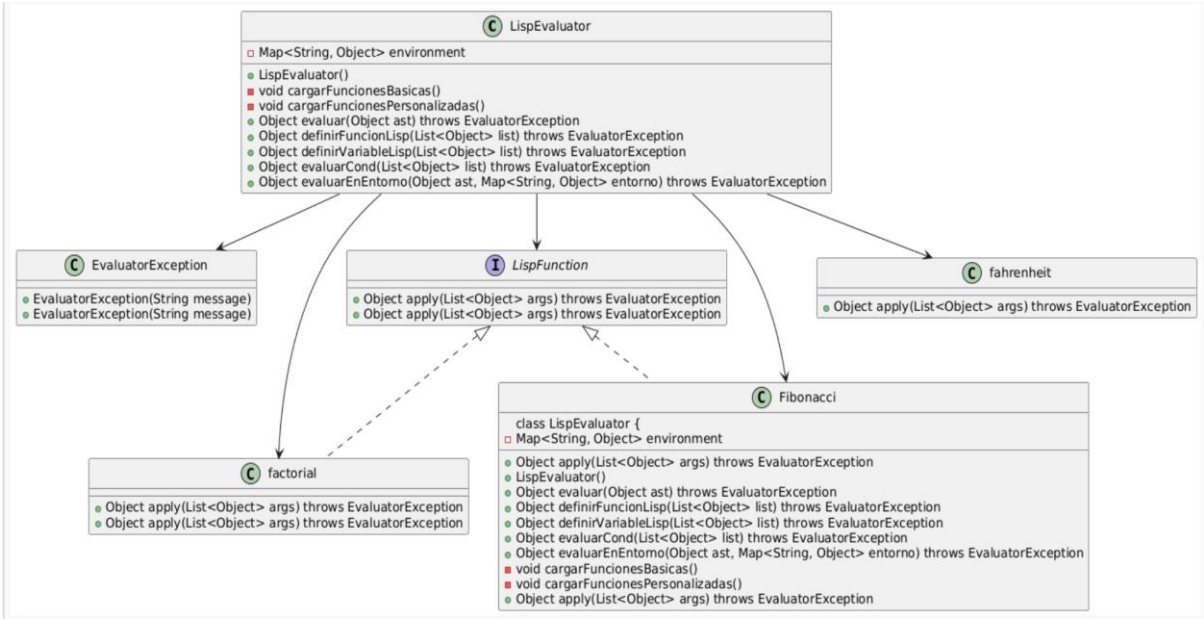


Diagrama de Uso:

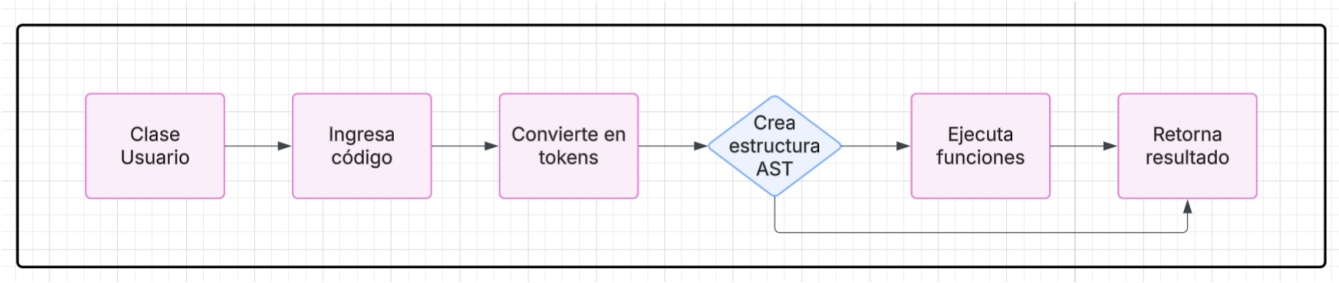
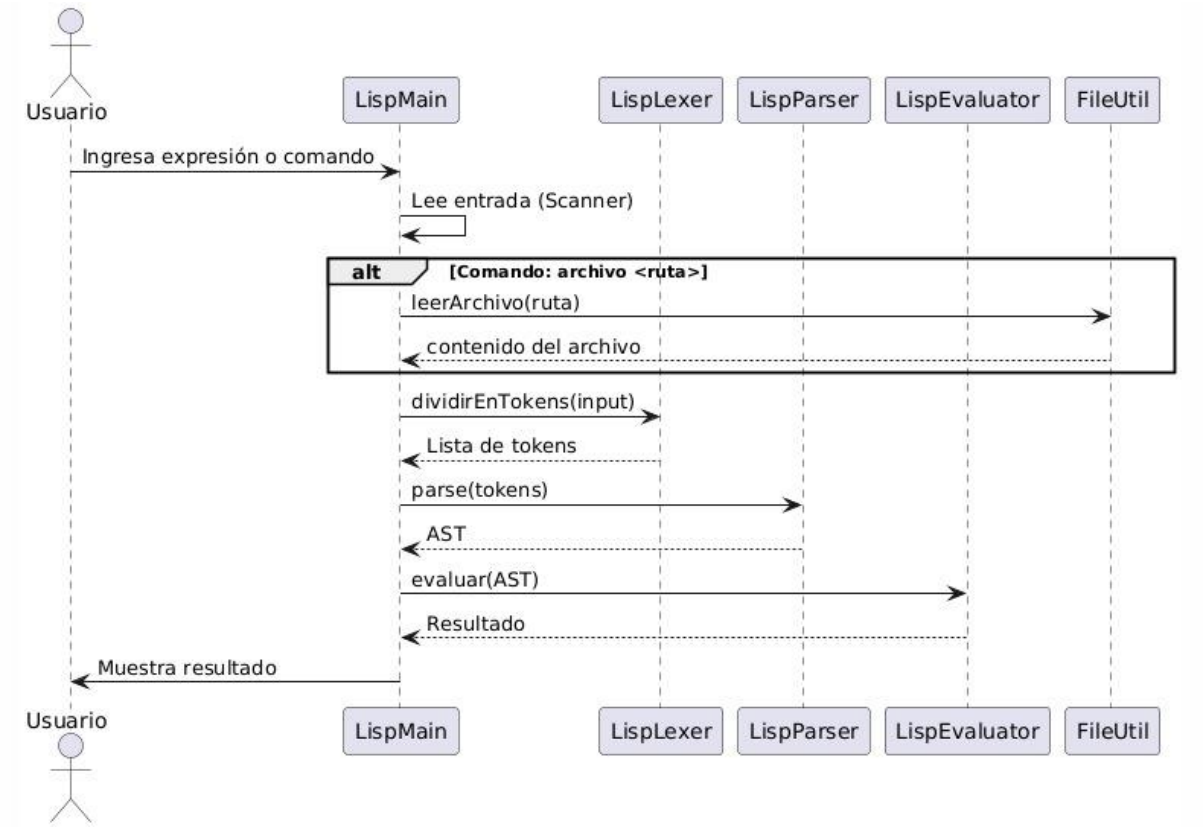


Diagrama de Secuencia:



Enlace de video:

https://www.canva.com/design/DAGihUyFK_I/f1jeeknZmoRmDOw1DTZIfg/edit?utm_content=DAGihUyFK_I&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Link de Git: https://github.com/RuanDaniela/Proyecto1