Proyecto No. 1

Realizar: un intérprete de LISP básico.
Realizarse: en grupo de 3 estudiantes**.**
Fecha de entrega:

FASE 1: en la semana del 6 al 11 de febrero 2023.
FASE 2: en la semana del 13 al 17 de marzo 2023.

Objetivos:

a. Utilizar el Java Collections Framework para desarrollo de aplicaciones.
b. Saber escoger estructuras de datos, con su complejidad en tiempo y espacio para la implementación de aplicaciones.
c. Identificar los principales elementos y usos de la programación Funcional (Functional Programming[1]).

Programa a realizar:

Desarrollo de un intérprete LISP para un subconjunto sencillo de instrucciones de alguno de los dos dialectos principales (**Common LISP y Scheme**). Deben implementarse como mínimo:

a) operaciones aritméticas,
b) instrucción QUOTE o **‘** (single quote, para interrumpir el proceso de evaluación de expresiones)
c) definición de funciones (DEFUN),
d) SETQ
e) predicados (ATOM, LIST, EQUAL, <, >),
f) condicionales (COND), nota: no es necesario implementar IF
g) paso de parámetros. (un parámetro puede ser incluso una función)

NOTA: recuerde que LISP es un lenguaje que permite recursividad, por lo que sus funciones deben soportarlo.

El intérprete desarrollado deberá permitir ejecutar programas sencillos en LISP.

El grupo debe mostrar la razón por la cual ha seleccionado las estructuras de datos utilizadas en el proyecto y la forma en que se relacionan para implementar el intérprete.

[1] https://www.tutorialspoint.com/functional_programming/functional_programming_introduction.htm

Entrega FASE 1:

Tareas:

a. Investigación corta de Lisp, su historia, sus características, donde es empleado. Incluya además conceptos de programación funcional y como se compara con Programación Orientada a Objetos.
b. Investigación corta sobre Java Collections Framework (especialmente la jerarquía de interfaces e implementaciones). Indicar cuales se usarán en el proyecto, indicando como se emplearán.
c. Colocar un ambiente de trabajo Lisp, de cualquiera de los dialectos principales, para poder aprender y practicar el lenguaje.
d. Ejecutar un pequeño programa en ese Lisp, como la conversión de grados Farenheit a centígrados.
e. Ejecutar un programa en Lisp para la producción del termino n de la serie de Fibonnacci y Factorial de un número.
f. Inicio del diseño de su propio Lisp: Diagramas UML que permitan conocer la estructura del intérprete (Clases, estados, secuencias, etc.)

Calificación: su programa debe funcionar para ser calificado.

Aspecto	Puntos
Investigación corta sobre Lisp	25
Diagramas UML que describen su proyecto. Por lo menos los diagramas de caso de uso, de clases y los de secuencia (de las principales operaciones).	25
Investigación corta de JCF e Indicación de las estructuras del Java Collections Framework utilizadas, indicando la razón y referencias para su uso.	30
Desarrollo de programa Fibonnaci y Factorial en forma recursiva. Incluya los programas Lisp desarrollados.	20
TOTAL:	100

Entrega FASE 2:

Tareas:

a. Diseño final del interprete Lisp: Diagramas UML que permitan conocer la estructura del intérprete (Clases, estados, secuencias, etc.)
b. Control de versiones de todos los documentos y código que seleccione el grupo.
c. Un esquema de pruebas unitarias de las principales estructuras y partes críticas del intérprete. Deben estar también bajo el control de versiones.
d. Cada dos semanas se deben mostrar en clase los avances realizados. Serán breves exposiciones de 5 o 10 minutos por grupo, mostrando productos.
e. Un pequeño programa ejecutándose en el ambiente Lisp implementado por el grupo. Debe contener solamente funcionalidad básica del lenguaje. Subirlo a Canvas.
f. Videos demostrativos de todo el proceso de desarrollo y del intérprete funcionando.

Calificación: su programa debe funcionar para ser calificado.

Aspecto	Puntos
Estilo de codificación: comentarios, identación, nombres de variables significativas. Documentación generada con Javadoc.	10
Diagramas UML que describen al intérprete de Lisp. Por lo menos los diagramas de caso de uso, de clases y los de secuencia (de las principales operaciones).	15
Casos de prueba en JUnit.	10
Indicación de las estructuras del Java Collections Framework utilizadas, indicando la razón y referencias para su uso.	10
Uso del repositorio del SCM. Estructurado y con seguimiento semanal de los productos realizados.	10
Video que muestre el funcionamiento completo del programa y del proceso de desarrollo	5
Funcionamiento del intérprete. Ejecución de un programa en LISP, como el Factorial o Fibonnaci (que sea en forma recursiva).	40
TOTAL:	100

LISP es acrónimo de List Processing (todo es una lista).

Por eso las etapas recomendadas para que desarrollen su propio LISP pueden ser:

- Separar una expresión LISP en sus componentes (llamados tokens). Ejemplo:

  (+ 5 (* 2 3))
  Los tokens son `(, +, 5, (, *, 2, 3, ), )`

- Luego ordenar estos elementos para formar una lista que la represente. Usaré aquí la notación de listas de Python, para facilidad de mostrar la expresión:

  ```
  [+ 5 [* 2 3] ]
  ```

  Notar que la expresión se convirtió en una lista que contiene otra lista.

- Ahora podemos evaluar, es decir ejecutar esta expresión. Esto implica evaluar cada una de las listas que compone la expresión.

  Para ello recordamos que LISP trabaja en notación prefix, y que el primer elemento en una lista indica la operación a realizar. Siguiendo nuestro ejemplo:

  a) Evaluamos la + de 5 y otra expresión: [* 2 3]
  b) Evaluamos + 5 y la evaluación de * con los operandos 2 y 3
  c) Evaluamos + 5 y 6
  d) Resultado: 11

- Y procedemos nuevamente a leer otra expresión LISP, extraer sus tokens, formar una lista que represente la expresión, evaluarla y devolver el resultado.

  A estas acciones se les llama REPL (Read, Evaluate, Print, Loop)

Cuando ya se tenga un REPL implementado, se podrá ver un poco más sobre como definir funciones, pasar parámetros y hacer que se soporte recursión.