UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

Guatemala 16 de Enero de 2024
Adrián López - 231361
Luis Fernando Mendoza

Algoritmos y Estructura de Datos

Proyecto 1: Intérprete de LISP

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).

"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)."

a. Investigación corta sobre LISP:

Lisp, es un lenguaje de programación desarrollado en la década de los años 50, el nombre Lisp es el acrónimo en inglés de LIST Processing. A lo largo de la historia se convirtió en uno de los lenguajes de programación que se sigue utilizando y su influencia en el campo de la Inteligencia Artificial y la programación funcional.

Historia:

Lisp fue desarrollado por John McCarthy a finales de los años 50 en el Instituto de Tecnología de Massachusetts (MIT), lo desarrolló con el objetivo de que fuera un

lenguaje de programación para la manipulación de listas y símbolos por lo que fue el lenguaje perfecto para la investigación en Inteligencia Artificial de ese tiempo.

Características:

- Listas como estructuras de datos fundamentales: En Lisp, las listas son una parte integral del lenguaje y se utilizan para representar tanto datos como código.
- Evaluación perezosa: Lisp utiliza una estrategia de evaluación perezosa, lo que significa que las expresiones no se evalúan hasta que sea necesario.
- Programación funcional: Lisp es un lenguaje funcional, lo que significa que trata las funciones como ciudadanos de primera clase y favorece el estilo de programación basado en la composición de funciones y la inmutabilidad de los datos.
- Macros: Lisp ofrece un poderoso sistema de macros que permite a los programadores extender el lenguaje escribiendo código que manipula el código Lisp directamente.
- Interactividad: Lisp es conocido por su capacidad para ser utilizado de manera interactiva, lo que facilita el desarrollo y la depuración de programas.

Aplicaciones:

Lisp ha sido utilizado en una amplia gama de aplicaciones, incluyendo:

- Desarrollo de software en inteligencia artificial y procesamiento del lenguaje natural.
- Sistemas expertos.
- Sistemas de simulación.
- Herramientas de desarrollo de software.
- Educación en informática.

Programación funcional vs. Programación Orientada a Objetos (POO):

- Enfoque en el estado: La programación funcional tiende a favorecer la inmutabilidad de los datos y evita el uso de variables mutables, mientras que la POO se centra en los objetos que encapsulan estado y comportamiento.
- Modularidad: La POO utiliza la encapsulación para agrupar datos y comportamiento relacionados en objetos, mientras que la programación funcional favorece la composición de funciones para construir sistemas modulares.
- Herencia vs. Composición: La POO se basa en la herencia para compartir comportamiento entre objetos, mientras que la programación funcional favorece la composición de funciones para reutilizar el código.

- Efectos secundarios: La programación funcional tiende a minimizar los efectos secundarios, lo que facilita el razonamiento sobre el código, mientras que en la POO los efectos secundarios son más comunes debido al uso de variables mutables y métodos con efectos colaterales.

b. Investigación corta sobre Java Collections Framework:

El Java Collections Framework (JCF) es una biblioteca de clases e interfaces en Java que proporciona implementaciones de estructuras de datos comunes, como listas, conjuntos y mapas. Está diseñado para ser eficiente, fácil de usar y flexible. La jerarquía de interfaces en el JCF establece un conjunto de comportamientos comunes para las diferentes estructuras de datos, lo que permite que las implementaciones sean intercambiables y se adapten fácilmente a las necesidades del proyecto.

La jerarquía de interfaces en el Java Collections Framework incluye:

- Collection: Es la interfaz raíz de la jerarquía de colecciones. Define métodos comunes para manipular grupos de elementos, como agregar, eliminar y consultar elementos.
- List: Extiende la interfaz Collection y representa una colección ordenada de elementos donde los elementos pueden estar duplicados. Permite el acceso a los elementos por índice.
- Set: También extiende la interfaz Collection pero representa una colección que no permite elementos duplicados. Los elementos en un conjunto no tienen un orden definido.
- Queue: Extiende la interfaz Collection y representa una colección diseñada para soportar operaciones de encolar y desencolar elementos.
- Map: Representa una colección de pares clave-valor, donde cada clave está asociada a un único valor. Las claves en un mapa no pueden estar duplicadas.

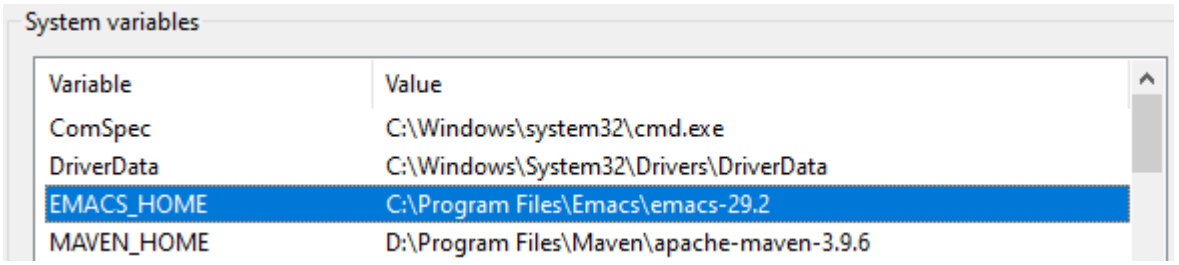
Las implementaciones concretas proporcionadas por el Java Collections Framework incluyen clases como ArrayList, LinkedList, HashSet, TreeMap, entre otras. Cada una de estas implementaciones tiene sus propias características y se adapta mejor a diferentes escenarios de uso.

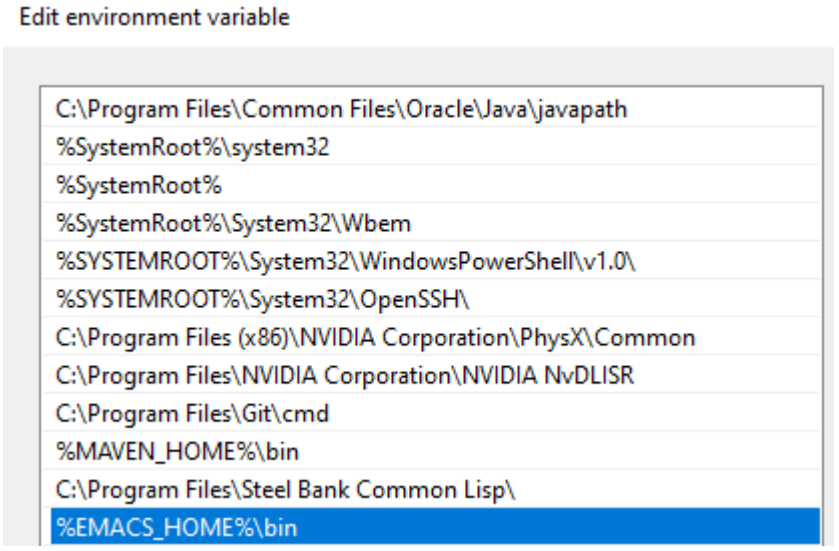
En un proyecto típico de Java, las implementaciones del Java Collections Framework se eligen según los requisitos específicos de la aplicación. Por ejemplo, si necesitas una colección ordenada que permita duplicados y un acceso eficiente por índice, puedes elegir ArrayList. Si necesitas una colección que garantice la ausencia de elementos duplicados, HashSet podría ser la elección adecuada. Para el proyecto, es posible que necesites usar varias implementaciones de la jerarquía de interfaces,

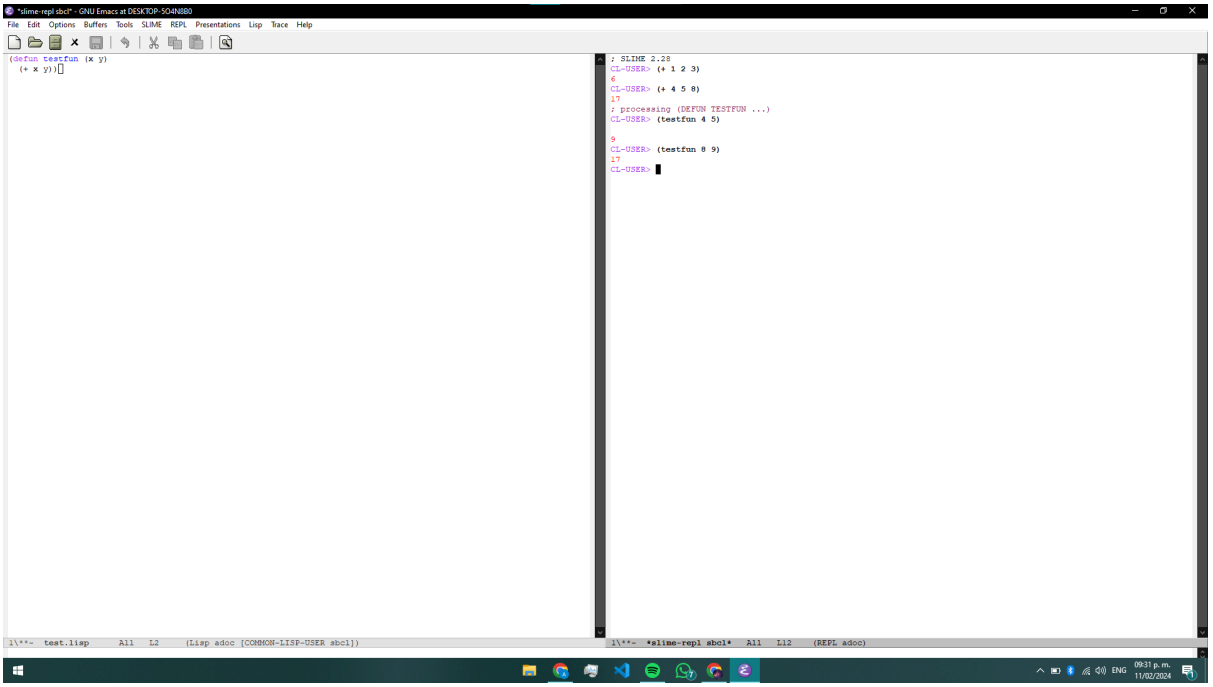
dependiendo de las estructuras de datos requeridas y las operaciones que se realizan sobre ellas.

Por ejemplo, si estás desarrollando un sistema de gestión de inventario, podrías usar un ArrayList para mantener una lista ordenada de productos, un HashMap para mapear códigos de producto a cantidades disponibles, y un HashSet para mantener una lista de categorías únicas de productos. Cada una de estas implementaciones se emplearía según las necesidades específicas de almacenamiento y acceso a los datos en diferentes partes del sistema.

c. Colocar un ambiente de trabajo:

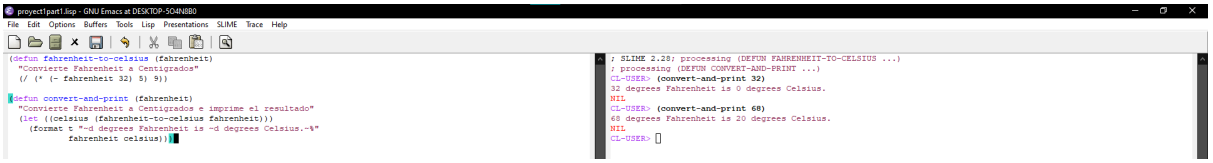






Archivo de prueba se puede encontrar en el github como “test.lisp”.

d. Ejecutar un programa sobre la conversión de Fahrenheit a Centígrados:



Archivo de programa se encuentra en el github como “proyect1part1.lisp”.

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

Archivo de programa se encuentra en el github como “proyect1part2.lisp”.

f. Inicio del diseño de su propio LISP, Diagramas UML: