Arquitectura Emmpresarial

Laboratorio 1
Introducción a MVN-GIT

Julián David Gutiérrez Vanegas

Agosto 2020

1. Introducción

En el siguiente artículo se busca presentar el modelo de un programa, desarrollado con Java y gestionado y construido utilizando Maven, cuyo objetivo es el cálculo de la media y la desviación estándar obtenidas de una serie de datos ingresados por el usuario en un formato de texto plano (.txt).

Para la elaboración de este programa, fue necesario implementar una estructura de datos llamada "lista doblemente enlazada". Esta permitió almacenar y mantener los datos ingresados por el usuario en una estructura que otorga la ventaja de: agregar, eliminar, obtener y remplazar un dato en tiempo lineal. Lo cual, permite que el programa entregue una respuesta de manera más rápida al usuario.

A continuación, empezaremos observando y explicando el modelo del programa con mayore detalle, seguido por un par de pruebas que comprueban su correcto funcionamiento, para finalizar entregando una conclusión sobre el desarrollo del programa.

2. Explicación del modelo

Dividiremos la explicación del modelo en dos partes:

- Primero, describiremos cómo está compuesta la estructura de datos utilizada, en este caso la lista doblemente enlazada
- Después, observaremos cómo está compuesto todo el programa. Cuál es la clase principal y como esta interactúa con las demás.

2.1. Implementación de la lista

Para la implementación de este programa se desarrolló una lista. Para esto, primero se implementó una interfaz que define qué funciones debe ofrecer una lista para el correcto desarrollo del programa. A continuación, se presenta el diseño de esta interfaz:

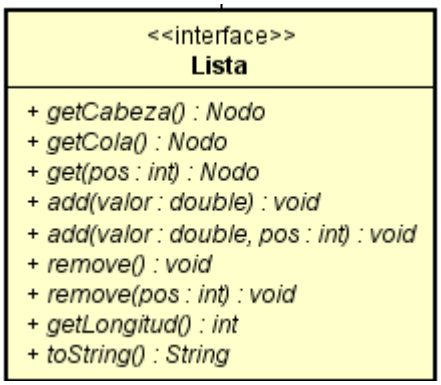

Figura 1: Interfaz de lista

Una vez se tuvo implementada la interfaz lista que define lo que se necesita para el desarrollo del programa, se implementa la clase "LinkedList"que es la lista que se utilizó para el desarrollo del programa. Esta clase, además de implementar los métodos definidos en la interfaz, tendrá los atributos de: cabeza (objeto de tipo nodo que es primera posición de la lista), cola (último nodo de la lista) y longitud (número de elementos que están en la lista). La conexión entre todos los elementos de la lista se hace gracias a los atributos "siguientez .ªnterior"(ambos de tipo nodo) que se encuentran en todos los elementos nodo que componen la lista. Además de estos dos atributos, todos los nodos tienen un atributo "valor", el cuál es un "double", para que los cálculos tengan una mayor precisión. A continuación, se presenta completamente la estructura "listaïmplementada para este programa:

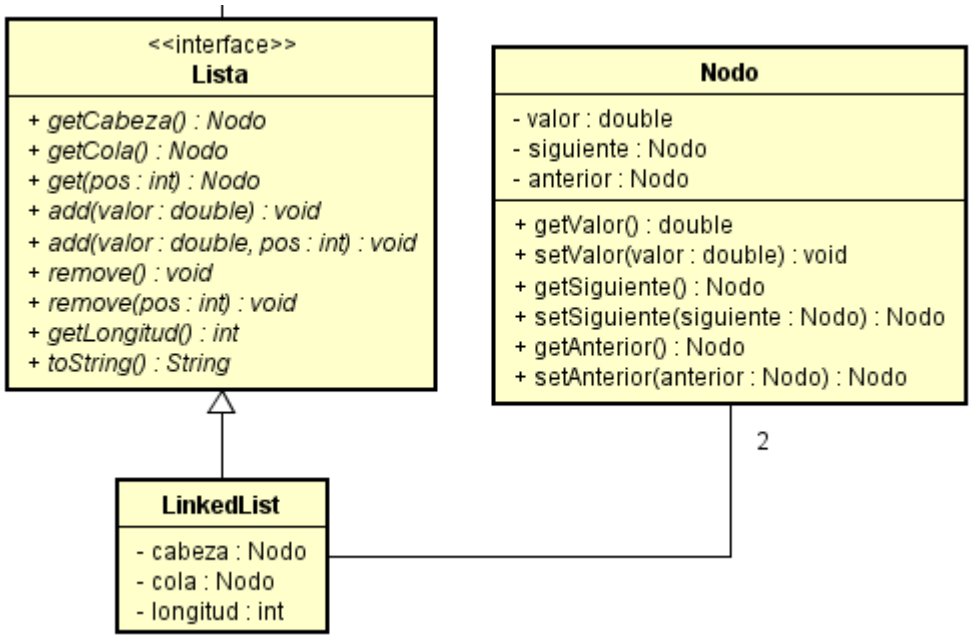

Figura 2: Implementacion de lista

Como podemos observar, este modelo, además de dar una mayor velocidad a la hora de administrar los datos que se ingresen, también es flexible. Esto permite que el programa, de ser necesario, utilice una implementación de "lista"distinta, que de tener cada uno de los métodos descritos en la interfaz bien implementados, no afectará el funcionamiento del programa.

2.2. Implementación de la clase principal

Una vez se tuvo implementada la lista que se va a usar para almacenar los datos ingresados por el usuario, se implementó la clase principal. En esta clase se leen los datos que se encuentran en el archivo ingresado por usuario, se almacenan y, finalmente, se ejecutan todos los cálculos necesarios para entregarle en pantalla al usuario la media y la desviación estándar obtenidas. Este programa está dividido en dos métodos: el principal "main", el cual recibe la ruta del archivo donde se encuentran los datos que se usarán para realizar los cálculos y luego muestra la información del resultado en pantalla, y el método çalcularMediaYDesviacion"donde se realizan todas las operaciones. A continuación, se presenta el diagrama de clases de todo el programa, donde se puede observar cómo interactúa esta clase principal con las demás:

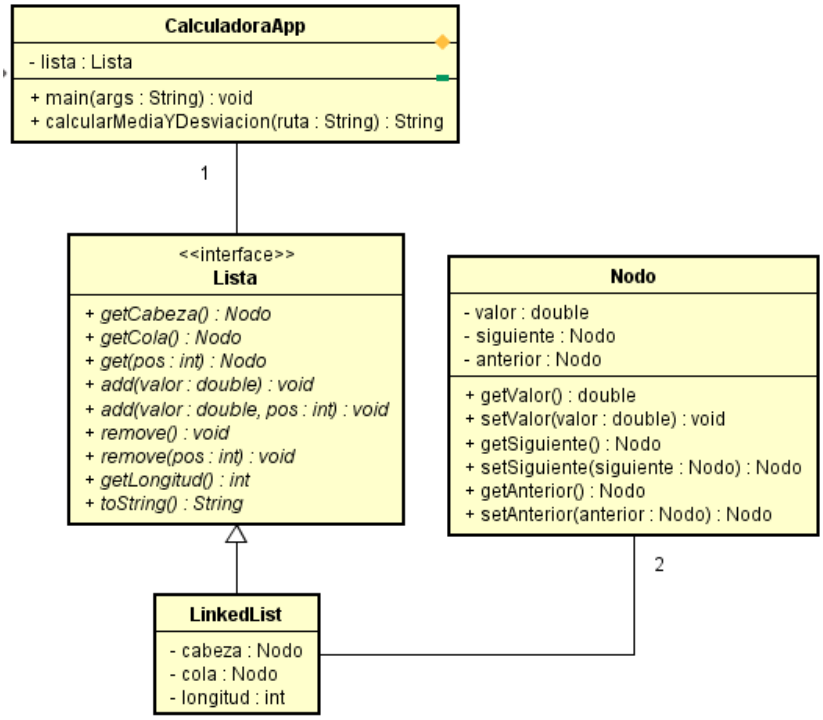

Figura 3: Diagrama de clases

3. Pruebas

Para verificar que el programa esté funcionando de manera correcta se realizaron dos pruebas: una con números enteros y otra con números decimales. A continuación, se listan los datos utilizados:

Prueba 1	Prueba 2
160	15.0
591	69.9
114	6.5
229	22.4
230	28.4
270	65.9
128	19.4
1657	198.7
624	38.8
1503	138.2

Cuadro 1: Valores de las pruebas

Con estos datos se calculó la media y desviación estándar utilizando este programa y se compararon estos resultados con los resultados que arrojó otro programa matemático:

Prueba	Valor Esperado		Valor Arrojado	
	Media	Des.Est.	Media	Des.Est.
Prueba 1	550.60	572.03	550.60	572.03
Prueba 2	60.32	62.26	60.32	62.26

Cuadro 2: Resultados

Como se puede observar, los resultados fueron los mismos, lo que significa que el programa está funcionando correctamente.

4. Conclusión

Para la implementación de una lista no es necesario tener una clase compleja. Esta implementación se puede dividir en dos clases, teniendo que la clase lista solo referencie a los nodos de los extremos de la lista, mientras que cado uno de los nodos referencia a sus vecinos. Esta implementación se vuelve más eficaz si los nodos referencian a dos vecinos (el anterior y el siguiente), en lugar de solo referenciar a uno.

5. Bibliografía

Germán Camaño, G. C. (s. f.). Estructuras de datos: listas enlazadas, pilas y colas. Recuperado 13 de agosto de 2020, de http://www.calcifer.org/documentos/librognome/glib-lists-queues.html