

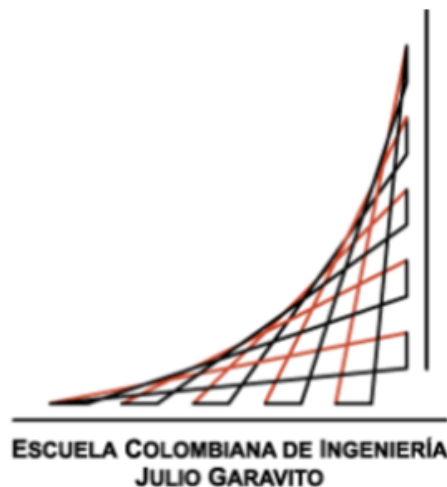
# Introducción a Sistemas Complejos, Java, Mvn Y Git

Daniel Felipe Walteros Trujillo

22 de Enero del 2020

**Profesor:**  
**Luis Daniel Benavides Navarro**

Arquitecturas Empresariales



# Tabla de Contenido

<b>1</b>	<b>Prerrequisitos</b>	<b>2</b>
<b>2</b>	<b>Introducción</b>	<b>2</b>
<b>3</b>	<b>Diseño</b>	<b>2</b>
3.1	Lista Encadenada . . . . .	3
<b>4</b>	<b>Pruebas</b>	<b>4</b>
<b>5</b>	<b>Conclusiones</b>	<b>4</b>
<b>6</b>	<b>Referencias</b>	<b>5</b>

# 1 Prerrequisitos

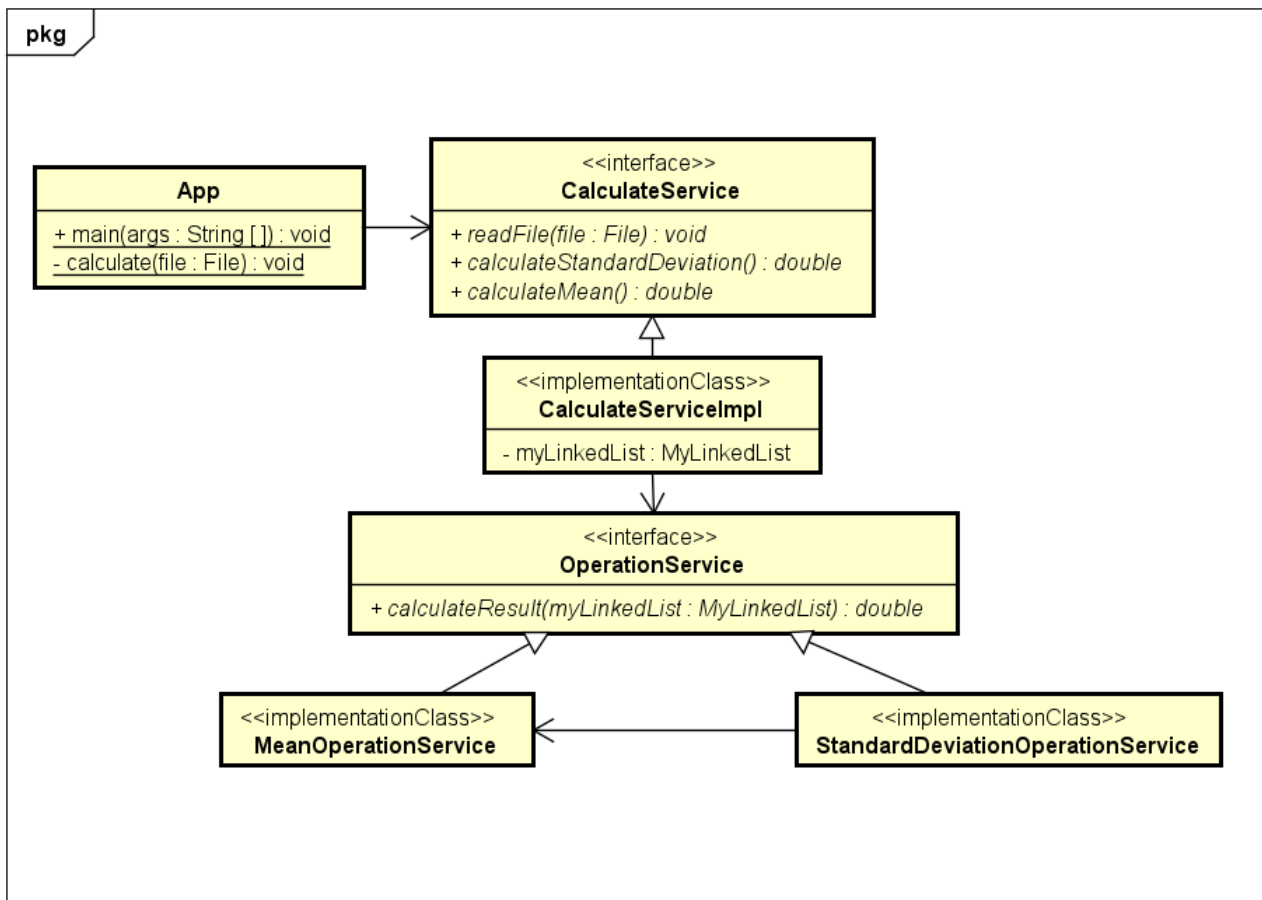
Para el desarrollo del programa se utilizó Maven como una herramienta para la gestión del ciclo de vida del software, el código fue desarrollado con el lenguaje de programación Java, por lo tanto para su ejecución se requiere:

- Java versión 8 o superior.
- Maven versión 3.5 o superior.

## 2 Introducción

En este trabajo, se desarrolló una aplicación que calcula el promedio y la desviación estándar de un conjunto de números reales, este conjunto es obtenido desde un archivo y para realizar los calculos se usa una Lista Encadenada de implementación propia que es compatible con la API de colecciones de Java [1].

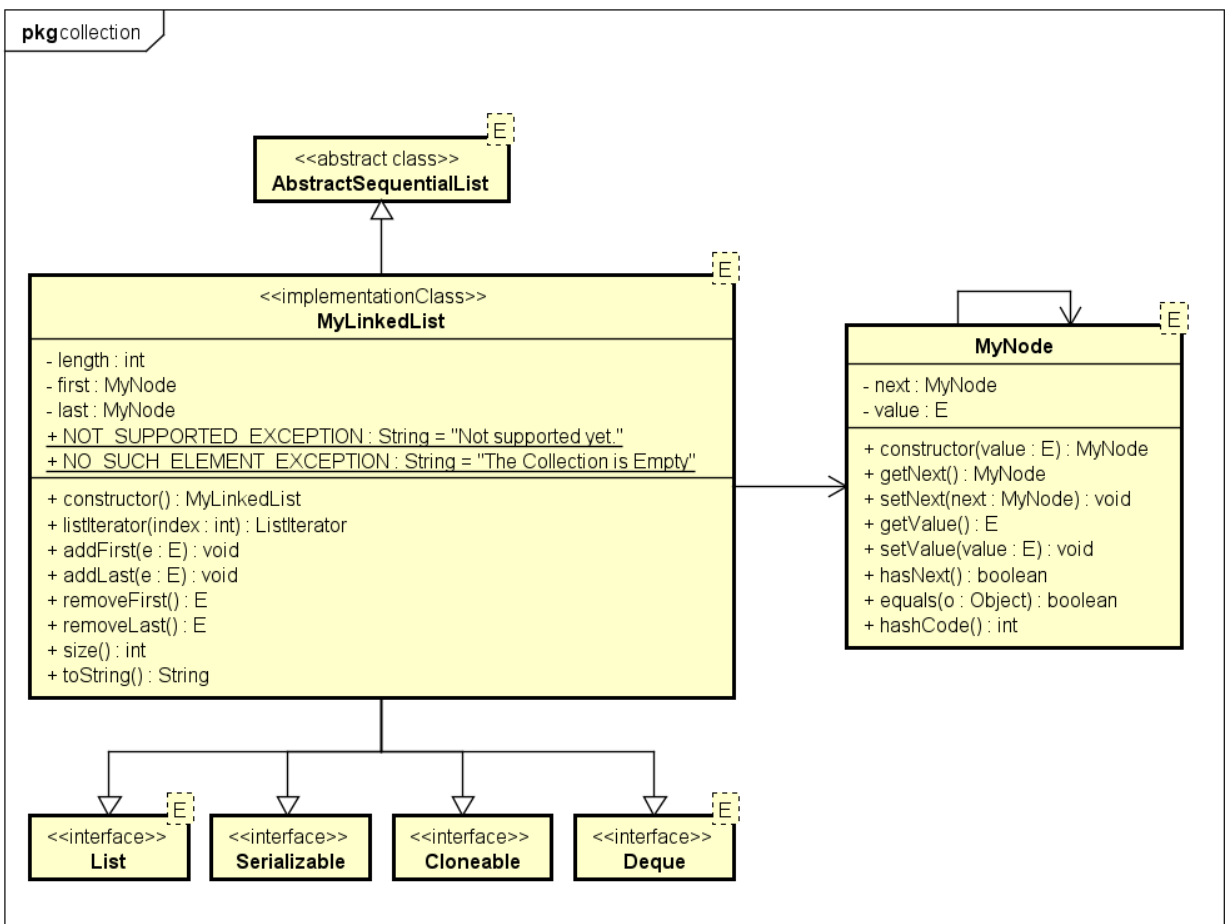
## 3 Diseño



El programa principal utiliza la interfaz CalculateService para realizar la lectura del archivo y las operaciones sobre la lista encadenada de implementación Propia. La implementación de esta interfaz utiliza la interfaz OperationService para realizar las operaciones de forma independiente de la lectura, lo que permite la reutilización de código en varias operaciones; un ejemplo de esto es la clase StandardDeviationOperationService que utiliza la clase Mean-OperationService para calcular el promedio del conjunto de npumeros, valor necesario para calcular la desviación estándar.

Otra ventaja de desacoplar las funcionalidades con interfaces, es que en el caso de que se quieran realizar otras implementaciones de las operaciones o de la lectura de archivos, basta con crear una clase que implemente la interfaz respectiva y asignarla en la clase principal App.

### 3.1 Lista Encadenada



La lista encadenada de implementación propia (`MyLinkedList`) extiende la clase `AbstractSequentialList` e implementa las interfaces `List`, `Deque`, `Cloneable` y `Serializable` para ser compatible con la API de colecciones de Java, cada elemento de esta colección es representado por la clase `MyNode`, el cual posee el valor asignado y una referencia al siguiente

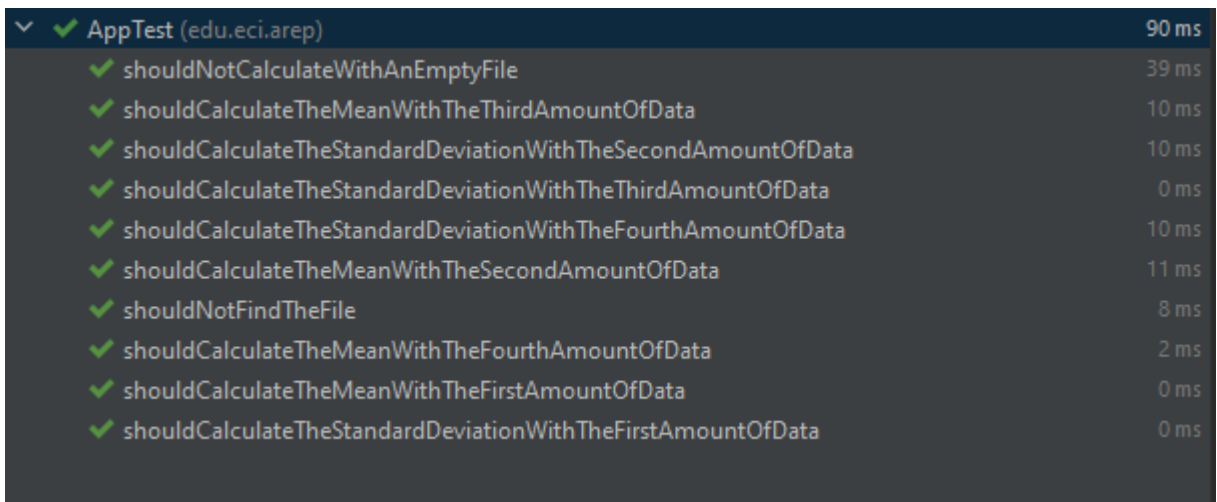
nodo de la lista, en caso de que el siguiente nodo sea nulo, se habrá recorrido toda la colección.

Las operaciones implementadas para esta colección (además de constructor vacío) fueron `listIterator`, `addFirst`, `addLast`, `removeFirst`, `removeLast` y `size`; todas las demás funcionalidades no fueron necesarias para realizar el programa y su uso retornara la excepción `UnsupportedOperationException`.

## 4 Pruebas

El programa fue probado con diez pruebas unitarias de JUnit donde se contemplaron los siguientes casos:

- Búsqueda de un archivo inexistente.
- Calculo con un archivo vacío.
- Calculo del promedio con cuatro conjuntos de información diferentes.
- Calculo de la desviación estándar con cuatro conjuntos de información diferentes.

A screenshot of a JUnit test runner interface showing the results of ten tests for a class named 'AppTest'. The interface has a dark theme. At the top, a summary line shows a green checkmark, the class name 'AppTest (edu.eci.arep)', and a total time of '90 ms'. Below this, a list of ten individual tests is shown, each with a green checkmark icon, the test name, and its execution time in milliseconds. All tests passed successfully.

✓ AppTest (edu.eci.arep)	90 ms
✓ shouldNotCalculateWithAnEmptyFile	39 ms
✓ shouldCalculateTheMeanWithTheThirdAmountOfData	10 ms
✓ shouldCalculateTheStandardDeviationWithTheSecondAmountOfData	10 ms
✓ shouldCalculateTheStandardDeviationWithTheThirdAmountOfData	0 ms
✓ shouldCalculateTheStandardDeviationWithTheFourthAmountOfData	10 ms
✓ shouldCalculateTheMeanWithTheSecondAmountOfData	11 ms
✓ shouldNotFindTheFile	8 ms
✓ shouldCalculateTheMeanWithTheFourthAmountOfData	2 ms
✓ shouldCalculateTheMeanWithTheFirstAmountOfData	0 ms
✓ shouldCalculateTheStandardDeviationWithTheFirstAmountOfData	0 ms

## 5 Conclusiones

- El programa calcula correctamente el promedio y la desviación estándar para todos los casos probados con un porcentaje de error de 0.01.
- El uso de interfaces para generalizar comportamientos dentro de la aplicación permite la extensión o cambio de código sin la necesidad de alterar múltiples archivos.

## 6 Referencias

- [1] Oracle. *Linked List Java*. URL: <https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html>. (entered: 22-01-2021).