**PRÁCTICA 24 | ÁRBOLES**

# Introducción

Volvemos al igual que en la práctica 22 a manejar expresiones aritméticas con árboles. Ahora con algo más de complejidad porque el árbol debe ser capaz de convertir una expresión matemática en un árbol de expresión, a diferencia de la práctica antepasada que se introdujo ésta misma de forma manual. Tengo la ventaja de ya haber creado una clase con este nombre **ArbolExpresion,** donde tengo pensado agregar la funcionalidad solicitada. Seguiré leyendo las instrucciones para conseguir transformar una expresión en árbol porque aún no las he entendido del todo bien, y en dado caso, identificar si necesito crear algo más.

# Desarrollo

Como dije en la introducción, me puse a leer más detenidamente las instrucciones de la práctica. Además de eso, conforme iba leyendo estaba haciendo un pseudocódigo que seguía los pasos que se indicaban. Finalmente siguiendo la lógica planteada, traté de dibujar el árbol que debería resultar de la expresión de ejemplo.

Después de hacer todo lo anteriormente dicho, comprendí completamente lo que se tenía que hacer y aparte ya tenía el análisis hecho. Para la parte del diseño demoré poco en hacerlo porque ya tenía una idea de lo que iba a codificar cuando estaba haciendo el análisis.

Finalmente me puse a programar todo. No demoré mucho en obtener buenos resultados de mi programa y confirmar lo que había planteado en mi análisis. Lo único que me demoró un poco fue una función que incluí en la clase **ExpresionAritmetica**;que, a partir de una cadena, extrae la variable más próxima ya sea que esté compuesta por una letra o varias. Lo que quería era que la función anteriormente mencionada quedara disponible para futuros usos y no se acoplara estrictamente a la clase “ArbolExpresion”, y el mejor lugar a mi parecer que encontré fue en la clase mencionada.

A continuación, se indican los archivos creados y modificados:

## ArbolExpresion.java (paquete ednolineal)

## ExpresionAritmetica (paquete matemáticas).

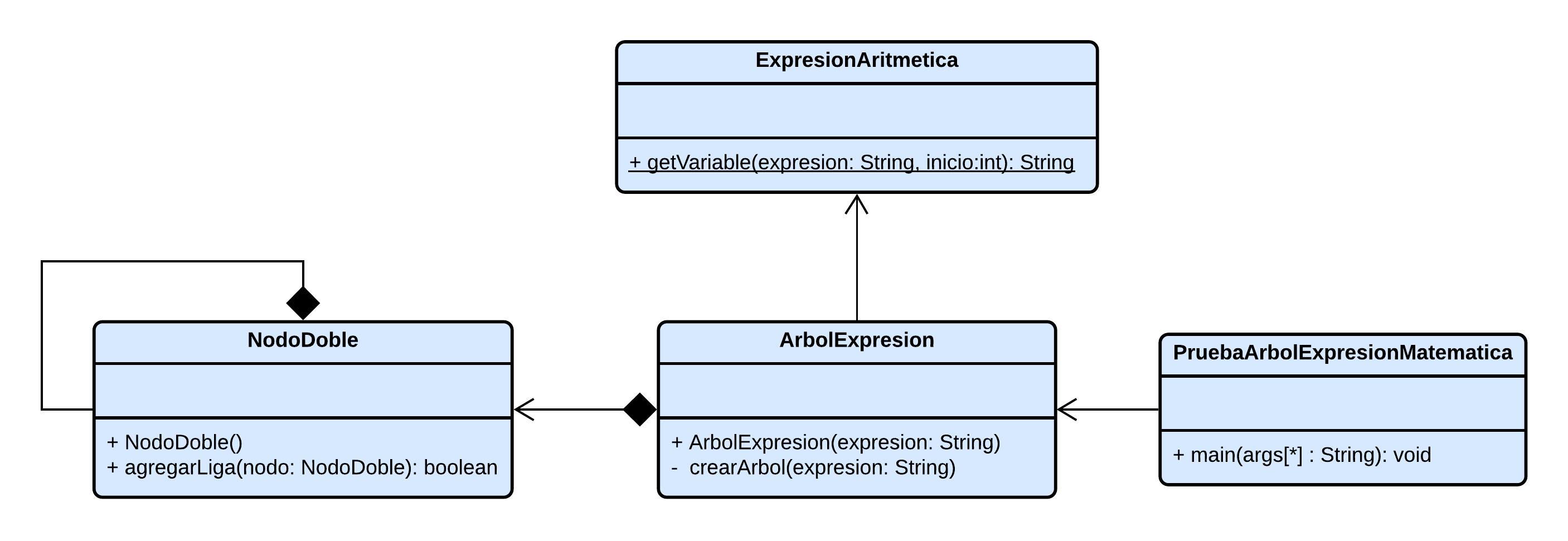
* ***NodoDoble.java (paquete catalogos).***

## PruebaArbolExpresionMatematica.java (paquete interfaces)

La clase anterior contiene las pruebas de esta práctica.

# Diagrama de clases

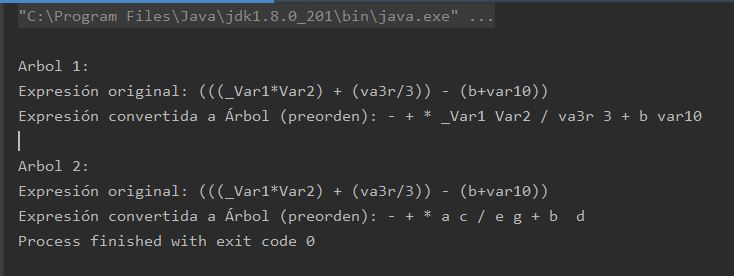
En el diagrama de clases se tiene una vista más cómoda de lo que se hizo en cada una de las clases mencionadas anteriormente. Cabe destacar que para las que ya existían en el proyecto solamente se ilustran los métodos o atributos que fueron agregados. En cambio, para las nuevas se muestra en su totalidad métodos y atributos que fueron incluidos.



**Si desea revisar la documentación de lo que fue agregado al proyecto vaya a las siguientes rutas:**

* **/docs/ednolineal/ArbolExpresion.html**
* **/docs/matematicas/ExpresionAritmetica.html**
* **/docs/catalogos/NodoDoble.html**
* **/docs/interfaces/PruebaArbolExpresionMatematica.html**

# Captura del programa funcionando



# Conclusiones

Este nuevo método para convertir una expresión matemática priorizada por paréntesis en un árbol de expresión para su posterior resolución, me parece más sencillo que el que convertía una cadena en notación polaca prefija o postfija para el mismo fin. Es interesante conocer otra forma en que una computadora puede almacenar un problema matemático para su posterior solución. La clase de estructura de datos ha sido complicada en numerosas ocasiones, pero también muy satisfactoria por todo el conocimiento adquirido.