Universidad Autónoma de Zacatecas

Unidad Académica de Ingeniería Eléctrica

Programa Académico de Ingeniería de Software.

Nombre de la Práctica Árboles.

Numero de Práctica 25

Nombre de la carrera Ingeniería de Software

Nombre de la materia Lab. Estructuras de Datos

Nombre del alumno Jesús Manuel Juárez Pasillas

Nombre del docente Aldonso Becerra Sánchez

Fecha: 25/10/2021

Práctica 25: Árboles.

Introducción:

Los arboles binarios de expresiones nos permiten almacenar una expresión en forma de un árbol, en donde cada uno de sus nodos es un operando o un operador. Esto nos permite sustituir variables de una forma más fácil, además de que se puede resolver la expresión de forma mas fácil.

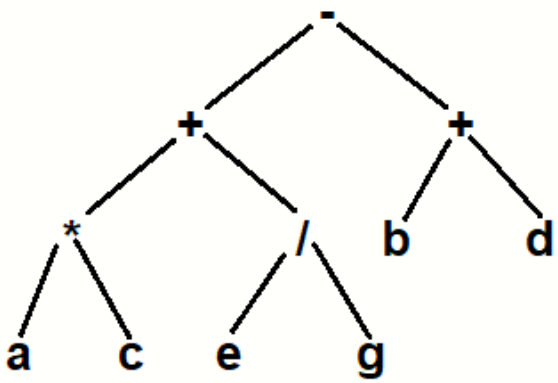
Desarrollo:

En esta practica se pide elaborar un método con el cual a partir de una expresión priorizada.

Las expresiones priorizadas son expresiones donde la prioridad se delimita por paréntesis y la expresión completa esta dentro de paréntesis, lo cual quedaría una expresión como la siguiente:

$$a * c + \frac{e}{g} - (b + d) \; ------> (\left((a * c) + \left(\frac{e}{g}\right)\right) - (b + d))$$

El método hará que la expresión priorizada resulte en el siguiente árbol:



Para crear este método se hizo una nueva clase llamada “ArbolExpresion” creada en el paquete “estructurasnolineales”, en esta clase se creó el método:

- Crea el árbol a partir de una expresión priorizada con paréntesis.
  public void expresionPriorizada(String expresion).

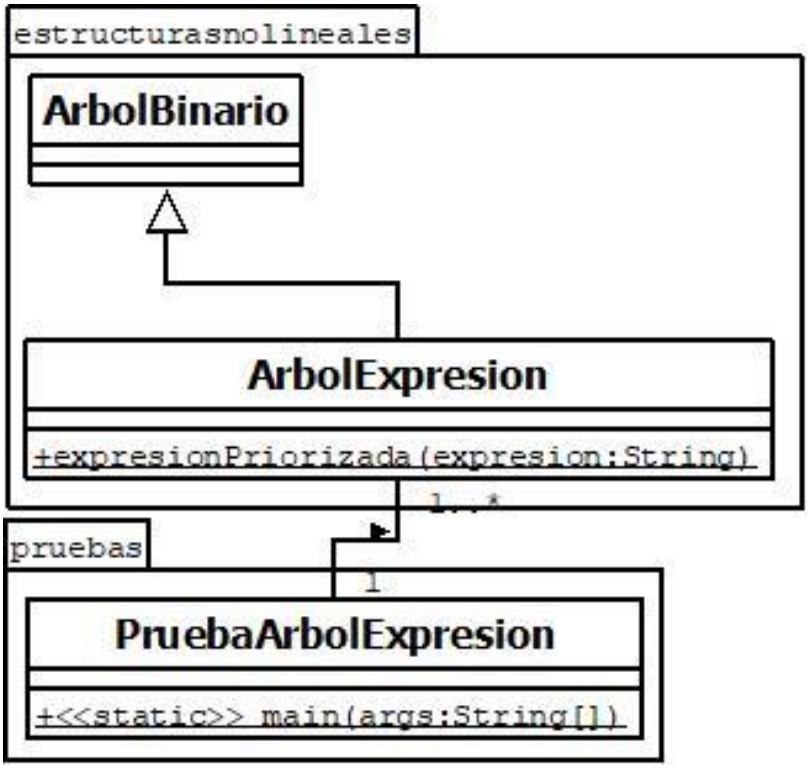
Este método no es recursivo.

Nota: Toda la documentación del proyecto esta agregada en la carpeta “doc” dentro de la carpeta del proyecto (“edylab_2021_25/doc”).

Capturas del programa funcionando:

```
Arbol expresion priorizada:
Inorden:
a * c + e / g - b + d
Postorden:
a c * e g / + b d + -
Preorden:
- + * a c / e g + b d
Process finished with exit code 0
```

Código agregado:



Pre-evaluación:

Pre-Evaluación para prácticas de Laboratorio de Estructuras de Datos	PRE-EVALUACIÓN DEL ALUMNO
CUMPLE CON LA FUNCIONALIDAD SOLICITADA.	Sí
DISPONE DE CÓDIGO AUTO-DOCUMENTADO.	Sí
DISPONE DE CÓDIGO DOCUMENTADO A NIVEL DE CLASE Y MÉTODO.	Sí
DISPONE DE INDENTACIÓN CORRECTA.	Sí
CUMPLE LA POO.	Sí
DISPONE DE UNA FORMA FÁCIL DE UTILIZAR EL PROGRAMA PARA EL USUARIO.	Sí
DISPONE DE UN REPORTE CON FORMATO IDC.	Sí
LA INFORMACIÓN DEL REPORTE ESTÁ LIBRE DE ERRORES DE ORTOGRAFÍA.	Sí
SE ENTREGÓ EN TIEMPO Y FORMA LA PRÁCTICA.	Sí
INCLUYE LA DOCUMENTACIÓN GENERADA CON JAVADOC.	Sí
INCLUYE EL CÓDIGO AGREGADO EN FORMATO UML.	Sí
INCLUYE LAS CAPTURAS DE PANTALLA DEL PROGRAMA FUNCIONANDO.	Sí
LA PRÁCTICA ESTÁ TOTALMENTE REALIZADA (ESPECIFIQUE EL PORCENTAJE COMPLETADO).	100%
Observaciones:	

Conclusión:

Los arboles binarios son de gran utilidad a la hora de separar expresiones aritméticas, ya que en cada nodo queda un elemento de esta. Y con esto es muy fácil identificar operandos y operadores.

Poder crear un árbol binario de una expresión priorizada es muy útil a la hora de crear un árbol para su manipulación, ya que no es necesario estar ingresando elemento por elemento.