Tarea 3: Expresiones Algebraicas

Profesores: Nelson Baloian
Patricio Poblete
Auxiliares: Manuel Cáceres
Sebastián Ferrada
Sergio Peñafiel

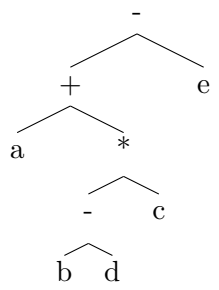
Fecha de Entrega: 2 de Mayo 23:59hrs

1 Introducción

Esta tarea consiste en implementar un tipo de dato abstracto (TDA) que permita representar fórmulas algebraicas, incluyendo las operaciones de crear e imprimir de estas expresiones.

Para implementarlo se le pide que utilice un "árbol binario de expresiones algebraicas" el cual tiene en sus hojas los operandos de la expresión y en los demás nodos los operadores.

A modo de ilustración la expresión "a + ((b - d) * c) - e" se representa con el siguiente árbol binario :



2 Implementación

2.1 El TDA Expresion

El TDA se debe llamar **Expresion** y provee al usuario las siguientes operaciones:

- Un constructor `Expresion(String formula)`: construye una representación de árbol para la expresión algebraica a partir de una fórmula que viene contenida en un String. La fórmula viene en notación postfijo, también conocida como notación polaca reversa. En la notación de postfijo primero se introducen los operandos y luego la operación respectiva. Por ejemplo: la expresión "a + b" en notación de postfijo corresponde a "a b +"; la expresión "a + ((b

- d) * c) - e" en notación postfijo corresponde a "a b d - c * + e -". Se utiliza también un operador "menos unario" que en notación de postfijo se representa como "_". Este operador toma solo un argumento. Por ejemplo, "-(1+2)" se representa como "1 2 + _".

- `String toSimpleString()`: genera una representación imprimible, en forma de un String conteniendo la fórmula en notación normal (infijo), usando paréntesis por cada operación realizada. Por ejemplo, para la fórmula "a b d - c * + e -" `toSimpleString` entrega "((a + ((b - d) * c)) - e)".

- `String toString()`: misma funcionalidad que `toSimpleString`, pero usando el **mínimo** número de paréntesis. Por ejemplo, para la fórmula "a b d - c * + e -" `toString` entrega "a+(b-d)*c-e".

2.2 Uso del TDA

En base a estas operaciones, implemente un transformador de expresiones algebraicas postfijo→infijo interactivo, que lea desde la entrada fórmulas en notación de postfijo, de a una por línea, y que responda escribiendo en la salida la misma fórmula traducida a notación de infijo con un paréntesis por operación, luego el signo igual ("=") y a continuación la fórmula en infijo con número mínimo de paréntesis.

2.3 Consideraciones y Guías

- Para simplificar el parsing de la entrada (fórmula descrita en el String formula), suponga que las variables son letras en $[a - z]$, separados por un espacio en blanco, y las operaciones permitidas son + (suma), - (resta), * (multiplicación), / (división) y _(menos unario). En caso de cualquier entrada que no cumpla con este formato, se debe mostrar en pantalla el error y abortar el programa (System.exit(-1)).

- Para almacenar la expresión algebraica, se debe utilizar como estructura de datos un árbol binario. La idea es que las variables son hojas y los operadores nodos, cuyos hijos izquierdo y derecho son los respectivos operandos (en caso del operador _ su hijo izquierdo es el operando correspondiente). El constructor debe parsear la fórmula en notación de postfijo y construir el árbol respectivo, usando una pila.

- Para los métodos `toSimpleString()` y `toString()`, se debe recorrer el árbol binario y generar la fórmula en notación normal (infijo) colocando paréntesis donde corresponda.

- Para minimizar el número de paréntesis tienen que comparar la prioridad del operador del padre con las de los operadores de los hijos en lo distintos niveles del árbol, para que expresiones como "a + b*c" no sean parentizadas pero "a* (b+c)" si. Además, considere reglas de asociatividad de manera que expresiones como "a+b+c" no sean parentizadas, pero "a-(b+c)" si.

3 Condiciones de Entrega

- Esta tarea debe ser resuelta en Java.
- Es obligatorio la entrega de un informe en formato pdf junto con su tarea (Ver siguiente sección).
- Esta tarea es de carácter individual, cualquier caso de copia se evaluará con la nota mínima.
- No olvide subir a U-cursos todos los archivos necesarios para que su tarea funcione correctamente.
- Debe subir los archivos de código fuente (*.java). Los archivos compilados (*.class) no serán evaluados.
- Cualquier duda respecto a la tarea puede ser consultada usando el foro del curso.
- **NO** se aceptarán atrasos.

4 Informe

El informe debe describir el trabajo realizado, la solución implementada, los resultados obtenidos y las conclusiones o interpretaciones de estos. Principalmente debe ser breve, describiendo cada uno de los puntos que a continuación se indican:

- **Portada:** Indicando número de la tarea, fecha, autor, email, código del curso, etc.
- **Introducción:** Descripción breve del problema y su solución.
- **Análisis del problema:** Exponga en detalle el problema, los supuestos que pretende ocupar, casos de borde y brevemente la metodología usada para resolverlo.
- **Solución del problema:** Indique claramente los pasos que siguió para llegar a la solución del problema. Muestre mediante figuras y ejemplos qué es lo que realiza su código. Evite copiar todo el código fuente en el informe, sin embargo, puede mostrar las partes relevantes de éste.
- **Modo de uso:** Explicando brevemente cualquier dato necesario para la compilación y ejecución de su programa.
- **Resultados:** Muestre ejemplos de entradas y salidas de su programa.
- **Discusión:** Discuta sobre implementaciones diferentes del TDA y compárelo con el uso de árboles binarios. Proponga nuevas operaciones que se podrían realizar sobre este TDA (las operaciones implementadas aquí fueron `toSimpleString` y `toString`).