Aula Prática 12

Resumo:
- Árvores Binárias de Procura;
- Árvores Binárias.

Exercício 12.1

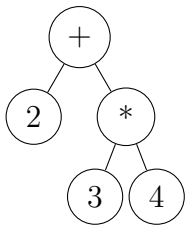
Implemente o módulo `AssociativeArray` com uma árvore binária de procura. Experimente o módulo com os exercícios da aula anterior.

Exercício 12.2

Podemos escrever expressões aritméticas três formas distintas consoante a posição do operador[1]:

- *Notação infixa*: `2 + 3` ou `2 + 3 * 4` ou `3 * (2 + 1) + (2 - 1)`
- *Notação prefixa*: `+ 2 3` ou `+ 2 * 3 4` ou `+ * 3 + 2 1 - 2 1`
- *Notação sufixa*: `2 3 +` ou `2 3 4 * +` ou `3 2 1 + * 2 1 - +`

Independentemente da notação utilizada, uma expressão aritmética pode ser representada por uma árvore binária em que os nós representam as (sub)expressões existentes (os números serão sempre folhas da árvore). Por exemplo a expressão (prefixa) `+ 2 * 3 4` é expressa pela árvore binária:



[1]No exercício 9.3 já utilizámos uma destas notações (sufixa[2]) como forma de simplificar o cálculo de expressões numéricas.

A geração desta árvore binária tendo como entrada uma expressão em notação *prefixa* é bastante simples (caso feita recursivamente):

```
createPrefix()
{
   if (in.hasNextDouble()) // next word is a number
   {
      // leaf tree with the number
   }
   else // next word is the operator
   {
      // tree with the form: operator leftExpression rightExpression
      // leftExpression and rightExpression can also be created with createPrefix
   }
}
```

a. Implemente um módulo – `ExpressionTree` – que cria uma árvore binária a partir de uma expressão em notação *prefixa* para os 4 operadores aritméticos elementares (`+`, `-`, `*` e `/`)[3];

b. Implemente um novo serviço no módulo – `printInfix` – que escreve a expressão (já lida) na notação *infixa*[4];

c. Implemente o módulo de uma forma robusta por forma a detectar expressões inválidas (note que a responsabilidade por uma expressão inválida é exterior ao programa pelo que deve fazer uso de uma aproximação defensiva);

d. Implemente um novo serviço no módulo – `eval` – que calcula o valor da expressão.

Nota: A criação da árvore binária com base numa expressão em notação *sufixa* é um pouco mais complexa. Se tiver essa curiosidade pode tentar fazer essa implementação.

[3]Considere que cada número e operador é uma palavra a ser lida no *standard input*

[4]Execute o programa em formato `jar` para ver o formato dessa escrita.