

Semana 5

Projecto 5 - Calculadora RPN



Cadeira de
Laboratório de Programação

2017

Laboratórios de Programação

Calculadora RPN

2017.03.14

Thibault Langlois

1 A notação Polaca inversa (RPN)

Existem duas maneiras de introduzir expressões aritméticas numa calculadora:

- Usando a notação “infix”, por exemplo:

$(3 + 5) * (5 - 3 * 4)$

onde os operandos aparecem à esquerda e à direita do operador. Esta notação tem a vantagem de ser “natural” pois é usada em matemática. Tem um defeito: obriga a definir a precedência entre os operadores. Por exemplo, se não houvesse precedência de operadores, a expressão $(5 - 3 * 4)$ deve ser interpretada como $((5 - 3) * 4) = 8$ ou $(5 - (3 * 4)) = -7$?

Um outro defeito é que, devido ao uso dos parêntesis, a análise da expressão (pelo computador) é mais difícil.

- Usando a notação Polaca inversa, a mesma expressão é calculada da seguinte forma:

$3\ 5\ +\ 5\ 3\ 4\ *\ -\ *$

Os operandos precedem os operadores. Esta notação tem o defeito de não ser muito legível por uma pessoa que não está habituada mas, em contrapartida, é fácil escrever programas para lidar com ela. No exemplo anterior, o mecanismo de avaliação é o seguinte:

- Os operandos são guardados à medida que são encontrados. O 3 e o 5 são os dois primeiros operandos encontrados e portanto guardados.
- Quando um operador é encontrado, no nosso exemplo o +, o número de operandos armazenados necessários são tirados, no exemplo os dois últimos operandos são tirados, a operação é efectuada e o resultado é guardado no lugar dos operandos. Aqui está a sequência dos valores guardados durante a avaliação da expressão $3\ 5\ +\ 5\ 3\ 4\ *\ -\ *$:

```
3 5
8
8 5 3 4
8 5 12
8 -7
-56
```

- O algoritmo constistete em:

1. ler cada *token* de entrada, considerando o espaço como separador.
2. caso o *token* corresponda a um número, colocá-lo numa pilha.
3. caso o *token* corresponda a uma operação:
 - (a) remover da pilha o número de operandos necessários,
 - (b) calcular o resultado da operação,
 - (c) colocar o resultado na pilha,

- repetir até não haver mais *tokens* na entrada.



2 Problema

O problema consiste em implementar uma calculadora que usa a notação RPN. A classe `Calc` deve possuir para além do constructor, um método `public String run(String input)` que recebe uma cadeia de caracteres que representa a entrada da calculadora. O valor retornado é uma `String` que corresponde ao valor no topo da pilha.

A classe `RunCalc` é responsável por receber e ler os comandos do teclado, linha a linha. Cada linha é executada usando o método `run` da classe `Calc`. O resultado obtido é mostrado no ecrã na classe `RunCalc`.

Exemplo de execução do programa:

```
> 4 5 * 7 /
2,8571
> 1 +
3,8571
> 5 -
-1,1429
> 12 # 2 *
-24,0000
> 17777 * #
426.648,0000
> e
Erro aritmetico: valor infinito.
> 1 +
-0,1429
>
```

Notas:

- Todos os valores são de tipo `double` (ou `Double`).
- Por omissão os valores são mostrados com 4 casas decimais.
- Em caso de erro, a pilha deve encontrar-se no estado anterior ao erro.

Os comandos que o programa deve implementar são:

- Os quatro operadores aritméticos.
- Funções: `tan`, `cotg`, `^`, `e` (e^x) e `ln`. O operador `^` calcula x^y onde x é o penúltimo número introduzido pelo utilizador e y o último.
- `#` é um operador que muda o sinal do último valor introduzido ou calculado.
- `dec` altera o número de casas decimais usado para mostrar os valores (ver a classe `NumberFormat`). Inicialmente o programa deve mostrar os valores com quatro casas decimais. Se o utilizador introduzir por exemplo `2 dec`, o programa passará a mostrar os valores com apenas duas casas decimais (como os valores presentes na pilha são de tipo `double` e portanto no exemplo guarda `2.0`, este comando usará a parte inteira do valor no topo da pilha, no caso `2`).

Em caso de erro aritmético (por exemplo divisão por 0) ou de introdução de comandos desconhecidos, o programa deve avisar o utilizador mas não deve terminar.

A classe `Calc` deve apanhar todas as exceções (deve usar as instruções `try ... catch`) ou fazer os testes necessários. Em caso de anomalia, o programa informa o utilizador (o método `run` deve retornar a mensagem de erro). Como a calculadora usa valores de tipo `double`, a exceção `ArithmeticException` não será lançada. Deve testar o resultado da divisão com o método `Double.isInfinite(x)`. Caso um valor negativo seja usado com o método `Math.log`, o valor retornado será `NaN` (Not A Number), a ocorrência pode ser detectada usando o método `Double.isNaN(x)`.

Se um operador é usado sem que haja o número de operandos suficientes na pilha, pode apanhar a exceção `EmptyStackException` para avisar o utilizador.

Duas opções podem ser adoptadas para a implementação da pilha da calculadora:

- caso saiba manipular classes genéricas, pode usar a classe `Stack` da API,
- em alternativa pode implementar a pilha usando um array de `double`. Neste caso o tamanho da pilha terá de ser igual a 10.

3 Entrega

Deve entregar pelo menos os seguintes ficheiros:

- **Calc.java** que contém a classe que implementa a calculadora.
- **RunCalc.java** que contém a função `main`, cria uma instância de **Calc** e corre a calculadora.

Antes de entregar o seu trabalho deve criar um ficheiro “zip” com um comando do tipo:

```
zip entrega.zip Calc.java RunCalc.java
```