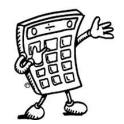
#### **DIFCUL**

## LabP 2013/2014

# Trabalho 3: Calculadora de Pilha

João Neto (adaptado por Ana Respício)



(todos os direitos reservados ao autor da imagem)

### Introdução

Quando escrevemos expressões matemáticas, estamos habituados à notação infixa, ou seja, a escrever os operadores entre os argumentos (e.g, 1+2). Neste trabalho escreveremos expressões com notação sufixa.

Na notação sufixa escrevemos os operadores e funções <u>depois</u> dos argumentos. Assim, 1+2 em notação infixa escreve-se 1 2 +.

Há duas vantagens na notação sufixa:

- (a) Não são precisos parêntesis. Por exemplo, (1+2)\*4 pode ser expresso por 1 2 + 4 \*
- (b) Podemos usar uma pilha de dados (*stack*) para avaliar expressões arbitrariamente complexas

Como se avaliam expressões numa pilha?

- 1) Avalia-se um número colocando-o na pilha
- 2) Avalia-se um operador binário (como o + e o \*) retirando dois elementos da pilha, fazendo a respectiva operação e colocando o resultado na pilha.

Por exemplo, vejamos a avaliação da expressão 1 2 + 4 \*:

Expressão por ler Conteúdo da Pilha (o topo está mais à direita)

Quando a expressão é inteiramente avaliada, a pilha contém no topo o resultado dessa avaliação.

Tente avaliar, para verificar se entendeu bem o processo, a expressão 4 1 2 + \* (o resultado é igualmente 12).

Nota: Existem linguagens de programação baseadas neste conceito de uso de pilhas para facilitar a avaliação de expressões. As mais conhecidas são as linguagens Forth - <a href="http://en.wikipedia.org/wiki/Forth\_(programming\_language">http://en.wikipedia.org/wiki/Forth\_(programming\_language)</a> - e PostScript - <a href="http://en.wikipedia.org/wiki/PostScript">http://en.wikipedia.org/wiki/PostScript</a>, esta última desenhada especialmente para aplicações gráficas.

### **Problema**

Pretende-se implementar uma calculadora que avalie expressões escritas na notação sufixa.

A calculadora trabalha somente com números inteiros. A calculadora pode usar variáveis de tipo inteiro. As variáveis são identificadas por apenas uma letra (cf. tabela seguinte para ver como se manipulam variáveis). A calculadora é *case sensitive*.

A calculadora suporta as seguintes operações:

Operador	Descrição	Exemplo			
+	adição	[ 4 2 -> [ 6			
-	subtracção	[ 4 2 -> [ 2			
*	multiplicação	[ 4 2 -> [			
%	resto da divisão inteira	[ 4 2 -> [			
/	quociente da divisão inteira	[ 4 2 -> [ 2			
	remover e imprimir o topo da pilha (mudando de linha)	[ 4 2 -> [ 4 (o 2 surge no ecrã)			
@x	remover o topo e guardá-lo na variável x (se a variável não existir, ela é criada e inicializada com este valor)	[ 4 2 -> [ 4 (x fica igual a 2)			
X	colocar o conteúdo da variável x (previamente inicializada) no topo da pilha (x permanece inalterada)	(se x=10) [ 4 2 -> [ 4 2 10			

e suporta as seguintes instruções:

dup	colocar o topo duas vezes na pilha	[	4	2 2	-> 2
swap	trocar o topo da pilha com o elemento sob ele	[	4 2	2 4	->
drop	apagar o topo da pilha	[	4 4	2	->

A calculadora reconhece números inteiros positivos e o zero. Para introduzir um número negativo deve-se usar a operação subtracção (por exemplo, 0 100 - coloca o valor - 100 no topo da pilha).

Todos os valores, variáveis, operadores e instruções devem ser separados uns dos outros por espaços.

Se uma linha começar com o *char* '-' deve ser assumida como um comentário e não ser avaliada. Ou seja, uma linha ou é um comentário ou é uma expressão (não há comentários na mesma linha de uma expressão).

Assuma que as expressões estão sintacticamente bem construídas e que os testes a aplicar nunca tentarão retirar valores de uma pilha vazia.

#### Classes a definir

Deverá criar uma classe Java Calculator. java que inclui:

- **public** Calculator(*BufferedWriter* out) -- construtor que estabelece a ligação entre o objecto e o fluxo de saída dado. O fluxo de saída será usado para escrever a informação impressa pelo operador '.'.
- **public void** processLine( *String* line ) -- método que processa uma linha de texto, modificando o estado actual da pilha de acordo com a expressão contida na *string* dada.

Tem disponível a classe <u>RunCalculator.java</u> que servirá de teste ao trabalho. Para instanciar a pilha da calculadora recorra ao interface para pilha <u>Stack.java</u> e a implementação de pilha com *array* que cresce <u>ArrayStack.java</u>, classes apresentadas em AED 2013/14 e disponíveis na respectiva página.

Estas classes não devem ser alteradas e devem ser submetidas juntamente com o vosso trabalho (cf. secção Entrega).

Pode usar os ficheiros data.txt (as linhas de texto com expressões a avaliar) e result.txt (os resultados esperados dessa avaliação) para testar o seu código, disponibilizados juntamente com o restante material deste trabalho.

#### **Entrega**

Deve criar o ficheiro projecto3.zip, contendo os ficheiros .java, com a instrução

zip projecto3.zip Calculator.java Stack.java ArrayStack.java RunCalculator.java

Este ficheiro deverá ser submetido através do formulário de submissão.