Departamento de Matemática da Universidade de Coimbra		
2015/2016	**Programação Orientada para os Objectos**	**Projecto 1**

Arithmetic Expressions Calculator

The reading and calculation of arithmetic expressions using infix notation is difficult to implement because it is an ambiguous notation. For example, how should we read and calculate the expression $x + y \times z$? The problem is the precedence of the operations that is implied but not explicitly written in the expression.

The polish notation[2], is an unambiguous alternative.

prefix notation (polish notation) $+ \times y\ z\ x$

postfix notation $x\ y\ z\ \ \times +$

In polish notation we have, always, an operator followed by two operands.

1. Implement, in *C++*, a class Integer Stack.
   Integer Stack = ({emptyStack,(Integer : Stack)}, {push, pop, top, empty})
   Consider the error situations.
2. Build a program in *C++* that reads a arithmetic expression, in Polish Notation, and calculates the result using a stack as a auxiliary data structure.

- Document your program. Internal and external documentation.
- The report (external documentation, max 5pp) should include the UML diagram of the class structure and a small user's manual. You should identify the group.
- You have to deliver (by electronic mail) one `zip` or `tar.gz` archive containing all the files related to the program (`Makefile, *.cpp, *.hpp`), and also the report (`PDF` format), up to the 24h00 hours of the project deadline.

Integer Stacks can be specified in the following form:

Elementos

$$\text{Stacks} = \begin{cases} \text{emptyStack}, & \text{empty stack} \\ \text{integer : Stack}, & \text{non-empty stack} \end{cases}$$

with ":" meaning concatenation.

Funções internas

$$\begin{array}{rccl}
\text{push :} & \mathit{Integer} \times \mathit{Stacks} & \longrightarrow & \mathit{Stacks} \\
 & (e,p) & \longmapsto & e : p \\
\\
\text{pop :} & \mathit{Stacks} & \longrightarrow & \mathit{Stacks} \cup \{\mathit{error}\} \\
 & p & \longmapsto & \begin{cases} p', & p = e : p' \\ \mathit{error}, & p = \text{emptyStack} \end{cases} \\
\\
\text{top :} & \mathit{Stacks} & \longrightarrow & \mathit{Integer} \cup \{\mathit{error}\} \\
 & p & \longmapsto & \begin{cases} e, & p = e : p' \\ \mathit{error}, & p = \text{emptyStack} \end{cases} \\
\\
\text{empty? :} & \mathit{Stacks} & \longrightarrow & \mathit{Bool} \\
 & p & \longmapsto & \begin{cases} \mathcal{F}, & p = e : p' \\ \mathcal{V}, & p = \text{emptyStack} \end{cases}
\end{array}$$

[2]Due to Polish mathematician Jan Łukasiewicz