Aula prática **#3**

Implementação e uso de Stacks:

1. Implemente a classe ArrayStack que tal como o nome indica providencia uma implementação do TAD Stack com arrays. O TAD é descrito na seguinte interface:

```
public interface Stack<E>{
        public void push(E o);
                public E top();
                public E pop();
                public int size();
                public boolean empty();
}
```

2. Como é do seu conhecimento, foi dado nas aulas teóricas um algoritmo que, fazendo uso duma stack, lhe permite fazer a avaliação de expressões em postfix.
   2.1. Use o algoritmo (no papel), para fazer a avaliação das expressões que se seguem, apresentando as operações sobre a stack, que são usadas em cada um dos casos. Por exemplo, se a expressão fôr: **12 34 + 5 ***, e a stack usada fôr **s**, deverá escrever:

```
s.push(12); s.push(34); int b=s.pop();
int a=s.pop(); s.push(a + b); s.push(5);
b=s.pop(); a= s.pop(); s.push(a*b) ; s.pop();
```

   2.1.1. **23 56- 3* 1 3 4/+/**
   2.1.2. **2 4 7 32/+ 7 5- 3*--**
   2.1.3. **2 6+ 4 7* 5-/ 6 9/* 4 9* 5 3/+-**

3. Pretende-se que avalie, se numa expressão com parêntesis, chavetas, colchetes, operadores e operandos, os parêntesis estão correctamente balanceados.
   3.1. Apresente em pseudo-código o algoritmo lhe permita concluir (ou não!) se uma está correctamente balanceada. A ideia básica do algoritmo é, fazendo uso duma Stack, ir avaliando os tokens da expressão da esquerda para a direita: se o símbolo lido for um parêntesis a abrir, ponha na stack, se for um parêntesis a fechar retire um símbolo da stack e "veja" se há concordância.

Aula prática **#3**

3.2. Implemente a classe ParentMatch que lhe permita ler do stdin um expressão e verifique se está ou não correctamente balanceada de parêntesis. É-lhe fornecido um esqueleto desta classe, que lhe permite implementar rapidamente o pedido. Use-a se quiser.