

Licenciatura em Tecnologias da Informação

2022/2023

Nesta primeira fase do projeto vão exercitar as seguintes matérias lecionadas em PCO: declaração de variáveis, atribuição de valores a variáveis, expressões, abstração procedimental (definição de métodos *static* e sua invocação), comandos condicionais, ciclos e utilização dos tipos de dados não primitivos `String` e `array`.

O *Columns* é um jogo de puzzle criado em 1989. Na sua versão original, o jogo usa uma área rectangular com 240 posições, distribuídas por 20 linhas com 12 colunas cada, na qual o jogador irá colocar peças. Cada peça consiste de 3 símbolos diferentes (cores diferentes no jogo original) que aparecem, uma de cada vez, no topo da área de jogo e caem até chegarem ao fundo da área ou chegarem ao topo de outras peças anteriormente colocadas.

Quando a peça “aterra”, se existirem 3 ou mais símbolos contíguos do mesmo tipo, quer na horizontal, na vertical ou na diagonal, então esses símbolos desaparecem. Peças que estejam colocadas em cima das que desaparecem, caem até atingirem o fundo da área de jogo ou até atingirem o topo de outras peças anteriormente colocadas. Se este reposicionamento colocar outras peças em posição de serem eliminadas, então o processo repete-se até não existirem mais peças que causem mais eliminações.

<http://www.youtube.com/watch?v=38xNl6BzIf0>

			%						
		\$							
		#							
	\$!	%	%	#	!	#	!	
	%	!	#	\$	%	#	\$	#	
\$	#	#	@	!	!	\$	%	#	

1

resultante de fazer a colocação da peça e as eventuais eliminações e reposições de acordo com o descrito acima, mas tendo só em conta a direção vertical.

A coluna de jogo vai ser representada por um *array* unidimensional (vetor) de caracteres; uma peça será representada por um *array* unidimensional de 3 caracteres. **ATENÇÃO:** O último elemento do vetor coluna representa o elemento “de baixo” da coluna. O mesmo para a peça.

São-vos fornecidas as seguintes duas classes:

- `TransformMethods`, onde devem colocar os métodos que vão implementar; esta classe já contém um método que invoca os métodos em falta – o resultado deste método `applyTransforms` é a coluna resultante de colocar uma dada peça numa dada coluna e aplicar as transformações necessárias;
- `PCOFase1` que faz várias invocações dos vários métodos a implementar e que podem usar para testar os vossos métodos.

Como poderão ver no método `applyTransforms` incluído na classe fornecida `TransformMethods`, é necessário implementar os métodos seguintes:

- a) `static char[] copyColumn(char[] col)`, que constrói e devolve uma cópia do vetor `col`;
- b) `static void placePiece(char[] col, char[] piece)` que modifica o vetor `col` “colocando-lhe por cima”, se for possível, os elementos do vetor `piece`; se `col` não tiver 3 ou mais elementos vazios, nada será modificado;
- c) `static boolean eliminate(char[] col)` que modifica o vetor `col`, substituindo todos os elementos que formam sequências de 3 ou mais caracteres iguais, pela constante **EMPTY**, e devolve `true` ou `false` conforme tenha feito alguma modificação ou não;
- d) `static void accomodate(char[] col)` que modifica o vetor `col`, acomodando todas as sequências contendo o carácter **EMPTY** de modo a que não fiquem “buracos” vazios na coluna (rever passos 2 e 3 das figuras acima);

Outros métodos privados podem ser criados, de modo a controlar a complexidade das vossas tarefas. Para verem os resultados dos vossos métodos, poderão executar a classe `PCOFase1`, comentando as instruções que não querem que sejam executadas.

Avaliação

Para efeitos de avaliação desta fase do projeto, os métodos a) e b) valem 50% e os métodos c) e d) valem 25% cada.

Reparem que a classe `PCOFase1` está preparada para ser possível testar somente os métodos a) e b) (*Nível 1*), testar esses dois e também o método c) (*Nível 2*) e finalmente testar todos os métodos (*Nível 3*).

Isto permite que os alunos que não consigam, por alguma razão, implementar todos os métodos, tenham hipótese de testar parte deles, desde que implementem pelo menos os métodos a) e b), ou os métodos a), b) e c). Basta para isso pôr em comentário, no método `main` da classe `PCOFase1`, as instruções correspondentes aos níveis que não desejam testar.

O que entregar?

Não há relatório a entregar porque o vosso *software* é a vossa documentação. Assim, têm que comentar condignamente todos os métodos da classe `TransformMethods`: incluir no início da classe um cabeçalho Javadoc com `@author` (número do grupo e nome e número dos alunos que compõem o grupo); para cada método definido, incluir a documentação Javadoc apropriada.

Apresentem código que siga as normas de codificação em Java aprendidas nas aulas, bem alinhado e legível.

Para entregar: Um ficheiro *zip* com a classe que compõe a vossa solução e com os ficheiros Javadoc que geraram a partir dela (relembrem como fazer isto no Guião do Eclipse).

O nome do ficheiro *zip* que contém o vosso trabalho deverá ter o formato `PCOxxx.zip` (onde `xxx` é o número do vosso grupo).

Como entregar o trabalho?

Através do Moodle de PCO. Às 23h55 do dia acordado para a entrega, 26 de Outubro, os trabalhos entregues serão recolhidos.

Atenção que ao entregar o trabalho está a comprometer-se com o seguinte:

- O trabalho entregue é atribuível única e exclusivamente aos elementos que constituem o seu grupo;
- Qualquer indício de plágio será investigado e poderá levar ao não aproveitamento dos elementos do grupo e conseqüente processo disciplinar.

Outputs esperados:

Pode ver os outputs esperados da execução do método `main` da classe `PCOFase1` no ficheiro `outputs.txt` fornecido.

Atenção que, por questões de não alongar muito o espaço ocupado para o output, as colunas são representadas na sua forma transposta, ou seja, “deitadas”.