

# Programação Orientada a Objetos

Pretende-se que o formando programe usando a linguagem JAVA, o jogo conhecido por **Connect-4 (ou quatro em Linha)**, seguindo o paradigma da programação orientada a objetos.

Para adquirir conhecimento das regras do jogo, sugere-se a visita do seguinte site:

[https://pt.wikipedia.org/wiki/Lig\\_4](https://pt.wikipedia.org/wiki/Lig_4)

De forma a orientar o formando nas boas práticas da programação orientada a objectos, de seguida apresentam-se as classes, atributos e alguns métodos a desenvolver pelo formando.

## Requisitos

### Enumerados

#### 1. Estado

Contem os vários estados que um jogo pode ter, e que são:

- ADECORRER
- TERMINADO\_EMPATE
- TERMINADO\_COM\_VENCEDOR

#### 2. Peça

Representa a cor das peças dos jogadores: vermelhas e azuis . Uma vez que o jogo será desenvolvido apenas em modo de texto, e não em modo gráfico, é requisito associar a cada elemento enumerado, o atributo **simbolo**, que representa o caracter das peças de cada jogador. Este atributo *simbolo*, deverá ser to tipo *char*.

O formando deverá considerar um elemento adicional, que representa uma posição vazia no tabuleiro do jogo. Ou seja, uma posição no tabuleiro, pode ter uma peça VERMELHA, AZUL ou VAZIA.

A correspondencia entre os enumerados e o atributo *simbolo* deverá ser a seguinte:

enumerado	simbolo
VERMELHA	'X'
AZUL	'O'
VAZIA	' '

Para além do construtor privado deste tipo enumerado, deve incluir também o

método ***getSimbolo()***.

## Classes

### 1. Classe Jogador

Atributos:

Nome	Tipo de Dados
<b>id</b>	int
<b>nome</b>	String
<b>nJogadas</b>	int
<b>peca</b>	Peca

Métodos a desenvolver: construtor, getters e setters, cujo nome siga a convenção. Exemplo: **getId()** para o getter do atributo **id**. Adicionalmente desenvolva também o método **toString**.

### 2. Classe Tabuleiro

Atributos:

Nome	Tipo de Dados
<b>nLinhas</b>	int
<b>nColunas</b>	int
<b>NUMEROEMLINHA</b>	int
<b>tabuleiro</b>	Peca[][]

Como pode verificar na tabela, o atributo **tabuleiro** deverá ser uma matriz do tipo **Peca**, e representa o tabuleiro do jogo.

A variável **NUMEROEMLINHA** representa o numero de peças que têm que estar seguidas, para se vencer um jogo. Ao declarar esta variável atribua o valor 4.

**Métodos.**

**Construtor: Tabuleiro(int nLinhas, int nColunas)**

Deverá ter como parametros o numero de linhas e de colunas do tabuleiro. É neste método que deverá criar o seu array (new...) atribuindo-o à variável **tabuleiro** (que consta da lista de atributos do objeto). Este método deverá chamar o método **inicializaTabuleiro()**, cuja especificação e logica encontra descritas de seguida.

**Método: inicializaTabuleiro()**

Este método é responsável por inserir em todas as posições da matriz **tabuleiro** a peça VAZIA.

Este método não tem parâmetros e deverá devolver *void*, ou seja, nada. Percorra todas as posições do tabuleiro com um ciclo *for*, e insira a PECA.VAZIA

**Método: boolean *poePeca*(Jogador jogador, int nCol)**

Este método representa a inserção de uma peça do jogador recebido no primeiro parametro, no coluna cujo numero é recebido no segundo parametro. A peça deverá ficar na primeira posição livre a contar de baixo do tabuleiro, ou seja, na primeira linha VAZIA, a contar do fim.

**Método: Peca[][] *getTabuleiro*()**

Este método é o getter standard para devolver o tabuleiro do jogo.

**Método: boolean *emLinha*(Peca peca)**

Depois de cada jogada este método é chamado para validar se o jogador fez um **4 em Linha**, ou seja, se possui 4 peças seguidas, numa qualquer linha do tabuleiro. O método apresenta o parametro *peca* (do ultimo jogador que jogou) para procurar em todas as linhas da matriz, as 4 peças seguidas iguais à reciba nesse parametro. Caso encontre, o método devolve *true*, caso contrário devolve *false*.

**Método: boolean *emColuna*(Peca peca)**

Igual ao anterior, mas para colunas.

**Método: boolean *emDiagonalDireita* (Peca peca)**

Igual ao anterior, mas para as diagonais direitas.

**Método: boolean *emDiagonalEsquerda* (Peca peca)**

Igual ao anterior, mas para as diagonais esquerdas.

**Método: boolean *existeVencedor*(Peca peca)**

Este método é chamado após cada jogada e deverá verificar se existe um vencedor. Para isso deve usar a seguinte lógica: caso algum dos métodos, *emLinha*, *emColuna*, *emDiagonalDireita* ou *emDiagonalEsquerda*, devolva *true*, então o método deverá devolver *true*, caso contrário devolve *false*.

**Método: boolean *empate*()**

Este método é chamado após cada jogada e deverá verificar se existe um empate no jogo. Um empate existe caso a sua matriz não tenha qualquer peça VAZIA. Nesse caso o método devolve *true*, caso contrário devolve *false*. Crie um *for* dentro de um *for*, para percorrer todas as posições da sua matriz tabuleiro, e verificar se existe alguma posição com a peça Peca.FAZIA. Caso exista o metodo deve devolver *false*, caso contrario *true*.

### **Método: Estado atualizaEstadoJogo(Peca peca)**

Este método é chamado após cada jogada e devolve o *estado* do jogo após essa jogada. Para isso chama o método *existeVencedor(peca)*. Caso exista, o método devolve o estado Estado.TERMINADO\_COM\_VENCEDOR. Dentro deste método deverá também fazer a chamada ao método *empate()*. Caso este devolva true, o método deverá devolver Estado.TERMINADO\_EMPATE. Caso contrário deverá devolver Estado.ADECORRER.

### **3. Classe Jogo**

#### **Atributos:**

Nome	Tipo de Dados
jogador1	Jogador
jogador2	Jogador
tabuleiro	Tabuleiro
estado	Estado

#### **Métodos.**

#### **Construtor: public Jogo(int nLinhas, int nColunas)**

Este construtor só tem que fazer a atribuição dos parametros recebidos aos respetivos atributos. Adicionalmente deverá atribuir ao **estado** o valor Estado.ADECORRER.

**Métodos:** getters e setters que sigam os nomes convencionados, para os atributos *jogador1*, *jogador2* e *estado*. Getter para o *tabuleiro*.

#### **Método: void atualizaEstadoJogo(Peca peca)**

Este metodo deverá atualizar o atributo *estado*, que pode ver na tabela de atributos, com o valor retornado pelo método atualizaEstadoJogo(peca) do objeto *tabuleiro*, também visível na tabela de atributos. Este método deverá ter apenas uma única linha de código.

#### **Método: boolean executaJogada(int idJogador, int nColuna)**

Lógica do método: caso o *idJogador* seja 1 o método deverá atribuir a uma nova variavel jogador (criada localmente neste método), o *jogador1* (atrituto do objeto jogo). Se não, o método deverá atribuir a essa nova variável o *jogador2*. Depois deverá chamar o metodo *poePeca* do objeto *tabuleiro*, passando-lhe os argumentos, *jogador* e *nColuna* (recebidos nos proprios parametros deste método). Ou seja, estamos a solicitar ao objeto tabuleiro, para por a peça do jogador na coluna especificada no parametro. O valor booleano retornado pelo método deverá ser guardado numa nova variavel declarada internamente neste metodo. Se o metodo *poePeca*, retornar true, então este metodo deverá chamar o metodo *atualizaEstadoJogo*, passando-lhe

como argumento a peça do jogador (use o respectivo getter do objeto jogador).  
No final o método deverá retornar o valor da variável booleana.

#### 4. Main

A classe main é responsável pela interação com os jogadores.

##### Exemplo de uma execução do programa.

##### Início do Programa

Indique o nome do jogador 1:

Ana

Indique a cor da peça: Vermelho (V) ou Azul(A)

V

Indique o nome do jogador 2:

João

	c1	c2	c3	c4	c5	c6
--	--	--	--	--	--	--

##### Exemplo de jogada da jogadora Ana na coluna 3

Ana, indique pf o nº da coluna onde quer jogar:

3

	c1		c2		c3		c4		c5		c6	
--		--		--		--		--		--		--
					X							

Exemplo de jogada do jogador João na coluna 1

João, indique pf o nº da coluna onde quer jogar:

1

	c1		c2		c3		c4		c5		c6	
--		--		--		--		--		--		--
	O				X							

Exemplo de final do Jogo com vitoria da jogadora Ana.

Ana, indique pf o nº da coluna onde quer jogar:

5

	c1	c2	c3	c4	c5	c6
	X					X
	O				X	O
	O			X	X	O
	O	O	X	X	X	O

Parabéns Ana. Você é o(a) grande vencedor(a)!

Quer jogar outra vez? (s/n)

Fim.

