



# Programação Estruturada

## Aula 13 - Desenvolvendo um jogo da velha



Material Didático do Instituto Metrópole Digital - IMD  
Versão 3.0 - Todos os Direitos reservados

## Apresentação

Na aula anterior, encerramos praticamente todo o conteúdo teórico de nossa disciplina. Vimos, ao longo de toda a disciplina de Programação Estruturada, como utilizar recursos disponíveis na linguagem de programação Java. Durante o nosso estudo, você teve a oportunidade de exercitar todos os conteúdos abordados de forma prática, visualizando o funcionamento de programas simples ou de alguns mais complicados, como o jogo do labirinto.

Nesta aula, vamos aplicar o conhecimento que adquirimos no estudo de nossa disciplina de uma forma interessante e divertida. Vamos desenvolver um jogo da velha. Desenvolveremos, ainda nesta aula, o jogo básico com instanciamento de jogadores e das posições de marcação no tabuleiro. Ao longo da implementação, você verá os trechos de código para cada funcionalidade e poderá implementar você mesmo o que aprender.



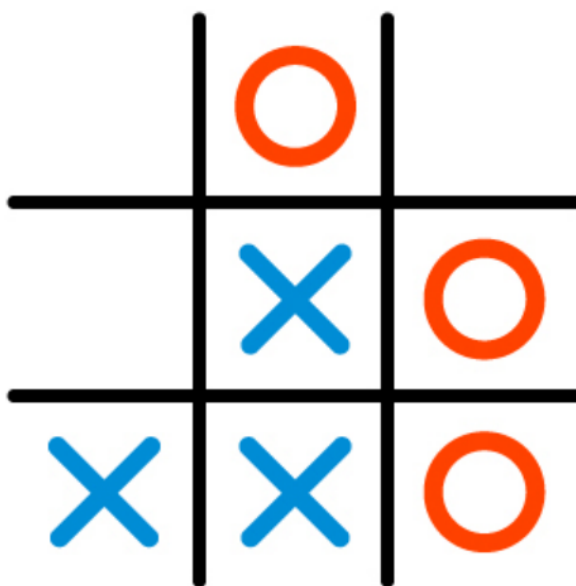
### Vídeo 01 - Apresentação

## Objetivos

- Desenvolver as funcionalidades mais básicas de um jogo da velha.
- Identificar, na implementação do jogo, o uso de conceitos aprendidos em aulas anteriores.

## 1. Jogo da Velha

Você já deve conhecer o jogo da velha. Trata-se de um jogo em turnos alternados entre dois jogadores que se passa em um tabuleiro de nove posições (3 x 3). No jogo, cada jogador, em sua vez, faz uma marcação em uma das posições do tabuleiro, sendo essa marcação o símbolo "X" ou o símbolo "O". Veja o tabuleiro do jogo da velha na Figura 1.



**Figura 1** - Tabuleiro do jogo da velha

A marcação, no jogo da velha, é feita de forma alternada, até que todo o tabuleiro seja preenchido (nesse caso, ocorre empate) ou até que, antes disso, um dos jogadores complete uma sequência de três símbolos em uma linha, coluna ou diagonal. Vence o jogador que completar primeiro a sequência. Na Figura 2, você pode ver algumas situações de vitória do jogo da velha.



**Figura 2** - Situações de vitória no jogo da velha

Você já deve ter jogado muitas vezes esse jogo. Tradicionalmente, para jogar, precisamos apenas de papel, lápis e dois jogadores dispostos a competir.

Mas e para implementar esse joguinho no computador em uma linguagem de programação? Como fazer?

Antes de começar a sair por aí escrevendo um monte de código, precisamos primeiro estruturar o que realmente queremos fazer. Quais são os elementos do jogo da velha? Dois jogadores e um tabuleiro de nove posições (3 x 3). Os requisitos para o jogo, dessa forma, são as variáveis para os jogadores, o desenho do tabuleiro e a lógica das jogadas.

Para isso, precisamos declarar variáveis para armazenar os valores que serão utilizados em todo o jogo, criar uma matriz de 3 x 3, para desenhar o tabuleiro e diversas rotinas com utilidades específicas, que serão chamadas por uma rotina principal, que executará o jogo. Vamos implementar? Mãos à obra!

## 2. Declaração das Variáveis

Pois bem, depois da inclusão das bibliotecas que iremos utilizar no jogo, precisamos declarar uma classe com as variáveis que armazenarão os valores referentes aos jogadores e ao tabuleiro. Veja:

```
1 public class JogoDaVelha {  
2     private static int jog;  
3     private static int[][] casa = new int[3][3];  
4     private static int linha, coluna, win;  
5     private static Scanner leitor = new Scanner(System.in);  
6 }
```

A variável *jog* do tipo `int` é declarada como variável estática dentro da classe *JogoDaVelha* para armazenar o jogador da vez. A matriz *casa*, de inteiros de 3 x 3 posições, é declarada para “desenhar” o tabuleiro para que possa ser visualizado pelos jogadores. Além disso, a matriz é sempre percorrida, a fim de verificar se há sequência de símbolos iguais nas suas linhas, colunas ou diagonais (sabemos que quando isso ocorre o jogo deve terminar e o vencedor é anunciado).

As variáveis do tipo `int` *linha* e *coluna* são declaradas para que seja feita a verificação dos valores escolhidos pelo usuário, na hora de definir em que campo do tabuleiro irá fazer sua marcação. Se o jogador escolher um campo inválido, uma mensagem de aviso será exibida (veremos isso daqui a pouco).

A variável *win*, também do tipo `int`, é declarada para armazenar o vencedor do jogo. Através do valor armazenado nessa variável, poderemos fazer a verificação de quem foi o vencedor. Por fim, temos a declaração de um leitor do teclado para ser utilizado ao longo do programa.

### 3. Desenhando o Tabuleiro

Após a declaração das variáveis globais, para armazenar dados referentes aos jogadores e ao tabuleiro, precisamos “desenhar” o tabuleiro. O procedimento *desenha*, representada a seguir, faz isso. Veja:

```
1 public static void desenha(int x, int y) {
2     if (casa[x][y] == 1) {
3         // campo marcado pelo jogador 1 aparece com "X"
4         System.out.print("X");
5     } else if (casa[x][y] == 2) {
6         // campo marcado pelo jogador 2 aparece com "O"
7         System.out.print("O");
8     } else {
9         // campo não marcado aparece em branco (" ")
10        System.out.print(" ");
11    }
12 }
```

Você viu que declaramos uma matriz de inteiros para construir o tabuleiro. Mas, no jogo da velha, não utilizamos números nas marcações. Utilizamos os símbolos "X" e "O". Na rotina *desenha*, dizemos basicamente que no campo de valor 1 da matriz que declaramos (escolhido pelo jogador 1) será marcado o símbolo "X" e no campo de valor 2 (escolhido pelo jogador 2) será marcado o símbolo "O". Note o uso de `System.out.print` ao invés de `System.out.println`, já que este último realiza uma quebra de linha, o que não é desejado para o procedimento *desenha*.

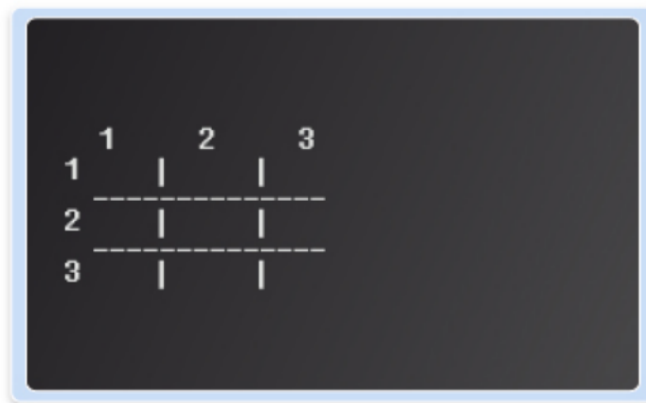
É importante determinar, no código, que os campos que não estiverem marcados ainda (ou seja, contiverem o valor inicial zero) devem ser exibidos com o caractere espaço em branco. Isso está sendo feito através do último comando *e/se* no código, que imprime um caractere em branco para qualquer valor diferente de 1 ou 2.

A rotina "desenha" é do tipo *void*, pois não retornará valor após a sua execução e recebe como argumentos dois valores inteiros, que são referentes à posição da matriz que se quer desenhar na tela.

Você já viu também que uma função pode ser chamada pela rotina *main* ou por outra função. É o que ocorre no procedimento *jogo*, apresentado a seguir, que chama o procedimento *desenha* para que o tabuleiro do jogo seja efetivamente exibido na tela. Veja:

```
1 public static void jogo() {
2     // aqui,são mostrados os números das colunas do tabuleiro
3     System.out.print("\n 1  2  3\n");
4     // aqui é mostrado o número da primeira linha
5     System.out.print("1 ");
6     // aqui é exibido o campo que cruza a linha 1 com a coluna 1
7     desenha(0, 0);
8     // caractere de divisão entre dois campos
9     System.out.print(" | ");
10    // aqui é exibido o campo que cruza a linha 1 com a coluna 2
11    desenha(0, 1);
12    // caractere de divisão entre dois campos
13    System.out.print(" | ");
14    // aqui é exibido o campo que cruza a linha 1 com a coluna 3
15    desenha(0, 2);
16    // desenha linha horizontal e mostra número da linha 2
17    System.out.print("\n ----- \n2 ");
18    // aqui é exibido o campo que cruza a linha 2 com a coluna 1
19    desenha(1, 0);
20    // caractere de divisão entre dois campos
21    System.out.print(" | ");
22    // aqui é exibido o campo que cruza a linha 2 com a coluna 2
23    desenha(1, 1);
24    // caractere de divisão entre dois campos
25    System.out.print(" | ");
26    // aqui é exibido o campo que cruza a linha 2 com a coluna 3
27    desenha(1, 2);
28    // desenha linha horizontal e mostra número da linha 3
29    System.out.print("\n ----- \n3 ");
30    // aqui é exibido o campo que cruza a linha 3 com a coluna 1
31    desenha(2, 0);
32    // caractere de divisão entre dois campos
33    System.out.print(" | ");
34    // aqui é exibido o campo que cruza a linha 3 com a coluna 2
35    desenha(2, 1);
36    // caractere de divisão entre dois campos
37    System.out.print(" | ");
38    // aqui é exibido o campo que cruza a linha 3 com a coluna 3
39    desenha(2, 2);
40 }
```

Veja que na linha que nomeamos “1”, são “desenhados” os campos (0,0), (0,1) e (0,2), separados pelo caractere “|”. Na linha 2, são “desenhados” os campos (1,0), (1,1) e (1,2), também separados por “|”. Na linha 3, por fim, temos os campos (2,0), (2,1) e (2,2). Veja os comentários no trecho de código exibido. Dessa forma, quando o jogo é carregado, o tabuleiro é exibido conforme a Figura 3.



	1	2	3
1			
2			
3			

**Figura 3** - abuleiro “desenhado” pela rotina *desenha*

Veja que, na Figura 3, o tabuleiro é mostrado com os números das colunas acima e os números das linhas ao lado. As linhas desenhadas para separar os campos na vertical são os caracteres “|” e as linhas horizontais foram desenhadas no momento da inserção dos números das linhas. Por exemplo, após completar o desenho da linha 1, temos `“printf(“\n ----- \n2 ”);”` que dá uma quebra de linha (causada pelo caractere especial “\n”), desenha a linha horizontal e dá outra quebra de linha para, então, exibir o número da linha seguinte.



**Vídeo 02** - imprime Tabuleiro

## Atividade 01

1. Além do jogo da velha, existem inúmeros outros jogos de tabuleiro que poderiam ser representados em Java através do uso de matrizes. Que tal desenharmos o tabuleiro de um desses jogos com o que vimos até aqui?

Sabendo que o tabuleiro de um jogo de damas tem 64 posições (8 x 8), desenha o tabuleiro para esse jogo, utilizando a mesma metodologia que acabamos de utilizar para o jogo da velha.



## 4. Fazendo e Validando as Marcações

Já desenhamos o tabuleiro. Como fazer, então, para definir de que forma serão feitas as marcações de cada jogador?

Vimos, há pouco, que precisamos definir uma variável (*jog*) para armazenar o jogador da vez.

Com base nessa variável é que o computador vai definir de quem é a vez e o símbolo que será marcado. A função *jogar*, cujo código completo você verá mais adiante, recebe como argumento uma variável do tipo `int`, a que chamamos *jogador*.

O programa lê o argumento e define, a partir dele, quem é o jogador da vez para, em seguida, informar isso aos usuários. Veja o trecho de código a seguir.

```
1 public static void jogar (int jogador) {  
2     if (jogador == 1) {  
3         jog = 1;  
4     } else {  
5         jog = 2;  
6     }  
7 }
```

O programa recebe o argumento no procedimento *jogar* e, caso o valor da variável *jogador* seja 1, esse valor é armazenado na variável *jog*. O mesmo ocorre se o valor da variável for 2. Esse valor é determinado na rotina principal (*main*, que veremos mais adiante) do programa, quando o procedimento *jogar* é “chamado”.

A variável *jog* será utilizada na mensagem que informa ao usuário de quem é a vez.

```
1 System.out.println("\n\n Vez do Jogador " + jog);
```

Bom, já sabemos como mostrar de quem é a vez de jogar. Mas ainda temos que determinar como serão feitas as marcações no tabuleiro. Precisamos, para isso, oferecer as opções para que o jogador escolha em

que campo do tabuleiro (em que linha e coluna da matriz) irá fazer a sua marcação.

As variáveis (*linha* e *coluna*) do tipo *int* que declaramos junto com a variável *jog* lá no início serão utilizadas nesse momento, para armazenar os valores das linhas e colunas digitados pelo jogador. Podemos fazer isso acrescentando ao código anterior a solicitação para que o jogador da vez escolha a linha e a coluna em que fará sua marcação. Veja:

```
1 System.out.print("Escolha a Linha (1,2,3):");
2 // lendo a linha escolhida
3 linha = leitor.nextInt();
4 System.out.print("Escolha a Coluna (1,2,3):");
5 // lendo a coluna escolhida
6 coluna = leitor.nextInt();
```

Mas não é só isso. Precisamos validar a escolha do jogador. Se a linha ou coluna escolhida for maior que 3 ou menor que 1, a marcação não pode ser feita, pois o nosso tabuleiro só tem linhas e colunas de 1 a 3.

É importante avisar ao jogador, caso isso ocorra. Veja o código a seguir, com o procedimento *jogar* ainda incompleto, mas um pouco mais elaborado, com a validação da escolha do usuário.