

# Programa o Estruturada

## Aula 13 - Desenvolvendo um jogo da velha

# 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 instanciação 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

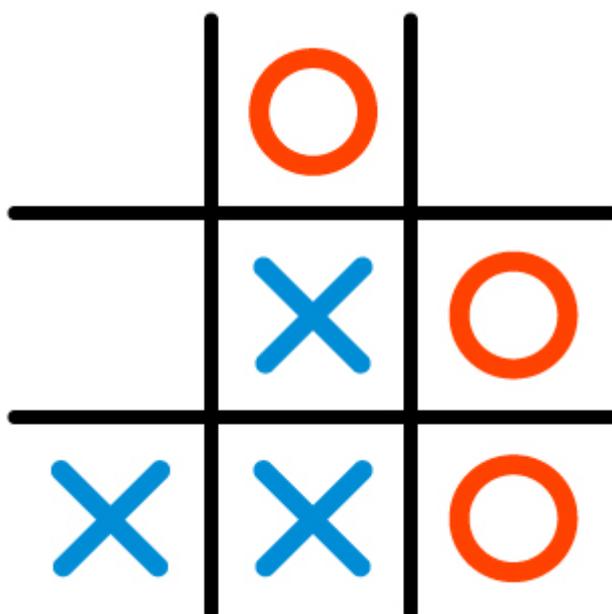
- Desenvolver as funcionalidades mais básicas de um jogo da velha.
- Identificar, na implementação do jogo, o uso de conceitos aprendidos em aulas anteriores.

# 1. Jogo da Velha

---

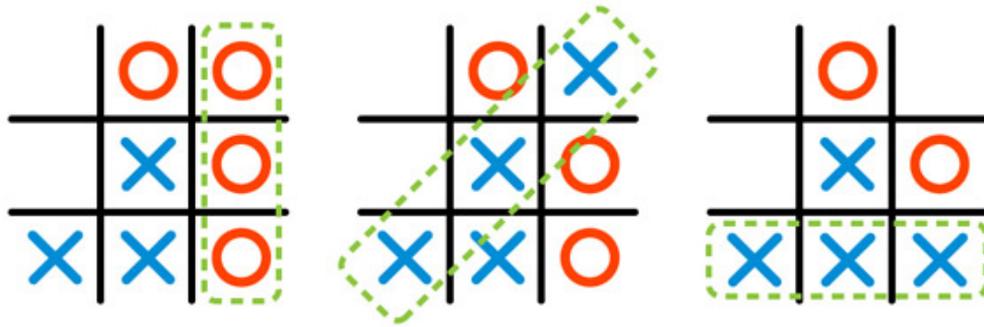
Você já deve conhecer o jogo da velha. Trata-se de um jogo em turnos alternados entre dois jogadores que se passa em um tabuleiro de nove posições (3 x 3). No jogo, cada jogador, em sua vez, faz uma marcação em uma das posições do tabuleiro, sendo essa marcação o símbolo "X" ou o símbolo "O". Veja o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

**Figura 01** - 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 02** - 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 *else* 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:

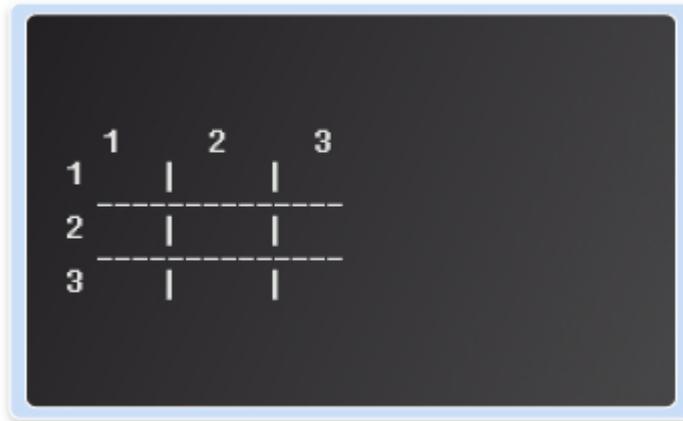
```

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.

**Figura 03** - Tabuleiro “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), desenhe 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.

```

1 public static void jogar(int jogador) {
2     // inicializando contador da estrutura while
3     int i = 0;
4     // definindo o jogador da vez
5     if (jogador == 1) {
6         jog = 1;
7     } else {
8         jog = 2;
9     }
10    System.out.println("\n\n Vez do Jogador " + jog);
11    while (i == 0) {
12        linha = 0; // inicializando valor da linha
13        coluna = 0; // inicializando valor da coluna
14        while (linha < 1 || linha > 3) {
15            System.out.print("Escolha a Linha (1,2,3):");
16            // lendo a linha escolhida
17            linha = leitor.nextInt();
18            // Aviso de linha inválida, caso o jogador tenha
19            // escolhido linha menor que 1 ou maior que 3
20            if (linha < 1 || linha > 3) {
21                System.out.println("Linha invalida! Escolha uma linha entre 1 e 3.");
22            }
23        }
24        while (coluna < 1 || coluna > 3) {
25            System.out.print("Escolha a Coluna (1,2,3):");
26            // lendo a coluna escolhida
27            coluna = leitor.nextInt();
28            if (coluna < 1 || coluna > 3) {
29                System.out.println("Coluna invalida! Escolha uma coluna entre 1 e 3.");
30            }
31        }
32        // Ajusta índices para começar do zero
33        linha = linha - 1;
34        coluna = coluna - 1;
35        (...)

```

Leia os comentários que estão no código (as linhas com //, lembra?).

Veja que, após a leitura da escolha do usuário (para a linha e para a coluna), temos um IF que valida a escolha, avisando que não é permitido escolher linha ou coluna menor que 1 ou maior que 3.

Veja, também, que a seleção da linha e a seleção da coluna estão inseridas, cada uma, em uma estrutura *while*. Isso é feito para que, enquanto o valor da linha e/ou da coluna digitado pelo jogador for inválido, a mensagem que solicita a escolha de uma linha e/ou coluna válida seja exibida.

Dessa forma, o programa só passa para a parte seguinte do código após a seleção validada desse ponto de vista (linha e coluna com valor entre 1 e 3). Feito isso, ainda falta uma validação importante: se o jogador tentar marcar um campo que não esteja vazio, ele deve ser avisado de que não pode fazer isso. Veja o trecho de código a seguir.

```
1 if (casa[linha][coluna] == 0) {
2     // se não estiver marcado
3     // marcar com o símbolo do jogador da vez
4     casa[linha][coluna] = jog;
5     i = 1;
6 } else { // se o campo escolhido já estiver marcado
7     System.out.println("Posição ocupada!");
8 }
```

O trecho de código analisa se o campo selecionado pelo jogador da vez não está marcado. Se não estiver, é feita a marcação com o símbolo do jogador da vez (*jog*) e o contador do laço *while* recebe o valor 1. Essa alteração no valor do contador ocorre para que, ao ser feita a marcação, a execução saia do laço (lembre-se de que a condição para o laço era o contador *i = 0*). Já que a marcação foi realizada, não há mais porque permanecer no laço.

Caso o campo escolhido pelo jogador já esteja marcado, o IF não é executado, mas sim o ELSE, em que uma mensagem avisa ao jogador para escolher outro campo. A jogada só é concluída quando a condição do IF é atendida. Só assim a marcação é realizada e o contador é alterado, para a execução sair do laço. Veja o código do procedimento *jogar* completo a seguir.

```

1 public static void jogar(int jogador) {
2     // inicializando contador da estrutura while
3     int i = 0;
4     // definindo o jogador da vez
5     if (jogador == 1) {
6         jog = 1;
7     } else {
8         jog = 2;
9     }
10    System.out.println("\n\n Vez do Jogador " + jog);
11    while (i == 0) {
12        linha = 0; // inicializando valor da linha
13        coluna = 0; // inicializando valor da coluna
14        while (linha < 1 || linha > 3) {
15            System.out.print("Escolha a Linha (1,2,3):");
16            // lendo a linha escolhida
17            linha = leitor.nextInt();
18            // Aviso de linha inválida, caso o jogador tenha
19            // escolhido linha menor que 1 ou maior que 3
20            if (linha < 1 || linha > 3) {
21                System.out.println("Linha invalida! Escolha uma linha entre 1 e 3.");
22            }
23        }
24        while (coluna < 1 || coluna > 3) {
25            System.out.print("Escolha a Coluna (1,2,3):");
26            // lendo a coluna escolhida
27            coluna = leitor.nextInt();
28            if (coluna < 1 || coluna > 3) {
29                System.out.println("Coluna invalida! Escolha uma coluna entre 1 e 3.");
30            }
31        }
32        // Ajusta índices para começar do zero
33        linha = linha - 1;
34        coluna = coluna - 1;
35        if (casa[linha][coluna] == 0) {
36            // se não estiver marcado
37            // marcar com o símbolo do jogador da vez
38            casa[linha][coluna] = jog;
39            i = 1;
40        } else { // se o campo escolhido já estiver marcado
41            System.out.println("Posição ocupada!");
42        }
43    }
44 }

```



## Vídeo 03 - rotina Jogar

## Atividade 02

---

1. Observando as regras do jogo de damas, preencha o tabuleiro que você criou na **Atividade 1**, crie uma função que determine as jogadas e as valide. **Dica:** É importante lembrar que a lógica do jogo de damas é diferente da lógica do jogo da velha.

**Lembre:** O código para o jogo de damas é um pouco mais complexo. Você terá que determinar a mudança de posição das peças, sem sobrepor (duas peças não podem ocupar uma mesma posição na matriz); terá que decrementar a quantidade de peças do jogador quando uma peça sua for “comida” por uma peça do adversário; terá que impedir o jogador de mover a peça para trás etc. Está lançado o desafio!

## 5. Verificando se Houve Vencedor

---

O jogo da velha termina no momento em que todas as posições do tabuleiro são preenchidas ou quando um dos jogadores completa uma sequência de símbolos em uma linha, em uma coluna ou em uma diagonal. Para o jogo finalizar, é importante que, durante toda a partida, seja feita a verificação do *status* do jogo. Ou seja, o programa deve verificar se, antes de todas as casas serem preenchidas, já existe um vencedor. Esse controle é realizado pelo seguinte procedimento:

```

1 public static void check() {
2     int i = 0;
3     //verificando se houve vencedor na Horizontal:
4     for (i = 0; i < 3; i++) {
5         if (casa[i][0] == casa[i][1] && casa[i][0] == casa[i][2]) {
6             if (casa[i][0] == 1) win = 1;
7             if (casa[i][0] == 2) win = 2;
8         }
9     }
10    //verificando se houve vencedor na Vertical:
11    for (i = 0; i < 3; i++) {
12        if (casa[0][i] == casa[1][i] && casa[0][i] == casa[2][i]) {
13            if (casa[0][i] == 1) win = 1;
14            if (casa[0][i] == 2) win = 2;
15        }
16    }
17    //verificando se houve vencedor na Diagonal de cima para baixo:
18    if (casa[0][0] == casa[1][1] && casa[0][0] == casa[2][2]) {
19        if (casa[0][0] == 1) win = 1;
20        if (casa[0][0] == 2) win = 2;
21    }
22    //verificando se houve vencedor na Diagonal de baixo para cima:
23    if (casa[0][2] == casa[1][1] && casa[0][2] == casa[2][0]) {
24        if (casa[0][2] == 1) win = 1;
25        if (casa[0][2] == 2) win = 2;
26    }
27 }

```

Lembre-se de que a variável *win* foi criada no início do nosso código para armazenar o vencedor (que pode ser o valor inteiro 1 ou 2). Cada estrutura *for*, na rotina *check*, verifica se houve vencedor em uma situação específica. Todas as situações de vitórias possíveis são checadas (na vertical, na horizontal e na diagonal).

## 6. Colocando o Jogo para Funcionar

---

Depois de escrevermos todas essas funções, você deve estar ansioso para ver o jogo funcionando, não é mesmo? Para isso, falta apenas a definição da rotina principal: a rotina *main*. Nela, são chamadas as outras funções do jogo para que ele funcione. Veja o código da função *main*:

```

1 public static void main(String[] args) {
2     int i = 0;
3     // percorre todo o tabuleiro, nas nove posições:
4     for (i = 0; i < 9; i++) {
5         jogo();// chama a rotina jogo(), que desenha o tabuleiro
6         if (i % 2 == 0) {
7             jogar(2);
8         } else {
9             jogar(1);
10        }
11        // chama a rotina check (), para ver se alguém ganhou
12        check();
13        if (win == 1 || win == 2) {
14            // sai do laço antes de completar o tabuleiro,
15            // se alguém tiver vencido
16            i = 10;
17        }
18    }
19    // chama a rotina jogo(), para desenhar novamente o tabuleiro
20    jogo();
21    // verifica se houve vencedor
22    System.out.println();
23    if (win == 1 || win == 2) {
24        // informa o vencedor
25        System.out.println("Jogador " + win + " é o ganhador!");
26    } else {
27        // se não houve vencedor
28        System.out.println("Não houve vencedor! O jogo foi empate!!");
29    }
30 }

```

Vamos agora implementar o jogo? Junte todas as funções em um código só e execute. Veja como fica a tela inicial do jogo:

```

  1   2   3
1  |  |  |
  ---
2  |  |  |
  ---
3  |  |  |

  Vez do Jogador 2
  Escolha a Linha (1,2,3):

```



## Vídeo 04 - Jogo da Velha Completo

Veja que aparece o tabuleiro, a informação de quem é a vez de jogar e a solicitação para a escolha da linha. Após a escolha da linha, o jogador escolhe a coluna e a marcação é feita. Veja a seguir a marcação feita para a linha 2 e coluna 2:

```
  1   2   3
1  |   |
--|---|
2  |   |
--|---|
3  |   |

Vez do Jogador 2
Escolha a Linha (1,2,3):2
Escolha a Coluna (1,2,3):2

  1   2   3
1  |   |
--|---|
2  | 0  |
--|---|
3  |   |

Vez do Jogador 1
Escolha a Linha (1,2,3):
```

Nesse momento, a marcação é feita na linha e coluna escolhidas e a vez de jogar passa para o jogador 2. As jogadas são realizadas, alternadamente, até que haja um vencedor ou o empate.

Agora que você implementou o jogo e ele funcionou, teste e divirta-se à vontade. Na próxima aula, incluiremos alguns adicionais ao jogo, como cadastro de jogadores, *ranking* e contagem de tempo. Até lá!

## Atividade 03

---

1. Implemente a rotina *main* para o jogo de damas.

## 6. Resumo

---

Nesta aula, desenvolvemos o passo a passo de um jogo da velha. Você viu as regras básicas do jogo, os requisitos para a sua implementação e desenvolveu diversas funções, cada uma com uma utilidade específica dentro do jogo. Você analisou, em detalhes, os aspectos funcionais de cada uma, como: desenho do tabuleiro na tela, determinação do jogador da vez, jogadas e suas validações e verificação de vitória ou empate. Ao final, viu como desenvolver a rotina principal (*main*, que chama as demais funções), implementar o jogo como um todo e testar. Divertido, não?

## 7. Autoavaliação

---

1. Desenvolva o jogo da velha que fizemos nesta aula. Tente elaborar a lógica de acordo com as regras do jogo e implemente, escrevendo seu próprio código, sem "colar" do código que vimos na aula. Você pode olhar, de vez em quando, caso tenha dúvidas, mas tente desenvolver a sua própria lógica.
2. Construa um jogo de damas completo, utilizando a mesma metodologia que utilizamos para o jogo da velha (com o uso de matrizes e funções).

## 8. Referências

---

ASCENCIO, Ana Fernanda Gomes; CAMPOS, Edilene Aparecida Veneruchi de. **Fundamentos da programação de computadores:** algoritmos, Pascal, C/C++ e Java. São Paulo: Editora Pearson, 2008.

JOGO da velha: entendendo um pouco mais da linguagem C: tutorial na internet. Disponível em: <[http://ube-164.pop.com.br/repositorio/12329/meusite/jogo\\_da\\_velha.pdf](http://ube-164.pop.com.br/repositorio/12329/meusite/jogo_da_velha.pdf)>. Acesso em: 20 jan. 2010.

RADTKE, Paulo V. W. **Jogo da velha**: projeto. Curitiba: PUC Paraná, 2006.  
Disponível em: <[http://www.ppgia.pucpr.br/~radtke/jogos/velha/projeto-jogo\\_da\\_velha.pdf](http://www.ppgia.pucpr.br/~radtke/jogos/velha/projeto-jogo_da_velha.pdf)>. Acesso em: 20 jan. 2010.

THE JAVA tutorials. Disponível em:  
<<http://download.oracle.com/javase/tutorial/>>. Acesso em: 6 dez. 2011.