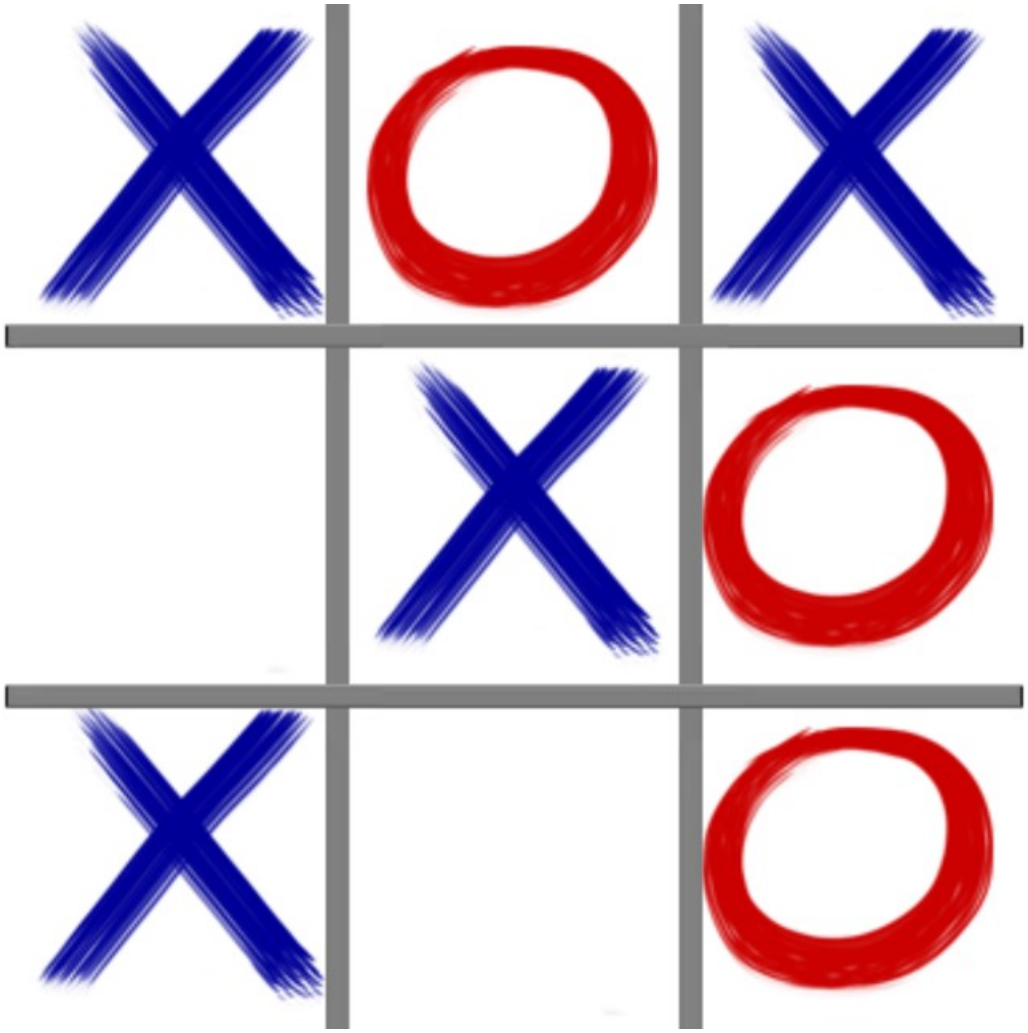
JOGO DA VELHA



Sumário

1. Objetivos do Projeto

Implementar uma versão em aplicativo para dispositivos móveis do mundialmente conhecido **Jogo da Velha**.

2. Sobre o Jogo da Velha

O jogo da velha é um jogo de estratégia disputado entre dois jogadores. Cada jogador marca com um símbolo X ou O (que eu chamarei aqui de marcadores) uma posição vazia num tabuleiro de 3x3 posições em turnos alternados. O objetivo do jogo é completar uma linha na horizontal, vertical ou diagonal do tabuleiro com três marcadores iguais. O jogador que completa uma linha dessa forma com os marcadores seus pontua no jogo.

Nesta versão do jogo da velha o jogador não disputará contra outra pessoa, mas contra o próprio dispositivo móvel. A habilidade do sistema numa jogada pode ser configurada antes de começar as partidas de acordo com a seguinte regra:

- ✔ ***NORMAL:*** Neste nível de dificuldade o sistema poderá cometer alguns erros de estratégia que se aproveitados levarão o jogador à vitória.

- ✔ ***EXPERIENTE:*** Neste nível de dificuldade o sistema cometerá poucos erros. Eventualmente é possível vencê-lo.

- ✔ ***INVENCÍVEL:*** Neste nível o sistema se torna invencível. A depender do nível de habilidade do jogador, ele empatará com o sistema em todas as partidas.

O jogador humano marcara o tabuleiro com o marcador **X** e o sistema com o marcador **O**. A sequência das jogadas após abrir o aplicativo segue como descrito:

1. Na primeira jogada o aplicativo escolhe o jogador que inicia a marcar o tabuleiro de forma aleatória.

2. O jogador selecionado marcará uma posição vazia do tabuleiro, e após fazê-lo, a vez passará para o adversário e depois deste marcar volta para o jogador novamente, nesta sequência, até alguém vencer o jogo ou se esgotar todas as casas vazias do tabuleiro.

3. Se ao longo do jogo algum dos jogadores completar uma linha com três marcadores seus na horizontal, vertical ou diagonais do tabuleiro ele pontuará. A partida é então finalizada.

4. Se nenhum dos dois jogadores pontuarem, assim que marcadas todas as 9 casas do tabuleiro a partida é finalizada com empate.

Ao iniciar uma nova partida as seguintes condições serão observadas:

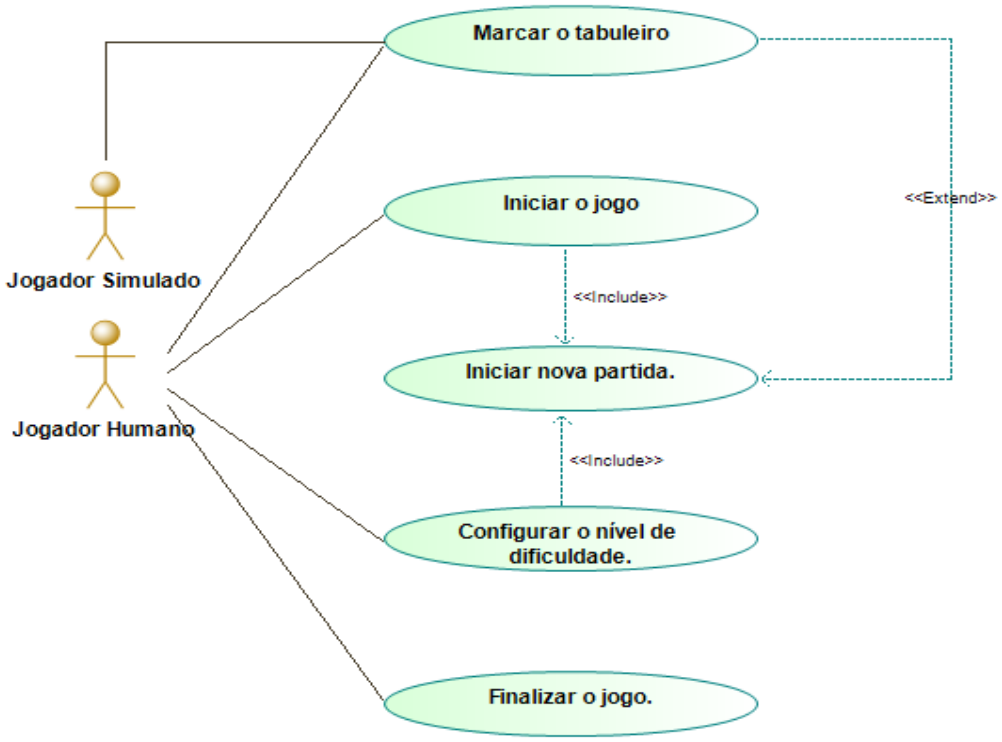
1. Se a partida anterior terminou em empate, inicia marcando o tabuleiro na nova partida o adversário do jogador que iniciou a partida anterior.

2. Se houve vencedor na partida anterior, inicia a partida o jogador que venceu.

Vence uma disputa quem pontuou o maior número de partidas no jogo. Em caso de não haver vencedores, termina em empate.

3. Casos de uso

O aplicativo é intencionalmente bem simples. O diagrama de casos de uso abaixo dá uma visão de alto nível das funcionalidades do mesmo:



3.1. Atores

Ator	Descrição
Jogador Simulado	Jogador que representa o próprio sistema em disputa.
Jogador Humano	Jogador humano que interage com o aplicativo.

3.2. Casos de Uso

Caso de Uso	Descrição
Marcar o tabuleiro	Marcar uma posição vazia do tabuleiro do jogo da velha. Se o jogador humano realiza este caso de uso, ela faz por meio da interação com a interface gráfica de usuário (GUI) do aplicativo. Caso seja o próprio sistema, ele executa um algoritmo para ler o estado do interno do tabuleiro e marcar. Uma nova partida é iniciada automaticamente em caso de vitória ou todas as casas do tabuleiro foram preenchidas.
Iniciar o jogo	Abrir o aplicativo para iniciar o jogo.
Finalizar o jogo	Fechar o aplicativo para finalizar o jogo.
Configurar o nível de dificuldade	Configurar o nível de dificuldade que o sistema oferecerá durante o jogo. Inicia-se uma nova partida quando muda o nível de dificuldade.
Iniciar nova partida	Iniciar uma nova partida de modo automático.

4. Estrutura do projeto

O projeto, que é implementado em linguagem de programação Kotlin, segue o paradigma de programação orientada a objetos. Como é pouco descritivo fazer aqui um diagrama de todas as classes e seus relacionamentos no código, resolvi explicar a lógica do programa dividindo-a em 3 categorias distintas:

- ✔ **Controle do jogo:** controle de alternância das jogadas, estado do placar e análise do estado do tabuleiro após cada jogada.

- ✔ **Interação com o jogador humano:** configuração de elementos de interface gráfica do usuário, tratamento de eventos dos componentes de interface e feedback.

- ✔ **Lógica do jogador simulado:** diz respeito a todas as etapas de análise que o sistema faz do estado do tabuleiro a fim de avaliar estratégias de jogada. No contexto do controle do jogo, o fato de o jogador ser humano ou simulado é indiferente. A implementação do jogador simulado se deu no contexto de dar a real impressão de se disputar uma partida com uma pessoa de verdade, inclusive simulando tempos de reação semelhantes aos que um jogador humano oferece.

4.1. Controle do Jogo

O controle do jogo é de responsabilidade da classe *GameController* (Controlador do Jogo). O controle se dá no seguinte contexto:

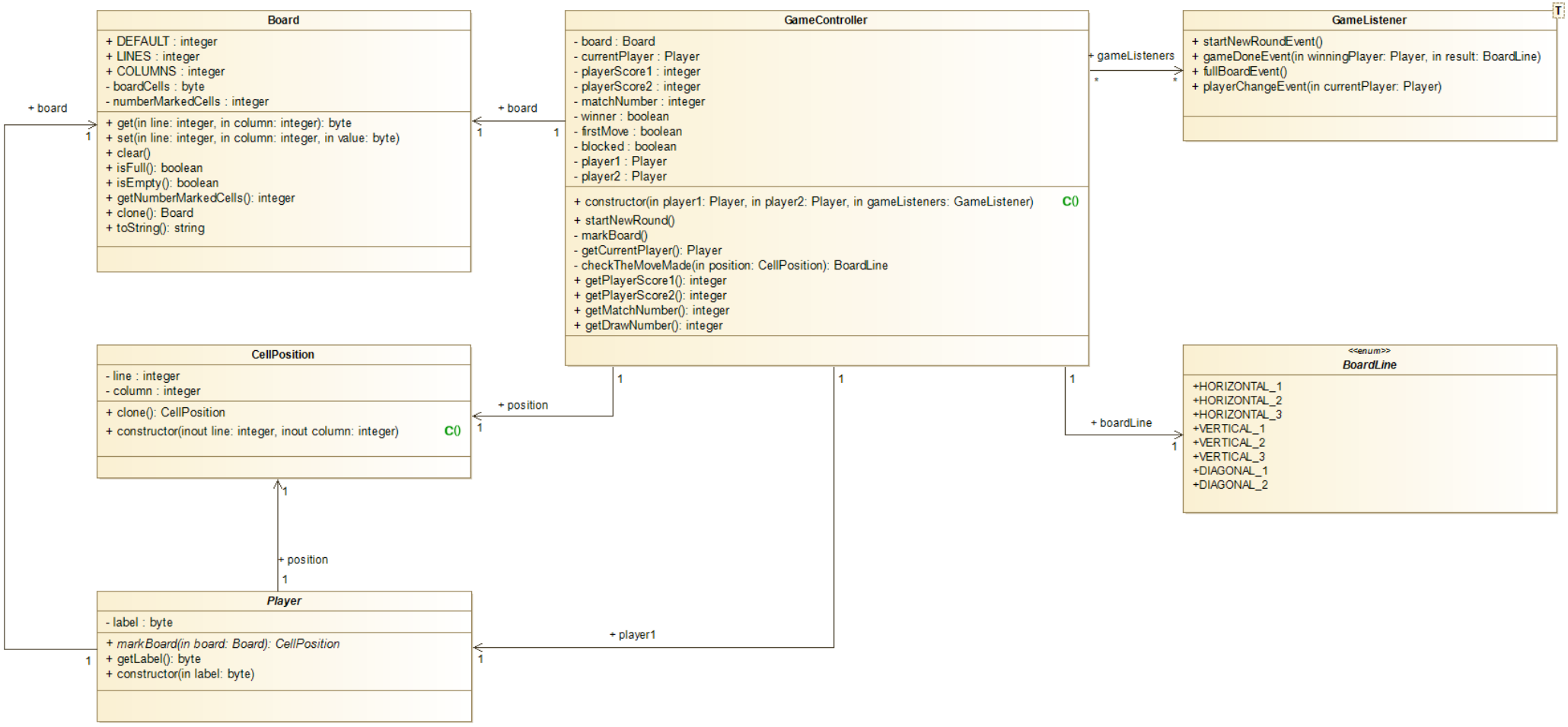
1. Inicializa o contexto do jogo assim que o aplicativo é inicializado. As pontuações de ambos os jogadores inicializa zerada e a ordem das partidas estará em 1. Notifica as instâncias de *GameListener* (Ouvinte do Jogo) de que uma partida foi iniciada.

2. Monitora o jogador a marcar o tabuleiro de acordo com as regras na seção 2. O jogador da vez deve marcar o tabuleiro. Se o jogador for o próprio sistema, ele executará o algoritmo imediatamente assim que selecionado. No caso do jogador humano, aguarda a interação do jogador com a interface gráfica do aplicativo.

3. Após o jogador marcar o tabuleiro, verifica se alguma linha na horizontal, vertical ou diagonal foi preenchida com o marcador do mesmo. Caso em alguma jogada tenha sido preenchida, notifica as instâncias de *GameListener* sobre a vitória do jogador e a identificação da linha preenchida de acordo com os valores da classe enum *BoardLines* (Linhas do Tabuleiro).

4. Caso o tabuleiro seja preenchido (as 9 casas marcadas), e não tenha vencedores (empate), notifica as instâncias de *GameListener* sobre a condição de tabuleiro cheio com o final da partida.

Na classe *GameController* não faz o início automático de uma nova partida após a vitória de algum dos jogadores ou a marcação de todas as casas do tabuleiro. Ela apenas monitora a alternância das jogadas e o estado do tabuleiro após cada jogada. Esta funcionalidade será delegada à classe que faz a chamada à *GameController*.


Board
+ DEFAULT : integer
+ LINES : integer
+ COLUMNS : integer
- boardCells : byte
- numberMarkedCells : integer
+ get(in line: integer, in column: integer): byte
+ set(in line: integer, in column: integer, in value: byte)
+ clear()
+ isFull(): boolean
+ isEmpty(): boolean
+ getNumberMarkedCells(): integer
+ clone(): Board
+ toString(): string
GameController
- board : Board
- currentPlayer : Player
- playerScore1 : integer
- playerScore2 : integer
- matchNumber : integer
- winner : boolean
- firstMove : boolean
- blocked : boolean
- player1 : Player
- player2 : Player
+ constructor(in player1: Player, in player2: Player, in gameListeners: GameListener) C()
+ startNewRound()
- markBoard()
- getCurrentPlayer(): Player
- checkTheMoveMade(in position: CellPosition): BoardLine
+ getPlayerScore1(): integer
+ getPlayerScore2(): integer
+ getMatchNumber(): integer
+ getDrawNumber(): integer
GameListener
T
+ startNewRoundEvent()
+ gameDoneEvent(in winningPlayer: Player, in result: BoardLine)
+ fullBoardEvent()
+ playerChangeEvent(in currentPlayer: Player)
CellPosition
- line : integer
- column : integer
+ clone(): CellPosition
+ constructor(inout line: integer, inout column: integer) C()
<<enum>>
BoardLine
+HORIZONTAL_1
+HORIZONTAL_2
+HORIZONTAL_3
+VERTICAL_1
+VERTICAL_2
+VERTICAL_3
+DIAGONAL_1
+DIAGONAL_2
Player
- label : byte
+ markBoard(in board: Board): CellPosition
+ getLabel(): byte
+ constructor(in label: byte)
+ board
+ board
+ gameListeners
+ position
+ boardLine
+ position
+ player1
1
*

Classe	Função
GameController	Faz o controle do contexto do jogo e arbitragem da partida.
TicTacToeBoard	Representa o tabuleiro do jogo da velha no sistema.
GameListener	Ouvinte de eventos do jogo. Vai se registrar como ouvinte elementos de mais alto nível como MainActivity que atualiza sua interface gráfica de usuário de acordo com as notificações recebidas da classe GameController.
CellPosition	Representa a localização de uma célula no tabuleiro do jogo da velha. Aqui é utilizada para receber a posição vazia do tabuleiro selecionada por algum dos jogadores.
Player	Representa um jogador. Serão dois jogadores (player1 e player2) que marcarão o tabuleiro alternadamente.
BoardLine	Representa uma determinada linha preenchida do tabuleiro. Está será notificada para as instâncias de GameListener quando algum dos jogadores vencer.

aaaa