

## Instituto Superior de Engenharia de Coimbra

## Licenciatura em Engenharia Informática Programação Avançada



Pedro dos Santos Alves | a2019112789

Laboratório P3 Trabalho Prático 1 Meta 2

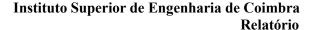
Coimbra, 13 de junho de 2021





# Índice

1.	Introdução	1
2.	Decisões tomadas na implementação	2
	Diagrama de Estados	
	Descrição e relacionamento das classes	
	4. 1. StateMachine	
	4. 2. GameData	
	4. 3. State	
	4. 4. Board	
	4. 5. Minigame	
	4. 6. Player	
	4. 7. StateMachineObservable	
5	Funcionalidades	
	Conclusão	
	anual de Utilizador	
	Iniciar a aplicação	
	Logs	
	Menu Principal	
	Configurar Jogo	
	Adicionar Jogador	
	Carregar Jogo	
	Replay Jogo	III
	Turno do Jogador	IV
	Guardar Jogo	V
	Fim do Jogo	V
	Jogar Mini-jogo	VI
	Mini-jogo	VI





## 1. Introdução

O presente relatório descreve o projeto desenvolvido no âmbito da disciplina de Programação Avançada da Licenciatura em Engenharia Informática do Instituto Superior de Engenharia de Coimbra.

Na primeira meta do trabalho prático pretende-se desenvolver uma versão do conhecido jogo quatro-em-linha, mas com a seguinte diferença: após quatro jogadas de cada jogador, é dada a hipótese de realizar uma atividade suplementar.

Na segunda meta do trabalho prático pretende-se desenvolver a interface de utilizador utilizando o JavaFX.

O trabalho consiste na implementação do padrão máquina de estados para concretizar a lógica do jogo. Algumas das funcionalidades para este jogo incluem: a possibilidade de voltar atrás nas jogadas, realizar atividades suplementares após um número de iterações, gravar um jogo para mais tarde continuar, carregar um jogo previamente guardado e manter um histórico dos últimos cinco jogos para poder ver o desenrolamento dele mais tarde.

O objetivo do presente trabalho é para consolidar todos os conhecimentos adquiridos nas aulas teóricas e práticas ao longo de todo o semestre.



## 2. Decisões tomadas na implementação

O trabalho consiste na implementação do padrão máquina de estados para concretizar a lógica do jogo.

Durante o decorrer do jogo, os jogadores humanos tem a hipótese de realizar uma atividade suplementar. A verificação para saber se o jogador pode ou não realizar essa atividade é efetuada num estado só para esse fim. Foram implementados vários estados que apenas realizam uma ação de forma automática com o intuito de facilitar o desenvolvimento da máquina de estados.

A funcionalidade de possibilitar de voltar atrás nas jogadas foi implementada com o auxilio de uma pilha com tamanho fixo, uma versão alterada do "Stack" do Java, permitindo apenas um certo número de itens nessa pilha. Cada jogada efetuada é guardada na pilha e assim que a pilha chega ao número de itens máximos, removerá o primeiro item da pilha, ou seja, o mais antigo. Quando um jogador desejar voltar atrás será removida um ou mais itens do topo da pilha.

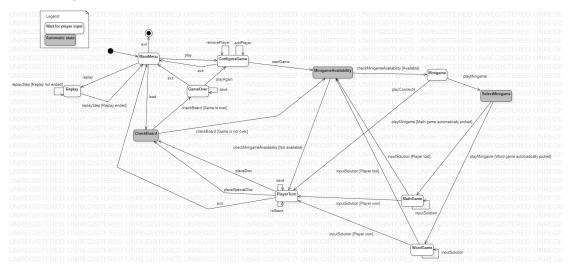
Para manter um histórico dos últimos cinco jogos jogados, o jogo é guardado automaticamente num ficheiro dentro de uma diretoria definida pelo programa, assim que esse jogo terminar. Os ficheiros são ficheiros binários com extensão ".pa", guardados com a ajuda das bibliotecas padrão do Java para *serializar* o jogo. O nome desses "replays" seguem o seguinte padrão: "yyyy-MM-dd\_HH-mm-ss\_Jogador1\_vs\_Jogador2.pa".

Foram criados ficheiros ".fxml", com o auxilio do "SceneBuilder", para a interface do utilizador em modo gráfico. Cada ficheiro ".fxml" tem um controlador associado que corresponde a um ou vários estados.

O padrão de notificações assíncronas foi implementado através de uma classe "observável" que observa as alterações de estado da máquina de estados, como estudado nas aulas. Foi criado uma interface "IController" em que dois dos métodos, o "show" e "hide", controlam o que é apresentado nas vistas assim que a aplicação seja notificada de uma alteração de estado. Em outras palavras, quando houver uma alteração de estado, a aplicação é notificada tendo acesso ao estado anterior e o estado atual, podendo assim esconder a vista relativa ao estado anterior e mostrar a vista relativa ao estado atual.



## 3. Diagrama de Estados



Acima é apresentado o diagrama de estados da máquina de estados implementado no trabalho.

Assim que a aplicação inicia, esta começa no estado "MainMenu", onde o utilizador poderá escolher uma de quatro opções. A máquina de estados tem apenas uma saída e está localizada neste estado.

A transição "replay" levará a máquina de estados a mudar para o estado "Replay". O estado "Replay" é um estado que não recebe nenhum *input* do utilizador, ou seja, é um estado automatizado, apenas realiza a transição "replayStep" até que tenha terminado todos os *replays*.

A transição "play" passa para o próximo estado: "ConfigureGame". Neste estado o utilizador poderá configurar algumas opções, como adicionar ou remover jogadores. Após introduzido pelo menos dois jogadores o jogo poderá ser iniciado. O utilizador pode também voltar para o estado "MainMenu" através da transição "exit".

A transição "startGame" inicia o jogo quatro-em-linha. Transita para o estado "MinigameAvailability", que verifica se pode ser feita uma atividade suplementar. É também um estado automatizado. Caso exista uma atividade suplementar disponível, ou seja o jogador é um jogador humano e tenha jogado quatro peças, passará para o estado "Minigame", se não passará para o estado "PlayerTurn".

No estado "Minigame" o utilizador poderá iniciar a atividade suplementar ou continuar a jogar o jogo quatro-em-linha. Ao iniciar a atividade suplementar a máquina de estados passa para o estado "SelectMinigame" onde seleciona automaticamente qual a atividade que o jogador jogará. Cada atividade suplementar tem o seu estado, que ficará à espera de *input* por parte do utilizador e passará para outro estado caso tenha efetuado essas atividades com sucesso ou ultrapassou o tempo necessário para as completar.

No estado "PlayerTurn" o utilizador tem a hipótese de escolher entre cinco opções. A transição "save" voltará ao mesmo estado, guardando o estado de jogo num ficheiro escolhido pelo utilizador. A transição "rollback" voltará também ao mesmo estado na máquina de estados, retornando para uma jogada anterior. A transição "exit" termina o jogo e volta para o estado "MainMenu". Finalmente as transições "placeDisc" e "placeSpecialDisc" colocam uma peça ou peça especial no tabuleiro respetivamente e alteram o estado para o estado "CheckBoard".

O estado "CheckBoard" é um estado automatizado que verifica o estado do tabuleiro e verifica se o jogo acabou ou não. Este estado pode também ser alcançado através do estado "MainMenu" pela transição "load", que carrega um jogo previamente guardado. Caso o tabuleiro esteja cheio ou um jogador tenha ganho o jogo, o jogo é guardado automaticamente num ficheiro numa pasta definida pelo programa e a máquina de estados passará para o estado "GameOver", caso contrário continua o jogo passando para o estado "MinigameAvailability".

No estado "GameOver" é dada a hipótese de guardar o jogo, jogar novamente um novo jogo ou voltar para o menu principal. A transição "save" guarda o jogo num ficheiro para mais tarde ser visto o seu *replay*. A transição "exit" muda o estado para o estado "MainMenu". E a transição "playAgain"

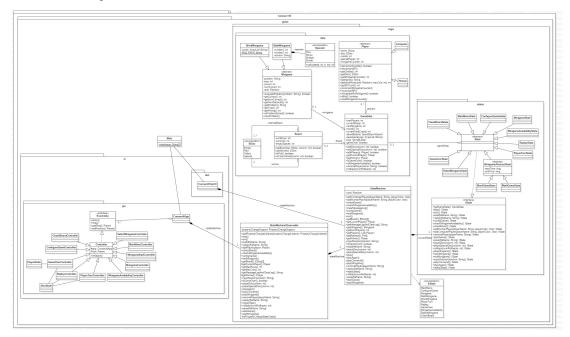


### Instituto Superior de Engenharia de Coimbra Relatório

passa o estado da máquina de estados para o estado "ConfigureGame" onde poderá iniciar um novo jogo.



## 4. Descrição e relacionamento das classes



Acima é apresentado um diagrama de classes do presente trabalho. Este diagrama contém as classes implementadas, porém não contem toda a informação, como por exemplo funções ou atributos "private", "protected" ou não muito relevantes para serem apresentadas.

#### 4. 1. StateMachine

A classe "StateMachine" é a class que representa a máquina de estados. Esta classe faz a ligação entre os estados e a interface. A classe disponibiliza várias funções para fazer avançar a máquina de estados, assim como vários dados como por exemplo o jogador atual, o vencedor, os logs, entre outros.

### 4. 2. GameData

A classe "GameData" é a classe que guarda todos os dados relativos ao jogo quatro-em-linha. É responsável por gerir o tabuleiro, os mini-jogos e os jogadores. Esta classe também guarda os dados necessários para que seja possível voltar atrás para uma jogada anterior.

#### 4. 3. State

A classe "State" é uma classe abstrata que implementa a interface "IState". A classe não faz nada de especial para além fazer de base para as classes derivadas e de implementar a função para poder guardar o jogo em qualquer estado. No entanto, as classes que derivam desta classe abstrata implementam as funções que cada uma delas necessitar. Por exemplo a classe "MainMenuState" implementa as funções "play", "exit", "load" e "replay" para poder iniciar o jogo e ir para as configurações de jogo, para sair do programa, para carregar um ficheiro e para visualizar um jogo previamente jogado, respetivamente.



### 4. 4. Board

A classe "Board" representa o tabuleiro de jogo. Esta classe apenas guarda os dados relacionados com o tabuleiro, como o número de linhas e colunas, o estado do tabuleiro e o vencedor. Esta classe também responsabiliza-se por gerir o estado do tabuleiro.

### 4. 5. Minigame

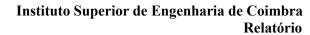
A classe "Minigame" é uma classe abstrata que representa as atividades suplementares. A classe guarda informação relativa aos mini-jogos como o problema, as tentativas, o número de respostas corretas e o número de respostas necessárias para ganhar o mini-jogo. As classes que derivam desta classe base são classes mais especializadas, por exemplo a class "MathMinigame" refere-se ao mini-jogo onde são propostos cálculos matemáticos e por isso contém os atributos necessários para fazer os cálculos.

### 4. 6. Player

A classe "Player" é uma classe abstrata que representa um jogador. Esta classe guarda informação sobre um jogador, como o nome, a peça, o número de créditos, o número de peças especiais e um contador para as atividades suplementares. As classes derivadas apenas implementam algumas funções da classe base.

#### 4. 7. StateMachineObservable

A classe "StateMachineObservable" é um *wrapper* da classe "StateMachine". A classe permite adicionar *listeners* que mais tarde serão notificados assim que haja uma mudança de estado.

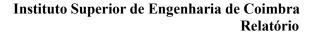




## 5. Funcionalidades

Funcionalidade	Implementado/Não implementado
Jogo quatro-em-linha	Implementado
Jogadores (Humanos e Virtuais)	Implementado
Máquina de estados	Implementado
Atividades suplementares (Mini-jogos)	Implementado
Peças especiais	Implementado
Rollbacks (voltar atrás)	Implementado
Guardar jogo	Implementado
Carregar jogo	Implementado
Histórico dos últimos 5 jogos	Implementado
Logs	Implementado
UI Texto	Implementado
UI JavaFX	Implementado
Notificações assíncronas (JavaBeans)	Implementado

Todas as funcionalidades previstas para a meta  $\bf 1$  e meta  $\bf 2$  do trabalho foram totalmente implementadas.





## 6. Conclusão

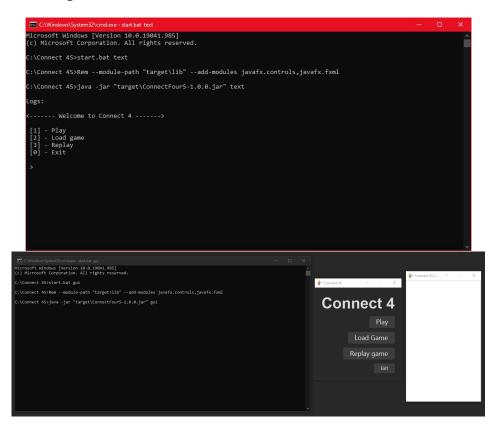
Com o desenvolvimento desta meta foi possível aprender muito sobre Java, máquinas de estado e JavaFX. Como não foram usados "mementos" ou "comandos" não foram possível pô-los em prática.

Durante o desenvolvimento desta meta foram surgindo problemas e desafios que foram superados com a ajuda dos professores da disciplina, os apontamentos por eles disponibilizados e da Internet.



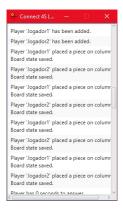
## Manual de Utilizador

## Iniciar a aplicação



A aplicação pode ser iniciada no modo de texto ou no modo gráfico. Por defeito, a aplicação é iniciada no modo gráfico.

### Logs



A janela de "Logs" é uma janela à parte que disponibiliza os logs.



## Menu Principal



- 1 Iniciar o jogo quatro-em-linha. (Configurar Jogo)
- 2 Carregar um jogo previamente guardado. (Carregar Jogo)
- 3 Visualizar um jogo terminado. (Replay Jogo)
- 4 Sair da aplicação.

### Configurar Jogo



- 1 Remover jogador.
- 2 Adicionar jogador. (Adicionar Jogador)
- 3 Iniciar jogo. (Turno do Jogador)
- 4 Voltar ao menu principal. (Menu Principal)

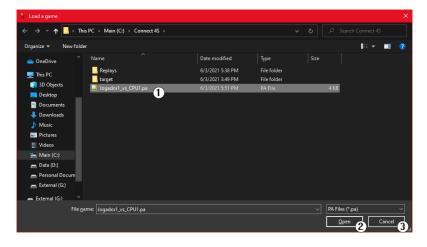


## Adicionar Jogador



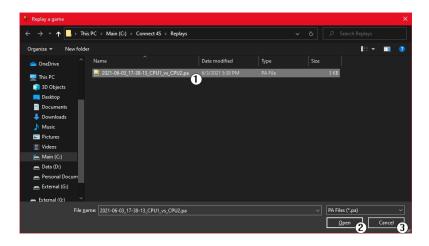
Janela para adicionar jogador.

## Carregar Jogo



- 1 Ficheiro contendo um jogo guardado previamente.
- 2 Carregar o ficheiro.
- 3 Cancelar carregamento de jogo.

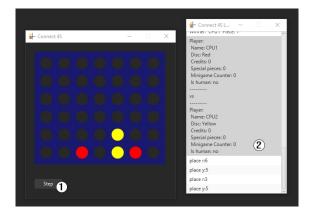
## Replay Jogo



- 1 Ficheiro contendo um jogo acabado.
- 2 Carregar o ficheiro.

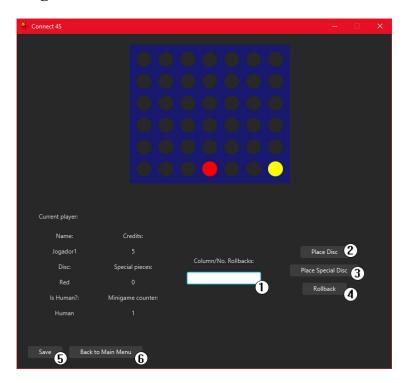


3 - Cancelar replay do jogo.



- 1 Avançar no replay.
  - O replay avança automaticamente de 1 em 1 segundo. O botão serve apenas para avançar mais rapidamente.
- 2 "Logs" relacionados com o replay.

### Turno do Jogador

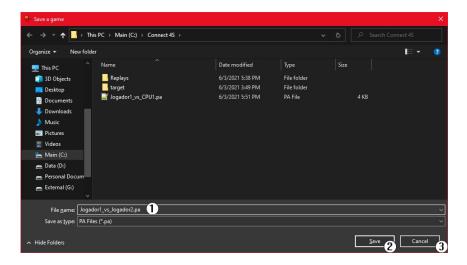


- 1 Caixa de texto
  - Introduzir um número depois clicar num dos botões: 2, 3 ou 4.
- 2 Botão para introduzir uma peça na coluna indicada na caixa de texto.
  - Botão esquerdo do rato nos círculos do tabuleiro também fará o mesmo.
- 3 Botão para introduzir uma peça especial na coluna indicada na caixa de texto.
  - Botão direito do rato nos círculos do tabuleiro também fará o mesmo.
- 4 Botão para voltar atrás o número de jogadas indicado na caixa de texto.



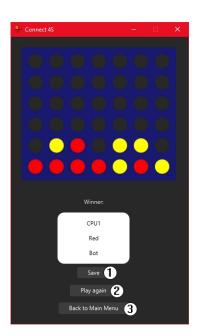
- 5 Guarda um jogo. (Guardar Jogo)
- 6 Voltar ao menu principal. (Menu Principal)

## Guardar Jogo



- 1 Nome do ficheiro do jogo a guardar.
- 2 Guardar o jogo num ficheiro para ser carregado mais tarde.
- 3 Cancelar guardar o jogo.

## Fim do Jogo



- 1 Guardar o jogo.
- 2 Jogar novamente. (Configurar Jogo)
- 3 Voltar ao menu principal. (Menu Principal)



## Jogar Mini-jogo



- 1 Nome do jogador.
- 2 Continuar a jogar o jogo quatro-em-linha. (Turno do Jogador)
- 3 Jogar mini-jogo. (Mini-jogo)

## Mini-jogo



- 1 Tempo restante do mini-jogo.
- 2 Caixa de texto
  - Introduzir solução. Clicar na tecla "Enter" aciona o botão 3.
- 3 Verificar solução.
- 4 Barra de progresso.
  - Mostra o progresso do mini-jogo, ou seja, número de perguntas certas em relação ao número de perguntas necessárias para vencer o mini-jogo.