



Enunciado do projeto prático

“Jogo de Tabuleiro - Xadrez”

Segunda Parte

Objetivos

Este projeto tem como objetivo o desenvolvimento de um programa (de agora adiante designado por jogo) usando a **linguagem Kotlin**.

O programa a desenvolver será uma versão do conhecido “**Jogo de tabuleiro - Xadrez**”.

Para alcançar estes objetivos, os alunos devem ter em conta tudo o que foram aprendendo em Fundamentos de Programação (FP).

Organização do Trabalho

Este trabalho está dividido em duas partes, sendo ambas de entrega obrigatória.

Este é o enunciado da Segunda Parte.

A não entrega de qualquer uma das partes do projeto implica reprovação na componente prática da disciplina.

Realidade / Domínio do Problema

Neste jogo de xadrez é realizado a gestão de todas as peças do tabuleiro e dos seus movimentos.

Universidade Lusitana de Humanidades e Tecnologias
LEI / LIG / LEIRT
Fundamentos de Programação
2020/2021
1º Semestre - 1ª época - 2ª parte
Bruno Saraiva, João Pedro Carvalho, Pedro Alves
v1.0.1

Dado que na primeira parte do trabalho foi realizado um tabuleiro dinâmico, na parte 2 deste projeto irão ser colocadas peças dinamicamente de acordo com as dimensões do tabuleiro.

Neste projeto o movimento das peças são as seguintes:

- Cavalo (H): Esta peça anda apenas em **L** em qualquer sentido com o máximo de 3 saltos (ex1: 2 casas para a frente e 1 para o lado; ex2: 2 casas para o lado e uma para trás);
- Torre (T) – Esta peça pode andar **N** casas na vertical (frente e trás) e horizontal (esquerda e direita). Pode também passar por cima de peças que estejam na sua direção;
- Bispo (B) – Esta peça apenas anda **N** casas na diagonal. Esta peça pode passar por cima de peças que estejam na sua direção;
- Rainha (Q) – Esta peça anda **N** casas na vertical, horizontal e diagonal. Pode também passar por cima das peças que estejam na sua direção;
- Rei (K) – Esta peça poderá andar **1** casa em qualquer direção.
- Peão (P) – Esta peça só pode andar **1** casa para a frente ou para trás.

Neste projeto, o jogo termina quando uma das equipas já não tiver nenhuma peça viva.

Objetivos da Segunda Parte

Nesta segunda parte do projeto pretende-se que os alunos realizem as seguintes tarefas:

- Alteração dos menus do jogo;
- Criação do tabuleiro inicial, com as peças posicionadas;
- Movimento das peças
- Detecção de jogo terminado

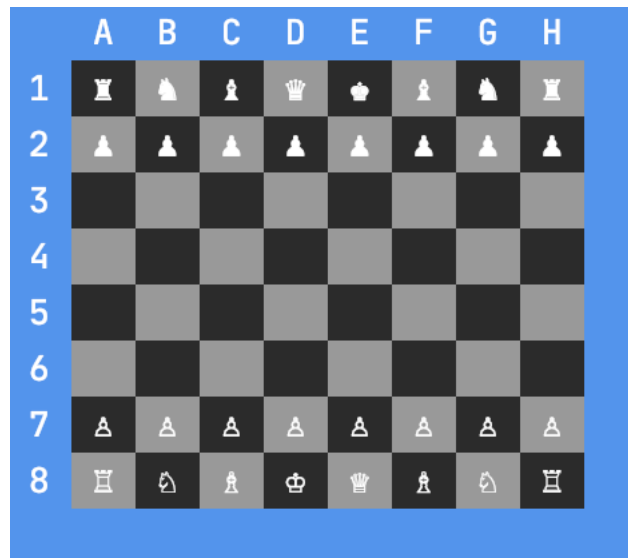
Menus / Écrans do Jogo

O menu principal deverá ter a seguinte estrutura:

```
Welcome to the Chess Board Game!  
1-> Start New Game;  
2-> Exit Game.  
  
>
```

O jogo deverá ser um ciclo que termina com a escolha **2**.

Ao escolher a opção **1** deverá ser apresentado inicialmente as mesmas perguntas que foram feitas no trabalho da primeira parte (nome dos jogadores, número de colunas e linhas e mostrar legendas e/ou peças do tabuleiro). Posteriormente será mostrado a tabela de xadrez (considerando o número de colunas e linhas igual a 8):



Nesta parte do projeto, irão existir 4 opções de tabuleiros iniciais:

8x8		7x7	
6x6		6x7	

4x4	
-----	--

- 8x8 – Contém todas as peças de um tabuleiro tradicional (16 peças brancas e pretas);
- 7x7 – Não tem a peça rainha em ambas as equipas e uma linha inteira foi retirada (14 peças brancas e pretas);
- 6x6 – Foram retiradas uma peça de cavalo e uma torre em cada equipa e 2 linhas inteiras do tabuleiro (12 peças brancas e pretas);
- 6x7 – Igual à 6x6 mas com uma linha a mais (12 peças brancas e pretas);
- 4x4 - Contém apenas 2 peças para cada equipa.

NOTA: Como se pode observar pelas figuras acima, existe um tabuleiro de 4x4. Isto quer dizer que o limite de linhas e de colunas já não é igual a **5** (como no enunciado da 1ª parte) mas sim igual a **4**.

Estrutura do Jogo

Nesta parte, terão que guardar numa estrutura a posição das várias peças no tabuleiro. Ou seja, haverá uma variável do vosso programa que permite saber em que posição está cada um dos peões, o rei, etc. Esta secção descreve essa estrutura.

Cada peça é representada por um Pair

Há 2 informações associadas a cada peça: o tipo e a cor. O tipo é uma String com um destes valores:

- P - Peão
- H - Cavalo
- K - Rei
- T - Torre
- B - Bispo
- Q - Rainha

A cor pode ser "b" (black/preto) ou "w" (white/branco).

Universidade Lusitana de Humanidades e Tecnologias
LEI / LIG / LEIRT
Fundamentos de Programação
2020/2021
1º Semestre - 1ª época - 2ª parte
Bruno Saraiva, João Pedro Carvalho, Pedro Alves
v1.0.1

Por exemplo, uma Torre preta é representada através de um `Pair("T", "b")` e uma Rainha branca através de um `Pair("Q", "w")`.

Isto significa que uma variável que represente uma peça pode ser declarada assim:

```
val peca : Pair<String,String> = Pair("Q", "w")
```

O tabuleiro é representado por um Array

Embora um tabuleiro tenha linhas e colunas, neste projeto irão representar o tabuleiro através de um único array (unidimensional) com a seguinte lógica:

tabuleiro 4x5:

1A	1B	1C	1D	1E
2A	2B	2C	2D	2E
3A	3B	3C	3D	3E
4A	4B	4C	4D	4E

array unidimensional que representa este tabuleiro:




1A	1B	1C	1D	1E	2A	2B	2C	2D	2E	3A	3B	3C	3D	3E	4A	4B	4C	4D	4E
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Ou seja, por exemplo, se a posição 1A está nas coordenadas (0,0) do tabuleiro e a posição 2C está nas coordenadas (2,1), já no array a posição 1A está no índice 0 e a posição 2C está no índice 7.

Colocar as peças no array

Como já vimos, cada peça é representada por um pair (tipo, côr) e existe um array com todas as posições do tabuleiro. Então, só falta colocar as peças dentro desse array. Para tal o array tem que ser de `Pair's nullable: Array<Pair<String,String>?>`. Caso uma certa posição não tenha nenhuma peça, deve ter o valor null.

Por exemplo, este tabuleiro "gráfico" (notem que não é possível ter um tabuleiro desta dimensão, é apenas para exemplificar):

Será na realidade o seguinte array:

Universidade Lusitana de Humanidades e Tecnologias
LEI / LIG / LEIRT
Fundamentos de Programação
2020/2021
1º Semestre - 1ª época - 2ª parte
Bruno Saraiva, João Pedro Carvalho, Pedro Alves
v1.0.1

Pair("P","b")	null	Pair("T", "b")	null	Pair("P","w")	null
---------------	------	----------------	------	---------------	------

A estrutura totalPiecesAndTurn

O jogo vai avançando por turnos (ora jogam as brancas, ora jogam as pretas) e há peças que vão sendo “comidas” até termos um vencedor (quando há apenas peças desse jogador). Para fazer essa gestão, usaremos um Array com 3 inteiros:

NUM_PEÇAS_BRANCAS	NUM_PEÇAS_PRETAS	TURNO ATUAL
-------------------	------------------	-------------

Em que:

- NUM_PEÇAS_BRANCAS é um inteiro com o número de peças brancas que ainda permanecem no tabuleiro. Por exemplo, no caso de um tabuleiro 8x8, começa por ter o valor 16.
- NUM_PEÇAS_PRETAS é um inteiro com o número de peças pretas que ainda permanecem no tabuleiro. Por exemplo, no caso de um tabuleiro 6x6, começa por ter o valor 12.
- TURNO_ATUAL é um inteiro com os valores 0 ou 1. Quando é a vez das peças brancas jogarem, estamos no turno 0. Quando é a vez das peças pretas jogarem, estamos no turno 1.

Esta estrutura e o array de peças vão mudando ao longo do jogo, logo têm que ser passadas por parâmetro para dentro das funções que fazem movimentar o jogo.

NOTA IMPORTANTE: Quando se passa um array para dentro de uma função e a essa função altera os seus valores, essas alterações refletem-se fora da função. Por exemplo:

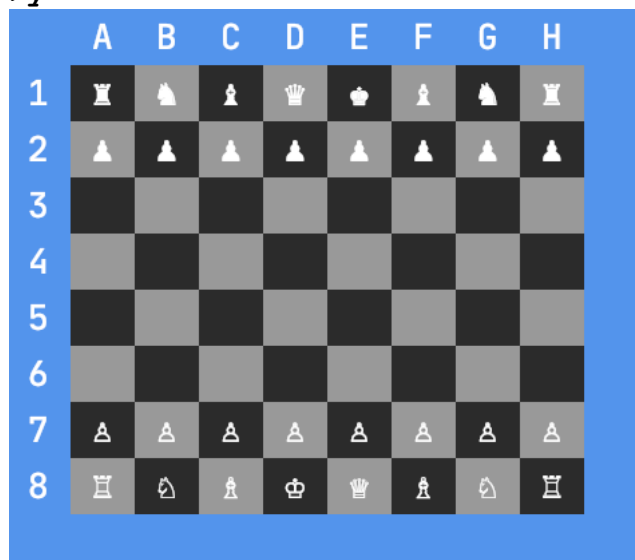
```
fun altera(numeros: Array<Int>) {  
    numeros[0] = 0  
}  
  
fun main() {  
    val numeros = arrayOf(3,4,5)  
    altera(numeros)  
    // numeros passou a ser { 0, 4 , 5 }  
}
```

O Jogo – Start New Game

Universidade Lusitana de Humanidades e Tecnologias
LEI / LIG / LEIRT
Fundamentos de Programação
2020/2021
1º Semestre - 1ª época - 2ª parte
Bruno Saraiva, João Pedro Carvalho, Pedro Alves
v1.0.1

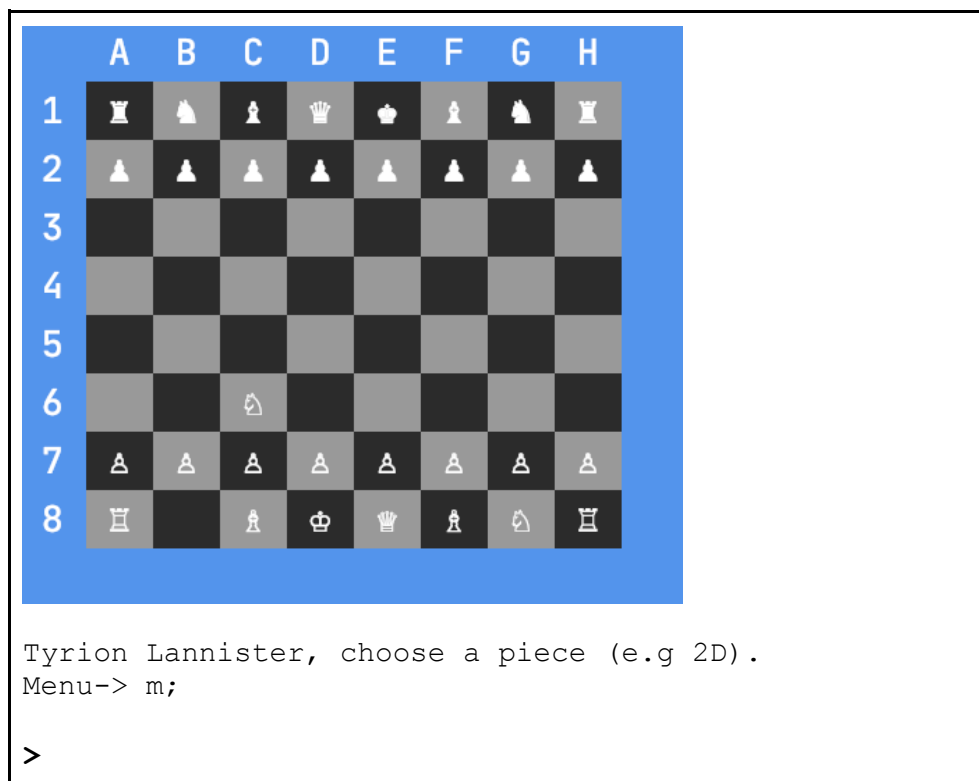
Com a opção 1 escolhida, entra-se num submenu que gera o curso do jogo. Um exemplo é então mostrado:

```
First player name?  
  
>Jon Snow  
Second player name?  
  
>Tyrion Lannister  
How many chess columns?  
  
>8  
How many chess lines?  
  
>8  
Show legend (y/n)?  
  
>y  
Show pieces (y/n)?  
  
>y
```



```
Jon Snow, choose a piece (e.g 2D).  
Menu-> m;  
  
>8b  
Jon Snow, choose a target piece (e.g 2D).  
Menu-> m;  
  
>6c
```

Universidade Lusitana de Humanidades e Tecnologias
LEI / LIG / LEIRT
Fundamentos de Programação
2020/2021
1º Semestre - 1ª época - 2ª parte
Bruno Saraiva, João Pedro Carvalho, Pedro Alves
v1.0.1



NOTA IMPORTANTE: Se a opção **m** for seleccionada deverá retornar ao menu principal.

Notem que os nomes dos jogadores introduzidos é depois usado nas jogadas.

Com esta informação, entra-se na parte mais importante da construção do jogo, ou seja, todo o algoritmo de verificações e controlo sobre o movimento das peças.

Como deve imaginar, o programa deve ter em consideração toda a validação correspondente ao movimento da peça dadas as coordenadas de partida e chegada da mesma. Com isto, pretende-se reforçar a importância de que as coordenadas estão por exemplo, dentro dos limites do tabuleiro, ou que as coordenadas de partida correspondam não só a uma peça do tabuleiro, mas também pertencente ao jogador do turno. Tenha então como referência os seguintes pontos:

- As coordenadas submetidas devem estar dentro dos limites do tabuleiro;
- As coordenadas de partida têm obrigatoriamente de conter uma peça da cor do jogador;
- O movimento das peças devem ser respeitadas. Por exemplo, o cavalo só poderá andar em L;
- A coordenada destino não poderá conter uma peça da mesma cor do jogador que está a jogar. Por exemplo, caso seja o turno do jogador das peças brancas, qualquer coordenada destino de uma peça branca não poderá conter nenhuma peça branca (será válido se a coordenada destino estiver vazia ou tiver uma peça preta);
- A peça escolhida para mover não for da mesma cor das peças do jogador. Por exemplo, o jogador das peças brancas não poderá mover uma peça de cor preta;

Universidade Lusitana de Humanidades e Tecnologias
LEI / LIG / LEIRT
Fundamentos de Programação
2020/2021
1º Semestre - 1ª época - 2ª parte
Bruno Saraiva, João Pedro Carvalho, Pedro Alves
v1.0.1

Qualquer resposta inválida, seja ela o movimento fora do tabuleiro, uma dimensão do tabuleiro inválida, um movimento inválido de uma peça, etc, terá de ser enviado para o utilizador a seguinte resposta:

`Invalid response.`

Note-se que esta resposta deve aparecer logo na linha a seguir à situação que gerou o erro.

Tenham atenção na introdução de colunas e linhas (terão de ser ambas maiores do que 4) e caso sejam, terá de ser validado se existe algum tabuleiro pré-definido com as dimensões seleccionadas.

Por fim, é necessário verificar que, quando um dos jogadores não tiver mais nenhuma peça em no tabuleiro, uma mensagem é retornada:

`Congrats! Jon Snow wins!`

E o programa volta ao menu principal.

Para mais detalhe, no fim deste documento encontram-se 2 exemplos de flows do projecto.

Funções de implementação obrigatória

Seguindo as boas práticas da programação, o projeto deverá fazer uso intensivo de funções de forma a facilitar a legibilidade e compreensão do mesmo.

Para facilitar esse processo, descrevem-se de seguida um conjunto de funções obrigatórias que deverão implementar e usar para atingir os objetivos enunciados na secção anterior.

Nota importante: Estas funções não devem escrever nada no écran (ou seja, não devem usar o `print/println`).

Funções obrigatórias

Assinatura	Comportamento
<pre>fun buildBoard(numColumns: Int, numLines: Int, showLegend: Boolean= false, showPieces: Boolean= false, pieces: Array<Pair<String, String>?>): String</pre>	Devolve uma string que contém a informação do tabuleiro. Devem adaptar a função que fizeram na parte 1 para construir o tabuleiro a partir do array de peças que é passado como parâmetro.
<pre>fun createInitialBoard(numColumns: Int, numLines: Int): Array<Pair<String, String>?></pre>	A partir do número de colunas e linhas, será necessário construir e retornar um array de Pair (peças do tabuleiro). Esta estrutura está explicada na secção "Estrutura do jogo".
<pre>fun createTotalPiecesAndTurn(numColumns: Int, numLines: Int): Array<Int>?</pre>	A partir do número de colunas e linhas, será necessário construir e retornar um array de 3 posições de Int. Na primeira

Universidade Lusitana de Humanidades e Tecnologias
LEI / LIG / LEIRT
Fundamentos de Programação
2020/2021
1º Semestre - 1ª época - 2ª parte
Bruno Saraiva, João Pedro Carvalho, Pedro Alves
v1.0.1

	posição é identificado o número de peças brancas, na seguinte o número de peças pretas e por fim turno atual (0 para branco e 1 para preto). Exemplo: arrayOf(16,16,0). Esta estrutura está explicada em detalhe na secção Estrutura do jogo.
<pre>fun convertStringToUnicode(piece: String, color: String): String</pre>	A partir do tipo de peça e da côr, devolve o Unicode respetivo. Caso a peça e/ou a côr sejam inválidos, deve retornar uma String com um espaço (" "). Verão que este espaço vai facilitar o desenho do tabuleiro.
<pre>getCoordinates (readText: String?): Pair<Int, Int>?</pre>	A partir de uma string (por exemplo "2a"), esta função converte as coordenadas em números (por exemplo return Pair(2,2) com a string de entrada de "2a"). Se as coordenadas forem inválidas, deve retornar null. A String passada deve ser <i>case insensitive</i> , ou seja, aceitar "2a" ou "2A".
<pre>fun checkRightPieceSelected(pieceColor: String, turn: Int): Boolean</pre>	Valida se a peça escolhida para mover é do respectivo jogador. Por exemplo, se o jogador das peças brancas (turno 0), escolheu bem a peça para mover (côr "w").
<pre>fun isCoordinateInsideChess (coord: Pair<Int, Int>, numColumns: Int, numLines: Int): Boolean</pre>	Valida se as coordenadas escolhidas estão dentro do tabuleiro.
<pre>fun isValidTargetPiece(currentSelectedPiece : Pair<String, String>, currentCoord : Pair<Int, Int>, targetCoord : Pair<Int, Int>, pieces : Array<Pair<String, String>?>, numColumns: Int, numLines: Int): Boolean</pre>	Valida se o movimento da peça é válido. Se sim retorna true e false caso contrário. (Dica: Esta função irá ser chamada dentro da função movePiece).
<pre>fun movePiece(pieces : Array<Pair<String, String>?>, numColumns: Int, numLines: Int, currentCoord: Pair<Int, Int>, targetCoord: Pair<Int, Int>, totalPiecesAndTurn : Array<Int>): Boolean</pre>	Esta função, irá alterar o valor do argumento pieces e do totalPiecesAndTurn. Só serão alterados caso o movimento das peças sejam válidas. Se forem alterados é retornado true e false caso contrário. (Dica: esta função irá ser chamada dentro da função startNewGame)
<pre>fun startNewGame (whitePlayer: String, blackPlayer: String, pieces : Array<Pair<String, String>?>,</pre>	É nesta função que é mostrado o tabuleiro e é onde pede constantemente para cada jogador colocar as coordenadas de partida

totalPiecesAndTurn : Array<Int?>, numColumns: Int,numLines: Int, showLegend: Boolean= false, showPieces: Boolean = false)	e de origem. Também é aqui que dá a opção de retornar ao menu principal, clicando na tecla “m”. É de notar que os argumentos whitePlayer e blackPlayer são os nomes dados pelos utilizador, em que o whitePlayer é o nome do 1º jogador e o blackPlayer é o nome do 2º jogador.
fun isHorseValid(currentCoord: Pair<Int, Int>,targetCoord : Pair<Int, Int>,pieces : Array<Pair<String, String>?>,numColumns: Int, numLines: Int): Boolean	Dado as coordenadas do cavalo (currentCoord), as coordenadas de destino (targetCoord) e as peças do tabuleiro (pieces) retornar se o movimento é válido (true) ou não (false) Esta função não deve alterar o tabuleiro, apenas verificar se o movimento é válido.
fun isKingValid(currentCoord: Pair<Int, Int>,targetCoord : Pair<Int, Int>,pieces: Array<Pair<String, String>?>,numColumns: Int,numLines: Int):Boolean	Dado as coordenadas do rei (currentCoord), as coordenadas de destino (targetCoord) e as peças do tabuleiro (pieces) retornar se o movimento é válido (true) ou não (false) Esta função não deve alterar o tabuleiro, apenas verificar se o movimento é válido.
fun isTowerValid(currentCoord: Pair<Int, Int>,targetCoord: Pair<Int, Int>,pieces: Array<Pair<String, String>?>,numColumns: Int,numLines: Int):Boolean	Dado as coordenadas da Torre (currentCoord), as coordenadas de destino (targetCoord) e as peças do tabuleiro (pieces) retornar se o movimento é válido (true) ou não (false) Esta função não deve alterar o tabuleiro, apenas verificar se o movimento é válido.
fun isBishopValid(currentCoord: Pair<Int, Int>,targetCoord: Pair<Int, Int>,pieces: Array<Pair<String, String>?>,numColumns: Int,numLines: Int): Boolean	Dado as coordenadas do Bispo (currentCoord), as coordenadas de destino (targetCoord) e as peças do tabuleiro (pieces) retornar se o movimento é válido (true) ou não (false) Esta função não deve alterar o tabuleiro, apenas verificar se o movimento é válido.
fun isQueenValid(currentCoord: Pair<Int, Int>,targetCoord: Pair<Int, Int>,pieces: Array<Pair<String, String>?>,numColumns: Int,numLines: Int):Boolean	Dado as coordenadas da Rainha (currentCoord), as coordenadas de destino (targetCoord) e as peças do tabuleiro (pieces) retornar se o movimento é válido (true) ou não (false) Esta função não deve alterar o tabuleiro, apenas verificar se o movimento é válido.
fun isKnightValid(currentCoord: Pair<Int, Int>,targetCoord: Pair<Int, Int>,pieces: Array<Pair<String, String>?>,numColumns: Int,numLines: Int):Boolean	Dado as coordenadas do Peão (currentCoord), as coordenadas de destino (targetCoord), as peças do tabuleiro (pieces), o turno actual

Universidade Lusitana de Humanidades e Tecnologias
LEI / LIG / LEIRT
Fundamentos de Programação
2020/2021
1º Semestre - 1ª época - 2ª parte
Bruno Saraiva, João Pedro Carvalho, Pedro Alves
v1.0.1

	(totalPiecesAndTurn), retornar se o movimento é válido (true) ou não (false) Esta função não deve alterar o tabuleiro, apenas verificar se o movimento é válido.
--	---

Podem e devem implementar funções adicionais se isso ajudar a organizar o vosso código. Funções com demasiadas linhas de código (incluindo o main()) levarão a penalizações na componente de qualidade de código.

Entrega e Avaliação

Artefactos a Entregar

A entrega deve ser feita através do Drop Project usando um ficheiro comprimido (formato **.zip**) dentro do qual se encontrem:

- Uma pasta com o nome “src” com todo o código necessário para compilar e executar o projecto.
- Um ficheiro de texto (chamado AUTHORS.txt) contendo os nomes e números de aluno).

Formatos de Entrega

O ficheiro .zip deve seguir a estrutura que se indica de seguida:

```
AUTHORS.txt (contém linhas NUMERO_ALUNO;NOME_ALUNO,
uma por aluno do grupo)

+ src
|--- Main.kt
|--- ... (outros ficheiros kotlin do projecto)
```

Restrições técnicas

Universidade Lusófona de Humanidades e Tecnologias
LEI / LIG / LEIRT
Fundamentos de Programação
2020/2021
1º Semestre - 1ª época - 2ª parte
Bruno Saraiva, João Pedro Carvalho, Pedro Alves
v1.0.1

A versão do Kotlin ensinada nas aulas e que é oficialmente suportada pelo Drop Project é a versão 1.3.X.

Como Entregar

O projecto será entregue via Drop Project (DP), através do link: <https://deisi.ulusofona.pt/drop-project/upload/fp-projecto-20-21-p2>

Não serão aceites entregas por outra via.

Os alunos são incentivados a testar o projecto no DP à medida que o vão implementando. Podem e devem fazer quantas submissões acharem necessárias. Para efeitos de avaliação será considerada a melhor submissão feita dentro do prazo.

Prazos de Entrega

A data limite de entrega é o dia **10 de Janeiro de 2021**, pelas **23h00** (hora de Lisboa).

Os alunos que entreguem depois do prazo, ainda durante o dia 10 de Janeiro, até às 23h30, **terão uma penalização de 5 valores** na nota final desta parte do projecto.

Não serão aceites entregas após dia 10 de Janeiro às 23h30. Nesse caso os alunos terão nota zero no projeto, **reprovando na componente prática da disciplina (1ª época)**.

Avaliação

A avaliação do projeto será feita considerando as entregas feitas pelos alunos em ambas as partes. A nota será divulgada no final de cada entrega.

Após a entrega final, será atribuída ao projecto uma nota final quantitativa, que será calculada considerando a seguinte fórmula:

$$0.25 * \text{NotaParte1} + 0.75 * \text{NotaParte2}$$

Cotações da Segunda Parte

Universidade Lusitana de Humanidades e Tecnologias
LEI / LIG / LEIRT
Fundamentos de Programação
2020/2021
1º Semestre - 1ª época - 2ª parte
Bruno Saraiva, João Pedro Carvalho, Pedro Alves
v1.0.1

A avaliação do projecto será feita através dos seguintes tópicos. Notem que esta tabela só se aplica a projetos que não tenham erros de compilação no Drop Project. Projetos com erros de compilação terão zero, implicando reprovação na componente prática (1ª época).

Tópico	Descrição	Pontuação (0..20)
Qualidade de código	O código cumpre as boas práticas de programação ensinadas nas aulas (nomes apropriados para as variáveis e funções em camelCase, utilização do val e do var, código bem indentado, funções que não sejam demasiado grandes, evitar usar o !!, etc.).	3
Testes automáticos	Será aplicada uma bateria de testes automáticos cujo relatório poderá ser consultado após cada submissão. Quantos mais testes passarem, melhor nota terão nesta componente.	14
Avaliação manual da aplicação	Os professores farão testes adicionais assim como inspeção de código para avaliar esta componente	3

Cópias

Trabalhos que sejam identificados como cópias serão anulados e os alunos que os submetam terão nota zero em ambas as partes do projeto (quer tenham copiado, quer tenham deixado copiar). Para evitar situações deste género, recomendamos aos alunos que nunca partilhem ou mostrem o código do seu projecto a pessoas fora do grupo de trabalho.

A decisão sobre se um trabalho é uma cópia cabe exclusivamente aos docentes da unidade curricular.

Outras Informações Relevantes

- Os projetos devem ser realizados em grupos de 2 alunos. Excepcionalmente poderão ser aceites trabalhos individuais, desde que seja pedida autorização prévia ao Professor das aulas práticas respetivo e desde que exista uma justificação razoável.

v1.0.1

- O grupo é formado automaticamente pelo Drop Project quando fazem a primeira submissão (através do ficheiro AUTHORS.txt). Submissões individuais que não tenham sido previamente autorizadas por um professor serão eliminadas do Drop Project.
- Existirá uma defesa presencial e individual do projeto, realizada após a entrega da 2ª parte. Durante esta defesa individual, será pedido ao aluno que faça alterações ao código para dar resposta a alterações aos requisitos. Da discussão presencial de cada aluno, resultará uma nota de 0 a 100%, que será aplicada à nota do projeto (primeira e segunda parte).
- Na segunda parte do projecto devem-se manter os grupos definidos para a primeira parte. Não será permitida a entrada de novos membros nos grupos.
- O ficheiro .zip entregue deve seguir escrupulosamente as regras de estrutura indicadas neste enunciado. Ficheiros que não respeitem essa estrutura terão nota zero.
- Trabalhos cujo código não compile e/ou não corra não serão avaliados e terão automaticamente nota zero.
- Grupos que não entreguem a primeira parte ou tenham nota zero na mesma (seja por cópia, não compilação ou outra das situações indicadas no enunciado), não podem entregar a segunda parte do projecto, e reprovam automaticamente na componente prática da disciplina (de primeira época).
- É possível que sejam feitas alterações a este enunciado, durante o tempo de desenvolvimento do projeto. Por esta razão, os alunos devem estar atentos ao Moodle de FP.
- Eventuais situações omissas e/ou imprevistas serão analisadas caso a caso pelos docentes da cadeira.

Exemplo de um possível flow do programa

Welcome to the Chess Board Game!

1-> Start New Game;

2-> Exit Game.

1

First player name?

João Carvalho

Second player name?

Bruno Saraiva

How many chess columns?

5

How many chess lines?

9

Invalid response.

How many chess columns?

6

How many chess lines?

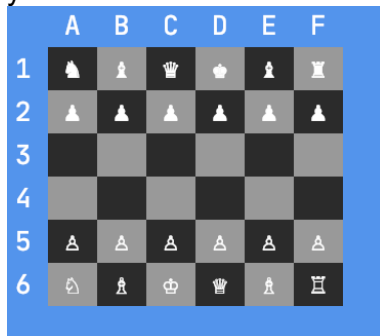
6

Show legend (y/n)?

y

Show pieces (y/n)?

y

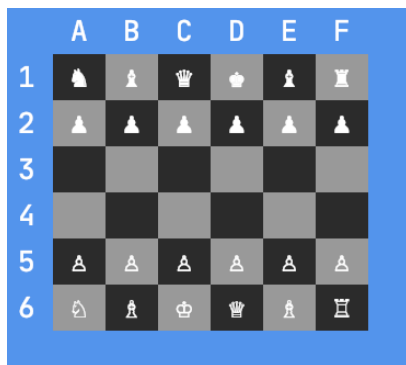


João Carvalho, choose a piece (e.g 2D).

Menu-> m;

2b

Invalid response.



João Carvalho, choose a piece (e.g 2D).

Menu-> m;

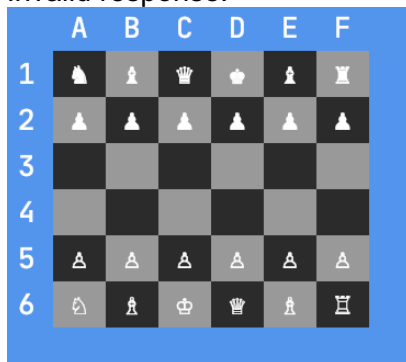
5b

João Carvalho, choose a target piece (e.g 2D).

Menu-> m;

4a

Invalid response.



João Carvalho, choose a piece (e.g 2D).

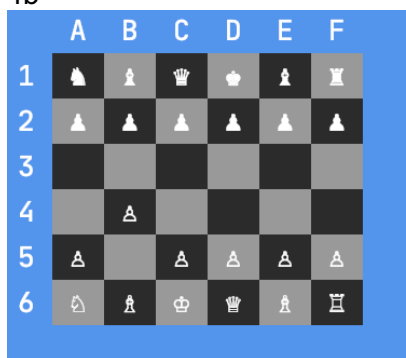
Menu-> m;

5b

João Carvalho, choose a target piece (e.g 2D).

Menu-> m;

4b



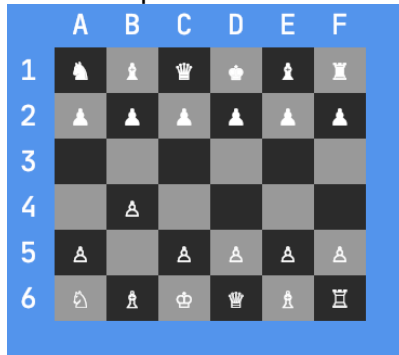
Universidade Lusitana de Humanidades e Tecnologias
LEI / LIG / LEIRT
Fundamentos de Programação
2020/2021
1º Semestre - 1ª época - 2ª parte
 Bruno Saraiva, João Pedro Carvalho, Pedro Alves
v1.0.1

Bruno Saraiva, choose a piece (e.g 2D).

Menu-> m;

6e

Invalid response.



Bruno Saraiva, choose a piece (e.g 2D).

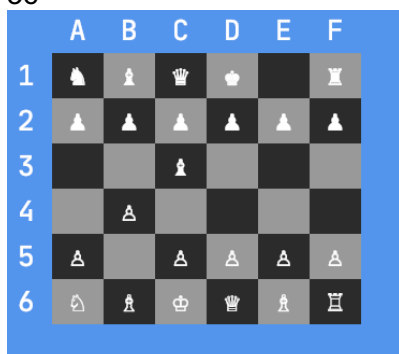
Menu-> m;

1e

Bruno Saraiva, choose a target piece (e.g 2D).

Menu-> m;

3c



João Carvalho, choose a piece (e.g 2D).

Menu-> m;

6a

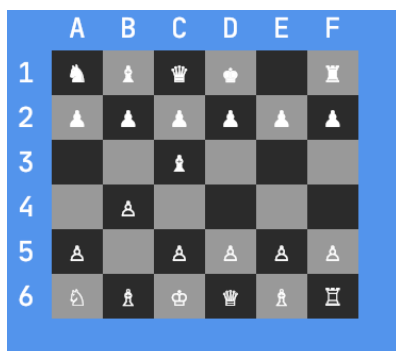
João Carvalho, choose a target piece (e.g 2D).

Menu-> m;

4a

Invalid response.

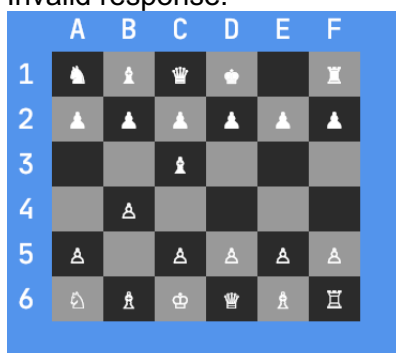
Universidade Lusitana de Humanidades e Tecnologias
LEI / LIG / LEIRT
Fundamentos de Programação
2020/2021
1º Semestre - 1ª época - 2ª parte
Bruno Saraiva, João Pedro Carvalho, Pedro Alves
v1.0.1



João Carvalho, choose a piece (e.g 2D).
Menu-> m;

6g

Invalid response.

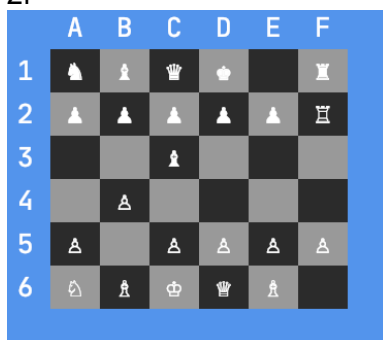


João Carvalho, choose a piece (e.g 2D).
Menu-> m;

6f

João Carvalho, choose a target piece (e.g 2D).
Menu-> m;

2f



Bruno Saraiva, choose a piece (e.g 2D).
Menu-> m;

3c

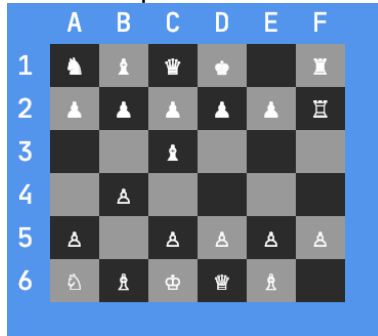
Bruno Saraiva, choose a target piece (e.g 2D).

Universidade Lusitana de Humanidades e Tecnologias
LEI / LIG / LEIRT
Fundamentos de Programação
2020/2021
1º Semestre - 1ª época - 2ª parte
 Bruno Saraiva, João Pedro Carvalho, Pedro Alves
v1.0.1

Menu-> m;

2d

Invalid response.



Bruno Saraiva, choose a piece (e.g 2D).

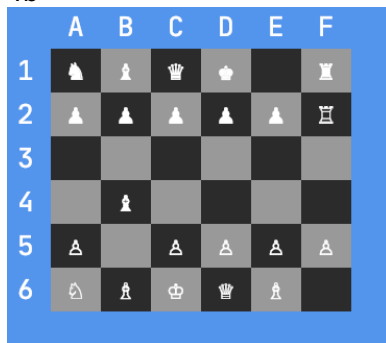
Menu-> m;

3c

Bruno Saraiva, choose a target piece (e.g 2D).

Menu-> m;

4b



João Carvalho, choose a piece (e.g 2D).

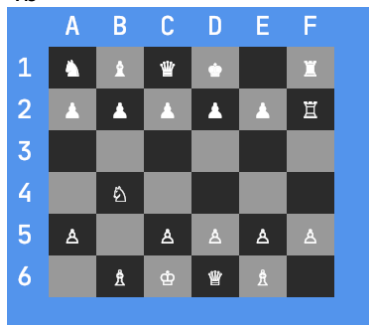
Menu-> m;

6a

João Carvalho, choose a target piece (e.g 2D).

Menu-> m;

4b



Universidade Lusitana de Humanidades e Tecnologias
LEI / LIG / LEIRT
Fundamentos de Programação
2020/2021
1º Semestre - 1ª época - 2ª parte
Bruno Saraiva, João Pedro Carvalho, Pedro Alves
v1.0.1

Bruno Saraiva, choose a piece (e.g 2D).
Menu-> m;

m
1-> Start New Game;
2-> Exit Game.

2

Exemplo de um outro possível flow do programa

Welcome to the Chess Board Game!
1-> Start New Game;
2-> Exit Game.

1
First player name?

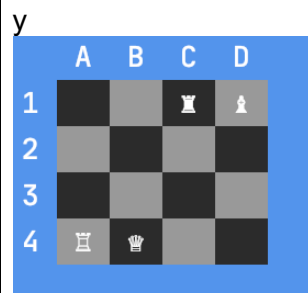
Pedro Alves
Second player name?

Bruno Saraiva
How many chess columns?

4
How many chess lines?

4
Show legend (y/n)?

y
Show pieces (y/n)?



Pedro Alves, choose a piece (e.g 2D).
Menu-> m;

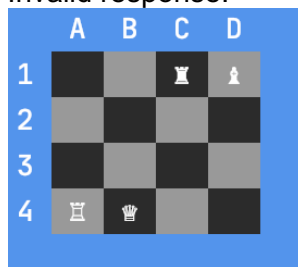
4b

Pedro Alves, choose a target piece (e.g 2D).

Menu-> m;

2a

Invalid response.



Pedro Alves, choose a piece (e.g 2D).

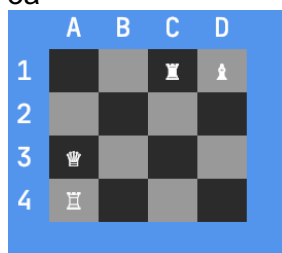
Menu-> m;

4b

Pedro Alves, choose a target piece (e.g 2D).

Menu-> m;

3a



Bruno Saraiva, choose a piece (e.g 2D).

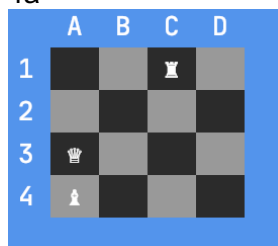
Menu-> m;

1d

Bruno Saraiva, choose a target piece (e.g 2D).

Menu-> m;

4a



Pedro Alves, choose a piece (e.g 2D).

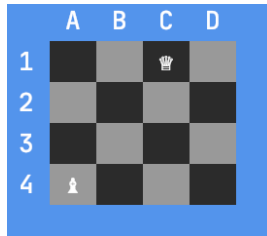
Menu-> m;

3a

Pedro Alves, choose a target piece (e.g 2D).

Menu-> m;

1c



Bruno Saraiva, choose a piece (e.g 2D).

Menu-> m;

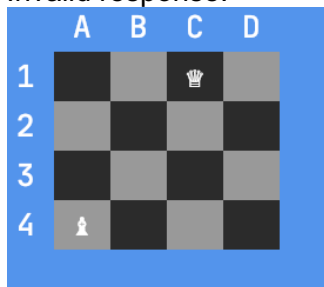
4a

Bruno Saraiva, choose a target piece (e.g 2D).

Menu-> m;

1c

Invalid response.



Bruno Saraiva, choose a piece (e.g 2D).

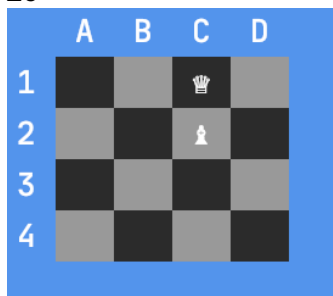
Menu-> m;

4a

Bruno Saraiva, choose a target piece (e.g 2D).

Menu-> m;

2c



Pedro Alves, choose a piece (e.g 2D).

Menu-> m;

1c

Pedro Alves, choose a target piece (e.g 2D).

Menu-> m;

2c

Congrats! Pedro Alves wins!

1-> Start New Game;

2-> Exit Game.

2