



UNIVERSIDADE
LUSÓFONA

Enunciado do projeto prático

“Jogo de Tabuleiro - Xadrez”

Primeira 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 Primeira Parte.

O enunciado da Segunda Parte será publicado em data posterior no *Moodle*.

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.

O jogo de xadrez é constituído por 32 peças, 16 para cada equipa branca (w) e 16 para a equipa preta (b). No entanto, nesta primeira parte do projeto irá ser apenas constituído por um tipo de peça (Peão neste caso). Também terá um tabuleiro dinâmico, isto é, o número de colunas e linhas podem variar consoante o pedido do utilizador.

Objetivos da Primeira Parte

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

- Criação dos menus do jogo;
- Recolha e validação dos nomes dos jogadores;
- Apresentação do tabuleiro de Xadrez com a opção de mostrar as peças e/ou legenda.

É de realçar que nesta primeira parte do trabalho apenas existirá a peça Peão (branco e preto) e não será implementado o movimento das peças.

Menus / Ecrãs do Jogo

O **menu principal** deverá ter a seguinte estrutura:

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

O jogo deverá ser um ciclo que termina com a escolha **0**.

Ao escolher a opção **1** deverá perguntado qual o nome do primeiro jogador:

```
First player name?  
  
>Jon Snow
```

A resposta deverá conter, no mínimo, 2 nomes (nome próprio e apelido) a começar com letra **maiúscula** em cada um. Caso um dos nomes não comece com letra maiúscula é avisado ao utilizador que inseriu incorretamente o nome e é perguntado novamente o nome do primeiro jogador. Exemplo:

```
First player name?
```

```
>Jon snow
```

```
Invalid response.
```

```
First player name?
```

```
>
```

```
First player name?
```

```
>Jon snow
```

```
Invalid response.
```

```
First player name?
```

```
>
```

Caso seja escrito corretamente o nome do 1º jogador, o programa deverá pedir o nome do 2º jogador:

```
First player name?
```

```
>Jon Snow
```

```
Second player name?
```

```
>Tyrion Lannister
```

Caso não seja inserido corretamente o nome do 2º jogador, é também enviado um alerta e o programa deverá voltar a perguntar o nome do segundo jogador:

```
First player name?
```

```
>Jon Snow
```

```
Second player name?
```

```
>Tyrion lannister
```

```
Invalid response.
```

```
Second player name?
```

```
>
```

Caso seja inserido o nome do 2º jogador corretamente, é então perguntado, por partes, quantas linhas, colunas, se irá ser mostrado legenda e as peças no tabuleiro:

First player name?

>Jon Snow

Second player name?

>Tyrion Lannister

How many chess columns?

>5

How many chess lines?

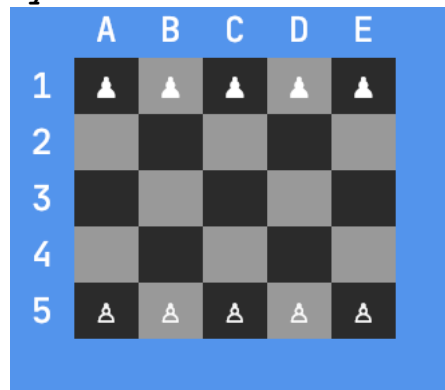
>5

Show legend (y/n)?

>y

Show pieces (y/n)?

>y



1-> Start New Game;

0-> Exit Game;

2 outros exemplos são também mostrados:

First player name?

>Jon Snow

Second player name?

>Tyrion Lannister

How many chess columns?

>5

How many chess lines?

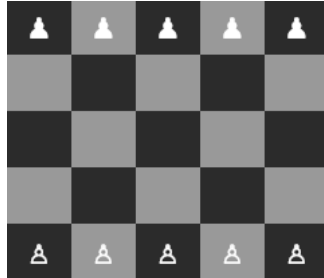
>5

Show legend (y/n)?

>n

Show pieces (y/n)?

>y



1-> Start New Game;

0-> Exit Game;

First player name?

>Jon Snow

Second player name?

>Tyrion Lannister

How many chess columns?

>5

How many chess lines?

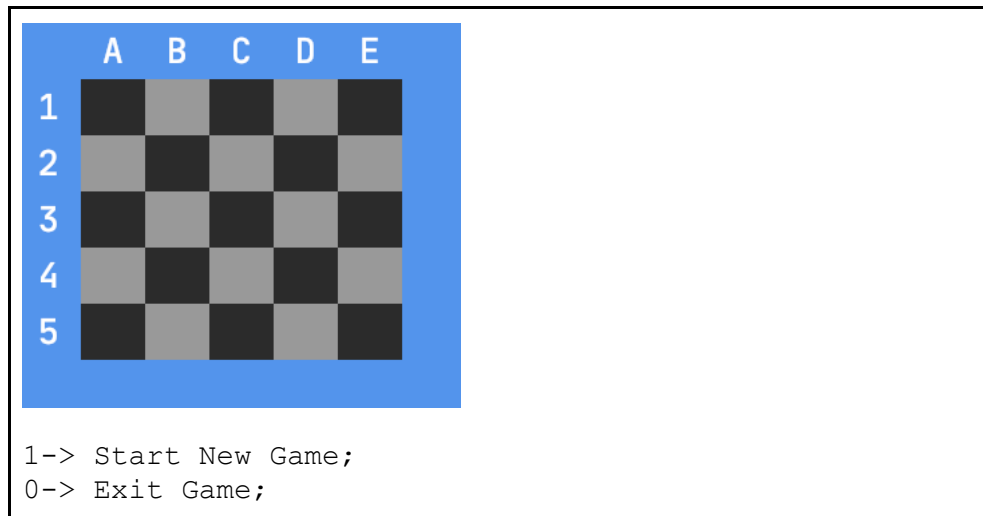
>5

Show legend (y/n)?

>y

Show pieces (y/n)?

>n



Em todos os casos será necessário verificar se o utilizador escreveu apenas números onde é pedido o número de colunas e linhas. Também será necessário verificar que apenas será respondido y/Y ou n/N nas perguntas de mostrar a legenda e peças no tabuleiro de xadrez. Caso as respostas não sejam nem números nem letras, nas respetivas perguntas, uma resposta inválida é gerada pelo computador e mostrado ao utilizador:

Exemplo1:

```
First player name?  
>Jon Snow  
Second player name?  
>Tyrion Lannister  
How many chess columns?  
>5  
How many chess lines?  
>h  
Invalid response.  
How many chess lines?  
>
```

Exemplo2:

```
First player name?  
  
>Jon Snow  
Second player name?  
  
>Tyrion Lannister  
How many chess columns?  
  
>5  
How many chess lines?  
  
>5  
Show legend (y/n)?  
  
>hey  
Invalid response.  
Show legend (y/n)?  
  
>
```

Nota: Sempre que se insere o número de colunas ou o número de linhas, é necessário verificar se o número não é inferior a 5. Caso seja inferior a 5, deverá ser gerado uma resposta inválida e o programa deverá perguntar um novo número de colunas ou linhas (onde o utilizador errou na resposta):

```
First player name?  
  
>Jon Snow  
Second player name?  
  
>Tyrion Lannister  
How many chess columns?  
  
>3  
Invalid response.  
How many chess columns?  
  
>5  
How many chess lines?  
  
>4  
Invalid response.  
How many chess lines?  
  
>
```

Tabuleiro

Sendo uma das partes mais importantes do trabalho (nesta primeira parte), é muito importante o tabuleiro estar bem apresentado. Para isso o aluno deverá ter atenção nas cores do tabuleiro e na peça escolhida. Relativamente à peça escolhida, nesta primeira parte será apenas tido em conta o peão. Para isso basta utilizar os Unicodes seguintes: **2659** para o peão branco e **265F** para o peão preto.

Relativamente às cores de fundo do tabuleiro, baseado no código *escape* (https://en.wikipedia.org/wiki/ANSI_escape_code), haverão **3** tipos de cores:

- `val startBlue = "$esc[30;44m"`
- `val startGrey = "$esc[30;47m"`
- `val startWhite = "$esc[30;30m"`

Onde a variável *esc* tem o seguinte valor:

- `val esc: String = Character.toString(27)`



Nota: como se pode verificar, para começar um código *escape*, será sempre necessário começar com o valor ASCII 27 seguido do símbolo '['. Depois destes 2 caracteres, existe uma lista de opções possíveis. No entanto, para este trabalho, apenas será modificado a cor do texto (representado pelo valor 30) e a cor de fundo (representado pelas cores 44, 47 e 30).

No fim de cada linha do tabuleiro deverá ser impresso um end de cores:

- `val end = "$esc[0m"`

O exemplo seguinte, mostra um print a um quadrado azul com um espaçamento de **3** e um cinzento com espaçamento de **5**:

- Cór azul com espaçamento de 3: `println("$startBlue $end")`
- Cór cinzenta com espaçamento de 5: `println("$startGrey $end")`

```
 <- Fundo azul com espaçamento de 3  
 <- Fundo cinzento com espaçamento de 5
```

O número do espaçamento, nesta primeira parte, deverá ser sempre igual a **3**.

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 ecrã (ou seja, não devem usar o `print()/println()`).

Funções obrigatórias

Assinatura	Comportamento
<code>fun buildMenu(): String</code>	Devolve uma string das opções do menu (start new game e exit).
<code>fun buildBoard(numColumns: Int, numLines: Int, showLegend: Boolean, showPieces: Boolean): String</code>	<p>É neste método que será construído o tabuleiro pedido. No fim deste método, será necessário retornar uma string que contém a informação toda do tabuleiro (incluindo as cores, legenda, peças, etc). Os parâmetros <code>showLegend</code> e <code>showPieces</code> refletem a resposta às respetivas perguntas que são feitas quando se inicia o jogo (ver "Menus/Écrans do Jogo").</p> <p>Importante: A função deve estar preparada para ser executada apenas com 2 ou 3 parâmetros, usando valores por omissão. Os valores por omissão para o 3º e 4º parâmetros são false.</p>
<code>fun checkIsNumber(number: String): Boolean</code>	Verificar se o argumento de entrada será um número ou não. Se sim retornar true e false caso contrário.
<code>fun checkName(number: String): Boolean</code>	Verificar se o argumento de entrada terá, no mínimo 2 nomes (primeiro e apelido) a começar com letra maiúscula cada um. Se sim, retornar true e false caso contrário.
<code>fun showChessLegendOrPieces(message: String): Boolean?</code>	Verificar se o argumento de entrada será y/Y, n/N ou outro caracter. Caso seja y ou Y deverá ser retornado true. Será retornado false caso seja n ou N e null para o resto.

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 projeto.
- 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 projeto)
```

Restrições técnicas

O projeto deverá utilizar apenas as instruções ensinadas nas aulas até à aula 9. Nomeadamente não deverão ser utilizados arrays, indexOf(), contains(), split(), listas, maps, Pairs, etc. Também não deverão criar classes novas. Projetos que usem estas instruções poderão ter fortes penalizações. Na dúvida, deverão contactar o vosso professor das aulas práticas.

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

Como Entregar

O projeto será entregue via Drop Project (DP), através do link: <https://deisi.ulusofona.pt/drop-project/upload/fp-projecto-20-21-p1>

Não serão aceites entregas por outra via.

Os alunos são incentivados a testar o projeto 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 **14 de Dezembro de 2020**, pelas **8h00** (hora de Lisboa).

Os alunos que entreguem depois do prazo, ainda durante o dia 14 de Dezembro, até às 23h30, **terão uma penalização de 5 valores** na nota final desta parte do projeto.

Não serão aceites entregas após dia 14 de Dezembro 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 projeto uma nota final quantitativa, que será calculada considerando a seguinte fórmula:

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

Cotações da Primeira Parte

A avaliação do projeto será feita através dos seguintes tópicos.

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. Quanto 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 projeto 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. Excecionalmente poderão ser aceites trabalhos individuais, desde que seja pedida autorização prévia ao Professor das aulas práticas respetivo e desde que exista uma justificação razoável.
- 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 projeto 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 projeto, 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.
- Todas as dúvidas devem ser colocadas no piazza.
- Eventuais situações omissas e/ou imprevistas serão analisadas caso a caso pelos docentes da cadeira.