DEISI Chess, Primeira Parte V1.0.0

Bruno Cipriano, Duarte Neves, Pedro Alves

Enunciado do projecto prático - Primeira Parte

Objectivo

Este projecto tem como objectivo o desenvolvimento de uma aplicação (programa) em **linguagem Java** (versão Java 17) e **Kotlin** (versão 1.9) aplicando os conceitos de modelação e Programação **Orientada a Objetos** (encapsulamento, herança, polimorfismo, etc.) e **Programação Funcional**.

O projecto está dividido em 2 partes:

- Primeira Parte - apresentada no presente enunciado, incide na modelação, e implementação do modelo e de um conjunto de funcionalidades associadas;

- Segunda Parte - onde se apresentam novos requisitos, que poderão (ou não) requerer a alteração do modelo submetido na Primeira Parte e implementação de novas funcionalidades. Inclui também a necessidade de utilização de conceitos e técnicas de Programação Funcional

As 2 partes são de entrega obrigatória e terão prazos de entrega distintos. O não cumprimento dos prazos de entrega de qualquer uma das partes do projecto levará automaticamente à reprovação dos alunos na avaliação de primeira época da componente prática.

DEISI Chess, Primeira Parte V1.0.0

Bruno Cipriano, Duarte Neves, Pedro Alves

Objectivos - Primeira Parte

Nesta primeira parte, os alunos terão de:

- Criar um **diagrama de classes** em **UML** que seja suficiente para modelar a realidade apresentada e responder aos requisitos funcionais apresentados ao longo do enunciado.
- Implementar o modelo proposto, usando a linguagem **Java 17**.
- Implementar algumas funcionalidades básicas, também em **Java 17**.
- Criar testes automáticos unitários para uma parte do modelo implementado, usando **JUnit 5**.

Restrições Técnicas - Primeira Parte

Nesta primeira parte do projecto, o modelo de dados (e respectiva implementação) **não pode usar** o mecanismo **de herança** e também não pode usar o mecanismo de ***Interfaces***.

Também não é permitida a utilização de streams.

Uma vez que estamos no mundo “puramente” orientado a objectos, desaconselhamos fortemente a utilização de variáveis ou funções static (globais)

Descrição do Projeto

O projecto encontra-se descrito em vídeo:

https://www.youtube.com/watch?v=EMKTKI4IZXI

Poderão ser identificados erros/omissões no vídeo. Devem estar atentos à errata que está publicada no primeiro comentário do vídeo.

Caso tenham dúvidas sobre o conteúdo do vídeo, devem colocá-las sob a forma de comentários no youtube, indicando o timestamp do vídeo ao qual diz respeito a pergunta.

Testes Unitários Automáticos

Nesta componente, os alunos devem implementar pelo menos **4 casos de teste** a métodos do GameManager que façam parte da API (ver secção API). Recomendamos que utilizem nos vossos testes apenas os métodos da API pois é isso que vai acontecer no Drop Project.

Estes casos de teste devem ser implementados usando **JUnit 5**, tal como demonstrado nas aulas.

Os testes devem ser implementados numa classe com o nome `TestGameManager`. Essa classe deve estar no mesmo package que o restante projeto.

Nota: para que os testes sejam considerado válidos, as condições seguintes têm de se verificar:

- A função de teste não pode estar vazia;
- A função de teste tem de ter pelo menos um assert;
- A função de teste tem de chamar código do projecto;
- O assert tem de comparar o resultado de uma função do projecto com um resultado esperado.

Caso precisem de ler ficheiros nos vossos testes, estes devem ser colocados na pasta test-files (ver secção “Estrutura do projeto” mais à frente). Devem aceder aos ficheiros usando caminhos relativos. Por exemplo:

```
File ficheiro = new File("test-files/somefile1.txt")
```

Criatividade

Nesta primeira parte, esta componente é opcional e não conta para a avaliação. No entanto, os alunos são convidados a “personalizar” o seu DEISI Chess, de várias formas:

- Podem definir novas imagens para as peças, através da função `getSquareInfo()`
- Podem usar a função `getAuthorsPanel()` para personalizar a janela de créditos.

Se realmente decidirem personalizar o vosso jogo, coloquem capturas de ecrã demonstrativas da vossa criatividade no ficheiro README.md (ver instruções mais à frente). Os trabalhos mais criativos serão divulgados numa aula teórica.

Formação dos grupos

Os projetos devem ser realizados em grupos de 2 alunos. Em situações excepcionais poderão ser aceites grupos de um aluno - quem o pretender fazer deve efectuar um pedido (justificado) aos docentes da cadeira através de e-mail. **Alunos que entreguem individualmente sem terem um pedido aceite por um professor terão nota zero.**

Os elementos do grupo devem pertencer à mesma turma prática. Isto facilita o acompanhamento do projeto por parte do professor respetivo.

Os elementos do grupo devem ter um nível de conhecimento mais ou menos equivalente. Alguns exemplos de grupos cuja formação deve ser evitada:

- Um dos elementos reprovou a FP e/ou AED mas o outro passou
- Um dos elementos faz FP e/ou AED com nota superior a 16 e o outro com nota inferior a 13
- Um dos elementos passou todos/a maioria dos testes do Mini-teste 1 e o outro elemento passou 1 ou menos testes

Os professores das práticas irão monitorizar a formação de grupos para garantir esta regra. Pela nossa experiência, quando há uma diferença grande de conhecimentos, o elemento mais fraco do grupo reprova na defesa. Queremos evitar isso.

Os grupos de alunos definidos para a primeira parte do projecto devem ser mantidos para a segunda parte do projecto. Não serão permitidas entradas de novos alunos nos grupos entre as duas partes do projecto.

Entrega

O que entregar

Nesta primeira parte do projecto, os alunos têm de entregar:

- Um diagrama de classes em **UML** (em formato `pdf` ou `png`), assim como eventuais comentários que os alunos decidam fazer no sentido de justificarem as suas escolhas de modelação;

- Ficheiros Java onde seja feita a implementação das diversas classes que façam parte do modelo.

- Ficheiros Java que implementem os testes unitários automáticos (JUnit).

Nota importante: O programa submetido pelos alunos tem de compilar e executar no contexto do visualizador e no contexto do Drop Project.

No visualizador:

- Carregar nos botões **não pode** resultar em erros de *runtime*. Projectos que não cumpram esta regra serão considerados como não sendo entregues. Isto significa que todos os métodos obrigatórios terão de estar implementados e a devolver valores que cumpram as restrições indicadas.

No Drop Project:

- Projectos que não passem as fases de **estrutura** e de **compilação** serão considerados como não entregues.

- Projectos que não passem a fase de **CheckStyle** serão penalizados em 3 valores.

Ficheiro README.md

O ficheiro README.md do repositório github deve contar os seguintes artefactos:

- Apresentação da imagem do diagrama UML do projecto;
- Apresentação (opcional) de capturas de ecrã, caso tenham personalizado o jogo. Também podem explicar a lógica por trás dessa personalização;
- Eventuais comentários que os alunos decidam fazer no sentido de justificarem as suas escolhas de modelação.

Para incluirem a imagem do vosso diagrama no README.md devem usar o código seguinte:

```
![](diagrama.png?raw=true "Diagrama UML")
```

Nota: Projectos que não tenham o ficheiro README.md ou cujo ficheiro README.md não inclua o diagrama UML terão uma penalização de 3 valores na nota final.

Estrutura do projecto

O projecto deve estar organizado na seguinte estrutura de pastas:

```
AUTHORS.txt  (contém linhas NUMERO_ALUNO;NOME_ALUNO, uma por aluno do grupo)
diagrama.pdf (ou.png)
+ src
|---+ pt
|------+ ulusofona
|---------+ lp2
|------------+ deisichess
|--------------- GameManager.java
|--------------- ...   (outros ficheiros java do projecto)
|--------------- TestGameManager.java
|---+ images   (devem colocar aqui imagens vossas para as peças)
+ test-files (devem colocar aqui os ficheiros que usem nos vossos testes)
```

Como entregar - Repósitorio git (github)

A entrega deste projecto deverá ser feita usando um repositório git. Não serão aceites outras formas de entrega do projecto.

Cada grupo deve criar um repositório git no *github* [github.com] e partilhar esse repositório com o Professor das aulas práticas.

Os vários alunos do grupo devem estar associados ao repositório *github* do grupo e devem usar o mesmo para trabalhar de forma colaborativa.

Todos os ficheiros a entregar (seja o relatório / UML, seja código) devem ser colocados no repositório git. O ficheiro que contiver o relatório / diagrama deve-se chamar diagrama.pdf (ou diagrama.png) e deve estar na **raiz** do projecto.

Notas:

- O user id / username de cada aluno no *github* tem de incluir o respectivo numero de aluno.

- Recomenda-se que o repositório tenha os números de aluno dos vários membros do grupo (p.e. "LP2 - 2100000 2100001").

A criação deste repositório e a sua partilha com o Professor é essencial para a entrega do projecto.

Os alunos terão também acesso a uma página no **Drop Project [DP]** a partir da qual poderão pedir para que o estado actual do seu repositório (ou seja, o *commit* mais recente) seja testado.

Nesta primeira parte do projecto, a página do DP a usar é a seguinte:

https://deisi.ulusofona.pt/drop-project/upload/lp2-2324-projeto-p1

Filosofia do uso do DP

O objectivo do DP não é servir de guião àquilo que os alunos têm que implementar. Os alunos devem implementar o projecto de forma autónoma tendo apenas em conta o enunciado e usar o DP apenas para validar que estão no bom caminho. Nas empresas nas quais um dia irão trabalhar não vão ter o DP para vos ajudar. Nesse sentido, decidimos limitar as submissões ao DP: passam a poder fazer uma submissão a cada 20 minutos (isto é, têm que esperar 20 min até que possam fazer nova submissão).

Regra dos Commits - Parte 1

Nesta primeira parte do projecto, deverão ser feitos (pelo menos) **dois (2) *commits* não triviais** (que tenham impacto na funcionalidade do programa), por cada aluno do grupo. Os alunos que não cumpram esta regra **serão penalizados em 3 valores** na nota final desta parte do projecto.

Prazo de entrega

A entrega deverá ser feita através de um *commit* no repositório git previamente criado e partilhado com o Professor. Recomenda-se que o repositório git seja criado (e partilhado) o mais rápido possível, de forma a evitar que surjam problemas de configuração no envio.

Para efeitos de avaliação do projecto, será considerado o último *commit* feito no repositório.

A data limite para fazer o último *commit* é o dia **6 de Novembro de 2023 (Segunda-feira)**, pelas **9h00m da manhã** (hora de Lisboa, Portugal). Recomenda-se que os alunos verifiquem que o *commit* foi enviado (*pushed*), usando a interface *web* do github. Não serão considerados *commits* feitos após essa data e hora.

Avaliação

A avaliação do projecto será dividida pelas 3 partes:

- Nas três partes existirão baterias de testes automáticos implementadas no sistema **Drop Project** ([DP]) que irão avaliar o projecto do ponto de vista funcional.
- Existirá uma nota em cada parte do projecto.
- Nesta primeira parte aplica-se a nota mínima de **oito (8)** valores
  - Quem não alcançar essa nota mínima ficará excluído da avaliação prática de primeira época.
- Após a entrega final, será atribuída ao projecto uma nota final quantitativa, que será calculada considerando a seguinte fórmula:
  - `0.2 * NotaParte1 + 0.8 * NotaParte2`
  - Na nota final existe a nota mínima de 9.5 valores.

DEISI Chess, Primeira Parte V1.0.0

Bruno Cipriano, Duarte Neves, Pedro Alves

Segue-se uma tabela de resumo dos itens de avaliação para a primeira parte do projecto:

Tema	Descrição	Cotação (valores)
Modelo de Classes	Análise do Diagrama UML por parte do Professor. O diagrama deve seguir as regras UML apresentadas nas aulas. Deve também estar em conformidade com o código apresentado.	2
Testes automáticos	Os alunos definem os casos de teste automáticos indicados para esta parte do projecto.	1
Avaliação funcional (DP)	O DP irá testar o funcionamento do jogo simulando diversos cenários, situações de jogada (quer válidas, quer inválidas), situações de detecção de fim de jogo, etc.	14
Avaliação funcional (Professores)	Os professores irão testar situações que não foram diretamente testadas pelo DP no ponto anterior, assim como validar a forma como implementaram certos comportamentos.	3

Avaliação - outras informações

Relativamente à análise do diagrama UML e do código do projecto:

- Serão valorizadas soluções que:
  - usem os mecanismos da programação orientada por objectos (encapsulamento, *method overloading, etc.)* nas situações apropriadas.

- Serão penalizadas soluções que:
  - não façam uso de mecanismos OO
    - (p.e. todo o projecto implementado em uma única classe, etc.).
  - cujo código Java que não corresponda ao modelo apresentado em UML.
  - tenham situações de acesso directo a atributos
    - (p.e. alteração directa de atributos de uma classe por parte de métodos de outra classe)
  - implementem métodos que não modificam nem consultam qualquer atributo/variável da classe respectiva.
  - não implementem construtores adequados

Cópias

Trabalhos que sejam identificados como cópias serão anulados e os alunos que os submetam terão nota zero (0).

Uma cópia numa das entregas intermédias afastará os alunos (pelo menos) da restante avaliação de primeira época, podendo inclusivamente ter efeitos nas restantes épocas.

Notem que, em caso de cópia, serão penalizados quer os alunos que copiarem, quer os alunos que deixarem copiar.

Nesse sentido, recomendamos que não partilhem o código do vosso projecto (total ou parcialmente). Se querem ajudar colegas em dificuldade, expliquem-lhes o que têm que fazer, não lhes forneçam o vosso código pois assim eles não estão a aprender!

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

Outras informações relevantes

– Não deve ser implementado qualquer menu (ou interface gráfica) para interação manual com o utilizador. A única interface gráfica prevista é o Visualizador Gráfico disponibilizado pelos docentes.

– Recomenda-se que eventuais dúvidas sobre o enunciado sejam esclarecidas (o mais depressa possível) no **discord** de Linguagens de Programação II 2023/2024:

– Existirá uma defesa presencial e individual do projeto. A defesa será feita após a segunda entrega. Durante esta defesa individual, será pedido ao aluno que faça alterações ao código do projecto, de forma a dar resposta a alterações aos requisitos.

– É possível que sejam feitas pequenas alterações a este enunciado, durante o tempo de desenvolvimento do projeto. Por esta razão, os alunos devem estar atentos ao **Moodle de LP II**.

- Qualquer situação omissa será resolvida pelos docentes da cadeira.

Anexo I - Instruções e Restrições sobre o Visualizador

Como foi referido, o projecto irá correr em cima de um visualizador gráfico, distribuído em forma de `.jar`. As instruções para configurar o Intellij de forma a utilizar esta biblioteca estão no Anexo IV.

Para que o visualizador funcione correctamente, é necessário respeitar as seguintes restrições:

1) É obrigatório criar as classes seguintes:
    - `GameManager` - responsável por gerir o jogo;
    - `Main` - apenas para ser aceite pelo Drop Project, pode estar vazia.
    - Nota: podem ser criadas mais classes além destas, se acharem que são úteis para a vossa implementação
2) Todas as classes têm de ser colocadas num package chamado:

    ```
    pt.ulusofona.lp2.deisichess
    ```

3) A classe `GameManager` tem de conter (pelo menos) o construtor sem argumentos.

Anexo II - API

A classe `GameManager` tem de conter (pelo menos) os métodos seguintes:

Assinatura
`boolean loadGame(File file)`
`int getBoardSize()`
`boolean move(int x0, int y0, int x1, int y1)`
`String[] getSquareInfo(int x, int y)`
`String[] getPieceInfo(int ID)`

`String getPieceInfoAsString(int ID)`
`int getCurrentTeamID()`
`boolean gameOver()`
`ArrayList<String> getGameResults()`
`JPanel getAuthorsPanel()`

A respetiva descrição encontra-se explicada no vídeo de descrição do projeto.

Anexo III - Exemplos

Neste anexo apresentam-se alguns exemplos de ficheiros de *input* validos.

Este exemplo deve ser usado pelos alunos para testarem o seu programa, antes de o submeterem para avaliação.

Ficheiro de *Input com tabuleiro 4x4*

```
4
6
1:0:0:Chefe
2:0:0:Selvagem
3:0:0:Grande Artista
4:0:1:O Maior
5:0:1:O Amigo
6:0:1:O Beberolas
0:1:0:2
0:0:3:0
0:6:0:0
0:5:4:0
```

Ficheiro de *input* com tabuleiro 8x8

```
8
10
1:0:0:Chefe
2:0:0:Selvagem
3:0:0:Grande Artista
4:0:0:Amante de Praia
5:0:0:Artolas
6:0:1:O Maior
7:0:1:O Amigo
8:0:1:O Beberolas
9:0:1:O Esperto
```

```
10:0:1:O Barulhento
0:1:0:2:0:0:0:0
0:0:3:0:4:0:5:0
0:0:0:0:0:0:0:0
0:0:0:0:0:0:0:0
0:0:0:0:0:0:0:0
0:0:0:0:6:0:0:0
0:0:7:0:0:0:8:0
0:0:9:0:10:0:0:0
```

(Recomenda-se que não criem o ficheiro através de copy&paste deste documento PDF pois isso pode resultar em problemas de *charset* / *encoding*.)

Anexo IV - Configurar o IDE para utilizar uma Biblioteca Externa em formato .jar

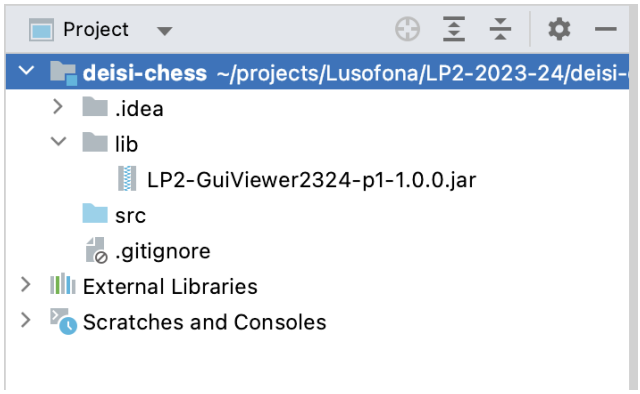
Como explicado anteriormente, o projeto deverá ser desenvolvido recorrendo a um Visualizador Gráfico, que foi disponibilizado num ficheiro .jar que está em moodle.

A versão actual tem o seguinte filename LP2-GuiViewer2324-p1-1.0.0.jar mas este jar pode ir mudando ao longo do projeto, pelo que devem estar atentos a atualizações no Moodle.

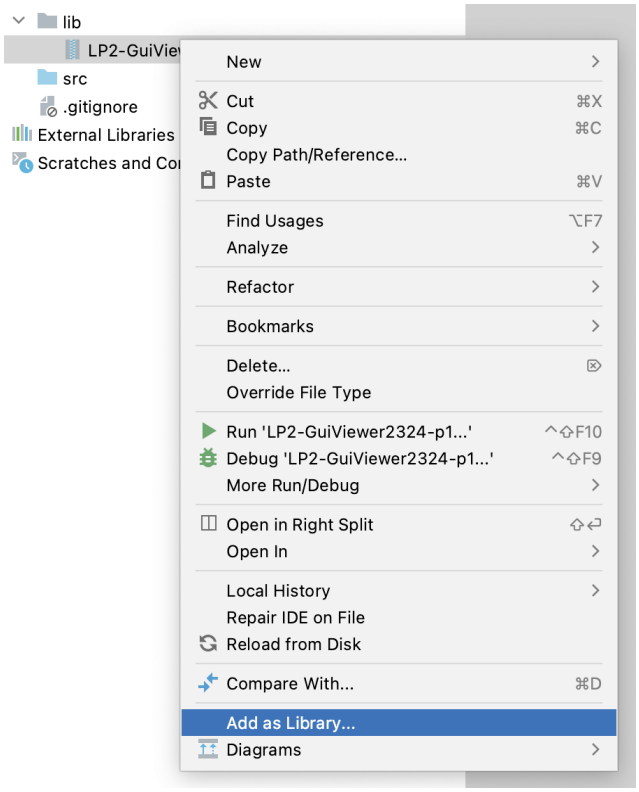
Para poderem usar este .jar no vosso projecto, precisam de configurar o vosso IDE nesse sentido.

Este .jar em particular já tem uma função/método main() que, como já sabem, é o ponto de arranque do projecto. Isto significa que o vosso projecto não deverá ter essa função/método.

Devem começar por descarregar o jar do Moodle e colocá-lo numa pasta lib (que terão que criar) na raiz do vosso projecto:

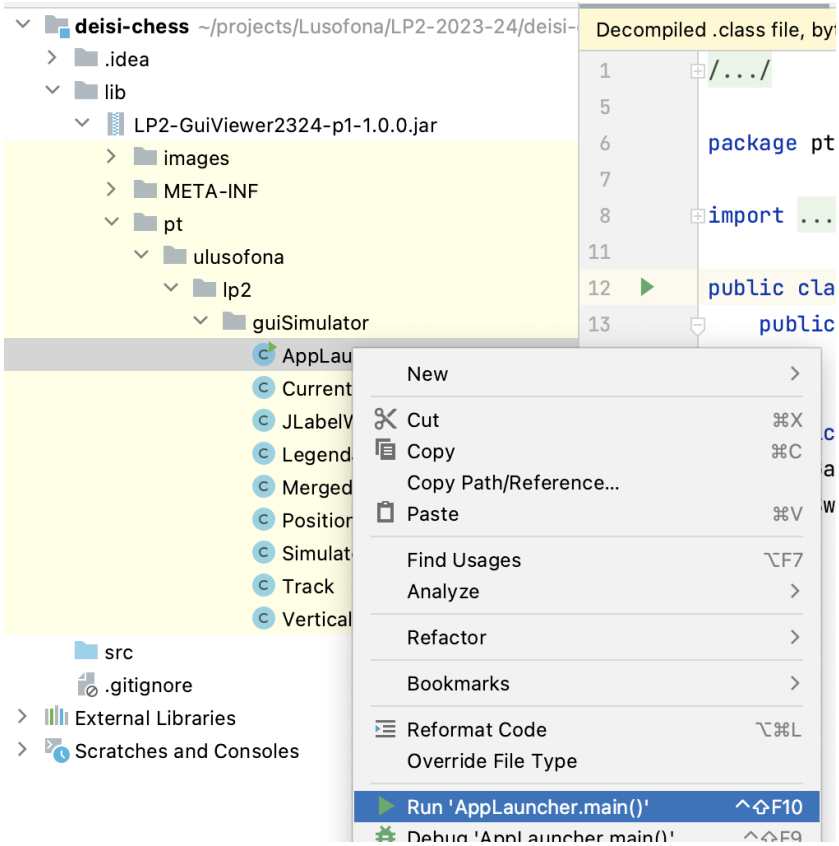


De seguida, com o botão direito em cima do ficheiro jar, escolhem a opção Add as Library:



No ecrã que se seguem, aceitam as opções. Com isto, passam a poder usar as classes que estão dentro deste jar. Em particular, podem correr a função main que está dentro de uma classe chama AppLauncher (no package pt.ulusofona.lp2.guiSimulator).

Para isso, devem espandir o item com o ficheiro jar na vista de projeto, fazer clique com o botão direito na classe AppLauncher e chamar o Run:



Notem que o AppLauncher espera que exista um GameManager com os métodos definidos na API, caso contrário vai dar um erro de execução.

FIM