1 2 9 0 FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE D COIMBRA
DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Relatório do projeto
"Gestor de publicações do CISUC"

Programação Orientada aos Objetos

Docente responsável
Marilia Curado

Alexy de Almeida, nº 2019192123
7 de janeiro de 2021

Índice

Introdução

O projeto consiste na criação de uma aplicação em Java que permita listar as publicações do CISUC (Centro de Informática e Sistemas da Universidade de Coimbra) de acordo com vários critérios de pesquisa. A aplicação "Gestor de publicações do CISUC" deve permitir gerir os grupos de investigação, investigadores e publicações. Esta aplicação tem como princípio o uso da programação orientada aos objetos, tendo sido usado o IDE IntelliJ IDEA 2020, da empresa JetBrains, para toda a sua execução.

Ao programa serão fornecidos dois ficheiros de texto. O ficheiro "Investigadores.txt" contém os dados relativos aos investigadores do CISUC, tais como o nome, email, grupo de investigação a que pertencem entre outros. O ficheiro "Publicacoes.txt" contém os dados relativos às publicações dos investigadores do CISUC, tais como o nome da publicação, a data de publicação, o tipo de publicação entre outros.

Funcionamento

No primeiro arranque, o programa verifica a existência dos dois ficheiros de texto. No final da execução e após sair da aplicação, o programa irá gerar dois ficheiros de objeto, "Investigadores.obj" e "Publicacoes.obj". A partir do segundo arranque, o programa deixará de receber os dados dos ficheiros de texto e passará a receber os dados dos ficheiros de objeto.

Em qualquer arranque, o programa adiciona, por ordem, os grupos de investigação, os investigadores e as publicações aos arrays indicados do CISUC. Se tudo correr bem, é mostrada a mensagem "Conteúdo carregado com sucesso!".

As classes estão divididas das seguintes formas:

1. "Main" - classe que o IDE irá executar para correr o programa.
2. "CISUC" - classe que contém todos os métodos essenciais à execução do programa, tais como leitura e escrita de ficheiros, prints de conteúdos, ordenação das publicações, entre outros.
3. "Publicacao" - classe abstrata, contendo informações relativas a uma publicação e com as seguintes classes descendentes:
    a. "ArtigoConferencia"
    b. "ArtigoConferenciaLivro"
    c. "ArtigoRevista"
    d. "CapituloLivro"
    e. "Livro"
4. "Investigador" - classe que contém informações relativas a um investigador e com as seguintes classes descendentes:
    a. "MembroEfetivo"
    b. "Estudante"
5. "GrupoInvestigacao" - classe que contém informações relativas a um grupo de investigação e com as seguintes classes descendentes:
    a. "AC"
    b. "CMS"

c. “ECOS”
d. “IS”
e. “LCT”
f. “SSE”

6. “Data” - classe que contém informações relativas a uma data.

Existe uma abordagem particular a ser mencionada, sendo ela a criação do objeto `grupos` da classe “GrupoInvestigacao”, na classe “CISUC”. A intenção é simular o CISUC como sendo um grupo de investigação que contém todas as listas de publicações e todos os investigadores do CISUC. A sua aplicação tem sobretudo a ver com os métodos criados para as alíneas 1 e 5, onde é pedido como argumento um objeto da classe “GrupoInvestigacao”.

A alínea 5 reaproveita os métodos da alínea 1, sendo que, na alínea 5, é passado como argumento cada objeto dos grupos de investigação (“AC”, “ECOS”, etc), um a um; e na alínea 1 é passado como argumento o objeto “grupos”, que equivale a passar como argumento cada objeto dos grupos de investigação de uma só vez.

Alguns métodos merecem alguma atenção particular na forma como funcionam.

```
public void lerObjPublicacoes()
```

Método que faz a leitura dos ficheiros de objetos. Como foi criado um objeto `grupos` da classe “GrupoInvestigacao”, na classe “CISUC”,, foi necessário fazer uma dupla leitura dos ficheiros de objetos, de forma a carregar o conteúdo dos objetos para os arrays de investigadores e de publicações tanto do CISUC como do objeto `grupos` do CISUC.

Os seguintes três métodos servem para apresentar no ecrã dados relativos às publicações, consoante o que a alínea do enunciado pede. As três são bastante parecidas na forma como se comportam, no entanto os outputs são completamente diferentes.

```
public void publUltimosCincoOrganizadas(String nomeGrupo)
```

Método criado para a alínea 2. As publicações encontram-se agrupadas, isto é, estão por ordem crescente alfanumérica de ano, seguindo de tipo de publicação e por fim de fator de impacto.

```
public void publicacoesInvestigador(String nomeInvestigador)
```

Método criado para a alínea 4. As publicações estão agrupadas (ao contrário da alínea 2 onde estão organizadas) por ano, tipo e fator de impacto. Ou seja, existe uma separação como se de categorias se tratassem.

```
public void numPublUltimosCincoOrganizadas(String acronimoGrupo)
```

Método criado para a alínea 5. Funciona de forma parecida ao método criado para a alínea 4, no entanto em vez de mostrar as publicações, mostra o número de publicações que existe para uma data com aquela categoria e aquele fator de impacto.

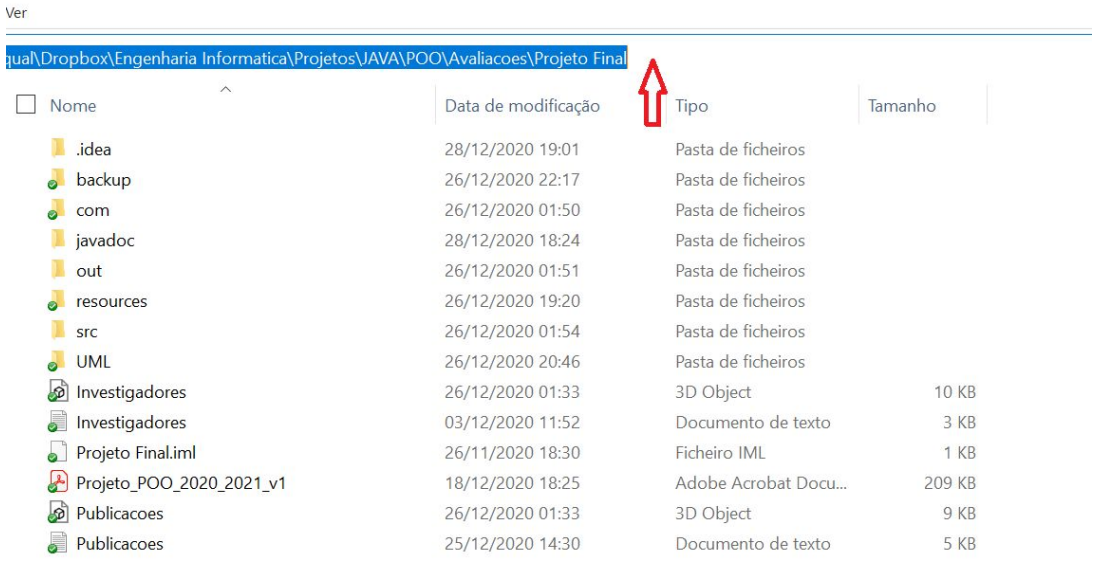
Também existem várias proteções para defender o programa contra eventuais interações com o utilizador, sendo elas:

- ➔ Verifica a existência de ficheiros de texto ou de objetos no início do programa (ex: se faltar um ficheiro txt avisa e o programa para).
- ➔ Verifica a validade dos dados do txt (ex: se o número de telefone do DEI de um investigador é apenas constituído por números inteiros ou se o grupo de investigação a que um investigador pertence existe realmente).
- ➔ Verifica a quantidade de dados necessários (ex: se a data de uma publicação está no formato dd//mm/aaaa ou se um investigador tem mais do que um nome).
- ➔ … outros tipos de verificações para o bom funcionamento do programa.

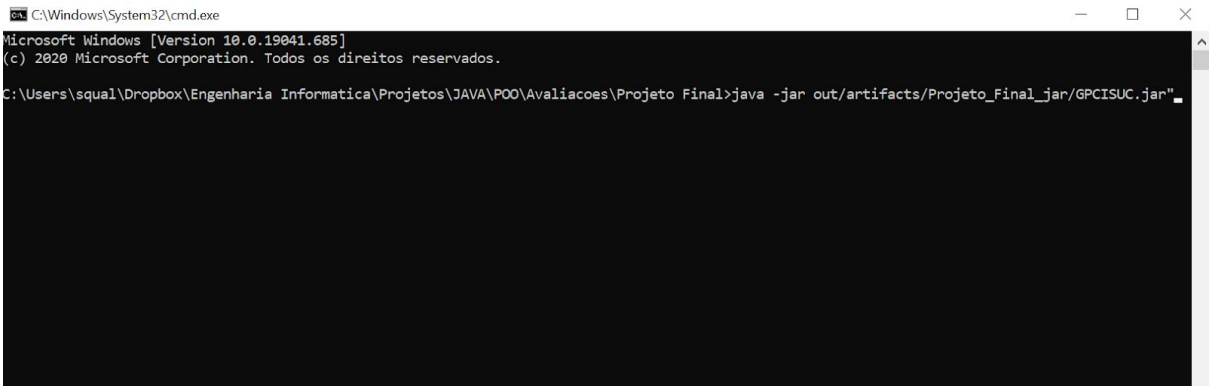
Manual de utilizador

O utilizador deverá correr o programa através do Command Prompt. Para isso, deve seguir os seguintes passos:

1. Na barra de endereços da pasta do projeto, escrever "cmd" e carregar em Enter.

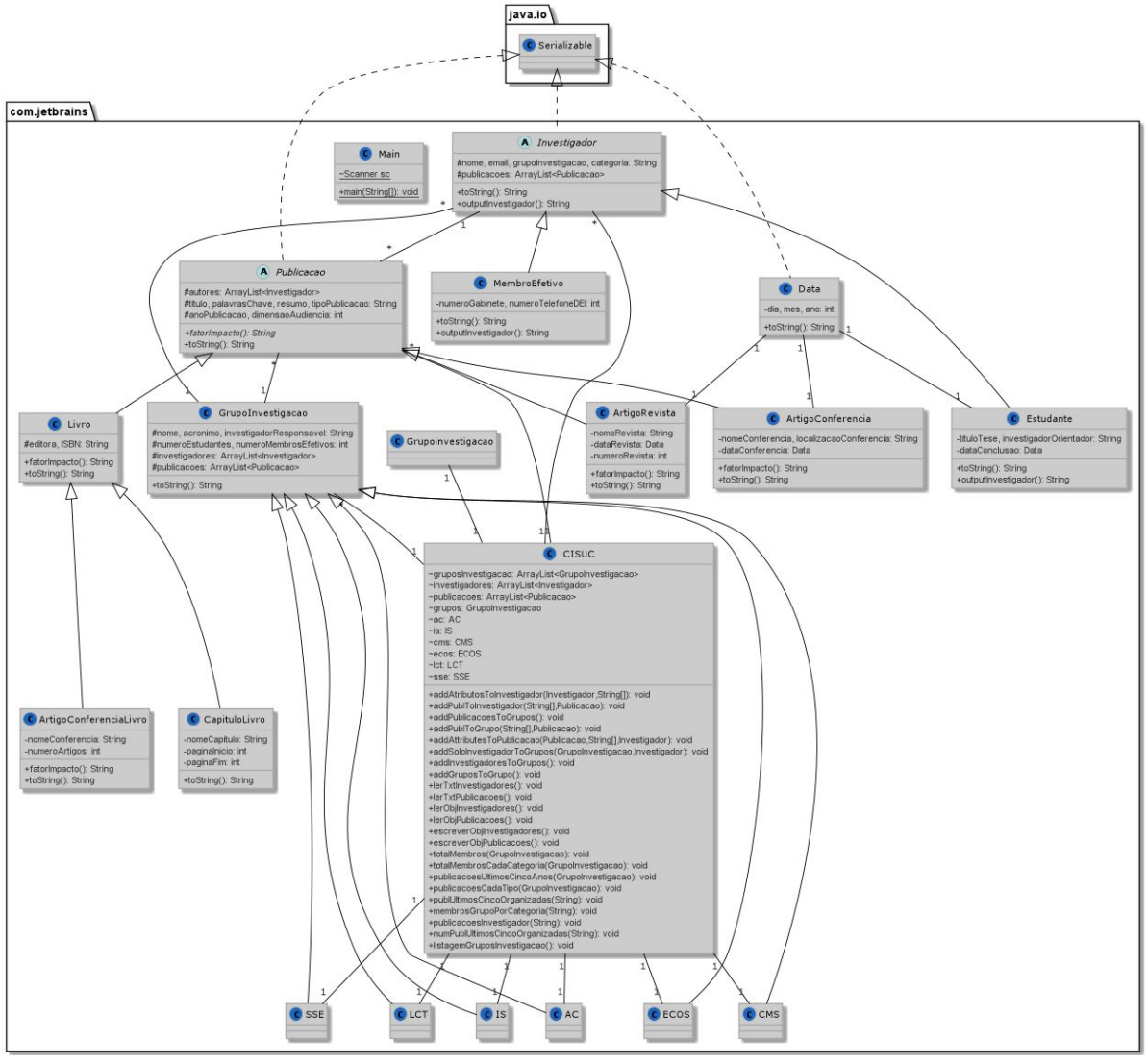


2. Dentro da linha de comandos, escrever <java -jar "caminhoDaPasta/GPCISUC.jar">.



O programa irá apresentar uma série de funcionalidades que o utilizador terá de escolher. Se o utilizador pretender sair do programa e guardar os dados, deverá escrever "0" e carregar em Enter.

UML



Gerado através do plug-in *PlantUML* no IDE *IntelliJ IDEA 2020*.

Conclusão

Após a conclusão deste projeto, queria expôr que a minha abordagem na realização deste mesmo poderia ainda ser melhorada em dois pontos.

O primeiro seria não criar seis classes para cada grupo de investigação, mas simplesmente atribuir ou aceder a um objeto da classe GrupoInvestigacao que tenha como nome o nome do grupo de investigação em questão. Através do método *getNome()* e de uma condição *if*, obtenho quaisquer parâmetros a que pretendo aceder daquele grupo de investigação, como faço normalmente em qualquer situação de outra classe.

O segundo ponto seria não criar um objeto *grupos* da classe GrupoInvestigacao, uma vez que o seu único propósito é servir como argumento dos métodos das alíneas 1 e 5.

A abordagem do projeto teria de ser completamente diferente com essas duas alterações. Tendo em consideração estes aspetos, posso concluir que, a meu ver, o programa tem uma proteção contra o utilizador bastante sólida, cumpre todos os requisitos do enunciado, tem uma documentação completa e faz uso dos bons princípios da programação orientada aos objetos.