Faculdade de Ciências e Tecnologia
Departamento de Engenharia Informática

PROJETO DE POO

Gestor de Publicações do CISUC

Programação Orientada aos Objetos (POO)

Tiago Oliveira Nº 2019219068

25 de dezembro de 2020

Índice

Introdução

Na disciplina de Programação Orientada aos Objetos foi-nos atribuída a função de criar um programa, em *Java*, que fizesse uma gestão das publicações feitas no site do CISUC e que pudesse ser usada por um funcionário do CISUC, tendo de ser intuitiva para todos.

O programa tem como objetivo aplicar os conhecimentos aprendidos na disciplina, de modo a realizar um programa totalmente funcional e estruturado de acordo com as normas impostas e, ou convenções adotadas pela linguagem Java.

Durante a realização do programa foram testadas todas, ou quase todas, as possíveis exceções que provocassem o mau funcionamento do mesmo.

O programa deve incluir uma listagem dos indicadores gerais do CISUC, das publicações de um determinado grupo/investigador existente, apresentar ainda uma lista de todos os membros de um grupo e uma lista de todos os grupos fazendo referência a alguns indicadores de cada grupo.

Foram também criados ficheiros de texto com exemplos de teste do programa, sendo entregues em anexo com o programa.

Descrição das classes / Métodos mais relevantes

Algumas das classes mais importantes do programa são as classes **CISUC**, **GrupoInvestigacao**, **Investigador** e **Publicacao**.

- **CISUC – Classe:** É a classe principal do projeto, onde se localiza o **main** e onde se faz a gestão das informações dos ficheiros.
  - **Variáveis:** **ArrayList<GrupoInvestigacao> grupos** – guarda todos os grupos existentes.
  - **Principais métodos:** **showMenuOptions()** e **menu()** – imprime todas as opções da interação com o utilizador pela interface de consola e permite a interação com o utilizador; **infoCISUC()** e **infoGrupos()** – mostra as informações gerais do CISUC/dos grupos; **listaPubGrupo()** e **listaPubInvestigador()** – mostra uma lista de publicações dos últimos 5 anos do grupo/todas as publicações do investigador escolhido pelo utilizador; **listaMembrosGrupo()** – mostra o nome dos elementos do grupo escolhido pelo utilizador; **openFichGrupos()**, **openFichInvestigadores()** e **openFichPublicacoes()** – é feita toda a gestão das informações dos ficheiros de texto.
- **GrupoInvestigacao – Classe:** É a classe que permite criar objetos do tipo **GrupoInvestigacao**.
  - **Variáveis:** **String nome** – nome do grupo; **String acro** – acrónimo do grupo; **Investigador responsavel** – investigador (efetivo) responsável pelo grupo; **ArrayList<Investigador> investigadores** – guarda todos os investigadores do grupo; **ArrayList<Publicacao> pubG** – guarda todas as publicações do grupo.
  - **Principais métodos:** **listaMembros()** – mostra todos os investigadores do grupo; **listaPublicacoes()** – mostra as publicações dos últimos 5 anos do grupo; **nPubPorAno()**, **nPubPorTipo()** e **nPubPorImpacto()** – retorna o número de publicações por ano/tipo/impacto; **nInvPorCategoria()** – retorna um **int array** com o número de investigadores por categoria.
- **Investigador – Classe:** Classe abstrata que permite criar objetos referentes às suas subclasses **IEfetivo** e **IEstudante**.
  - **Variáveis:** **String nome** – nome do investigador; **String email** – email; **GrupoInvestigacao grupo** – grupo do investigador; **ArrayList<Publicacao> publ** – guarda todas as publicações do investigador.
  - **Principais métodos:** **listaPublicacoes()** – mostra todas as publicações do investigador; **categoria()** – método abstrato que retorna uma **String** com a categoria do investigador (**"Efetivo"** ou **"Estudante"**).
- **Publicacao – Classe:** Classe abstrata que permite criar objetos referentes às suas subclasses **AConferencia**, **ARevista** e **Livro** (e às suas subclasses **Capitulo** e **LivroConferencia**).
  - **Variáveis:** **String titulo** – titulo da publicação; **String keywords** – keywords da publicação; **String resumo** – resumo da publicação; **int anoPub** – ano de publicação da publicação; **int dimensão** – dimensão da audiência da publicação; **ArrayList<Investigador> autores** – lista de autores;
  - **Principais métodos:** **tipo()** – método abstrato que retorna uma **String** com o tipo da publicação; **calculaImpacto()** - método abstrato que retorna uma **String** com o impacto causado pela publicação.

Manual de Utilizador

No caso de os dados serem carregados dos ficheiros de texto vão ter de ser seguidas algumas regras para o programa ler corretamente os dados. Em todos os ficheiros os vários dados estão separados por espaço # espaço. No caso dos ficheiros dos investigadores e das publicações antes dos dados referentes ao(à) mesmo(a) existe um *token* para definir o tipo da publicação (no caso do ficheiro das publicações) e a categoria do investigador (no caso do ficheiro dos investigadores).

Para correr o programa é necessário colocar todos os ficheiros (.txt ou .obj) na pasta do programa. Assim que é corrido irá verificar se já existe algum ficheiro objeto. No caso de isso não acontecer irá ler todos os ficheiros de texto. Assim que isso é feito (frações de segundo) é apresentados um menu, na consola, com todas as opções que o programa realiza, indicando um número específico para cada opção. É pedido ao utilizador que introduza um desses números para executar a opção desejada. Caso seja escolhido uma opção relativa a apenas um grupo ou um investigador é perguntado ao utilizador o acrónimo do grupo ou o nome do investigador. Tanto o nome como o acrónimo podem estar em minúsculas ou maiúsculas.

Assim que o programa acaba de realizar uma opção é apresentado o menu e é pedido uma nova opção. Assim que seja escolhida a opção de sair (0) é criado um ficheiro objeto que guarda todos os grupos e os dados associados aos mesmos (toda a informação do programa).

Conclusão

Neste trabalho pude ampliar os meus conhecimentos relativos à linguagem de programação Java, podendo colocar em prática toda a matéria aprendida nas aulas da disciplina.

A implementação do programa foi baseada na minha interpretação do enunciado que nos foi entregue, podendo não coincidir na totalidade com o que os professores tinham idealizado. Talvez faltem algumas proteções, como por exemplo as proteções para os Integer.parseInt(), e certamente existem implementações mais eficientes que a minha.

Referências

affiliates, O. a. (1993). *Class IOException*. Obtido de Java™ Platform: https://docs.oracle.com/javase/7/docs/api/java/io/IOException.html

affiliates, O. a. (1993). *Package java.time*. Obtido de Java™ Platform, Standard Edition 8: https://docs.oracle.com/javase/8/docs/api/java/time/package-summary.html

SINGH, C. (2012). *Java ArrayList of Object Sort Example (Comparable And Comparator)*. Obtido de BeginnersBook: https://beginnersbook.com/2013/12/java-arraylist-of-object-sort-example-comparable-and-comparator/

Slides da disciplina