



Universidade do Minho

## Projecto Java de Laboratório de Informática Relatório

Discentes:  
Axel Ferreira - a53064  
João Rua - a41841

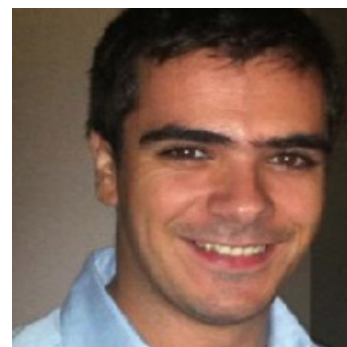
Docentes:  
F. Mário Martins  
João Miguel Fernandes  
João Luís Sobral

June 15, 2013

### Grupo



(a) nome : Axel Ferreira  
número : 53064  
mail : axelferreira@me.com



(b) nome : João Rua  
número : 41841  
mail : joaorua@gmail.com



## Contents

<b>1</b>	<b>Introdução</b>	<b>2</b>
1.1	Estrutura do Relatório . . . . .	2
<b>2</b>	<b>Classes e Estruturas de Dados</b>	<b>3</b>
2.1	Classes Criadas . . . . .	3
<b>3</b>	<b>Consultas Estatísticas</b>	<b>4</b>
3.1	Dados do último ficheiro lido . . . . .	4
3.2	Consultas Interativas . . . . .	4
<b>4</b>	<b>Consultas Interativas</b>	<b>4</b>
4.1	. . . . .	4
<b>5</b>	<b>Medidas de Performance</b>	<b>4</b>
<b>6</b>	<b>Conclusão</b>	<b>4</b>

### Abstract

Foi desenvolvido um programa no âmbito da U.C. de Laboratório de Informática III , capaz de utilizar o conteúdo processado pelo anterior programa desenvolvido em C, também nesta UC, capaz de ler um ficheiro contendo um conjunto de autores e respetivos co-autores, e responder a alguns queries interativos sobre estes dados. Estes dados de autorias, e co-autorias são retirados do website [DBLP](#).

## 1 Introdução

No âmbito da unidade curricular Laboratórios de Informática III foi proposta a realização de um projeto que dá continuidade ao projeto anteriormente desenvolvido em C nesta mesma UC. Este novo projeto conta com essencialmente duas partes. A primeira diz respeito á leitura de dados de memória secundária e população de estruturas de dados em memória central, gravação destas estruturas de dados em memória persistente em modo binário, bem como a criação de alguns queries de forma a permitir uma consulta interativa aos dados. Desenvolveram-se ainda alguns métodos que permitem consultas sobre as estruturas de dados. A Segunda parte prevê o teste de performance do código e respetivas estruturas de dados criados na 1ª parte, relativamente a estruturas de dados alternativas. De forma a facilitar esta segunda parte, o grupo teve o cuidado de criar uma interface cada vez que foi utilizada uma estrutura de dados do Java.Collections, de forma a permitir alterar as estruturas utilizadas alterando apenas, e se necessário, esta interface.

### 1.1 Estrutura do Relatório

Este relatório inicia-se com uma capa, incluindo o título do projeto, a data, a identificação dos autores e da equipa docente que acompanhou o projeto. Segue-se o Índice, o Abstract que resume o projeto, a Introdução ao mesmo (onde se explicita o objetivo a atingir) e a Estrutura do Relatório.



## 2 Classes e Estruturas de Dados

### 2.1 Classes Criadas

Na criação deste programa foram desenvolvidas alguns classes

- Anos - Esta classe armazena toda a estrutura de dados. Contém um TreeMap em que são inseridos todos os anos. Cada ano é inserido, utilizando como chave a String contendo a numeração do ano. A escolha do TreeMap deveu-se ao facto de manter a ordem do ano. Esta ordem facilita a impressão e travessia ordenada necessária para algumas queries.

```
private TreeMap<String,Ano> anos;
```

- Ano - Classe criada para guardar o conteúdo de cada Ano, contém uma String com o nome do ano e um HashMap com os Autores. Cada autor é inserido utilizando como chave a String com o nome do mesmo. A utilização do HashMap deve-se ao facto de não haver vantagem associada á ordem de armazenamento.

```
private String ano;  
private HashMap<String,Autor> autores;
```

- Autor - Classe criada para guardar o nome do Autor, numero de publicações, e rede de co-autores. Esta última é armazenada num HashMap de co-autores. Cada co-autor é inserido utilizando como chave a String com o nome do mesmo. A utilização do HashMap deve-se ao facto de não haver vantagem associada á ordem de armazenamento.

```
private String nome;  
private int artigos;  
private HashMap<String,Coautores> coautores;
```

- Coautor - Classe criada para guardar o conteúdo de cada co-autor, resumindo-se ao nome e numero de artigos publicados em comum com o respetivo autor.

```
private String nome;  
private int artigos;
```

- FileInput - Esta classe faz todo o parsing e leitura dos dados de ficheiros com que posteriormente as estruturas de dados são povoadas. Contém apenas como variável de classe o nome do ficheiro que deve ler sempre. Contém ainda um método de classe que devolve o nome do ficheiro. Bem como dois métodos que devolvem o conteúdo dos ficheiros.

```
public static final String ficheiro ="publicx.txt";
```



### 3 Consultas Estatísticas

#### 3.1 Dados do último ficheiro lido

Quando o ficheiro é lido, é apresentado no ecrã o nome do ficheiro, seguido do número total de artigos, e número total de nomes lidos, número total de nomes distintos bem como o intervalo fechado de anos em que os artigos foram lidos. É ainda apresentada alguma informação respeitante aos dados atuais da estrutura de dados, nomeadamente nº total de autores, nº total de artigos, de um só autor, e nº de autores que apenas publicaram a solo. Finalmente é ainda apresentada toda a sequência ordenada de anos seguido do respetivo número de publicações.

#### 3.2 Consultas Interativas

### 4 Consultas Interativas

#### 4.1

### 5 Medidas de Performance

*Nesta fase foi efetuado o re-factoring e implementado o buffer dinâmico. Garantindo – se desta forma que*

### 6 Conclusão

*A principal dificuldade que foi ultrapassada, nomeadamente na 2ª Fase, foi a implementação da estrutura de dados, faltando apenas implementar a estrutura de dados – autoria.*