



## **Trabalho Prático 2 - Programação Orientada a Objetos**

### **Query's de ficheiros CSV**

#### **Relatório de Programação Orientada a Objetos**

**Gustavo Paulino, n.º 21805593**

**Leandro Fonseca, n.º 22001805**

**Pedro Pinto, n.º 22000888**

Docente:

Prof. Doutor Tiago Candeias

2021

# Resumo

Este trabalho foi realizado com o objectivo de implementar encapsulamento, herança, polimorfismo, expressões regulares, estruturas de dados, metodologia ETL, testes unitários, exceções e JavaDoc, estudados no âmbito do paradigma de programação orientada a objetos.

Para implementar os conhecimentos, foi pedido para ler qualquer ficheiro com o formato .csv, guardá-lo em memória e aplicar operadores de filtragem e agregação sobre os dados.

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>2</b>
1.1	Descrição do Problema . . . . .	2
<b>2</b>	<b>Abordagem</b>	<b>3</b>
<b>3</b>	<b>Diagrama de Classes</b>	<b>5</b>
<b>4</b>	<b>Testes Unitários</b>	<b>6</b>
<b>5</b>	<b>Exceções</b>	<b>7</b>
<b>6</b>	<b>Resultados</b>	<b>8</b>
<b>7</b>	<b>Conclusão</b>	<b>10</b>
	<b>Bibliografia</b>	<b>11</b>

# Introdução

## 1.1 Descrição do Problema

Pretende-se ler qualquer ficheiro com o formato .csv, guardá-lo em memória e aplicar operadores de filtragem e agregação sobre os dados.

Um dataset intitulado Customer Data.csv é disponibilizado num ficheiro com o formato csv. A funcionalidade deve ser genérica e não deve estar limitada a este dataset, no entanto a utilização deste dataset é um importante ponto de partida. Uma amostra do ficheiro é apresentado na figura seguinte.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
2	1	15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101348,88	1
3	2	15647311	Hill	608	Spain	Female	41	1	83807,86	1	0	1	112542,58	0
4	3	15619304	Onio	502	France	Female	42	8	159660,8	3	1	0	113931,57	1
5	4	15701354	Boni	699	France	Female	39	1	0	2	0	0	93826,63	0
6	5	15737888	Mitchell	850	Spain	Female	43	2	125510,8	1	1	1	79084,1	0
7	6	15574012	Chu	645	Spain	Male	44	8	113755,8	2	1	0	149756,71	1
8	7	15592531	Bartlett	822	France	Male	50	7	0	2	1	1	10062,8	0
9	8	15656148	Obinna	376	Germany	Female	29	4	115046,7	4	1	0	119346,88	1
10	9	15792365	He	501	France	Male	44	4	142051,1	2	0	1	74940,5	0
11	10	15592389	H?	684	France	Male	27	2	134603,9	1	1	1	71725,73	0
12	11	15767821	Bearce	528	France	Male	31	6	102016,7	2	0	0	80181,12	0
13	12	15737173	Andrews	497	Spain	Male	24	3	0	2	1	0	76390,01	0
14	13	15632264	Kay	476	France	Female	34	10	0	2	1	0	26260,98	0
15	14	15691483	Chin	549	France	Female	25	5	0	2	0	0	190857,79	0
16	15	15600882	Scott	635	Spain	Female	35	7	0	2	1	1	65951,65	0
17	16	15643966	Goforth	616	Germany	Male	45	3	143129,4	2	0	1	64327,26	0
18	17	15737452	Romeo	653	Germany	Male	58	1	132602,9	1	1	0	5097,67	1
19	18	15788218	Henderson	549	Spain	Female	24	9	0	2	1	1	14406,41	0
20	19	15661507	Muldrow	587	Spain	Male	45	6	0	1	0	0	158684,81	0
21	20	15568982	Hao	726	France	Female	24	6	0	2	1	1	54724,03	0

**Figura 1.1** – Fichero Customer\_Data.csv

# Abordagem

A abordagem do grupo está centralizada no processo ETL.

O que é ETL? ETL do inglês Extract, Transform e Load, é o processo que combina mover a data(informação) de uma/várias base de dados para outra usando três processos chave:

E –Extract ou Extração : Permite extrair a informação da primeira base de dados, pode ser um ficheiro, cloud, entre outros.

T-Transform ou Transformação : Ao termos a informação queremos normalizar a informação na nossa base de dados, é neste processo que existe a padronização, por exemplo passar todos os tipos de informação para o mesmo formato, o cleansing, ou seja resolver inconsistências e imprecisões entre bases de dados, em resumo é neste processo que aplicamos o nosso conjunto de regras(formatação, conversão, etc) na base de dados original para a base de dados onde vamos trabalhar.

L – Load ou Carregamento : Processo final de ETL em que consiste em carregar/enviar a informação já normalizada e trabalhada para outras bases de dados ou usuários.

Para responder à extração, na classe Extract, primeiro normalizá-mos os dados, lendo os mesmos a partir do ficheiro e introduzindo-os num array de duas dimensões, procurando generalizar a funcionalidade.

No armazenamento, que está presente na classe Transform, guardamos os dados numa ArrayList<HashMap<String,String>, onde o título de cada coluna, presente no ficheiro, é a chave dos HashMaps e os values são os valores de cada coluna com a respetiva chave.

Para cada operação foram criadas classes, cada uma com a sua função de filtragem ou agregação dos dados.

Quanto ao problema que nos foi proposto no que toca a conseguir ler uma query inserida a partir da consola, usámos uma expressão regular para filtrar a informação útil e invertendo-a para usar os operadores começando do fim para o início.

Para imprimir tabelas foi criada uma classe TableFormatter que calcula tudo o que é preciso para imprimir tabelas de dimensões dinâmicas.

```
java -jar /bin/jar/19k-15.0.2/bin/19k.jar
Customer_Data.csv
FILTER(Customer_Data, Customer_Data[Age]=18)
```

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
747	15787619	Hsieh	844	France	Male	18	2	160980.03	1	0	0	145936.28	0
1620	15770309	McDonald	656	France	Male	18	10	151762.74	1	0	1	127014.32	0
1679	15569178	Kharlanov	570	France	Female	18	4	82767.42	1	1	0	71811.9	0
2022	15795519	Vasiliev	716	Germany	Female	18	3	128743.8	1	0	0	197322.13	0
2137	15621893	Bellucci	727	France	Male	18	4	133559.67	1	1	1	46941.41	0
2142	15758372	Wallace	674	France	Male	18	7	0	2	1	1	55753.12	1
3331	15657439	Chao	738	France	Male	18	4	0	2	1	1	47799.15	0
3513	15657779	Boylan	806	Spain	Male	18	3	0	2	1	1	86994.54	0
3518	15757821	Burgess	771	Spain	Male	18	1	0	2	0	0	41542.95	0
3687	15665327	Cattaneo	706	France	Male	18	2	176139.5	2	1	0	129654.22	0
4557	15796231	Nwankwo	681	France	Female	18	1	98894.39	1	1	1	9596.4	0
4717	15805764	Hallahan	646	France	Male	18	10	0	2	0	1	52795.15	0
7335	15759133	Vaguine	616	France	Male	18	6	0	2	1	1	27308.58	0
7723	15570086	Lynch	684	Germany	Male	18	9	90544	1	0	1	4777.23	0
8523	15619892	Page	644	Spain	Male	18	8	0	2	1	0	59172.42	0
9830	15722701	Bruno	594	Germany	Male	18	1	132694.73	1	1	0	167689.56	0
9582	15634146	Hou	835	Germany	Male	18	2	142872.36	1	1	1	117632.63	0
9521	15673180	Onyekaozulu	727	Germany	Female	18	2	93816.7	2	1	0	126172.11	0
9527	15665521	Chiazagomekepele	642	Germany	Male	18	5	111183.53	2	0	1	10863.75	0
9573	15641688	Collier	644	Spain	Male	18	7	0	1	0	1	59645.24	1
9783	15728829	Weigel	509	France	Male	18	7	102983.91	1	1	0	171778.58	0
9933	15813451	Fleetwood-Smith	677	Spain	Male	18	8	134796.87	2	1	1	114858.9	0

```
Process finished with exit code 0
```

**Figura 2.1** – Print de tabela de output

# Diagrama de Classes

Antes de começar a desenvolver o programa foi feito o diagrama de classes com o que iria ser preciso implementar.



**Figura 3.1** – Diagrama de Classes Link

# Testes Unitários

Na parte de desenvolvimento foi seguida a metodologia TDD(Test-Driven Development), ou seja baseado em testes. Foi utilizada a framework JUnit5 para a realizar os testes unitários, para poder validar os métodos das classes implementadas, que estão localizados em `'/src/JUnit'`.

Foram aplicados testes unitários para todas as classes do package *helperClasses*, excepto as Classes All e Table Formatter devido a apenas terem métodos para printar.



# Exceções

Para evitar que o programa termina-se subitamente devido a erros durante a execução foram implementadas classes que estendem a classe `Exception` para ser possível o catch dos erros. As Exceções implementadas neste programa foram:

- `NoLinesTableException` que é provocado quando se tenta dar print de uma tabela que têm 0 linhas .
- `NoNumberFieldException` que é provocado quando se tenta calcular `AVERAGE/SUM` de colunas que não são reconhecidas com valores numéricos.
- `ImpossibleConditionException` que é provocado se tenta executar algum comando inexistente ou inválido.
- `ImpossibleCalculateException` que é provocado quando o primeiro argumento do comando `CALCULATE` não é um comando de agregação.

# Resultados

1. Qual o número de clientes adultos ativos?

COUNTROWS(FILTER(Customer\_Data, Customer\_Data[Age]>=18  
&& Customer\_Data[IsActiveMember]=1))

```
/usr/lib/jvm/jdk-15.0.2/bin/java ...  
Customer_Data.csv  
AVERAGE(Customer_Data[Tenure])  
A média dos elementos da coluna Tenure é: 5.0128  
  
Process finished with exit code 0  
|
```

**Figura 6.1** – Average Query.csv

2. Quantos países distintos pertencem os clientes?

CALCULATE(DISTINCTCOUNT(Customer\_Data[Geography]), ALL(Customer\_Data))

```
/usr/lib/jvm/jdk-15.0.2/bin/java ...  
Customer_Data.csv  
SUM(Customer_Data[Balance])  
A soma dos elementos da coluna Balance é: 7.648588928799961E8  
  
Process finished with exit code 0  
|
```

**Figura 6.2** – Sum Query.csv

3. Qual a média de anos de fidelização?

AVERAGE(Customer\_Data[Tenure])

```
/usr/lib/jvm/jdk-15.0.2/bin/java ...  
Customer_Data.csv  
COUNTROWS(FILTER(Customer_Data, Customer_Data[Age]>=18 && Customer_Data[IsActiveMember]=1))  
  
A tabela têm 5151 linhas!  
  
Process finished with exit code 0
```

**Figura 6.3** – Countrows Query.csv

4. Qual o saldo total?

SUM(Customer\_Data[Balance])

```
/usr/lib/jvm/jdk-15.0.2/bin/java ...  
Customer_Data.csv  
CALCULATE(DISTINCTCOUNT(Customer_Data[Geography]), ALL(Customer_Data))  
A coluna Geography têm 3 valores distintos!  
  
Process finished with exit code 0  
|
```

**Figura 6.4** – Calculate Query.csv

# Conclusão

Concluío com este trabalho, que através do planeamento através do desenho do diagrama de classes, a implementação das classes, testes unitários e exceções foi conseguido aplicar todo o conhecimento adquirido nas aulas Teórico-práticas da disciplina de Programação Orientada a Objetos [1].

Foi um trabalho com algum grau de dificuldade e que foi necessário pensar muito bem na sua implementação e concepção, podendo implementar encapsulamento, herança, polimorfismo, expressões regulares, estruturas de dados, metodologia ETL, testes unitários, exceções e JavaDoc e que com pequenas mudanças no código pode-se utilizar mais de uma tabela em memória.

O trabalho foi implementado no IDE IntelliJ IDEA Ultimate utilizando o JDK na versão 15 no sistema Operativo Linux baseado em Ubuntu e Windows 10 e pode ser acedido através do zip enviado ou pelo Repositório no GitHub.

# Bibliografia

- [1] Prof. Doutor Tiago Candeias, 2021/2022, aulas Teórico-práticas da disciplina de Programação Orientada a Objetos, 2º ano, 1º semestre da licenciatura em Engenharia Informática do Instituto Superior Manuel Teixeira Gomes.
  
- [2] K. Berry. Tex live documentation. TeX Live has been developed since 1996 by collaboration between the TeX user groups. [Online]. Available: <https://tug.org/texlive/>