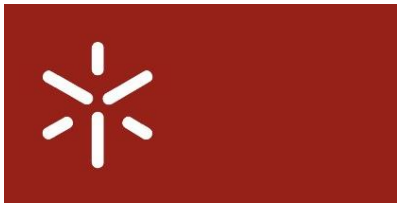


Universidade do Minho

Mestrado Integrado em Engenharia Informática

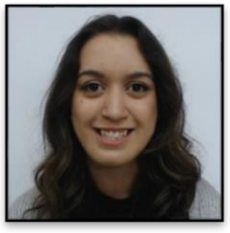
Projeto de Java: GestVendas

Laboratórios de Informática III (LI3)

- Henrique Veiga da Paz, A84372

- Joana Isabel Afonso Gomes, A84912

- Susana Vitória Sá Silva Marques, A84167



ÍNDICE

CONTEÚDO ANEXADO: Resultado dos testes obtidos.

´

DESCRIÇÃO DAS CLASSES

- Classe Cliente e classe Produto: Classes cujas variáveis de instância são respetivamente uma `String clie` que representa o código do cliente e uma `String prod` que representa o código do produto.

- Classe CatClientes: Classe que contém um `SortedSet<String> clie` que guarda a lista de clientes ordenada por ordem alfabética.

- Classe CatProdutos: Classe que contém um `SortedSet<String> prods` que guarda a lista de produtos ordenada por ordem alfabética.

- Classe Faturacao: Classe que guarda uma lista de vendas em que a chave de procura é o código do produto (v.i.: `Map<String,List<Venda>> vendas`).

- Classe Filial: Classe que tem 3 `Map<String,List<Venda>>` cuja chave é o código de cliente e guarda a lista de vendas para cada uma das filiais.

- Classe Venda: As suas variáveis de instância representam o produto comprado (`String p`), o preço (`double preco`), a quantidade do produto comprado (`int quant`), o modo promoção ou normal (`char promnorm`), o cliente que efetuou a compra (`String c`), o mês em que foi efetuada (`int mes`) e a respetiva filial (`int filial`), todos correspondentes às entradas de uma linha de vendas. Para além dos constrututores, *getters* e *setters*, o método `public boolean isVenda_valida(String v, InterfCatProd cp, InterfCatClie cl)` verifica se uma venda é válida e `public Venda getVenda_valida(String v)` dá como resultado uma Venda válida a partir de uma linha de vendas.

- Classe Menu: Classe onde criamos e apresentamos a estrutura de um Menu. Com o método `private int lerOpcao()` sabemos também a opção escolhida pelo utilizador aquando apresentação do menu.

- Classe GereVendasModel: Classe onde se lê os ficheiros e que contém os métodos para cada *querie*:

    - Querie Estatística 1.1. - A *querie* apresenta ao utilizador os dados referentes ao último ficheiro de vendas lido. Para que tal se possa suceder, a GereVendasModel disponibiliza métodos para cada um dos respetivos dados, passando por `public String nome_fich ()` (nome do ficheiro), `public int venda_errados()` (número total de registos de venda errados), `public int total_prod()` (número total de produtos), `public int total_prod_dif()` (total de diferentes produtos comprados, `public int nunca_comprados()` (total de não comprados), `public int tot_clie()` (número total de clientes), `public int total_clie_dif()` (total dos que realizaram compras), `public int clie_naocomp()` (total de clientes que nada compraram), `public int tot_comp_igual0()` (total de compras de valor total igual a 0.0) e `public double faturacao()` (facturação total).

    - Querie Estatística 1.2. - Esta *querie* apresenta em ecrã ao utilizador os números respeitantes aos dados registados nas estruturas. Assim, a GereVendasModel contém métodos que calculam: o número total de compras por mês (métodos auxiliares `public int nrTotalComprasMesF1(int mes)` (e o mesmo para as outras filiais)), faturação total por mês (valor total das compras/vendas) para cada filial `(public double faturacaoPorMesF1(int mes),` (…)) e global (`public double faturacaoPorMes(int mes)`) e o número de distintos clientes que compraram em cada mês filial a filial (`public int nrClientesDifCompraramF1(int mes)`, (…)).

    - Querie Interativa 1 - Para podermos listar alfabeticamente com os códigos dos produtos nunca comprados e o seu respectivo total usamos o método `public SortedSet<String> lista_nunca_comprados()`. Escolhemos utilizar um Set para não existir elementos repetidos.

    - Querie Interativa 2 – O objetivo será, dado um mês válido, determinar o número total global de vendas realizadas e o número total de clientes distintos que as fizeram, a nível global e a nível das filiais. Nos métodos que criamos para tal, utilizamos exceções, como por exemplo no `public int numero_global_vendas(int mes)throws mesException`, que não permite a inserção de um inteiro que não entre 1 e 12 aquando do input do mês.

    - Querie Interativa 3 – Para, dado um código de cliente, determinar, para cada mês, quantas compras fez, usamos os métodos auxiliares `public`

`int qtsComprasF1(String c, int mes)throws mesException,clienteException, (…)` (em que mais uma vez lançamos a exceção do mês inválido, juntamente com a que não permite a inserção de um código de cliente inválido, e analogamente, métodos para calcular quantos produtos distintos comprou e quanto gastou no total.

- Querie Interativa 4 – Para, dado o código de um produto existente, determinar, mês a mês, quantas vezes foi comprado, utilizamos o método `public int qtsVezesFoiComprado(String p, int mes) throws produtoException`, lançando uma nova exceção para o *input* de um produto inválido. Temos métodos análogos para determinar por quantos clientes diferentes foi comprado e o total facturado mas não lançamos exceções dado que o input é o mesmo para os três, aquando do *input* pelo *user* (o que é determinado na *GereVendasView*).

- Querie Interativa 5 - Para determinar a lista de códigos de produtos que um dado cliente mais comprou (e quantos) existe o método `public SortedSet<ParVendaDouble> listaord(String c)throws clienteException,` utilizando para o efeito de ordenação por ordem decrecente de quantidade (e, para quantidades iguais, por ordem alfabética dos códigos) um `ParVendaDoubleComparator`, classe que criamos para o efeito.

- Querie Interativa 6 – Para determinar o conjunto dos X produtos mais vendidos em todo o ano temos o método `public SortedSet<ParStringInt> produtos_maisvend(int y)`, em que mais uma vez recorremos a um comparator para obter a ordem desejada, desta vez `ParVendaInt8Comparator(),` e para indicar o número total de distintos clientes que o compraram o método p`ublic int numero_clie(String p).`

- Querie Interativa 7 - Para determinar, para cada filial, a lista dos três maiores compradores em termos de dinheiro facturado, usamos o método `public SortedSet<ParVendaDouble> lista_paracadafilial1()` (e análogos para cada filial).

- Querie Interativa 8 – Para determinar os códigos dos X clientes (sendo X dado pelo utilizador) que compraram mais produtos diferentes utilizamos o método `public SortedSet<ParVendaInt> clientes_maiscomprad(int y).` Para que a ordem seja pedida recorremos novamente ao *Comparato*r `ParVendaInttComparator()` (reversed).

- Querie Interativa 9 – Com vista a determinar o conjunto dos X clientes que mais o compraram e, para cada um, qual o valor gasto há o método `public SortedSet<ParVendaDouble> clientes_valorgasto(String p, int y)throws produtoException.` Sendo a ordem pedida a mesma que na querie 5, mais uma vez recorremos ao *Comparator* `ParVendaDoubleComparator().`

- Querie Interativa 10 – Para determinar mês a mês, e para cada mês filial a filial, a facturação total com cada produto, existe o método `public double fact_tot(String produto, int mes, int filial)throws produtoException.`

- Classe GereVendasView: Classe que interage com o utilizador, através de *prints* de Menus, dos resultados das *queries* (métodos respetivos) ou da apresentação de tabelas, juntamente com todo o mecanismo de *back and forth* da aplicação.

- Classe GereVendasController: Faz a ligação entre a GereVendasView e a GereVendasModel (que contém os métodos das queries).

- Classe GereVendasAppMVC: É responsável pela implementação do modelo MVC. É a classe que faz a ligação entre as classes `GereVendasModel`, `GereVendasView`, `GereVendasController`.

Todas estas classes implementam uma respetiva interface, em que o nosso trabalho se baseia. Usamos também as classes `Input` (realizar uma utilização segura do Scanner) e `Crono` (calcular os tempos de execução de cada *querie* e realizar uma utilização segura do Scanner) disponibilizadas.

Criamos classes como por exemplo `ParVendaComparator` que nos ajudou a responder a *queries* que nos pediam resultados de duplos ou triplos por ordem decrescente.

APLICAÇÃO

A aplicação do nosso projeto inicia com uma saudação de boas vindas ao programa, seguindo-se de um Menu que inicialmente apresenta duas opções de escolha: as consultas estatísticas e as consultas interativas.

```
**********************
BEM VINDO À GESTVENDAS!
**********************


      MENU:
**********************
**********************

Escolha a funcionalidade a que pretende aceder:
1 - Consultas Estatísticas.
2 - Consultas Interactivas.
3 - Sair.
Para fechar e guardar a aplicacao digite 0.
Opção:
```

Selecionada qualquer uma das opções, as queries são apresentadas no ecrã, permitindo ao utilizador escolher qual pretende consultar.

```
Escolha a funcionalidade a que pretende aceder:
1 - Consulta da lista ordenada alfabeticamente com os códigos dos produtos nunca comprados e o seu respectivo total.
2 - Consulta do número total global de vendas realizadas e o número total de clientes distintos que as fizeram
3 - Cosulta de quantas compras um cliente fez, quantos produtos distintos comprou e quanto gastou no total.
4 - Consulta, mês a mês, quantas vezes um produto foi comprado, por quantos clientes diferentes e o total faturado.
5 - Consulta dos produtos que um cliente mais comprou (e quantos).
6 - Consulta dos X produtos mais vendidos em todo o ano (em número de unidades vendidas),indicando o número total de clientes que os comprararam.
7 - Consultar, para cada filial, a lista dos três maiores compradores em termos de dinheiro faturado.
8 - Consultar os códigos dos X clientes que compraram mais produtos diferentes.
9 - Consultar o conjunto dos X clientes que mais compraram um produto e, para cada um, qual o valor gasto
10 - Consultar mês a mês, e para cada filial. a faturação total com cada produto.
11 - Sair.
Para fechar e guardar a aplicacao digite 0.
Opção:
```

O tempo de execução é sempre apresentado antes do resultado das queries.

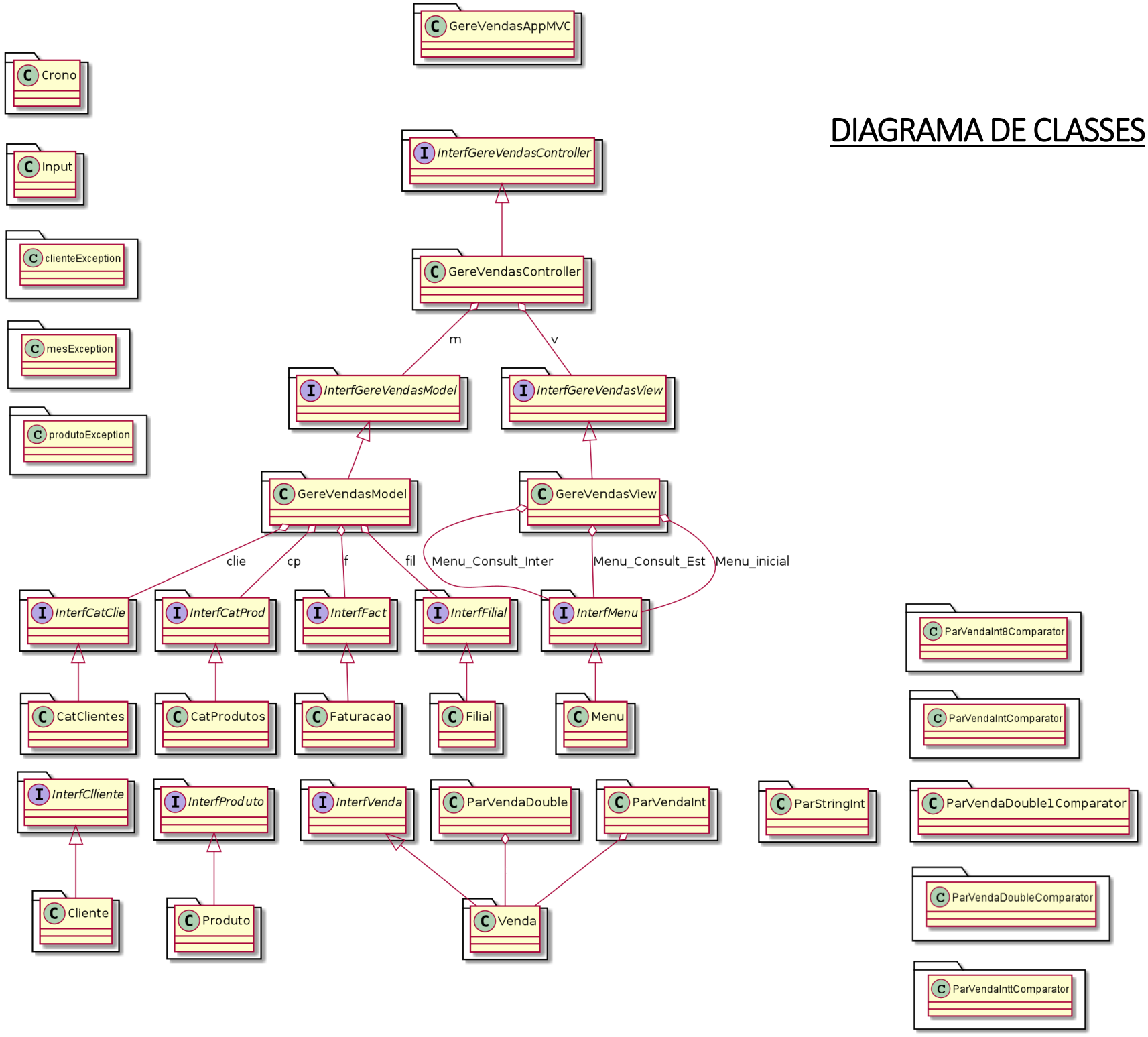
```
Tempo de execução: 0.115416174

********************************************************************************************************
O número total de vendas no mês 4 na filial 3 foi 24846.
O número de clientes diferentes que compraram neste mês na filial 3 foi 12779.
********************************************************************************************************
Carregue em qualquer tecla para voltar ao menu!
```

Dependendo da consulta, o output das consultas pode ser apresentado em terminal ou em tabela, numa janela pop-up.

NÚMERO DE CLIENTES DIFERENTES QUE COMPRARAM EM CADA MÊS

	FILIAL 1	FILIAL 2	FILIAL 3
JANEIRO	12699	12800	12892
FEVEREIRO	12807	12745	12793
MARÇO	12789	12644	12758
ABRIL	12808	12830	12779
MAIO	12737	12744	12794
JUNHO	12736	12784	12772
JULHO	12864	12746	12843
AGOSTO	12818	12673	12821
SETEMBRO	12704	12758	12720
OUTUBRO	12810	12733	12754
NOVEMBRO	12708	12793	12700
DEZEMBRO	12795	12721	12812

DIAGRAMA DE CLASSES



CONCLUSÃO (COMPLEXIDADE DAS ESTRUTURAS E OTIMIZAÇÕES REALIZADAS)

As nossas estruturas basearam-se sempre em *HashMap*, *HashSet* e *TreeSet*.

De acordo com os testes que foram realizados e apresentados no *GestVendas*, para otimizarmos o nosso trabalho concluímos que estas estruturas eram as mais indicadas para aumentarmos o nível de performance.

Nas *queries*, quando nos pedem organização por ordem decrescente usamos *comparators* que nos ajudaram nessa mesma ordem.

Como tínhamos *queries* que nos pediam pares ou triplos criamos classes que devolviam esses pares ou triplos.