Universidade do Minho

Escola de Engenharia

Licenciatura em Engenharia Informática

Unidade Curricular de

Laboratórios de Informática III

Ano Lectivo de 2014/2015

Gesthiper-OO

Pedro Cunha, a67677 **Stéphane Fernandes, a67681** **Filipe de Oliveira, a67686**

Grupo 21

Junho, 2015

Índice

Índice	2
1. Introdução	3
1.1. Apresentação do Caso de Estudo	3
2. Arquitectura e Descrição da Aplicação	4
2.1. Interface com o Utilizador	6
3. Estruturas de Dados	7
3.1. Módulo de Catálogo	7
3.2. Módulo de Contabilidade	7
3.3. Módulo de Compras	8
4. Testes de Performance	9
4.1. Teste 1 – Tempos de Leitura	9
4.2. Teste 2 – Tempos de Execução de queries	10
5. Conclusão	11

1. Introdução

1.1. Apresentação do Caso de Estudo

O projecto de Java da disciplina de LI3 tem como objectivos consolidar os conhecimentos teórico-práticos adquiridos sobre o paradigma orientado aos objectos, de forma a desenvolver um programa de inserção e consulta de dados sem recurso a bases de dados.

O programa deverá ser capaz de ler três ficheiros.txt e armazenar eficazmente os dados em colecções da Java Collections Framework, de forma a realizar as operações pretendidas com os mesmos, após uma breve análise dos três ficheiros recebidos (FichProdutos.txt, FichClientes.txt e Compras.txt) chegou-se aos seguintes pontos:

- **FichProdutos.txt** cada linha representa o código de um produto vendável no hipermercado, cada código é formado por duas letras maiúsculas e quatro dígitos.
- **FichClientes.txt** cada linha representa o código de um cliente identificado do hipermercado, cada código é formado por duas letras maiúsculas e três dígitos;
- **FichCompras.txt** cada linha representa o registo de uma compra efectuada no hipermercado, nestas linhas está representado um código de produto, um preço unitário, o número de unidades compradas, o tipo de compra (N - Normal ou P - Promoção), o código de cliente e o mês da compra (1..12);

Objectivos/Requisitos:

- Desenhar e implementar estruturas de dados que suportem a aplicação a desenvolver;
- Desenvolver métodos de leitura, povoamento e persistência de dados de forma a tornar a aplicação fiável;
- Implementar métodos de consulta à estrutura de dados que mantenham o encapsulamento da mesma.
- Testar a aplicação em termos de resultados e performance

2. Arquitectura e Descrição da Aplicação

A aplicação foi desenvolvida utilizando o IDE Netbeans. Foi utilizada uma arquitectura baseada na composição de implementações das seguintes Interfaces, na classe Hipermercado:

- ICatalog – Interface que corresponde ao módulo de Catálogo.
- IContabilidade – Interface que corresponde ao módulo de Contabilidade.
- IComprasDB – Interface que corresponde ao módulo de Compras.

A arquitectura não apresenta mecanismo de herança e hierarquia.

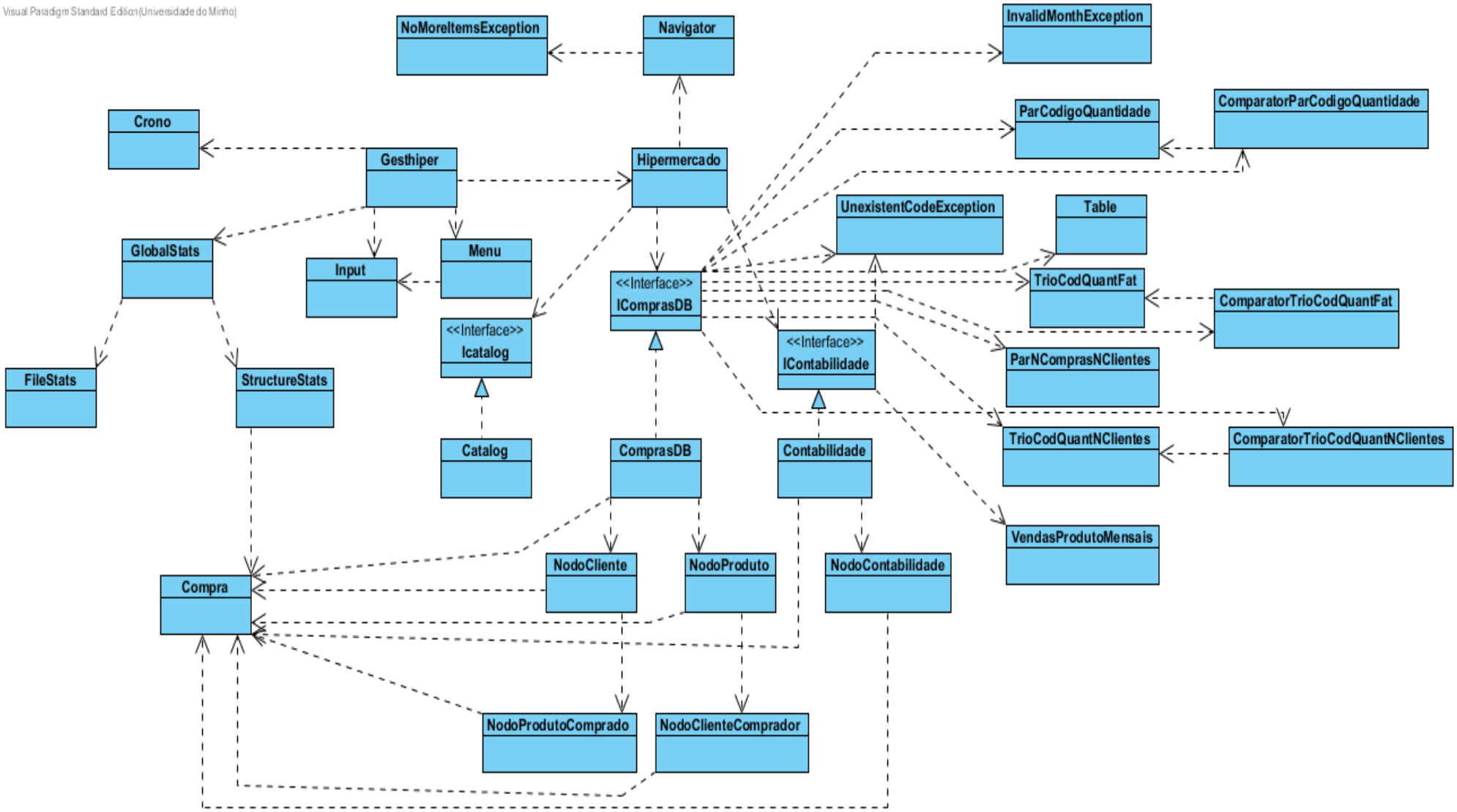
O executável da aplicação foi definido na classe Gesthiper, juntamente com todas as funções auxiliares para implementação dos requisitos. Como dependências directas para a classe Gesthiper foram definidas as seguintes classes:

- Classe Menu, para implementação dos menus de navegação;
- Classe Input, para gerir os erros de introdução de dados pelo utilizador, da autoria dos docentes da unidade curricular;
- Classe Crono, para medição de tempos de execução, da autoria dos docentes da unidade curricular.

Adicionalmente foram definidas classes e comparadores auxiliares de pares de resultados (ex: pares de Código e Quantidade).

A documentação do código desenvolvido encontra-se na directoria *"./dist/javadoc".*

A figura abaixo contém o diagrama de classes da aplicação, com todas as dependências identificadas:



2.1. Interface com o Utilizador

De forma a permitir uma interface intuitiva e simples com o utilizador, foram utilizados menus de opções. Ao iniciar o programa, o utilizador terá primeiro que carregar dados para que a aplicação possa correr (Ilustração 1). Os menus foram implementados através da classe Menu, que permite criar menus de uma forma genérica através de arrays de Strings.

```
1 - Carregar a partir de ficheiros de texto
2 - Carregar a partir de ficheiro de Objectos
0 - Sair
Opção:
```

Ilustração 1 - Menu de Carregamento de Dados

```
************ GESTHIPER ************
1 - Estatísticas da estrutura de dados
2 - Queries interactivas
3 - Guardar Estado
4 - Recarregar Estruturas
0 - Sair
Opção:
```

Ilustração 2 - Menu Inicial

A qualquer momento o utilizador pode voltar atrás nos menus e sair da aplicação no Menu Inicial (Ilustração 2) ou no início da aplicação (Ilustração 1).

3. Estruturas de Dados

No desenvolvimento das estruturas de dados foram instanciadas na Classe Hipermercado implementações das interfaces ICatalog, IContabilidade e IComprasDB.

3.1. Módulo de Catálogo

A implementação deste Módulo seguiu a implementação utilizada no TP de C: um mapeamento de um Character (Primeira letra do Código) para uma árvore de Strings (os códigos dos clientes/produtos/outros.):

```
private HashMap<Character, TreeSet<String>> catalogo;
```

3.2. Módulo de Contabilidade

A implementação deste Módulo seguiu a implementação utilizada no TP de C: um mapeamento de uma String (Código de Produto para um Nodo que contém os campos abaixo assinalados):

```
private Map<String, NodoContabilidade> contabilidade;
```

A classe NodoContabilidade contém as seguintes variáveis de instância:

```
private String codigo;
private int qtdVendidaN[], qtdVendidaP[], nVendasN[], nVendasP[];
private float faturacaoN[], faturacaoP[];
```

Estes arrays referem-se às relações mensais de quantidade vendida de um produto em modo normal ou promocional, de vendas em modo normal e promocional e de facturação em modo normal ou promocional.

3.3. Módulo de Compras

A implementação deste Módulo tem diferenças relativamente ao módulo de compras encontrado no TP de C, existem dois mapeamentos, um de código de Cliente para um nodo de Cliente, e um de código de Produto para um nodo de Produto:

```
private Map<String, NodoCliente> clientes;
private Map<String, NodoProduto> produtos;
```

Os nodos Cliente e Produto têm, respectivamente as seguintes variáveis de instância:

```
private String codigoC;
private int compraMes[];
private int nCompras;
private Map<String, NodoProdutoComprado> prodComprados;
```

//

```
private String codigoP;
private int nVezesComprado;
private int qtdComprada;
private int compradoMes[];
private Map<String, NodoClienteComprador> clientesComp;
```

Os nodos de Produto Comprado por parte de um Cliente e de Cliente Comprador de um Produto têm respectivamente as seguintes variáveis de instância:

```
private String codigoP;
private int qtdCompN[], qtdCompP[];
private float valorN[], valorP[];
```

//

```
private String codigoC;
private int qtdCompradaN[],qtdCompradaP[];
private float valorN[],valorP[];
```

4. Testes de Performance

Para os testes abaixo foi utilizada uma máquina com as seguintes especificações técnicas:

Processador:	Intel(R) Pentium(R) Dual CPU T3200 @ 2.00GHz 2.00 GHz
Memória (RAM):	4,00 GB

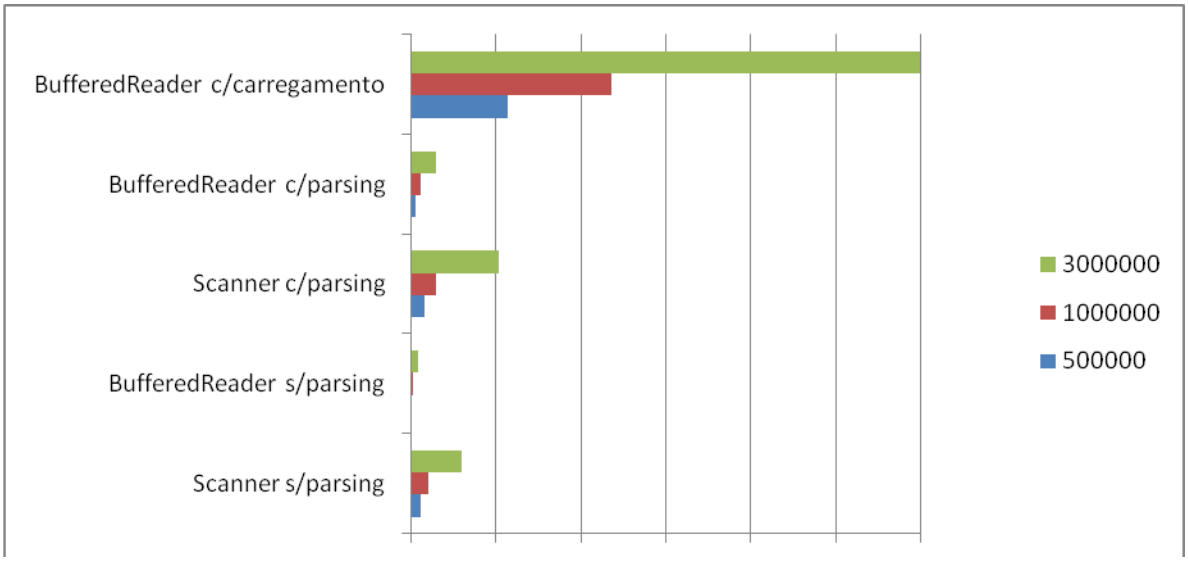
Sistema Operativo: Windows Vista Professional 32 bit

Compilador de Java: JDK7

4.1. Teste 1 – Tempos de Leitura

Tamanho Ficheiro	Scanner s/parsing	BufferedReader s/parsing	Scanner c/parsing	BufferedReader c/parsing	BufferedReader c/carregamento
Compras	2,41398837	0,29452486	3,16948076	1,09708552	19,61533854
	2,10583552	0,25571406	3,22342393	1,05375935	37,45488212
	2,25132189	0,26717278	3,13820835	1,26303874	19,70380956
	2,58744645	0,233405084	2,99647162	1,11005589	21,91460256
	2,22230228	0,22795495	3,27284445	0,93950907	15,24999434
Compras1	4,40038608	0,57148744	5,88062729	2,31156586	49,50132573
	4,12409454	0,60151891	5,68424004	2,60389966	39,10569556
	4,12369589	0,60667906	6,23713852	1,94064469	43,51265091
	4,33207349	0,51771748	5,70296420	1,90409899	56,38700282
	4,17428751	0,646219282	5,89204494	2,02968646	47,73764878
Compras3	12,78837729	1,91474028	17,93383070	6,09814292	120,76542345
	11,76151969	1,42279955	21,41463957	6,03112630	117,33606427
	11,69493497	1,47184740	20,65341908	5,91808931	123,54327200
	11,61437054	1,57950857	24,51593708	5,90135534	130,05862304
	11,39144444	1,59930438	19,70976144	6,23642893	112,03450983

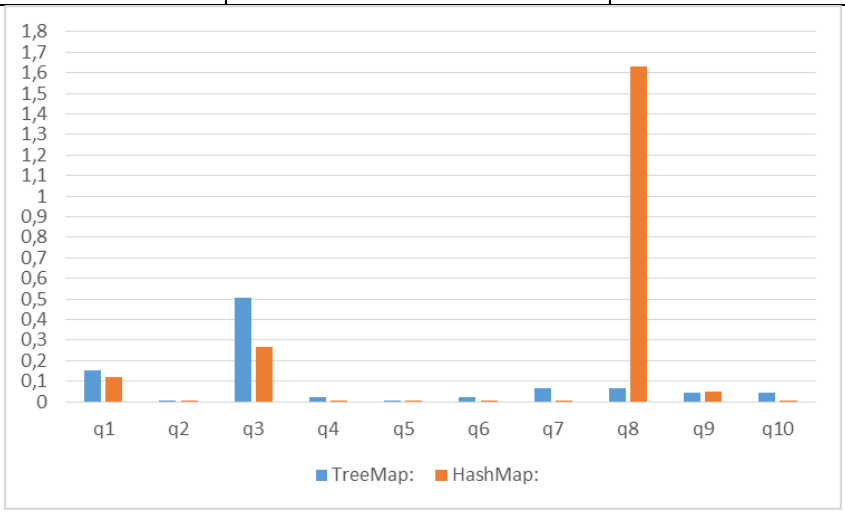
Abaixo encontra-se um gráfico comparativo para cada uma das medidas de teste



4.2. Teste 2 – Tempos de Execução de queries

Produto: JD4775; Cliente: AH849

	TreeMap	HashMap
Q1	0,154389454	0,118068637
Q2	0,004990858	0,00159322
Q3	0,507541601	0,266374916
Q4	0,023024155	0,002986693
Q5	0,000117054	0,000093587
Q6	0,019668422	0,002862375
Q7	0,064630002	0,008128687
Q8	0,064630002	1,631797896
Q9	0,04099711	0,050789416
Q10	0,045982939	0,050789416



5. Conclusão

Neste momento o programa desenvolvido cumpre os requisitos funcionais enunciados, bem como as regras definidas referentes a encapsulamento e modularidade.

Ao carregar os ficheiros para as estruturas notámos uma diferença enorme de tempo de execução entre a utilização do BufferedReader com parsing e a utilização da mesma classe, parsing e posterior envio para as estruturas. Tal ocorre devido à replicação de dados que ocorre no módulo de Compras (Um produto deverá guardar todos os seus clientes compradores e um cliente terá de guardar todos os produtos comprados). Achámos que o tempo despendido compensa pois no momento de realizar as queries interativas os resultados são apresentados quase imediatamente.

No momento de carregamento do ficheiro também o módulo estatístico será carregado, com recurso a pequenas queries à estrutura de dados para obter as várias relações mensais bem como pequenos dados quantitativos. O carregamento do módulo estatístico dá-se quase instantaneamente, dada a estrutura interna do programa.

Ao guardar os dados internos para um ficheiro de objectos notámos um grande tempo de espera ao realizar esta opção (aproximadamente 4 minutos). Achámos esta situação aceitável já que a escrita de ficheiros em modo objecto é muito mais pesada que uma escrita em modo texto para um ficheiro.

No geral fazemos uma apreciação positiva do trabalho realizado mas notámos alguns pontos a melhorar para aprimorar os resultados obtidos:

- Realizar uma melhor partição das classes existentes, tentando introduzir uma maior noção de separação de “camadas”.
- Ao realizar a navegação sobre listas, permitir ao utilizador escolher o tamanho da “página” a apresentar.
- Melhorar a interface com o utilizador, bem como a apresentação das queries, tanto estatísticas como interactivas.