Márcia Teixeira – 201706065 – up201706065@fe.up.pt

Pedro Esteves – 201705160 – up201705150@fe.up.pt

09/01/2019

Turma 5 – Grupo 3

Faculdade de Engenharia da Universidade do Porto  
Mestrado Integrado em Engenharia Informática e Computação – 2º Ano

TEMA 8 – cADEIA DE fARMÁCIAS

Algoritmos e Estruturas de Dados – Trabalho Prático

Índice

[1. Descrição do tema 2](#_Toc534725518)

[2. Implementação – Classes/Estruturas Implementadas 3](#_Toc534725519)

[3. Casos de utilização 6](#_Toc534725520)

[4. Dificuldades 7](#_Toc534725521)

[5. Esforço dos membros do grupo 8](#_Toc534725522)

# Descrição do tema

O objetivo do trabalho desenvolvido é complementar a aplicação para a gestão de uma cadeia de farmácias desenvolvida na primeira parte do trabalho prático. Esta nova aplicação utiliza estruturas de dados não lineares, mais especificamente árvores binárias, filas de prioridade e tabelas de dispersão.

Uma cadeia é composta por farmácias, funcionários e clientes. Uma farmácia tem um stock, encomendas pendentes e um histórico de vendas. Nos funcionários da cadeia estão todos os funcionários que trabalham ou trabalharam na mesma, tendo cada funcionário um salário e exercendo um cargo numa farmácia. Os clientes vivem num determinado distrito e têm um histórico de vendas.

# Implementação – Classes/Estruturas Implementadas

* 1. CadeiaFarmacias

Um objeto da classe CadeiaFarmacias representa uma cadeia de farmácias, armazenando todas as farmácias (num vetor), clientes (numa árvore binária, um set) e funcionários (numa tabela de dispersão). Esta classe contém principalmente métodos para alterar ou obter os seus atributos, adicionar e remover farmácias, funcionários e clientes.

* 1. Data

Um objeto da classe Data corresponde a uma data, caracterizada pelo dia, mês e ano. A data pode tanto ser definida pelo utilizador como corresponder à data atual do sistema (usando a biblioteca ctime).

* 1. Hora

Um objeto da classe Hora corresponde a uma data, caracterizada pela hora, minutos e segundos. A hora pode tanto ser definida pelo utilizador como corresponder à hora atual do sistema (usando a biblioteca ctime).

* 1. Farmacia

Um objeto da classe Farmacia corresponde a uma farmácia, caracterizada por um nome, morada, gerente e diretor técnico. Cada farmácia tem uns certos produtos em stock (priority\_queue stock, com os produtos e as respetivas quantidades disponíveis), um historial de vendas (vetor vendas) e encomendas pendentes (priority\_queue encomendas). Esta classe contém métodos para alterar e obter os seus atributos, obter informações sobre estes (por exemplo, número de vendas total ou em datas específicas), adicionar e remover vendas ou produtos e ordenar as vendas.

* 1. FarmaciaInexistente

Quando se tenta remover uma farmácia que não existe na cadeia, é lançada uma exceção do tipo FarmaciaInexistente. Um objeto desta classe é caracterizado pelo nome da farmácia que se tentou remover.

* 1. Pessoa

Um objeto da classe Pessoa corresponde a uma pessoa, caracterizada pelo seu nome, morada e número de contribuinte. Esta classe contém métodos para alterar e obter os seus atributos.

* 1. Funcionario

A classe Funcionario é derivada publicamente da classe Pessoa. Um objeto desta classe representa um funcionário, que tem como atributos, para além dos da classe base, o seu salário, farmácia em que trabalha e cargo exercido. Esta classe contém principalmente métodos para alterar e obter os seus atributos.

* 1. FuncionarioInexistente

Quando se tenta remover um funcionário que não existe na cadeia, é lançada uma exceção do tipo FuncionarioInexistente. Um objeto desta classe é caracterizado pelo número de contribuinte do funcionário que se tentou eliminar.

* 1. FuncPtr

A estrutura FuncPtr contém um apontador para um funcionário e um booleano que indica se é funcionário atual ou antigo da cadeia.

* 1. Cliente

A classe Cliente é derivada publicamente da classe Pessoa. Um objeto desta classe representa um cliente, que tem como atributos, para além dos da classe base, o seu historial de compras e o distrito. Esta classe contém principalmente métodos para alterar e obter os seus atributos, bem como informações sobre os mesmos (por exemplo, número de vendas).

* 1. Produto

Um objeto da classe Produto representa um produto, caracterizado pelo seu código, nome, preço e descrição. Um produto pode ser ou não passível de ter receita e, caso seja, pode ou não ser possível a sua venda sem receita, e a sua venda com receita tem associado um valor de comparticipação. Esta classe contém métodos para alterar e obter os seus atributos.

* 1. ProdutoInexistente

Quando se tenta remover um produto que não existe no stock da farmácia, é lançada uma exceção do tipo ProdutoInexistente. Um objeto desta classe é caracterizado pelo nome do produto que se tentou remover.

* 1. ProdutoStock

Um objeto da classe ProdutoStock representa um produto em stock, caracterizado por um apontador para o produto e a sua quantidade em stock. Esta classe contém métodos para obter e alterar os seus atributos, bem como um overload do operador <.

* 1. Receita

Um objeto da classe Receita representa uma receita, caracterizada pelo seu número, pelo nome do médico que a prescreveu e pelo cliente ao qual a receita foi prescrita. A receita tem produtos receitados em determinada quantidade. Esta classe contém métodos para alterar e obter os seus atributos e informação sobre os mesmos.

* 1. Venda

Um objeto da classe Venda representa uma venda, caracterizada pelo seu código, data e hora. Uma venda tem associado um cliente que realizou a venda e pode ter também associada uma receita. Uma venda tem vários produtos vendidos, numa determinada quantidade e com determinados valores de IVA e comparticipação (este valor será 0 se o produto não for passível de receita ou se, caso seja e seja possível a sua venda sem receita, não conste na receita associada à venda). Esta classe contém métodos para alterar e obter os seus atributos e informações sobre os mesmos, bem como um método para adicionar produtos à venda.

* 1. Encomenda

Um objeto desta classe representa uma encomenda. Contém uma string com o nome fornecedor, um código da encomenda, que é sequencial, o código do produto a encomendar, a quantidade do produto a encomendar e a quantidade do produto em stock

# Casos de utilização

Ao começar o programa, o utilizador tem duas opções (para além da opção de sair): criar uma nova cadeia de farmácias ou importar uma cadeia já existente num ficheiro. De seguida tem um menu de gestão da cadeia de farmácias, onde tem mais opções: ver, adicionar e remover (despedir, no caso dos funcionários) farmácias, funcionários ou clientes e ordenar farmácias (por vários critérios e ordem crescente ou decrescente). Relativamente aos funcionários podem ver-se todos os funcionários, apenas os atuais ou apenas os antigos. Pode também escolher ver os dados da cadeia de farmácias, gerir uma farmácia, funcionário ou cliente específico.

Na gestão de uma farmácia é possível ver os dados da farmácia, mudar o gerente ou o diretor técnico, consultar e alterar o stock (adicionar, alterar quantidades e remover produtos), ver historial de vendas (completo ou entre duas datas específicas). Pode também realizar uma venda. Esta venda pode ou não ter associada uma receita; caso tenha, são pedidos os dados da mesma. Pode ainda realizar ou receber encomendas e ver a lista de encomendas pendentes.

Na gestão de um funcionário é possível ver os dados do funcionário, alterar a farmácia em que trabalha, cargo, salário ou morada.

Na gestão de um cliente é possível ver os dados do cliente, ver o seu historial de compras e alterar a sua morada e distrito.

# Dificuldades

Na implementação deste trabalho, uma vez que tivemos mais tempo e era baseado na parte anterior, não encontrámos tantas dificuldades.

O mais difícil foi implementar a priority\_queue (fila de prioridade), devido à interpretação do enunciado e à própria dificuldade de manipulação desta estrutura de dados. Desta forma, optamos por implementar duas filas de prioridade, uma para as encomendas e outra para os produtos em stock.

# Esforço dos membros do grupo

As tarefas desta parte do projeto foram igualmente divididas pelos dois membros do grupo, tendo o esforço de ambos sido equivalente.