Faculdade de Engenharia da Universidade do Porto

MIEIC - Algoritmos e Estruturas de Dados

Relatório trabalho prático - Parte 1

Cadeia de Farmácias

22/11/2018

Turma 5 – grupo 1

David Luís Dias da Silva – up201705373@fe.up.pt

Gaspar Santos Pinheiro – up201704700@fe.up.pt

Luís Pedro Pereira Lopes Mascarenhas Cunha – up201706736@fe.up.pt

Índice

Descrição do tema

O tema escolhido de entre os propostos para o trabalho prático consiste na realização de um programa para gerir uma cadeia de farmácias, fornecendo ao utilizador funcionalidades básicas que lho permitam. Os principais elementos da cadeia de farmácias são as farmácias, os clientes, os empregados e as vendas (nas quais receitas dos pacientes podem ser aviadas).

As principais funcionalidades presentes no programa permitem lidar com estes elementos e são realizar vendas, alterar dados referentes aos empregados e clientes da cadeia, adicionar farmácias à cadeia e gerir o seu *stock*, bem como consultar dados relativos a todos os elementos da cadeia.

Outra funcionalidade é a de ser possível gerir diferentes cadeias (em utilizações diferentes do programa) e guardar os dados relativos à cadeia em ficheiros de texto para utilização futura.

Solução implementada

O objetivo do trabalho era modelar o problema, neste caso, o de gerir uma cadeia de farmácias, e construir uma solução recorrendo ao paradigma de Programação Orientada por Objetos e às ferramentas da linguagem C++.

De forma a alcançar o objetivo, criámos classes que representassem as informações essenciais de cada um dos elementos. As classes criadas são:

- Cadeia:
  - A classe cadeia guarda a informação relativa a toda a cadeia de farmácias. É composta por apontadores para os seus empregados, clientes, farmácias, e vendas, guardados em vetores.
  - Os empregados e clientes estão ordenados pelo seu ID único, o que permite pesquisa binária e evita ambiguidades.
  - As farmácias estão ordenadas por nome (não são permitidas farmácias com o mesmo nome) e as vendas por data e hora em que foram realizadas.
  - A cadeia tem também um nome para permitir que seja identificada quando se quer carregar os seus dados a partir de ficheiros.
- Farmácia:
  - A classe farmácia tem atributos que representam o seu nome e a sua morada e é composta pelos seus empregados, sendo um deles, o gerente, fundamental, pelas suas vendas e os produtos em stock.
  - As farmácias guardam os seus empregados também num vetor, ordenados pelo seu ID.
  - O *stock* é guardado por uma estrutura *map*, onde são guardados os produtos (sendo a *key* um apontador para o produto e o *value* a quantidade).
  - Em relação aos produtos, é guardado um apontador para permitir utilizar capacidades do paradigma de orientação a objetos, como o polimorfismo, pois a classe Produto tem a classe Medicamento como classe derivada.
- Venda:
  - A classe venda é constituída pelos nomes e IDs do empregado e cliente responsáveis pela venda e pelos produtos (guardados tal como na Classe farmácia).
  - Nesta classe, é feito o uso da classe Timestamp, que serve, na sua essência, para guardar a data e hora a que foi realizada uma dada venda, para que possa ser identificada.
  - A venda possui ainda métodos que permitem adicionar-lhe receitas ou produtos e calcular o valor total a pagar, consoante os impostos e descontos que possam existir.
- Receita:
  - A classe receita é constituída pelo nome e NIF do paciente, bem como o nome do médico.

- A classe receita guarda ainda os produtos receitados do mesmo modo que os produtos são guardados na classe Venda e Farmácia.

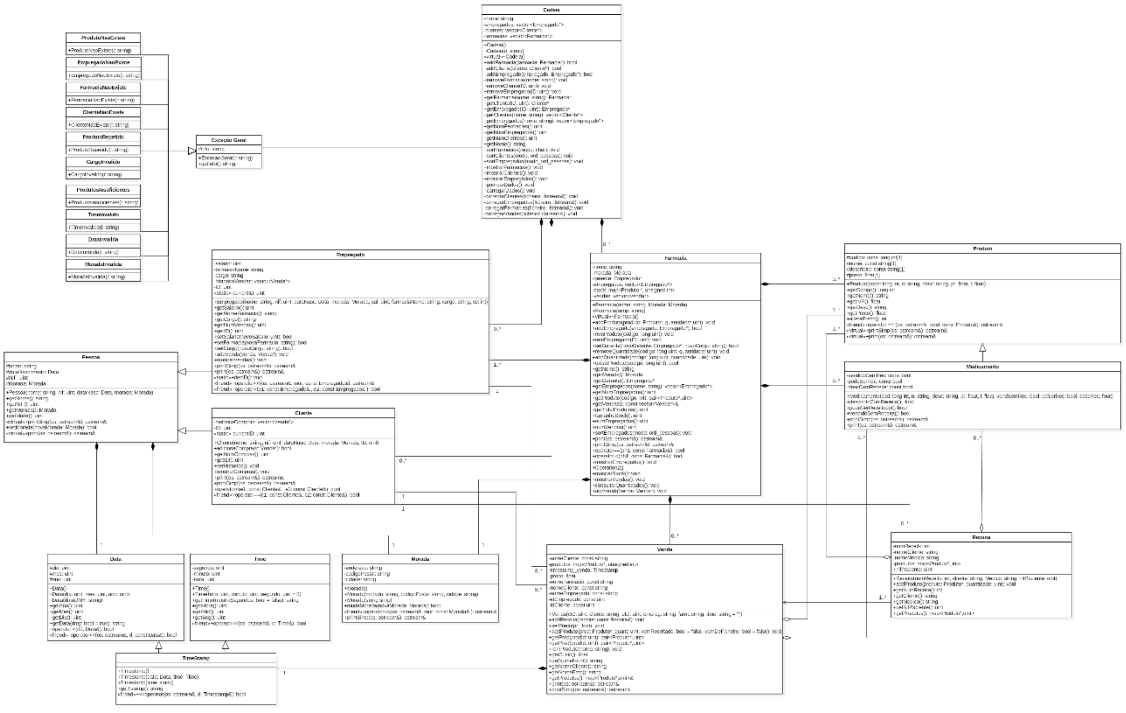
- Produto:
  - A classe produto é composta pelo nome, código, descrição, imposto e preço base.
- Medicamento:
  - Classe derivada de Produto, possui, para além dos atributos em comum com Produto, o valor de desconto no caso de o medicamento ser receitado e variáveis booleanas que indicam se o Medicamento pode ser receitado ou vendido sem receita médica.
- Pessoa:
  - A classe Pessoa é constituída pelo nome, NIF, data de nascimento e morada da pessoa cada objeto representa.
- Empregado:
  - A classe Empregado deriva da classe Pessoa e adiciona a esta atributos próprios de um empregado de uma farmácia, como o seu historial de vendas, cargo, salário e nome da farmácia onde trabalha.
  - Para além disso, cada objeto desta classe tem um ID único, definido através da utilização de um atributo estático.
- Cliente:
  - A classe Cliente deriva da classe Pessoa e adiciona a esta o historial de compras do respetivo cliente que o objeto representa.
  - Tal como a classe Empregado, cada objeto desta classe tem um ID único, definido através da utilização de um atributo estático.
- Morada:
  - A classe Morada é utilizada para representar a morada quer de pessoas quer de farmácias, e é constituída por código-postal, endereço e localidade.
- Data:
  - Classe utilizada para representar um dia do calendário, em que o construtor padrão inicializa um objeto com a data atual.
- Time:
  - Classe utilizada para representa horas, minutos e segundos, em que o construtor padrão inicializa um objeto com a hora atual.
- Timestamp:
  - Classe derivada das classes Data e Time que representa a data e hora. O construtor padrão inicializa um objeto com a data e hora atuais.
- ExcecaoGeral:
  - Esta classe foi criada para criar facilmente exceções úteis para a implementação do programa, pois a funcionalidade desejada para cada exceção era idêntica. Desta classe derivam algumas classes para representar exceções mais específicas, cuja funcionalidade é a mesma da classe base.

De modo a utilizar as classes de forma a permitir ao utilizador efetivamente gerir uma farmácia, bem como demonstrar as suas capacidades, construímos uma *interface* simples na linha de comandos, baseada em menus, nos quais o utilizador é confrontado com várias opções possíveis de entre as quais deve escolher uma.

Para implementar a funcionalidade de aviar receitas, importante numa farmácia, e por forma a tentar replicar aquilo que nos parece ser o funcionamento de uma farmácia real (em que muitas das receitas têm formato digital), introduzimos as receitas no programa através de ficheiros de texto, que são lidos pelo programa de modo a realizar vendas.

Diagrama de Classes UML

- Para ver em mais detalhe consulte o ficheiro UML.png ou UML.jpg



Casos de utilização

O programa inicia dando duas hipóteses ao utilizador, criar uma nova cadeia de farmácias ou carregar os dados de uma cadeia de farmácias. Para a segunda opção, o utilizador deve indicar o nome da cadeia que pretende continuar a gerir.

De seguida, no menu principal, o utilizador tem quatro opções, que consistem em gerir clientes, empregados e farmácias e realizar vendas.

Dentro da funcionalidade realizar venda, o utilizador deve especificar a farmácia na qual a venda está a ser realizada, tal como o empregado responsável. O utilizador deve ainda indicar o cliente, devendo criar uma ficha de cliente para o caso deste ainda não existir. Depois de especificados estes dados, o utilizador pode adicionar produtos e receitas à venda, bem como consultar o "saco de compras" até ao momento. À medida que produtos e receitas são adicionados à venda, a possibilidade do *stock* da farmácia satisfazer a procura é verificada. Para terminar, é apresentado ao utilizador o custo da venda, de modo a este proceder à confirmação.

A funcionalidade gerir farmácias permite:

- Ver um resumo geral das farmácias existentes na cadeia.
- Consultar informações relativas a cada farmácia, como os empregados, vendas e produtos.
- Adicionar farmácias novas à cadeia, sendo para isso necessário criar um gerente para a nova farmácia.
- Gerir o *stock* de cada farmácia, permitindo adicionar produtos e remover produtos
- Alterar o gerente de uma farmácia.

A funcionalidade gerir clientes permite:

- Ver uma listagem com informações dos clientes da cadeia, ordenados por um critério à escolha pelo utilizador.
- Gerir um cliente em específico, onde se pode consultar as compras de um determinado cliente, bem como alterar a sua morada.

A funcionalidade gerir empregados permite:

- Ver uma listagem com informações dos empregados da cadeia, ordenados por um critério à escolha pelo utilizador.
- Gerir um empregado, podendo alterar informações relativamente a este, tal como o cargo, salário, morada e farmácia onde trabalha.
- Adicionar e remover empregados.

Dificuldades encontradas

As principais dificuldades encontradas no decorrer do projeto foram a gestão de tempo e organização. Em relação à procura de uma solução para o tema proposto, encontrámos algumas dificuldades na decisão sobre estruturas de dados a utilizar e situações não especificadas pelo enunciado. A complexidade gerada pela procura de representar a situação-problema detalhadamente sob a forma de classes mostrou-se como um desafio, obrigando-nos a fazer sacrifícios e a procurar pensar na perspetiva do "cliente". Para além disso, consideramos que em situação real, a comunicação com o cliente seria essencial, pois é ele que tem uma melhor capacidade de avaliar as suas necessidades e as funcionalidades essenciais do programa, bem como o design que leve a uma utilização mais eficaz e fluida.

No que refere à linguagem usada, não encontrámos dificuldades, pois não é a primeira vez que contactamos com ela.

Esforço dedicado

O esforço dedicado pelos intervenientes no trabalho foi equiparável, pois inicialmente procedemos à divisão de tarefas e todos os membros cumpriram os seus deveres. Não obstante desta divisão, a construção do projeto não foi efetuada de forma estanque, tendo os membros do grupo contribuiram de uma forma proativa para a resolução do problema, cooperando e discutindo abordagens, soluções e melhorias. No entanto, consideramos que o trabalho não está perfeito, e que há espaço para uma otimização e correção de eventuais problemas.