Gestor de compras *online*

Programação Orientada aos Objetos

2021/2022

Nome Estudante:

Hugo Filipe Amaral dos Santos Priva

Número Estudante:

2020220399

Email Estudante: uc2020220399@student.uc.pt

Data de entrega: 8 de dezembro de 2021

Data a entregar: 9 de dezembro de 2021

Conteúdo

1 Introdução

Para projeto final da cadeira Programação Orientada aos Objetos (POAO) foi nos dado um enunciado com o objetivo de o reproduzir usando a linguagem de programação ensinada, java.

O enunciado propõe a produção de um programa capaz de gerir os clientes, os produtos, as suas vendas e ainda promoções. O programa deve ainda propocionar ao cliente um sistema de login, a possibilidade de fazer uma compra e consultar as mesmas.

2 Desenvolvimento

2.1 Início

De forma a gerir os parametros previamente enunciados, procedeu-se à criação de diversas classes de forma a optimizar e organizar o código. A primeira classe a ser usada é 'Supermercado', esta é considerada o *main*, onde vão ser chamadas todas as classes criadas e os respetivos métodos.

No caso de ser a primeira vez que o programa é executado, ou seja o ficheiro *.obj* ainda não existe, o ficheiro *.txt* é lido através do método *lineParsing* e os dados são subdivididos por *ArrayLists* do determinado *Object*, e de seguida são usados para definir o construtor do tipo *FileDealer* que irá conter, e partilhar informação ao longo de todo o programa. Se o ficheiro *.obj* já existir, só se diferencia a forma como se obtêm os dados, neste caso é executado o método *fromObjectFile*. Este método recorre às bibliotecas *FileInputStream* e *ObjectInputStream* para ler os dados do ficheiro de objetos e, por fim, retorna o construtor *FileDealer* com a informação lida.

2.2 Classe *Cliente* e *Compras*

2.2.1 *Cliente*

Esta classe tem como função a representação de cada cliente. Cada um deste clientes é caracterizado pelo seu nome, morada, mail, número de telefone e data de nascimento. No entanto, de modo a facilitar a implementação do programa e diferenciação entre clientes *frequentes* e *regulares*, o construtor toma como argumento um *Boolean*: *freq*.
Toma ainda uma *ArrayList«Compras»* que funciona como um carrinho de compras, em que são guardadas todas as compras efetudas pelo cliente.

2.2.2 *Compras*

Esta classe representa a maioria das compras efetuadas, em que cada uma é composta por um *ArrayList«Produtos»* e pelo precoTotal. Nesta classe existem três métodos de maior relevância, que são : *buyMobilia*, *buyLimpeza* e *buyAlimentares*. Estes métodos recebem como parâmetros o construtor que guarda a informação de cada *compraAtual*, um *Boolean* que caracteriza o tipo de cliente, duas *ArrayLists* uma do tipo de *Objeto* a ser comprado e uma de *Produtos* que guarda o stock da aplicação, recebe ainda um valor inteiro *promo* que define a promoção a ser aplicada.

2.3 Login

Na execução do programa é pedido a inserção da data corrente e login. O *login*, que se encontra na classe 'Supermercado', recebe o input do utilizador e verifica se este encontra no formato certo para ser considerado um email. De seguida, é comparado com os restantes emails proporcionados pelos *ArrayLists* de *Clientes* frequentes e regulares. Por fim, se encontrar o *Cliente* retorna-o, caso contrário pede novamente input até poder progredir.

2.4 Menu

Após a recolha de dados e formalidades iniciais, o utilizador tem a opção de :

2.4.1 Proceder a uma compra

Como foi mencionado, no inicio, é pedida a data corrente ao utilizador. Se esta data estiver dentro do periodo, predefinido, de promoções temporárias o programa irá definir um dos tipos de produtos para que lhe seja aplicado uma das promoções.

A modalidade 'pague-menos', em que a cada um de stock do produto será deduzido 5% até um máximo de 50%. A outra modalidade é 'leve 4 pague 3', em que tal como o nome sugere a cada 4 de stock do produto é retirado 1 ao cálculo do preço.

Com ou sem promoção o procedimento é bastante semelhante. Os três métodos previamente referidos na secção 2.2.2 apenas diferem num dos parâmetros que é a receção da *ArrayList«Objeto»* em que o tipo de *Objeto* difere perante o método chamado.

Estes métodos funcionam recebendo o número de itens a comprar e comparando com o stock disponivel, e se o número for aceitável, procede a calcular o preço da transação. É de seguida atualizado o construtor da compra atual, a *ArrayList* de *Produtos*.

2.4.2 Mostrar compras efetuadas

Esta opção executa o método *printAll* que percorre a *ArrayList«Compras»* associada ao cliente atual e mostra o seu conteúdo. Esta função é ainda usada para mostrar o stock de cada tipo de produto antes de cada compra. Caso esta função seja chamada sem que nenhuma compra esteja registada, o utilizador recebe uma mensagem informativa do sucedido.

2.4.3 Sair do Programa

No fim de cada compra, fim do programa, a *ArrayList«Compras»* é atualizada com a nova *Compra*. Através do contrutor *FileDealer fd1* é chamada a função *writeToObjectFile* que através das bibliotecas *FileOutputStream* e *ObjectOutputStream*, e dos parâmetros associados: *File* representa o ficheiro associado ao *path* do ficheiro em que vai ser guardada a informação guardada pelo segundo parâmetro, o construtor *FileDealer* e do cliente atual *cl*; com a variável *cl* encontra-se o cliente atual numa das *ArrayList* do tipo *Cliente* e atualiza-se o seu parâmetro de *ArrayList«Compras»*; por fim, toda a informação guardada em *fd1* é escrita no ficheiro de objetos.

3 Conclusão

Referências