FCTUC FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Licenciatura em Engenharia Informática

2º Ano | 1º Semestre | 2021/2022

Unidade Curricular de Programação Orientada a Objetos

Projeto

Online shop

Autores:

André Colaço| 2020220301

Lino Varela | 2020220433

1 Índice

2 Introdução

Este projeto tem como objetivo desenvolver uma aplicação que permita a organização de vendas de produtos de uma cadeia de supermercados. A aplicação terá de permitir realizar certas operações como realizar o *login*, efetuar compras e consultar as compras realizadas.

Foi desenvolvida na linguagem java e para estruturar o projeto elaborou-se um UML (diagrama de classes).

3 CLASSES E MÉTODOS

Para desenvolver a aplicação foram utilizadas classes responsáveis por um objetivo específico. Estas têm atributos e métodos que permitiram gerir e operar o que se pretende com cada classe.

Product - superclasse abstrata com os atributos: *"name"*, *"identifier"*, *"stock"* e *"price"*. Estes permitem descrever os produtos. Contém o método *"createProduct"* que irá ser usado nas subclasses. Nestas, dependendo do tipo de produto, definem-se os atributos característicos do tipo de produto.

Subclasses de **Product**:

Em cada uma existe um método "*create*", que serve para criar cada tipo de produto.

- **Food** - Sendo uma subclasse, terá os mesmos atributos da sua superclasse. Esta também terá os atributos *"fatPercentage"* e *"calories"*.

- **Cleaning** - Além dos atributos comuns a **Product**, também constitui *"toxicityDegree"*.

- **Furniture** - *"height"," length", "depth"," weight"* são os atributos adicionais.

Client - Superclasse abstrata para caracterizar os clientes. Tem como atributos: *"name", "address", "email", "phone", "birthday", "type"* e *uma arrayList "carts".* Identicamente à classe **Product**, existe um método que irá ser usado nas subclasses, "*createClient*". Outro método existente é *"furnitureTransport"* que calcula o peso total dos produtos do tipo *"furniture".* Dependendo deste valor, adiciona-se ao preço total o valor de 10€. Também existe um método chamado "*addCart*", que irá adicionar as compras à arrayList "carts".

O atributo *"email"* permite fazer o *login* na aplicação por parte do cliente. O *"type"* distingue o cliente entre regular e frequente, estes serão explicados mais à frente. Por fim, *"cart"* é uma arrayList que irá guardar as compras do cliente.

As subclasses de **Client** são duas, **Frequent** e **Regular**, ambas com os mesmos atributos que a superclasse e, para além desses, com o atributo "*transportPrice".* Este é importante para distinguir o preço total de uma compra feita pelo cliente, pois para um cliente frequente, se este efetuar compras acima de 40€, terá o transporte ao domicílio gratuito (15€ para o transporte se as compras forem inferiores a esse valor). Já para os clientes regulares, existe um preço fixo de 20€. Para isto existe um método *"transport"*, nas duas classes (método abstrato) que calcula o preço do transporte para cada caso.

Também existe em cada subclasse um método "*create*", para criar os mesmos.

Promotion - superclasse abstrata que permite conceder aos produtos promoções. Os atributos desta são: *"product", "startDate", "finishDate"* e *"type".* Esta apenas tem o método abstrato *"discount",* com o parâmetro *"quantity",* que permitirá aplicar ao produto uma promoção. Com o atributo *"type"* diferencia-se o tipo de desconto que.

As subclasses existentes são **PaytThreeGetFour** e **PayLess**. Ambas com atributo *"discountprice"*. Estas classes têm, como mencionado, o método *"discount"* que, dependendo da quantidade de produto comprada, calcula o preço final deste com o desconto aplicado.

Purchase - classe bastante importante que tem como atributos o *"product"* e *"quantity"*. Sendo estes o produto que se compra e a quantidade do tal.

Esta classe tem o método *"buy"* que permite fazer uma compra. Recebendo como parâmetros a compra efetuada (*"Purchase p"*) e o carrinho a que se adiciona a compra (*"Cart cart"*), este método irá retirar ao stock de produto existente a *"quantity"* que se pretende comprar (se a quantidade que se quer comprar for maior que o stock, considera-se que se quer comprar tudo e produto deixa de ser disponibilizado) e adicionar ao carrinho a compra realizada.

Cart - classe que contém as compras realizadas guardadas. Os atributos são: a arrayList *"purchases"* e *"totalPrice"*.

O método *"addPurchase",* desta classe, guarda os produtos comprados no carrinho (arrayList *"purchases"*) e vai calculando o preço total da compra tendo em conta se o produto comprado tem promoção e de que tipo.

Date - classe com os atributos *"day"*, *"month"* e *"year".* Tem os métodos *"newDate", "checkDate"* e *"dataComparation"*. O primeiro permite criar uma nova data, o segundo verifica se a data é valida e, por fim, o último método verifica se a data é maior ou igual que a data recebida como parâmetro.

File - classe que trata dos ficheiros utilizados na aplicação. Tem apenas o atributo *"name"*, que serve para indicar o nome do ficheiro. Esta classe tem três métodos importantes:

- Primerio temos o método *"readTextFile"* que é utilizado no primeiro arranque do programa, pois este lê o ficheiro de texto que contém os dados (clientes, produtos, promoções) predefinidos e os coloca em arrayLists (diferentes arrayLists para o tipo de dado recebido da leitura do ficheiro) que são depois colocados no ficheiro de objetos.

- Os dados do ficheiro de texto são analisados e adicionados às respetivas arrayLists através do método *"parseLine"*. A informação guardada no ficheiro de texto está organizada da seguinte forma:
  - Em cada linha do ficheiro está a informação completa de um produto, cliente frequente, cliente regular ou promoção.
  - Essa informação está separada por "+". Por exemplo: "Food+pizza+…" ("Food" – produto, "pizza" – nome).
  - Quando existe uma data, que é no caso da data de nascimento do cliente e nas datas da promoção, esta está separada por "/".

Dividindo cada linha em partes (cada parte com informação diferente), cria-se um novo objeto (ex: novo produto) e adiciona-se a arrayList correspondente.

- Posto isto, estas arrayLists são adicionadas ao ficheiro de objetos pelo método *"writeObjFile"* que cria o próprio ficheiro e adiciona lá a informação.

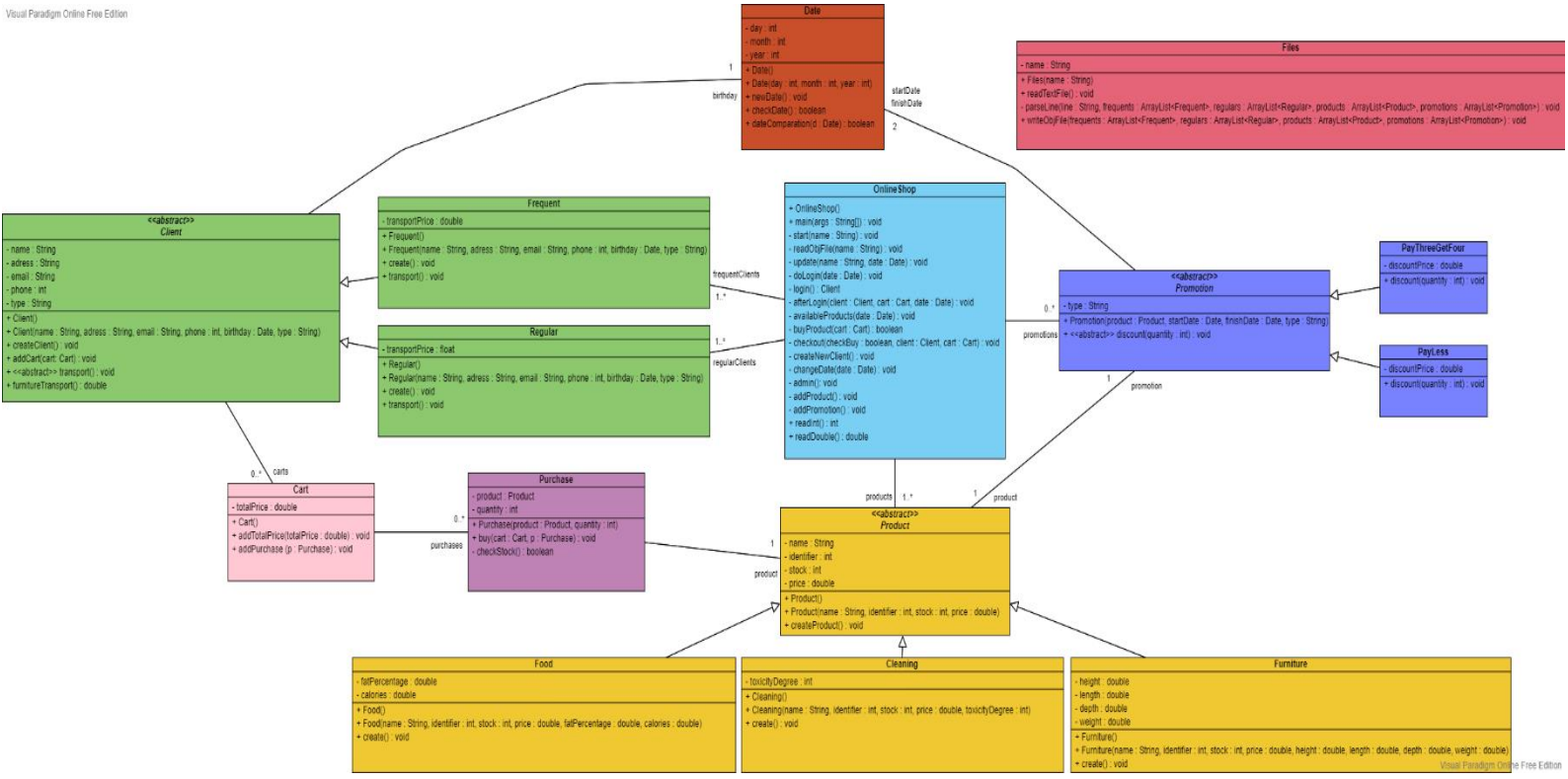
OnlineShop – classe principal onde está o programa organizado, contém bastantes métodos que permitem uma fácil leitura e compreensão. Os métodos que a classe tem (além do "*main*"que chama o "*start*") são:

- "*start*" – começa-se por ler o ficheiro de texto, na primeira execução, que contém os dados predefinidos da aplicação. Lido o ficheiro, são adicionados ao ficheiro de objetos os dados. Após guardar a informação, começa-se as interações com o utilizador, em que este pode decidir o que pretende fazer com o programa.
- "*readObjFile*" – realiza a leitura do ficheiro de objetos para guardar a informação nas respetivas arrayLists.
- "*update*" – verifica as promoções existentes nessa data e atualiza o ficheiro de objetos com as arrayLists.
- "*doLogin*" e "*login*" - permitem ao utilizador fazer login na aplicação com o uso do email, verificando se este existe e se sim, cria um carrinho (*"cart"*) para as compras que realizar.
- "*afterLogin*" – menu após *login*, apresenta ao utilizador as ações que o cliente pode realizar.
- "*availableProducts*" – método que mostra ao cliente as promoções e os produtos disponíveis.
- "*buyProduct*" - adiciona ao carrinho a compra que se realiza e retira ao produto a quantidade comprada.
- "*checkout*" – método utilizado no final das compras, caso o cliente tenha comprado algum produto. Adiciona à arrayList *"cart"* (do cliente) o carrinho, calcula o preço do transporte tendo em conta o tipo de cliente e adiciona esse preço ao preço total do carrinho. Por fim, mostra a lista de compras que o cliente fez com o respetivo preço.

- "*createNewClient*" – cria um novo cliente (frequente ou regular) e adiciona a respetiva arrayList.
- "*changeDate*" – alterar a data utilizada no programa.
- "*Admin*" – método com o Menu que permite adicionar produtos e promoções.
- "*addProduct*" – cria um novo produto (**Food**, **Cleaning** ou **Furniture**).
- "*addPromotion*" – cria uma nova promoção (**PayLess** ou **PayThreeGetFour**) para um produto existente que não tenha uma promoção.
- "readInt" e "*readDouble*" – método que pede ao utilizador um valor *int* ou *double e* verifica se existem exceções.

4 UML

Como mencionado na introdução, foi utilizado um UML (diagrama de classes) para estruturar o projeto.



5 Execução

Ao executar a aplicação são dadas ao utilizador as seguintes opções:

```
(1)-Login.
(2)-Create new client.
(3)-Change the date.
(4)-Admin.
(5)-Exit.
Current Date-4/12/2021
Choose what you want to do by the number ->
```

(1) - Inserir email (cliente);
(2) - Criar cliente com novos dados (nome, adereço…);
(3) - Alterar data;
(4) - Admin (permite ao utilizador criar um novo produto e/ou promoção).

Ao dar *login* o utilizador pode ver os produtos disponíveis e realizar compras, sendo que ao dar "*check out*" terá a lista de compras realizadas.

```
(1)-View available products.
(2)-Buy a product.
(3)-Checkout.
```

5 Conclusão

Com o desenvolvimento desta aplicação foi possível aprofundar e adquirir novos conhecimentos sobre questões fundamentais da programação orientada a objetos. Este projeto foi elaborado de forma que os utilizadores não tenham dificuldades na realização das funcionalidades.