2021/2022

Relatório projeto POAO

Gestor de compras online

Rui Santos - 2020225542
Ricardo Guegan - 2020211358

Introdução

O objetivo do projeto “Gestor de compras online” é simular uma loja online. O funcionamento da loja é o seguinte, o utilizador passa por uma fase de login, na qual tem de inserir o seu mail para entrar, e dependendo se esse email está ou não registado na base de dados, é aberta a sua conta ou então é criada uma nova. Uma vez conectado, o cliente prosseguirá para o menu principal e terá quatro opções: visualizar o catálogo, adicionar produto ao carrinho, finalizar a compra, ver as compras já efetuadas e cancelar a compra. Uma vez efetuada a compra, é apresentada a fatura e as compras serão enviadas para a morada do cliente.

Na fase de login temos também a opção de se conectar sendo administrador, e este poderá efetuar operações como adicionar novos produtos à loja e alterar os stocks de produtos já existentes.

Classes e métodos

Classe GestorDeVendas:

Esta classe terá a função de coordenar todas as classes do programa e de permitir ao utilizador de realizar a suas compras.

Esta classe tem informação sobre todos os clientes já registados e todos os produtos disponíveis para venda e respetiva quantidade em stock.

Métodos:

login() - é chamada apenas uma vez por execução e é a primeira interação com o utilizador. É pedido o email ao cliente, se este já existir o login é realizado com sucesso, caso contrário é criado um novo cliente com o uso do método criaCliente() e no final retorna um cliente.

criaCliente() – pede os dados necessários para a criação de um novo cliente e retorna-o.

menu() – permite escolher entre 4 opções, que são listar os produtos disponíveis, adicionar produto, finalizar compra, mostrar compras realizadas e cancelar compra.

listaProd() – percorre a arraylist de produtos e chama o toString de cada um deles.

compraProd() – pede ao cliente o nome do produto que quer comprar e a respetiva quantidade e verifica se o produto está disponível com recurso ao método procuraProd() e finalmente retorna um objeto do tipo PreparaProduto.

finalizaComp() – calcula o preço final da compra com recurso ao método calculaPreco() que devolve o preço dos produtos que o cliente quer comprar já com as promoções aplicadas e de seguida é determinado o preço de transporte que varia de clientes frequentes para não frequentes e com recurso ao método transporteMobilia() caso o cliente compre algum tipo de mobília.

listaCompras() – mostra todas as compras que o cliente já efetuou na loja.

menuAdmin() – permite escolher entre 2 opções, adicionar novo produto e aumentar o stock.

addProduto() – o método addProduto() pede um a um, todos os elementos necessários para criar um novo produto, com recurso ao método criaPromo() para criar promoções caso seja desejado.

aumentaStock() – pede o nome do produto e a quantidade de stock a aumentar e caso o produto exista aumenta.

pedeInteiro() – pede de forma segura um inteiro ao utilizador. Caso este insira um número inválido é lhe pedido um novo número.

pedeFloat() - pede de forma segura um float ao utilizador. Caso este insira um número inválido é lhe pedido um novo número.

Classe Cliente:

Representa cada cliente da loja. Estes são definidos por 6 atributos, o seu nome, contacto, morada, email, data de nascimento e número de cliente. Na ficha de cada cliente é guardada uma lista com todas as compras já efetuadas na loja.

Classe Compra:

Caracteriza cada compra efetuada na loja. Esta contém o preço final da compra(já com o preço de transporte incluído) a data da compra e todos os produtos e respetivas quantidades comprados, representados na arraylist prod.

Métodos:

A classe compra é capaz de representar tudo aquilo que ela contém, os produtos e o preço final, num método chamado printFatura() ao finalizar a compra, ou num método chamado printListaCompras().

Superclasse Produto:

Define aquilo que é comum a todos os produtos, ou seja, o seu nome, o seu preço, o seu identificador e, caso exista, a sua promoção. O identificador permitirá obter informação sobre o tipo de produto e o tipo de promoção associada(o algarismo mais significativo representa o tipo de promoção(0-não tem promoção, 1-promopagamenos, 2-leve4pague3) e o menos significativo o tipo de produto(1-alimentar, 2-limpeza, 3-mobilia)).

Subclasse Alimentar:

Define tudo aquilo que é comum num produto alimentar, ou seja, o número de calorias e a percentagem de gordura.

Subclasse Limpeza:

Define tudo aquilo que é comum num produto de limpeza, ou seja, o grau de toxicidade.

Subclasse Mobilia:

Define tudo aquilo que é comum num produto de mobília, ou seja, o peso, e as suas dimensões. Este peso será útil para calcular o preço de transporte.

Superclasse *Promocao*:

É uma classe abstrata que define aquilo que é comum a todo o tipo de promoções, a sua data de inicio e a sua data de fim.

Métodos:

Esta classe possui o método procuraPromo(), que recebe a data em que está a ser efetuada a compra e verifica se um produto está ou não em período de promoção. Possui também um método abstract calculaPromo(), que devolverá o preço do produto quando aplicada a promoção que lhe corresponde.

Subclasse Promo4para3:

Esta classe representa um tipo de promoção em que por cada 4 produtos comprados, apenas são pagos 3.

Subclasse PromoPagaMenos:

Esta classe representa um tipo de promoção em que por cada produto comprado, o preço decrescerá em 5% sendo que o primeiro é pago na totalidade até a um máximo de 50% de decréscimo.

Classe StockProduto:

Representa cada produto existente na loja e a sua respetiva quantidade.

Métodos:

Esta classe possui o método toString() que é usado para imprimir o nome do produto em questão, o seu preço, a quantidade em stock e o tipo de promoção, caso exista. Este método será usado para imprimir o catálogo.

Classe PreparaProduto:

Representa cada produto existente no carrinho do cliente e a sua respetiva quantidade.

Métodos:

Esta classe possui o método calculaFatura() que retorna uma String com cada produto comprado e a respetiva quantidade.

Classe Data:

Representa uma data através dos atributos dia, mês e ano.

Métodos:

Esta classe possui o método verifica() que verifica se a data inserida pelo utilizador é válida, o método comparaDatas() verifica se uma data é anterior ou posterior a outra e o método inicializa() que pede os atributos da data ao utilizador.

Classe Ficheiros:

Esta classe interage com os ficheiros de texto e objetos de forma a inicializar todas as variáveis necessárias para o decorrer do programa.

Métodos:

Esta classe possui métodos para trabalhar com ficheiros de texto:leFichClientes() que lê um ficheiro de texto com as informações dos clientes e guarda esta informação numa arraylist, o método leFichCompras() que lê um ficheiro de texto com as compras que cada cliente já realizou e conecta-as com o cliente que as realizou, o método leFichProd() que lê um ficheiro de texto com as informações de cada produto e a respetiva quantidade em stock e que servirá para inicializar uma arraylist StockProduto, um método que lê os um ficheiro de texto com as promoções de cada produto e vai associá-las aos respetivos produtos.

Possui também, métodos para trabalhar com ficheiros de objetos que serão inexistentes na primeira execução do programa. Este métodos são: gravaInfo() que é chamado no final de cada execução do programa para escrever os arraylists clientes e produtos da classe GestorDeVendas em dois ficheiros de objetos separados, e os métodos lerStock() e lerClientes() para ler estes mesmos ficheiros e inicializar os arraylists no inicio da execução do programa.

Organização dos ficheiros de texto

Para a boa execução do programa foi necessário organizar os ficheiros de modo a facilitar a sua alteração, leitura e a correlação com outros ficheiros. Quando os ficheiros vão ser lidos começamos por ler os ficheiros clientes.txt e produtos.txt que são independentes um do outro, no qual cada atributo está separado pelo sinal “+”, tal como nos outros ficheiros. Uma vez lidos estes ficheiros e as arraylists clientes e produtos da classe GestorDeVendas inicializadas vamos ler o ficheiro compras.txt. Este ficheiro já depende dos dois anteriores uma vez que precisa do número de cliente para associar a compra ao cliente e o identificador do produto para associar os produtos a cada lista de compras. Finalmente lemos o ficheiro promocoes.txt que está dependente dos produtos, uma vez que precisa destes para poder inicializar a promoção.