UNIVERSIDADE DO MINHO
LICENCIATURA EM ENGENHARIA INFORMÁTICA

PROGRAMAÇÃO ORIENTADA AOS OBJETOS

Trabalho Prático

GRUPO 71

Sistema de marketplace Vintage

Luís Castro
Rodrigues Caetano
a100893

Mariana Filipa
Morais Gonçalves
a100662

Maya Gomes
a100822

14 de maio de 2023

Conteúdos

1 Introdução

Neste contexto, o nosso projeto revela-se organizado tendo em conta o MVC (Modelo-Vista-Controlador) e respeitando os princípios básicos de modularidade e encapsulamento.

Naturalmente, a nossa arquitetura de classes vai ao encontro do requisitado: a construção de um programa de marketplace Vintage que permita a compra e venda de artigos de vários tipos.

Deste modo, o intuito principal deste trabalho é alcançar os seguintes objetos:

- Fortalecer os conceitos de Modularidade e Encapsulamento;
- Aprofundar a compreensão sobre conceitos de abstração de dados, hierarquia de estrutura e a utilização de API na linguagem escolhida;
- Desenvolver um projeto com possibilidade de "ajustes"(facilidade de inclusão de novas funcionalidades).

2 Descrição da arquitetura de classes utilizada (MVC)

O modelo MVC permite uma maior organização, eficiência, segurança e facilidade quando necessário realizar alterações ao código.

Ao seguir a metodologia MVC (*Model-View-Controller*), pudemos dividir o nosso projeto em três grupos principais que compartilham características semelhantes. Mantendo sempre estas três bases do trabalho em destaque, abaixo descrevemos de forma geral cada uma delas:

- *Model*: Responsável pelo armazenamento de dados e gerência do programa e todos os processos relacionados a ele, como a manipulação de artigos e encomendas;
- *View*: Processos que permitem a comunicação com o utilizador;
- *Controller*: Comunicação entre o Modelo e a Vista.

Em termos de ordem de trabalho, começamos por construir o código do Modelo, onde desenvolvemos as diversas classes sobre os Artigos, as Transportadores e os Utilizadores. Depois, avançamos para o *Controller* e a *View*.

2.1 Classes que integram a *Model*

Para além dos respetivos atributos, as classes da Model possuem alguns métodos específicos mas também métodos semelhantes como os getters e setters, o toString e o clone.

2.1.1 Artigo

É uma classe abstrata que serve de molde e que contém todas as características comuns ao diversos artigos, independente do seu tipo (t-shirt, sapatilha ou mala).

Desta forma um artigo contém diversos atributos:

- private String nome: nome próprio;
- private String usado: "S"caso o artigo seja usado, "N"caso não seja;
- private String descricao: descrição sobre o artigo;
- private String marca: marca do artigo;
- private String codigo_alfanumerico: código alfanumérico do artigo, funciona como um ID e é gerado aleatoriamente;
- private float preco_base: preço base atribuído ao artigo;
- private float correcao_preco: correção de preço atribuída ao artigo (desconto ou acréscimo);
- private AvaliacaoEstado avaliacao_estado: avaliação de estado do artigo com base na classe enum AvaliacaoEstado (PESSIMO, MAU, MEDIO, BOM, EXCELENTE);

- private int num_donos: números de donos que já teve o artigo;
- private String vendido: "S"caso o artigo seja vendido, "N"caso não seja;
- private String ID_T: nome da transportadora associada ao artigo (funciona como um ID);
- private String email_vendedor: email do vendedor do artigo.

2.1.2 Sapatilha

É uma classe que estende a classe abstrata Artigo, que representa o tipo de artigo sapatilha e que também tem algumas características particulares.

- private int tamanho: indica o tamanho da sapatilha;
- private String indicacao: refere se as sapatilhas possuem atacadores/atilhos;
- private String cor: referente a cor das sapatilhas;
- private int lancamento: ano de lançamento da coleção a que pertencem as sapatilhas.

A classe Sapatilha é tambem superclasse da classe PremiumSapatilha.

2.1.3 T-shirt

É uma classe que estende a classe abstrata Artigo, que representa o tipo de artigo t-shirt.

- private String tamanho: indica o tamanho da t-shirt(S,M,L,XL);
- private Padrao padrao: indica o padrão da t-shirt(liso, riscas, palmeiras).

2.1.4 Mala

É uma classe que estende a classe abstrata Artigo, que representa o tipo de artigo mala.

- private Dimensao dimensao: dimensão da mala com base na classe Dimensao que possui os seguintes atributos:
  - private float largura;
  - private float altura;
  - private float comprimento;
- private String material: material da mala;
- private int ano_colecao: ano de coleção;
- private MalaTipo tipo: tipo da mala com base na classe enum MalaTipo (CARTEIRA, ESTOJO, BOLSA, MOCHILA, VIAGEM);

A classe Mala é tambem superclasse da classe PremiumMala.

2.1.5 Utilizador

É a classe referente aos utilizadores e suas características.

- private String codigo_sistema: código no sistema do utilizador;
- private String email: email do utilizador, funciona como um ID;
- private String nome: nome do utilizador;
- private String morada: morada do utilizador;
- private String num_fiscal: número fiscal do utilizador;
- private String palavra_passe: palavra passe da conta do utilizador;
- private float saldo: saldo do utilizador no programa;
- private List<String> lista_artigos: lista dos artigos que o utilizador vende ou já vendeu;
- private List<String> lista_encomenda: lista das encomendas (compras) do utilizador.

2.1.6 Encomenda

A classe Encomenda é referente às encomendas que os utilizadores realizam. Uma encomenda naturalmente é composta por um ou vários artigos.

- private String ID_E: ID da encomenda;
- private List<String> lista_artigos: lista dos diversos artigos da encomenda;
- private DimensaoEncomenda dimensao: referente a classe enum DimensaoEncomenda (PEQUENO, MEDIO, GRANDE);
- private float preco_artigos: preço dos artigos;
- private float preco_final: preço final da encomenda (artigos + portes + imposto);
- private EstadoEncomenda estado: referente a classe enum EstadoEncomenda (PENDENTE, FINALIZADA, EXPEDIDA);
- private LocalDate dataCriacao: data de criação da encomenda;
- private LocalDate dataFinalizacao: data em que a encomenda é concluída;
- private LocalDate dataExpedicao: data em que a encomenda é expedida;

2.1.7 Transportadora

A classe Transportadora inclui todas os atributos relativos as transportadoras:

- private List<String> lista_artigos: lista de artigos de uma determinada transportadora;
- private float margem_lucro: margem de lucro da transportadora;
- private String ID_T: ID da transportadora;
- factor multiplicativo da margem de lucro com base no tamanho da encomenda:
  - private float valor_pequeno = 25;
  - private float valor_medio = 30;
  - private float valor_grande = 40;
- private float valorFaturado: valor faturado pela transportadora;
- private String criador: email do utilizador que criou a transportadora;

A classe Transportadora é também superclasse da classe PremiumTransportadora.

2.2 Classes que integram a *View*

2.2.1 View

A classe *View* é responsável por imprimir os diversos menus ao utilizador, nomeadamente o menu principal, o menu das métricas ou ainda o menu criado para editar a conta do utilizador. Para além disso, a *View* disponibiliza os comandos disponíveis, contendo assim a interface em modo texto do projeto.

A classe *View* contém uma função imprime que nos permite imprimir uma String. Essa função será utilizada no *Controller* de forma a respeitar o MVC.

2.3 Classes que integram o *Controller*

2.3.1 Classe Controler

A classe Controler é responsável por interpretar os pedidos do utilizador e devolver à Vista o resultado desses diversos pedidos. Assim sendo, o Controler contém:

- private String user_atual: email (ID) do user atual;
- private int diasFrente: número de dias que se deseja avançar;
- private Map<String, Utilizador> user: map com os diversos utilizadores (a chave é o email);
- private Map<String, Encomenda> hash_encomenda: map com as diversas encomendas (a chave é o ID_E);
- private Map<String, Transportadora> hash_transportadora: map com as diversas transportadoras (a chave é o ID_T);
- private Map<String, Artigo> hash_artigo: map com os diversos artigos (a chave é o código alfanumérico);
- private transient Scanner scanner = new Scanner(System.in): scanner do controler;
- private Encomenda carrinho: encomenda atual denominada de carrinho;
- private float imposto = 5: imposto cobrado pela Vintage;
- private float lucroVintage: lucro da Vintage;

2.4 Diagrama de Classes

2.4.1 Diagrama genérico de pacotes

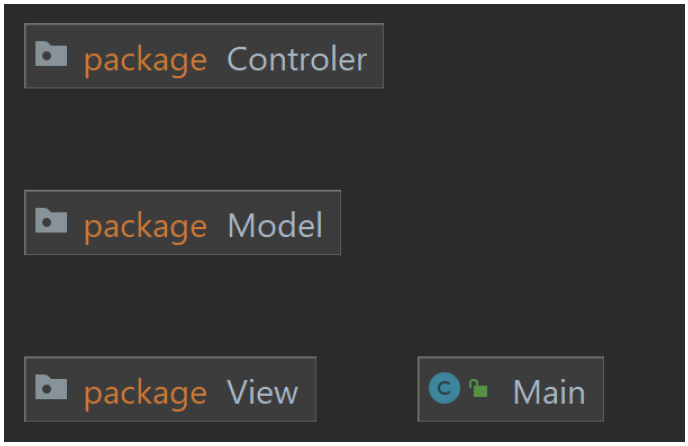

Figura 1: Diagrama genérico de pacotes.

2.4.2 Diagrama genérico do projeto

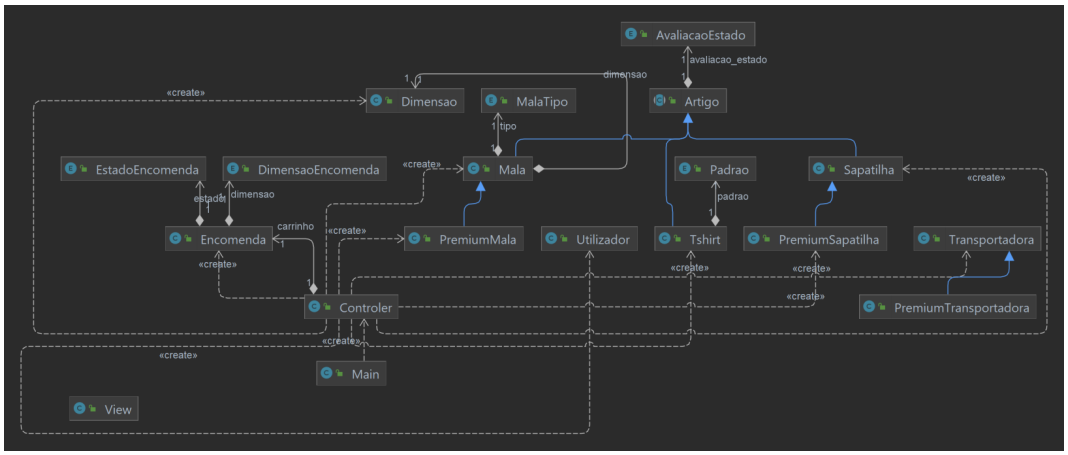

Figura 2: Diagrama genérico do projeto.

3 Descrição da aplicação desenvolvida

O programa começa por perguntar se o utilizador pretende criar conta ou fazer *login*. De seguida, o utilizador escolhe uma das opções criando conta ou fazendo *login*, sendo posteriormente apresentado o menu principal.

Através do menu principal, o utilizador tem várias opções que pode escolher. É perguntado se este pretende:

- visualizar artigos,
- visualizar os seus artigos,
- visualizar as suas vendas,
- editar o seu perfil,
- visualizar as suas encomendas,
- criar um novo artigo,
- criar uma nova transportadora,
- realizar métricas,
- avançar no tempo,
- ver carrinho,
- editar transportadora,
- fazer *logOut*.

Seguidamente, com base na resposta do utilizador é chamado a respetiva função ou um novo menu.

3.1 Ver artigos

Caso o utilizador escolha a opção de visualizar artigos, ser-lhe-ão apresentados todos os artigos a venda com a exceção dos seus, pois um utilizador não pode comprar os seus próprios artigos. É chamada a *View* de forma a imprimir os diversos artigos e seus respetivos atributos de 10 em 10. De seguida é apresentado um novo menu ao utilizador onde é lhe perguntado se deseja avançar (de modo a ver mais artigos), recuar (de forma a ver artigos para trás) ou comprar. Caso o utilizador escolha a opção comprar é lhe pedido que artigo deseja comprar e se o utilizador tiver saldo suficiente é efetuada a compra.

3.2 Meus artigos

Caso o utilizador deseje visualizar os seus artigos é chamada a *View* de forma a imprimir os diversos artigos desse utilizador e os seus respetivos atributos de 10 em 10. Sendo também possível avançar, recuar e sair. Caso o utilizador não possua nenhum artigo, uma mensagem será enviada para informá-lo sobre essa condição.

3.3 Minhas Vendas

Tal como na subsecção anterior, mas agora com os artigos vendidos pelo utilizador, são imprimidos 10 artigos com a possibilidade de avançar, recuar e sair. Os artigos à venda ou vendidos são diferenciados via o atributo *private String vendido* que é "S"caso o artigo seja vendido e "N"caso não seja. Caso o utilizador não possua nenhuma venda, uma mensagem será enviada para informá-lo sobre essa condição.

3.4 Editar perfil

Nesta opção é imprimido um novo menu onde surgem três novas opções:

- Alterar palavra passe,
- Alterar morada,
- Editar saldo
- Sair.

3.5 As minhas encomendas

Nessa opção é permitido visualizar as diversas encomendas tal como acontece com os artigos referido anteriormente. Caso o utilizador não possua nenhuma encomenda, uma mensagem será enviada para informá-lo sobre essa condição.

3.6 Criar um novo Artigo

Nesta secção são feitas diversas perguntas sobre o artigo que se deseja criar de forma a preencher os atributos todos. Uma vez todas as informações recolhidas é criado o novo artigo.

3.7 Criar uma nova transportadora

Tal como na subsecção anterior, mas aqui com as transportadoras, são realizadas as perguntas necessárias à criação de uma nova transportadora.

3.8 Métricas

Ê apresentado o seguinte menu:

- Vendedor que mais faturou,
- Vendedor que mais faturou num intervalo de tempo,
- Transportadora que mais faturou,
- Listar as encomendas emitidas por um vendedor,
- Maior vendedor,
- Maior comprador,
- Lucro da *Vintage* no seu funcionamento,
- Sair.

Desta forma, podem ser realizadas diversas métricas à escolha do utilizador.

3.9 Avançar no tempo

É perguntado ao utilizador de quantos dias ele deseja avançar e é atualizado o atributo *private int diasFrente* do Controler.

3.10 Ver carrinho

Permite visualizar a encomenda que esta a decorrer denominada de carrinho.

3.11 Editar transportadora

Permite ao criador e somente ao criador da transportadora alterar certos dados da mesma.

3.12 LogOut

É terminada a sessão do utilizador.

4 Conclusão

Relativamente a nossa primeira experiência com a programação em JAVA, acreditamos que conseguimos superar o desafio do projeto prático. No entanto, reconhecemos que pode haver maneiras mais eficientes ou melhores de abordar certas situações como por exemplo com o uso mais recorrente de métodos do streams para filtrar e mapear. Neste caso, só foram usados algumas vezes, pois esses novos conceitos foram aprendidos mais tarde na realização do projeto.

No decorrer do trabalho surgiram algumas dificuldades nomeadamente na construção e organização das classes ou ainda na realização das funções mais trabalhosas como a de realizar uma compra.

Apesar disso, neste projeto concretizamos grande parte dos requisitos pedidos no enunciado (todos com a excessaõ do último: "Automatizar a simulação") e é importante mencionar que respeitamos os conceitos de encapsulamento e modelo MVC apresentados nas aulas teóricas.