Universidade do Minho

Licenciatura em Engenharia Informática

Ano letivo 2022/2023

POO – Trabalho prático

Grupo 38

Uma imagem com pessoa, homem, interior, pousar

Descrição gerada automaticamenteUma imagem com pessoa, parede, interior, homem

Descrição gerada automaticamenteUma imagem com pessoa, interior

Descrição gerada automaticamenteAntónio Silva(A100533), José Matos(A100612), Pedro Silva(A100745)

**Índice**

1. **Introdução**
2. **Classes**
   1. Artigo
   2. Sapatilhas
   3. TShirt
   4. Mala
   5. Utils (Artigos)
   6. Utilizador
   7. Encomenda
   8. Utils (Encomendas)
   9. Transportadora
   10. Utils(Transportadora)
   11. SaveLoad
   12. Clock
   13. UI
   14. Info
   15. InfoUtils
   16. Manage
   17. ManageUtils
   18. Stats
   19. StatsUtils
   20. ControlCenter
   21. TimeTravel
   22. LocalDateTimeTypeAdapter
   23. AutoRun
   24. ErrorCode
   25. ErrorHandler
   26. Vintage
   27. Utils (Vintage)
   28. Main
3. **Diagrama de classes**
4. **Aplicação desenvolvida e bibliotecas utilizadas**
5. **Conclusão**

**Capítulo 1**

**Introdução e principais objetivos**

**Capítulo 2**

**Classes**

**2.1.** Artigo

Quando nos foi proposto o projeto, foi recomendado que fosse possível uma futura implementação de outros tipos de artigos e nós para facilitarmos essa implementação criamos a classe abstrata Artigo que teria todas as características comuns a todos os artigos.

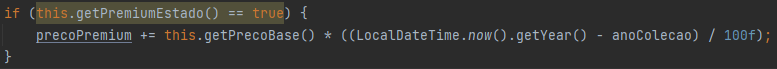
Inicialmente temos três constantes inteiras(MALA, SAPATILHAS e TSHIRT) para identificar os tipos de artigos que temos atualmente. As outras características comuns são:

* private int tipo
* private float estadoUtilizacao
* private int numDonos
* private String descricao
* private String marca
* private int codigo
* private float precoBase
* private int codigoVendedor
* private Transportadora transportadora
* private boolean premiumEstado

Futuramente iremos falar do que são as Transportadoras e os artigos que não são supostos ter premium irão sempre devolver false não interferindo de qualquer forma nos seus cálculos e estatísticas.

Criamos um construtor de artigo de forma a que as subclasses que sejam extensões do artigo poupem trabalho na sua criação, informamos que todas têm de ter uma função de calcularPreco e calcularCorrecao dependo da transportadora, por exemplo. Tem a função de calcularCorrecaoPremium já implementada, porque esta será igual para todos que utilizem premium.

A forma que arranjamos de calcular a evolução do preço de um artigo premium foi:



Dá para compreender pela função que quanto mais antigo for o Artigo, maior será o preço do Artigo premium.

Finalmente, tem os getters e setters de todos os atributos.

**2.2.** Sapatilhas

O primeiro artigo que implementamos foram as sapatilhas. Como mencionamos anteriormente, todos os artigos vão ser uma extensão da classe Artigo, sendo que cada subclasse tem as suas características específicas. No caso das sapatilhas, seria ter atacadores ou atilhos, o tamanho delas, a cor e o ano da coleção.

Estes atributos estão guardados do seguinte modo:

* public static final int ATACADORES = 0;
* public static final int ATILHOS = 1;
* private int tamanho;
* private int atacadores;
* private String cor;
* private int anoColecao;

As contantes ATACADORES e ATILHOS servem para facilitar a compreensão do tipo de atacadores.

Além disso têm o seu próprio construtor, utilizando a funcionalidade super com os atributos da classe Artigo que são usados nas sapatilhas, a sua própria função de calcularPreco que utiliza o precoBase, a calcularCorrecaoPremium, pois sapatilhas podem ser premium ou não e a sua própria calcularCorrecao.

No caso das sapatilhas, a calcularCorrecao,além de se adicionar o preço da Transportadora, caso não sejam novas, ou seja, estado de Utilização diferente de 1, sendo que estes podem ser 0.25,0.5,0.75 ou 1, vão sofrer uma redução de preço quão pior for o seu estado(Pouco Uso 25%, Muito Uso 50% e Estragado 75%) e caso sejam novas e de tamanho maior de 45 recebem ainda mais uma dedução de 25% também.

No nosso caso, o estado nunca terá nada a ver com o número de donos pois achamos que isso é uma correlação pouco baseada, porque dependendo do tipo de dono, o artigo pode não diminuir no seu estado de utilização, por exemplo, algo pode ter 3 donos e continuar praticamente novo e também ter 3 donos e ficar completamente estragado.

Por fim, tem os getters e os setters e na sua toString utiliza funções de utils que serão todas explicadas no seu devido tempo. Todos os tipos de artigo terão sempre no final do ficheiro estas funções, sendo a principal a toString que utiliza as funções dos ficheiros utils.

**2.3.** TShirt

Sendo a TShirt outra extensão da classe Artigo, também só terá de implementar as suas características específicas. No caso da TShirt são o tamanho e o seu padrão.

Para facilitar a identificação de cada usamos as seguintes constantes:

Uma imagem com texto

Descrição gerada automaticamente

private int padrao;

private String tamanho;

Depois, tal como nas sapatilhas, criamos o construtor utilizando novamente o mesmo método super.

O calcularPreco é implementado, mas ao contrário das Sapatilhas, as TShirts nunca serão premium, logo não utilizará a calcularCorrecaoPremium, e, por fim, tal como as Sapatilhas, tem a sua implementação da calcularCorrecao.

O calcularCorrecao, no caso das TShirts, funciona mais focado no seu padrão. Além de adicionar o valor da transportadora, caso o padrão seja Liso não há mais nenhum desconto, caso contrário, se a TShirt não for nova, independente do padrão, exceto o Liso, teria um desconto fixo de 50%.

**2.4.**  Mala

O último tipo de artigo que implementamos foi a Mala, este tipo de artigo caracteriza-se principalmente pelo facto de o seu preço acabar por depender além das suas características específicas também muito pela sua dimensão.

Este artigo tem várias constantes, sendo as primeiras utilizadas para o tipo de material(TECIDO, PELE, LONA ou VELUDO), as segundas para facilitar a encontrar as dimensões no array delas(COMPRIMENTO, LARGURA e ALTURA) e por fim duas constantes para o calculo do preço (CONSTANTE\_CORRECAO e MARGEM\_ERRO).

Além disso as suas características específicas são:

* private float[] dimensao; [COMPRIMENTO, LARGURA, ALTURA]
* private int material;
* private int anoColecao

Tal como todas as outras o construtor continua a funcionar da mesma forma, tal como a implementação da calcularPreco, que nas malas também utiliza calcularCorrecaoPremium, porque uma mala pode ser premium ou não.

O calcularCorrecao da mala funciona pegando nas suas dimensões, calcula-la em volume utilizando a função calcularDimensao, adiciona o preço da Transportadora e depois com esta formula:



A constante de correção foi adicionada, para o caso de as dimensões forem inválidas, por exemplo, dava um preço negativo, o programa possa avisar que essa mala não pode ser adicionada ao programa.

Por fim, tem todos os getters e setters como todos os artigos.

**2.5.** Utils(Artigos)

Quando havia necessidade de utilizar funções que seriam partilhadas por várias classes, fomos criando os Utils. Os utils são pequenas classes com algumas funções que servem para corrigir cálculos, facilitá-los ou ajudar na designação de certos parâmetros.

No caso do Utils dos Artigos, este só tem três funções, sendo as duas primeiras para facilitar a demonstração no toString (arredondarDecimas e arredondarCentesimas), ou seja, apesar de o programa calcular com todas as casas decimais que lhe estão disponíveis, dificultando erros de cálculo, quando demonstrarmos no ecrã, só iremos mostrar as casas decimais principais que desejamos, por exemplo, no preço apresentado, mesmo que este dê 35,1253264, só iremos mostrar 35,13.

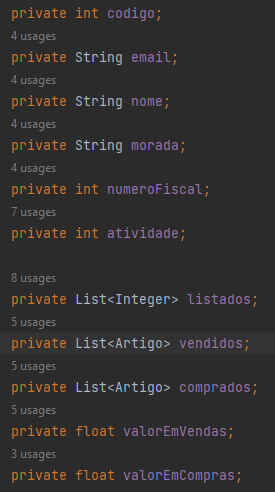


A terceira função é a calcularPercentagem que trata do valor que é preciso acrescentar ao artigo por causa do valor de expedição que a transportadora cobra pelo transporte.

**2.6.** Utilizador

Depois de termos implementados os três tipos de artigo pedidos, iniciamos a criação da classe Utilizador. Os utilizadores serão as pessoas que são ambos compradores e vendedores que podem comprar e por à venda os seus artigos. Pode-se criar utilizadores a qualquer momento e apagar, consequentemente perdendo os seus artigos listados, a não ser que estejam numa encomenda. Quando se apaga um utilizador, simplesmente colocamos a conta INATIVA, para não perder informação das encomendas previamente feitas, mas deixamos de lista-lo nos utilizadores na UI.

Por este motivo, inicialmente temos as duas constantes ATIVA e INATIVA e depois os atributos de um utilizador:

* O código de cada utilizador atribuido pelo sistema, o seu email nome, morada, número fiscal e asua atividade.
* As três listas com os seus artigos listados, os que já foram vendidos e os que ele comprou
* Por fim guardamos o valor que ele ganhou em vendas e quanto gastou em compras.

No construtor, iniciamos o utilizador com três ArrayList para os listados, vendidos e comprados e com os dois valores a zero.

Temos 4 funções que tratam da listagem, da compra e da venda de artigos:

* A função criarListagem recebe um artigo e adiciona-o aos listados do utilizador, caso este for ativo
* A função removerListagem que remove o artigo que é pedido dos listados de um utilizador
* A função venderArtigo que remove um artigo dos listados, acrescenta-o aos vendidos e aumenta o valorEmVendas do valor do artigo, excluindo a parte da transportadora
* A função comprarArtigo recebe a lista de utilizadores e o artigo que vai ser comprado, adiciona o lucro à transportadora usada, descobre qual é o utilizador vendedor, vende-lhe o artigo e depois adiciona este artigo aos comprados do utilizador comprador e aumenta o seu valorEmCompras

No final tem todos os getters e setters e a toString do utilizador também utiliza Utils, sendo que algumas já foram explicadas e outras ainda não.

**2.7.** Encomenda

Depois de criados os artigos e os utilizadores, finalmente podemos criar a forma como os artigos vão ser vendidos aos utilizadores, em forma de encomendas.

Todas as encomendas têm três estados: pendentes, expedidas ou finalizadas, algo que definidas com constantes.

Depois de criadas, todas as encomendas têm dois dias para serem reembolsadas, ou será impossível retirar o artigo da encomenda, mesmo que ainda seja pendente.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente As encomendas terão vários atributos:

* Um código que as identifica, um código que identifica o utilizador que a faz e uma lista com o código dos artigos que vão ser comprados
* Vai ter três tipos de dimensão: Grande com 10 artigos, Média com 5 artigos e Pequena com 1 artigo, podendo sempre ter menos
* Além disso, tem o estado, o seu preço, quando foi criada e até quando é a sua entrega, que depende da sua dimensão

No seu construtor, quando a encomenda é criada, o código dela é atribuído pelo sistema, inicia o ArrayList dos artigos, o seu estado será PENDENTE, o seu preço é 0 e a dataCriacao será a data dada pelo sistema.

As funções na classe Encomenda já utilizam o ErrorCode que explicaremos mais à frente.

Na função adicionarArtigo, recebemos a lista de artigos, de encomendas e o código do artigo que queremos adicionar. Testando se a encomenda já foi expedida, se já não está cheia, se o código daquele artigo existe, se não é um artigo do próprio comprador, se este artigo já não está noutra encomenda e só por fim é que sabemos que não nenhum problema e o adicionamos à encomenda. Recalculamos o valor da encomenda e dizemos que não existe qualquer erro.

A segunda função removeArtigo, recebe a lista de artigos e o código do artigo que pedem para ser removido. Se a encomenda já não estiver pendente, já não podemos remover o artigo, se tiver vazia não há nada para remover, se o artigo não existir também não é possível e no fim verifica se o artigo existente está na encomenda. Caso esteja, é removido da encomenda e recalculasse o preço.

Por fim, apesar dos getters e setters serem como o habitual, na toString das encomendas tivemos de meter um if no caso de a encomenda ainda não ter sido expedida, pois nesse caso ainda não temos data de entrega, logo só aparece “Por expedir”, enquanto que no outro caso já aparece a data em concreto.

**2.7.** Utils(Encomendas)

No utils das encomendas, simplesmente tem a função que determina o tempo de expedição das encomendas. No caso que ela seja grande, ou seja, tamanho 10, serão 14 dias, tamanho médio(5), 7 dias e uma encomenda pequena demora somente 3 dias a ser entregue.

**2.8.** Transportadora

Por fim, apesar de já termos falado delas, só implementamos as transportadoras no fim e depois é que as adicionamos aos artigos.

Todos os artigos têm a sua transportadora específica, tendo esta de ser especificada no momento que o artigo é criado, uma encomenda pode ter artigos de transportadoras diferentes e, caso uma transportadora seja apagada, todos os artigos que a têm como a sua transportadora, não são apagados, pois a transportadora encontra-se obrigada a terminar o seu trabalho, neste caso, até estes artigos serem vendidos ela continua a “existir”.

As transportadoras também poderão ser premium e artigos premium terão de ser entregados obrigatoriamente por estas transportadoras. Para as diferenciar, estas transportadoras também terão uma margem extra além da margem de lucro habitual.

Os atributos são então:

* Uma imagem com texto, captura de ecrã, Tipo de letra

  Descrição gerada automaticamenteTodas as transportadoras têm o seu nome e a sua margem de lucro, mas só as premium têm a margem extra
* O valor de expedição é depois calculado com umas constantes que estão nas utils e as margens
* Sempre que um artigo é vendido guarda-se o lucro que a transportadora teve
* Por fim, guarda-se se uma transportadora é premium ou não

No construtor, além de se guardar os parâmetros todos calcula-se o valor de expedição.

A função calcularValorExpedicao utiliza as tais constantes e as margens sendo a fórmula deste modo:

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente Tem também a função calcularEntrega que atualiza o lucro da transportadora.

A última função que falta falar das transportadoras é a atualizarValores que autoriza uma transportadora de alterar o valor das suas margens se achar necessário. Não dá para ver na função em concreto, mas nós só autorizamos as transportadoras premium a alterar os seus valores, pois achamos que não faz sentido qualquer transportadora mudar o seu valor de um momento para o outro.

No final, igualmente os getters e setters e a toString utilizando novamente outras funções das Utils, sendo primeiramente introduzidos à que trata do premium.

**2.8.** Utils(Transportadora)

No caso das utils da transportadora, esta simplesmente guarda as duas constantes que são utilizadas no cálculo do valor de expedição, o VALORBASE e o IMPOSTO.

**2.9.** SaveLoad

Foi a partir daqui que começamos a pensar na formulação da nossa UI, mas inicialmente tivemos de pensar numa forma de guardarmos o estado do programa e carregarmos esse estado caso fosse necessário futuramente, duma certa forma tal e qual como deveria funcionar um website real.

Para conseguirmos fazer o Save e o Load do programa optamos por utilizar a biblioteca Gson e guarda num ficheiro json o estado do programa e consegue posteriormente lê-lo de forma a obter toda a informação que estava guardada.