

Projeto de Programação Orientada aos Objetos

Grupo 14

André Carvalho da Cunha Martins A89586

António Jorge Nande Rodrigues A89585

Rui Emanuel Gomes Vieira A89564



A89586



A89585



A89564

Introdução

Como trabalho prático da UC Programação Orientada aos Objetos, foi-nos proposto o desenvolvimento de um programa para realizar encomendas ao domicílio. Todo o programa previa a existência de 4 tipos de utilizadores diferentes, os utilizadores domésticos, que iriam efetuar as encomendas, empresas transportadoras e voluntários, que iriam fazer as entregas dessas mesmas encomendas, que iria, conter produtos das lojas que fizessem parte do programa, sendo elas o quarto tipo diferente de utilizador. Entre várias funcionalidades do programa, é possível ao utilizador doméstico realizar uma encomenda, médica ou não, sendo esta depois preparada por uma loja e, posteriormente, entregue por um voluntário, sem custos adicionais, ou por uma empresa de transportes, tendo esta custos que podem, ou não, serem aceites pelo utilizador, aquando da realização da encomenda.

Descrição das classes utilizadas

1 Classe Abstrata UtilizadorSistema

Como foi dito anteriormente, o programa prevê a existência de 4 tipos diferentes de utilizadores. Contudo, esses 4 tipos diferentes de utilizadores têm parâmetros em comum, tais como, a existência de um código que os identifica, de um nome, da sua latitude e longitude, e de um email e password, necessários para fazer login no programa. Estes parâmetros representam as variáveis de instância da superclasse Utilizador do Sistema, que será depois especificada por cada um dos utilizadores. Esta classe é abstrata por conter um método abstrato, o método clone(), que será depois especificado por cada uma das subclasses.

```
public abstract class UtilizadorSistema implements Serializable {
    private String email;
    private String password;
    private String typeUser;
    private String codigo;
    private String nome;
    private double latitude;
    private double longitude;
}
```

SuperClasse UtilizadorSistema

2 Classe Utilizador

A classe Utilizador herda a superclasse UtilizadorSistema, tendo, depois, mais uma variável de instância, uma List de Encomendas, que representa todas as encomendas realizadas por um utilizador. Para além dos getters e setters, equals, clone e toString, esta classe possui métodos que devolvem, em forma de String, as encomendas já recebidas no domicílio e as encomendas que ainda se encontram por entregar. Possui, também, métodos que tratam da atualização da v.i. encomendas-realizadas, tais como, métodos que adicionam uma nova encomenda, e 3 métodos que atualizam os 3 estados de uma dada encomenda, podendo esta estar preparada, levantada ou entregue. Possui ainda mais um método que devolve uma encomenda com um dado código.

```
public class Utilizador extends UtilizadorSistema implements Serializable {
    private List<Encomenda> encomendas_realizadas;
```

Classe Utilizador

3 Voluntário

A classe utilizador também herda a superclasse UtilizadorSistema e possui mais 9 variáveis de instância. As variáveis disponível e transporteMedico indicam, respetivamente, se o voluntário está disponível para o transporte de encomendas e se este aceita o transporte de encomendas médicas. As variáveis velocidade e minutosDeEspera serão importantes para depois fazer uma previsão do tempo que demorou uma encomenda a ser entregue a um utilizador, sendo que a variável minutosDeEspera é atualizada com o tempo de espera que o voluntário teve numa determinada loja, em função do tempo médio de atendimento e do número de pessoas que se encontravam na fila. Possui também uma variável que guarda a hora de registo de um voluntário. A variável raio-acao servirá para saber a área na qual o voluntário se pode deslocar a uma loja e pode realizar uma entrega. Depois tem as variáveis classificacao, que representa o rating médio desse voluntário, e avaliacoes, que indica quantas avaliacoes já foram feitas a esse voluntário, número esse que será usado sempre que se realiza uma nova avaliação, por parte de um utilizador, para poder depois a média das classificações ser atualizada. Tal como um utilizador, também possui uma List de Encomendas com todas as encomendas que esse voluntário efetuou ou irá efetuar. Tal como um utilizador, também tem métodos para atualizar os diferentes estados de uma encomenda e métodos, que retornam uma String, as encomendas, dependendo do estado em que se encontram. Possui o método updateRate, que serve para atualizar a sua classificação. Por fim, e como forma de representar alguma aleatoriedade do estado do tempo, existe o método calculaAtrasos, que devolve o número de minutos perdidos, devido às condições atmosféricas.

```
public class Voluntario extends UtilizadorSistema implements Serializable
    private boolean disponivel;
    private boolean transporteMedico;
    private int velocidade;
    private int minutosDeEspera;
    private LocalDate horaDeRegisto;
    private double raio_acao;
    private double classificacao;
    private int avaliacoes;
    private List<Encomenda> historico;
```

Classe Voluntario

4 Loja

Esta classe, para além das variáveis herdadas, possui mais 3 variáveis de instância, sendo elas a variável tempo-espera, que representa o tempo médio que cada cliente demora a ser atendido, a variável nrPessoasEmFila, para saber quantas pessoas se encontram na fila da loja, importante para depois se calcular todo o tempo de espera numa loja e um ArrayList com todas as encomendas recebidas pela loja. Também possui métodos de atualização do estado das encomendas, e métodos, que devolvem na forma de String, encomendas, dependendo do estado pretendido.

```
public class Loja extends UtilizadorSistema implements Serializable {  
    private double tempo_espera;  
    private int nrPessoasEmFila;  
    private ArrayList<Encomenda> encomendas_recebidas;
```

Classe Loja

5 EmpresaTransportes

Esta classe especifica mais 11 variáveis de instância. Tal como foi explicado na classe voluntário, as variáveis minutosDeEspera, velocidade, raioDeAcao, classificacao, avaliacao e registos funcionam de forma análoga, tendo também métodos semelhantes aos da classe Voluntario. Para além dessas, possui a variável nif, local, meramente informativas e a variável que distingue a classe EmpresaTransportes da classe Voluntário, que é o custo-km, a partir do qual se irá calcular o preço do transporte de uma encomenda, ao contrário do transporte efetuado por um voluntário, que é gratuito. Para calcular o preço de um encomenda, consideramos que todas as empresas transportadores aplicam a taxa de 0,2 por cada unidade de peso, variando apenas o custo por km de cada empresa. Para além dos métodos de atualização de classificação, de estado das encomendas e de retorno de String com as encomendas desejadas, esta classe possui métodos tais como, getFaturacao, que devolve o total faturado por essa empresa entre 2 datas, o método getKms, que devolve o total de kms percorridos pela empresa para entregar as encomendas (Aqui excluimos a entrega utilizando uma rota, simplesmente somamos as distancias percorridas para entregar cada encomenda, partindo sempre da localização da empresa transportadora). O método getRota devolve um ArrayList com todas as encomendas já levantadas, para que depois sejam entregue aos seus compradores. O método distanciaEntreLojas calcula a distância total percorrida para se ir a todas as lojas levantar as encomendas.

Nota: Para efetuar a entrega de N encomendas utilizando uma rota, definimos que primeiro se iria a todas as lojas levantar as encomendas todas, e só,

depois, é que seria feita a sua entrega, daí ser necessário calcular a distância entre lojas.

```
public class EmpresaTransportes extends UtilizadorSistema implements Serializable {  
    private int nif;  
    private double custo_km;  
    private int velocidade;  
    private int minutosDeEspera;  
    private String local;  
    private double raioDeAcao;  
    private double classificac;  
    private int avaliacoes;  
    private ArrayList<Encomenda> registos;  
    private boolean transporteMedico;  
    private boolean disponivel;  
}
```

Classe EmpresaTransportadora

6 Encomenda

A classe Encomenda possui 12 variáveis de instância, sendo elas o código da encomenda, o código do seu comprador, o código da loja que vendeu, o peso total da encomenda (Para ajudar a calcular o preço do transporte com uma empresa), o nome do comprador, o nome da loja, a hora a que foi feita a encomenda, 3 variáveis que representam os diferentes estados da encomenda, preparada, levantada e entregue, um Map que para cada código de Linha de Encomenda, faz corresponder a respetiva Linha de Encomenda, compondo assim todos os produtos que fazem parte da encomenda e, por fim, uma variável que identifica a encomenda como sendo, ou não, uma encomenda médica.

7 LinhaEncomenda

Esta classe possui apenas 4 variáveis de instância, sendo elas o código que identifica o produto, o nome do produto, a quantidade desse produto e o seu valor unitário.

As classes BD

8 A classe BDGeral

As classes descritas anteriormente apenas representam uma entidade singular. Para que fosse possível o programa funcional, seriam necessárias classes que agregassem todas as entidades que compõem o programa. Esse é o papel das

classes BD. A classe BDGeral é a classe agregadora de BDProdutos, BDVoluntarios, BDLojas, BDUtilizadores, BDTransportes e EncomendasAceites (Consideramos uma encomenda como aceite sempre que um utilizador conseguiu realizar o registo dessa encomenda até ao fim, isto é, quando a essa encomenda ficou também atribuído uma empresa transportadora ou um voluntário). Nesta classe podemos dividir os métodos em 4 tipos. Temos os métodos de atualização, que servem para ir atualizando as diferentes BD à medida que um elemento é adicionado ou modificado. Todo o programa se baseia na alteração dos estados dos diferentes elementos (Adicionar encomendas, atualizar encomendas, atualizar classificações, atualizar filas, etc...) portanto, é necessária uma constante atualização dos Map que compõe as diferentes BD. Como o email de um elemento se mantém imutável, a atualização de um elemento é realizada de uma forma simples, bastando fazer um get desse elemento, fazer as alterações pretendidas e depois voltar a fazer um put desse elemento. Como o email é o mesmo, irá ocorrer a substituição do elemento antigo pelo elemento novo e atualizado.

Nota: As classes BD são compostas por Maps, cujas keys são o email. Utilizamos esta estratégia para facilitar o desenvolvimento do mecanismo de login, visto que as credenciais de login são o email e a respetiva password. Cada classe BD possui também um set com os diferentes códigos dos elementos que os constituem para, aquando do registo de um novo elemento, seja possível gerar um código que ainda não exista.

Existem depois os métodos que usamos para efetuar o login, cada um desses métodos dá throw de uma exception, específica para cada utilizador, caso as credenciais estejam erradas.

Existem depois os métodos top10Encomendas e top10Kms que devolvem um Set<Pair>, ordenado, consoante o método, e que indicam as entidades que realizaram mais encomendas e as empresas que mais kms percorreram, respetivamente.

Nota: A classe Pair foi uma classe auxiliar que definimos que tem apenas 2 variáveis de instância, uma String que guarda o nome e um int que guarda ou o nr de Encomendas ou o nr de Kms percorridos. O Set é ordenado com o comparador ComparaQuantidadePair.

Finalmente, existem os métodos gravarFicheiro e lerFicheiros, usados para gravar e carregar o estado da aplicação, guardado num ficheiro binário.

9 As classes BDVoluntarios, BDTransportes, BDUtilizadores e BDLojas

Tal como dito em cima, estas classes têm métodos definidos que tratam da atualização ou adição de novos elementos. Para além desses métodos, as classes BDTransportes e BDVoluntarios também têm métodos definidos que devolve uma List com os elementos disponíveis para realizar uma encomenda. Mas, num contexto mais geral, os métodos destas classes servem o propósito da atualização dos Map à medida que são realizadas encomendas, ou é feita uma nova avaliação, ou é levantada uma encomenda, ou é entregue uma encomenda, etc...

Podemos dizer que são estas classes que formam o cerne do programa, visto que é a sua constante atualização que permite que o programa flua como é suposto.

10 A classe BDProdutos

Esta classe agrega toda a informação sobre os produtos que é possível encomendar, tendo métodos que fazem a distinção entre o que é um produto médico e o que é um produto dito normal.

Como estratégia, definimos que 4 produtos seriam catalogados como produtos médicos (Álcool, Desinfetante, e os 2 tipos de sacos do lixo) e que os restantes produtos seriam considerados como produtos normais. Quando uma encomenda possui algum dos produtos médicos, ela é catalogada como sendo encomenda médica.

Esta classe tem métodos definidos que devolvem sobre a forma de String 2 catálogos, o de produtos normais e o de produtos médicos, sendo possível fazer uma encomenda tendo os 2 tipos de produtos.

11 A classe Parse

Esta classe foi utilizada para fazer a leitura do ficheiro logs.txt, que depois irá popular as outras classes do programa, tornando este funcional. Nesta classe estão definidos métodos que fazem a leitura do ficheiro txt e, posteriormente, o split e criação de todos os objetos necessários.

12 Descrição da aplicação desenvolvida (ilustração das funcionalidades)

Ao iniciar o programa, é pedido se pretende ler o ficheiro logs.txt ou carregar um ficheiro. Após esse passo, aparecem 4 opções diferentes. Registo, onde é possível registar uma nova entidade, Login, para efetuar o login no programa, gravar estado, para gravar o estado do programa num novo ficheiro binário e carregar estado, para carregar um novo ficheiro.

Na opção login é necessário escolher o tipo de entidade que pretende fazer login.

Após o login efetuado, é apresentada a cada entidade um leque de opções diferentes

NOTA PARA TESTAR O PROGRAMA: De maneira a ser relativamente simples efetuar o login com cada uma das entidades presentes no ficheiro logs.txt, atribuiu-se a mesma password a todas as entidades ("12345") e o email de cada entidade é o seu código + @gmail.com (Exemplo: Voluntário com o código v33. Email: v33@gmail.com Password: 12345) Assim também se garante a existência de uma email diferente para cada entidade.

13 Funcionalidades de um Utilizador

Após o login, são apresentadas a um utilizador doméstico 11 funcionalidades.

As funcionalidades Nova Encomenda e Nova Encomenda Médica servem para realizar uma nova encomenda, sendo que a opção Nova Encomenda não permite a existência de um produto médico, enquanto que a Nova Encomenda Médica tem obrigatoriamente de ter um produto médico, podendo ter mais produtos normais. Após o preenchimento dos campos da encomenda, esta será automaticamente atribuída a um voluntário, caso haja algum disponível, ou, caso isso não seja possível, serão apresentados ao utilizador as diferentes empresas transportadoras disponíveis. Aqui, o utilizador decide se aceita pagar o custo do transporte ou se cancela a encomenda.

As funcionalidades histórico apresentam ao utilizador as suas encomendas que se encontram por entregar e aquelas que já foram entregues. Estas são apresentadas devido à presença das variáveis de instância na Encomenda, que permitem a distinção das várias encomendas.

A funcionalidade ver estado de encomenda permite, após ser introduzido o código da encomenda que se pretende analisar, ver o estado em que se encontra uma encomenda, ou seja, se esta é médica ou não, se já está preparada pela loja, se já está a caminho e se já foi entregue.

As funcionalidades de avaliação consistem na escolha do código da entidade que se pretende avaliar e na introdução de uma avaliação (1 a 10). A classificação dessa entidade será depois atualizada com a nova classificação média.

As funcionalidades 8 e 9 apresentam o número de encomendas realizadas por um voluntário e empresa transportadora, respetivamente, entre 2 datas e o número de lojas diferentes envolvidas.

As funcionalidades 10 e 11 apresentam uma lista com os top10 de mais encomendas e de mais kms percorridos por empresa transportadora, respetivamente.

```
-----USER-----
1. Nova encomenda
2. Nova encomenda Médica
3. Histórico de encomendas recebidas
4. Histórico de encomendas por entregar
5. Ver estado de encomenda
6. Avaliar um voluntário
7. Avaliar uma transportadora
8. Informação sobre entregas de voluntário
9. Informação sobre entregas de Empresa Transportadora
10. TOP10 mais encomendas efetuadas
11. TOP10 mais kms percorridos por transportadoras
0. Retroceder

Insira uma opção --> 
```

Funcionalidades para um Utilizador

14 Funcionalidades de um Voluntário

Após o login, são apresentadas 7 funcionalidades a um voluntário.

As primeiras 4 funcionalidades servem para alterar o estado de um voluntário, ou seja, se este se encontra disponível ou não para transportar encomendas, e se este se encontra, ou não, disponível para transportar encomendas médicas. Estes estados podem ser alterados a qualquer altura, sendo depois esse voluntário atualizado na BDGeral e na BDVoluntários.

A funcionalidade 5 imprime todas as encomendas que esse voluntário recebeu, independentemente do estado das mesmas.

A funcionalidade 6 serve para um voluntário sinalizar que levantou uma encomenda numa loja, estando esta, agora, pronta para ser entregue. Um voluntário só pode entregar uma encomenda depois de esta estar levantada, sendo este passo obrigatório para se efetuar a entrega de uma encomenda. Contudo, o voluntário só pode levantar uma encomenda que já se encontre preparada, sendo este passo do domínio da loja em questão.

Por fim, a funcionalidade 7 serve para o voluntário sinalizar a entrega de uma encomenda a um utilizador. A encomenda só pode ser dada como entregue depois de ter sido preparada e levantada. Esta funcionalidade apresenta também uma previsão do tempo que a encomenda demorou a ser entregue, desde do momento em que foi emitida, utilizando a velocidade do voluntário, o tempo de espera na loja, a distância e a aleatoriedade do clima como fatores para efetuar os cálculos.

Aparece sempre uma atualização constante de quantas encomendas esse voluntário tem por levantar, e quantas encomendas já estão prontas a serem entregues.

A funcionalidade 8 serve para fazer um `clearScreen()` e voltar a imprimir o menu.

```
-----VOLUNTÁRIO-----
1. Declarar-se disponível para ir buscar encomenda
2. Declarar-se indisponível para ir buscar encomenda
3. Declarar-se disponível para transportar encomendas médicas
4. Declarar-se indisponível para transportar encomendas médicas
5. Histórico total de encomendas.
6. Sinalizar levantamento de encomenda em loja
7. Sinalizar entrega de encomenda a utilizador
8. Imprimir menu.
0. Retroceder

-> Tem 1 encomendas por levantar
-> Tem 0 encomendas prontas a entregar

Insira uma opção --> 
```

Funcionalidades para um Voluntário

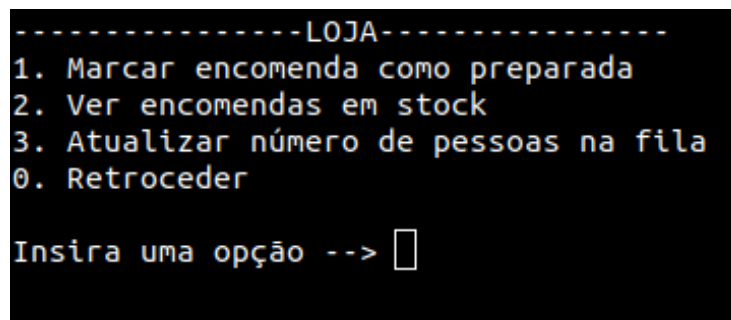
15 Funcionalidades de uma Loja

Após o login efetuado, são apresentadas 3 funcionalidades a uma loja.

A primeira funcionalidade serve para uma loja sinalizar uma encomenda como estando pronta, ou seja, essa encomenda já pode ser levantada por um voluntário ou por uma empresa transportadora.

A segunda funcionalidade imprime as encomendas que ainda não foram levantadas e que ainda se encontram na loja.

A última funcionalidade serve para haver a atualização do número de pessoas na fila da loja, número esse que irá, depois, afetar o tempo que demora uma encomenda a ser entregue.



```
-----LOJA-----
1. Marcar encomenda como preparada
2. Ver encomendas em stock
3. Atualizar número de pessoas na fila
0. Retroceder

Insira uma opção --> 
```

Funcionalidades para uma Loja

16 Funcionalidades de uma Empresa Transportadora

Após o login efetuado, são apresentadas 10 funcionalidades a uma Empresa Transportadora

As primeiras 4 funcionalidades funcionam de maneira análoga a um Voluntário, servindo para atualizar o diferente estado de uma empresa transportadora

A funcionalidade 5 apresenta quanto custou a um utilizador efetuar uma dada encomenda nessa empresa transportadora.

A funcionalidade 6 serve para sinalizar uma encomenda como estando levantada. Tal como nos voluntários, uma encomenda só poderá ser levantada depois de se encontrar preparada.

Se existir apenas uma encomenda levantada, a empresa apenas pode utilizar a funcionalidade 7, que marcará uma encomenda como entregue e apresentará o tempo que demorou essa entrega, da mesma forma como nos voluntários.

Se existirem mais de 1 encomenda levantadas, a empresa assume a noção de rota e irá entregar todas as encomendas de uma vez. Também será apresentado o tempo que demorou a entrega a cada utilizador e, ainda, os kms percorridos durante as entregas.

A funcionalidade 9 imprime todas as encomendas recebidas por essa empresa, independentemente do seu estado.

A funcionalidade 10 apresenta o total faturado por essa empresa entre 2 datas.

```
-----EMPRESA TRANSPORTADORA-----
1. Declarar-se disponível para transporte de encomenda
2. Declarar-se indisponível para transporte de encomenda
3. Declarar-se disponível para transportar encomendas médicas
4. Declarar-se indisponível para transportar encomendas médicas
5. Apresentar preço de transporte de encomenda
6. Sinalizar levantamento de encomenda na loja
7. Anotar tempo e entrega de uma encomenda
8. Entregar todas as encomendas
9. Histórico de encomendas
10. Total faturado entre 2 datas
11. Imprimir menu
0. Retroceder

-> Tem 4 encomendas por levantar.
-> Tem 0 encomendas prontas a entregar

Insira uma opção --> 
```

Funcionalidades para uma Empresa Transportadora

17 Como se procede o registo e entrega de uma Encomenda

Para que uma nova encomenda seja efetuada e entregue ao utilizador é necessária a intervenção direta de 3 utilizadores, ou seja, é necessário efetuar 3 logins diferentes. Um utilizador doméstico, que irá registar a nova encomenda. Uma loja, que irá marcar a encomenda como estando preparada. E um voluntário/Empresa de Transportes, que irá fazer o levantamento da encomenda e a sua entrega. Tal como descrito anteriormente, uma encomenda só pode ser levantada depois de ser sinalizada como preparada, e esta só poderá ser entregue depois de ter sido levantada.

18 Factor de aleatoriedade no clima

Como descrito acima, as classes Voluntário e EmpresaTransportes possuem métodos que tentam efetuar uma previsão do tempo perdido devido às condições atmosféricas. Estes métodos são bastante simples, gerando um número aleatório entre 0 e 100. Se esse número se localizar entre 0 e 75, inclusive, significa que estão boas condições atmosféricas, ou seja, o tempo do transporte não será afetado. Se o número estiver entre 76 e 94, inclusive, quer dizer que não estão condições muito favoráveis ao transporte, sendo este atrasado em 30 minutos. Se

o número estiver entre 95 e 100, estão condições muito adversas, e o transporte sofre um atraso de 60 minutos. Este algoritmo foi baseado nas probabilidades de acontecimento de certas condições atmosféricas não serem iguais, podendo haver assim uma distinção entre os 3 diferentes possíveis estados do clima.

19 Conclusão

Concluindo este trabalho, consideramos que foi importante para ajudar a consolidar conhecimentos na programação orientada aos objetos, sendo possível aplicar os conhecimentos adquiridos ao longo do semestre nas aulas teóricas e práticas. Tivemos mais dificuldades em saber como implementar o uso de interfaces, e consideramos que esse aspeto poderia ter sido melhorado, mas procuramos sempre utilizar as diferentes soluções que nos foram apresentadas ao longo do semestre.

20 Nota explicativa

Ao ler o ficheiro update(Ficheiro binário onde estão guardados os dados apresentados no logs.txt), e ao entregar as encomendas já previamente carregadas, vão ser apresentados valores de tempo passado até à entrega muito elevados. Isto deve-se ao simples facto das encomendas pré-carregadas apresentarem data de emissão de 11/06/2020 às 14h15, ou seja, será sempre esse o valor usado como base para os cálculos. Efetuando uma nova encomenda, já será apresentado um valor mais realista.

21 Diagrama de Classes

