Universidade do Minho
Escola de Engenharia

Projeto
Paradigmas da Programação I

Nomes: Júlio Ismael Barroso de Oliveira (A89393); José Miguel Faria Pedroso(A89395); Sérgio Miguel Pereira Carvalho (A89313)

Instituição: Universidade do Minho

Curso: MIETI- Mestrado Integrado em Telecomunicações e Informática

Unidade curricular: Paradigmas da Programação I

Guimarães, 8 dezembro 2019

Índice

Índice de Figuras

Introdução

O presente trabalho consiste na resolução de um problema proposto pelo docente, no âmbito da unidade curricular de Paradigmas da Programação I, da Universidade do Minho. Este projeto representa o culminar de todo o trabalho desenvolvido durante o semestre, e como tal, todos os nossos conhecimentos apreendidos até o momento, devem ser aqui aplicados. Tivemos a ajuda do docente durante as aulas práticas, onde fomos tirando dúvidas e fomos delineando a nossa visão do projeto. Conseguimos aplicar devidamente, todas as ferramentas que a linguagem nos oferece, e que foram lecionadas nas aulas, no decorrer do primeiro semestre.

Um dos objetivos com a realização deste projeto é conseguirmos conciliar o objetivo do projeto, com a nossa criatividade.

Descrição do Enunciado

Na cadeira de Paradigmas da Programação I fomos desafiados a criar um programa intitulado de EntregasExpress, um projeto com o objetivo de ajudar nos serviços de entregas ao domicílio, uma vez que estes cada vez são mais utilizados.

Cada transportadora disponibiliza de apenas um tipo de serviço e veículo, o que simplifica bastante o projeto. A empresa deve fornecer todo o tipo de informações acerca do serviço que irá oferecer, como o preço por quilometro, a autonomia, a velocidade, entre outros, indicando também detalhes sobre que tipo serviço fornece. Neste mesmo contexto, o cliente ao registar-se deve informar ao sistema a localização final dos serviços que usufrui, podendo esta ser alterada mais tarde caso deseje. Ao efetuar um serviço, o cliente deve informar que tipo de produtos pretende que sejam transportados, a aplicação deve verificar o pedido feito e selecionar as empresas mais adequadas, dando opção de escolha.

Apos o serviço ser concluído, o sistema deve registar a operação, podendo ser acedida através do histórico de pedidos. Ao efetuar um pedido os clientes podem solicitar a entrega de um item através de uma escolha automática do transportador mais rápido ou mais barato, ou pode ser-lhes oferecida uma lista com os transportes disponíveis que podem efetuar este pedido, ou então caso o cliente deseje um custo máximo e tempo máximo, o sistema deve oferecer uma lista de transportes capaz de efetuar o desejado.

A aplicação deve considerar diversos tipos de serviços de entrega, como pessoas, bus, big e urgentes, uma vez que se pretende adicionar novos tipos de serviço, deve-se projetar o código de modo a que seja possível adicionar sem alterar o código já existente.

Descrição da Arquitetura de Classes

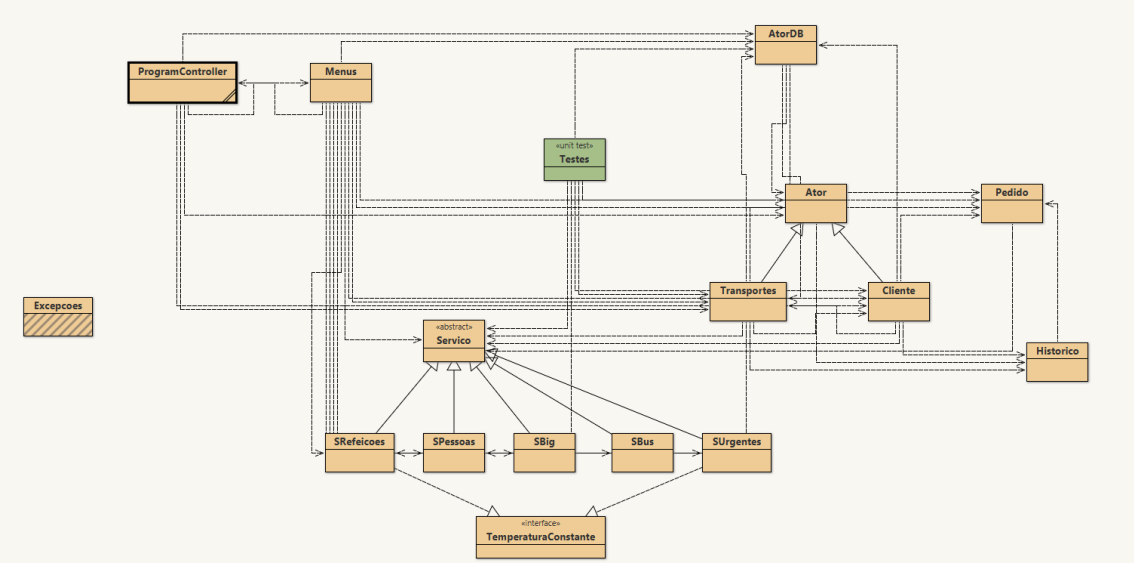

Figura 1- Arquitetura de Classes elaborada pelos discentes.

Nesta proposta utilizamos vários tipos de classes com diferentes atributos, desde classes concretas a classes abstratas, e implementação de exceções e interfaces (ver figura 1). Por questões de necessidade de gravação do estado da aplicação, as classes que contêm dados que devem ser gravados, têm em comum a implementação da interface *Serializable*, oferecendo, assim, um método simples de gravação dos objetos com todos os seus estados atuais em ficheiro, para posterior consulta.

- Classe AtorDB

A base da aplicação pois é nesta classe que são gravados, através da implementação de um HashMap, onde como key temos o e-mail e como value o Ator (Cliente ou Transportador) correspondente. É também aqui que se verifica a existência de um ator e onde se faz a autenticação durante o login e onde a cada interação no menu é verificado o estado de cada pedido que existe no sistema.

- Classe Ator

Possui as informações que as classes Cliente e Transportadores têm em comum. Estes partilham email, nome, password, morada e data de nascimento, também como um histórico de pedidos e as suas respetivas coordenadas x e y.

- Classe Cliente

Deriva da classe Ator, contendo por isso todos os atributos da mesma. É nesta classe que esta implementado o método de adicionar um pedido ao histórico do cliente e do respetivo transportador, isto é claro após todo o processo que leva à escolha de um determinado transportador, uma vez que este pedido se trata da mesma instância para os dois.

- Classe Transportes

Deriva também da classe Ator, mas difere da classe Cliente pois possui alguns atributos adicionais, nomeadamente, o preço por km, o tempo por km, a autonomia, o preço extra noturno, a percentagem de desconto (se a transportadora o pretender), a classificação da transportadora (avaliada pelos clientes após a entrega de um produto) e a disponibilidade do transportador. É nesta classe que se tratam todos os métodos de filtrar, consoante as necessidades e desejos do cliente, o(s) transportador(es) recorrendo a vários recursos como Iterators, TreeSets e Lists, após os respetivos cálculos terem sido feito tendo em conta a distancia, o tempo e o preço que são influenciados, primeiramente pela posição do cliente e do produto e pela disponibilidade de cada transportadora mas também por variáveis aleatórias, como por exemplo, a existência de trânsito.

- Classe Serviço

A classe Serviço é uma classe abstrata que serve de base para as subclasses dos vários serviços existentes sendo elas SBig, SBus, SPessoas, SRefeicoes e SUrgentes. Onde esta solução permite adicionar algum novo tipo de serviço de maneira a não alterar uma linha de código. De resto esta classe apenas contém um atributo que é comum a todas as subclasses, que é o limite de carga para cada tipo de serviço, definido pelo transportador.

- Classes SPessoas, SBus e SBig

Classes que derivam da classe Serviço, onde as duas primeiras têm como atributo a disponibilidade para o transporte de crianças, consoante a transportadora.

- Classes SRefeições e SUrgentes

Classes que derivam da classe Serviço, mas com a particularidade de implementarem a interface TemperaturaConstante, onde cada transportador pode definir se garante ou não a temperatura constante dos produtos a ser transportados.

- Classe Pedido

Classe que contem as informações de um pedido realizado pelo cliente, e tem como atributos o tipo de serviço, a data e hora inicial e final, preço associado, se já foi concluído e uma classificação que servirá para determinar a classificação de cada transportador.

- Classe Historico

Classe que possui como atributo um ArrayList de pedidos, que esta associado a cada ator existente na aplicação, onde cada Pedido, como foi mencionado em cima na classe Cliente, é adicionado ao histórico do cliente e transportador correspondente sendo a mesma instância para ambos.

- Classe Menus

Classe onde são feitas todas as interações com o utilizador da aplicação, e onde se encontram os três menus existentes (Principal, Cliente e Transportador), bem como onde se invocam todos os métodos disponíveis. A partir daqui o utilizador da aplicação poderá optar pelas várias opções que o programa oferece.

- Classe ProgramController

Classe que serve como controlo de toda a aplicação, sendo a partir dela que são invocados os menus, consoante os dados introduzidos.

- Exceções

Tendo em conta os requisitos da aplicação criamos cinco exceções que melhor correspondem aquilo que pode acontecer durante a execução do programa. Com isto temos:
-NoAtorException (quando se faz uma procura e não se encontra nenhum ator que corresponda);

-ExistingAtorException (quando se está a fazer um registo e já existe um ator com o mesmo e-mail introduzido);

-NoExistentServiceException (quando se escolhe um serviço que não corresponde a nenhum dos existentes);

-NoStoredDataException (quando se efetua uma visualização do histórico e este esta vazio);

-NoSuportedException (quando o utilizador efetua alguma operação inválida).

Descrição da Aplicação

Com o desenvolvimento deste projeto, a aplicação começou a tomar grandes proporções, pelo que foi necessária uma boa estruturação. A aplicação está dividida em três importantes menus, o "Menu Principal" trata-se do primeiro menu à qual o utilizador se depara.

No "Menu Principal" (ver figura 2) o utilizador pode fazer o *login* ou o registo de uma conta, sendo que nesta última pode-o fazer como cliente ou transportadora. Ao fazer o *login*, caso o utilizador introduza as credenciais certas, este será dirigido para um dos outros dois menus, para o "Menu Cliente" (ver figura 3) caso seja um cliente ou para o "Menu Transportes" (ver figura 4) caso seja uma transportadora.

Ao aceder o "Menu Cliente", caso o utilizador tenha efetuado algum pedido recente, este poderá avaliar a transportadora que realizou o seu pedido, após isso poderá escolher entre diversas opções, a primeira é a efetuar pedido, o utilizador tem de inserir o serviço que pretende e as coordenadas para onde a transportadora se deve dirigir. Apos a escolha do serviço, o utilizador terá diversas opções, umas mais rápidas, onde a aplicação escolhe automaticamente a transportadora mais rápida, a mais barata ou a que tem melhor classificação, ou simplesmente ver todas as transportadoras disponíveis. O utilizador pode também escolher entre uma lista de transportadoras com um preço e tempo máximo escolhido pelo mesmo.

Voltando ao "Menu Cliente" na segunda opção o utilizador pode ver o histórico de pedidos concluídos, onde mostra informações como a transportadora, o serviço desta, a que horas o pedido foi efetuado e entregue, tendo também o preço deste (exemplo: Uber SPessoas 02-12-2019 11:51 02-12-2019 12:05 8,49). Na terceira opção a aplicação mostra todos os dados da conta e o utilizador pode alterar caso deseje. Na quarta opção o utilizador tem acesso a todas as transportadoras que já utilizou e pode escolher novamente uma se assim o desejar. Na quinta e última opção o utilizador tem acesso a uma lista das cinco transportadoras com mais serviços concluídos por ordem decrescente.

No "Menu Transportes" na primeira opção o utilizador pode verificar os cinco pedidos mais recentes (concluídos ou não), na segunda opção mostra todos os pedidos concluídos e na terceira o utilizador pode verificar o total faturado. Na opção quatro, tal como no "Menu Cliente" a aplicação mostra todos os dados da conta e o utilizador pode alterar caso deseje, porém aqui o utilizador também pode alterar o serviço a que está

associado. Na quinta o utilizador pode criar uma campanha de descontos ou alterar uma existente. Na sexta opção o utilizador pode verificar a média da classificação dada pelos seus clientes.

Sempre que o utilizador sai de um destes menus volta para o "Menu Principal".

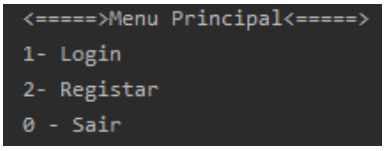

Fig.2 – Menu Principal. Retirado da aplicação.

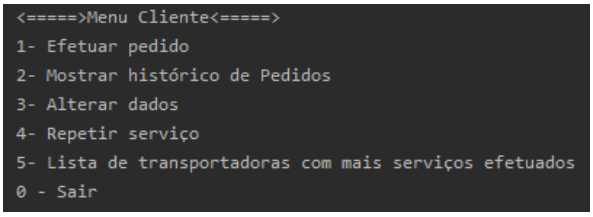

Fig.3 – Menu de Cliente. Retirado da aplicação.

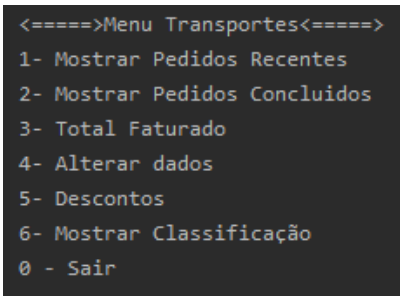

Fig.4 – Menu de Transportes. Retirado da aplicação.

Como seria possível incluir novos tipos de viaturas na aplicação

Para se poder incluir novos tipos de viaturas na aplicação, seria necessária uma nova classe Veículo. Esta classe possuiria os atributos de cada veículo da transportadora, podendo o atributo de autonomia, de tempo por km e disponibilidade passar para esta classe. E na classe Transportes, poderíamos ter como atributo uma List dos veículos de uma determinada transportadora, sendo fácil de alterar e aceder a informação contida em cada elemento presente. Para verificar a disponibilidade bastaria percorrer esta List e verificar se existe algum veículo que cumpra os requisitos do cliente e que esteja disponível.

Conclusão

O presente trabalho iniciou-se com o planeamento do projeto, bem como as suas diretrizes seguindo-se a elaboração do respetivo código em linguagem Java. Tivemos sempre em conta alguns pontos que foram abordados ao longo das aulas teóricas, tais como a reutilização de código, a abstração de dados e a importância de termos um código que se fosse preciso alguma alteração, só precisaríamos de trabalhar numa zona especifica de maneira que o resto da aplicação não fosse afetada.

Ao longo do processo, foram mostradas diversas maneiras de resolver os vários requisitos da aplicação, consequentemente desenvolvemos a nossa capacidade de decidir qual a melhor opção consoante o contexto em que nos situamos.

Em suma, este foi um projeto ambicioso que acompanhou sempre a nossa evolução ao longo da UC, como tal fez-nos desenvolver capacidades de superação e trabalho de equipa, bem como os nossos conhecimentos de POO, onde foi posto à prova o nosso espírito de entreajuda, e onde tivemos de aplicar todos os conhecimentos adquiridos ao longo do semestre. Como tal, sentimos que demos o nosso melhor na execução deste projeto.