Programação Orientada aos Objetos
UMeR

André Rodrigues Freitas A74619
Carlos Daniel Leitão da Silva Vieira A73974
Sofia Manuela Gomes de Carvalho A76658

3 de Junho de 2017



Conteúdo

1 Introdução

Este projeto tem como objetivo a criação de uma aplicação que permita a gestão de um serviço de transporte de passageiros muito parecido com a conhecida aplicação UBER. Pretende-se que este projeto seja desenvolvido usando uma linguagem orientada aos objetos sendo, neste caso, utilizada a linguagem JAVA.

2 Desenvolvimento

2.1 Descrição do problema

O nosso trabalho consiste num serviço de gestão de transporte de passageiros. Deste modo, pretende-se e supõe-se que dê suporte a toda a funcionalidade que permite a que qualquer utilizador realizar uma viagem usando esta aplicação. Para isto, deve abranger tudo o que seja relacionado com uma viagem: o utilizador, o motorista, as viaturas e a marcação de viagens bem como o respetivo preço.

Por último, e não menos importante, devemos guardar todas as operações efetuadas de modo a que UMeR possa manter um registo de todas as suas atividades, até para que os próprios utilizadores possuam um historial que possam consultar.

Para aumentar a dificuldade do nosso trabalho, foi possível que certas empresas pudessem usar este serviços, tendo eles próprios os motoristas e as viaturas. Essa gestão é feita por essas mesmas empresas.

2.2 Arquitetura de classes

Para a realização deste trabalho, decidimos que precisávamos da seguintes classes(sendo que algumas são superclasses de outras classes):

- Utilizador

Nesta classe Utilizador, estão as variáveis de instância pedidas no enunciado do projeto como email, nome, morada, data de nascimento. Além disso, criamos um Map de viagens, que é inicializado a nulo, e uma password.

A classe Utilizador será superclasse de Cliente e Motorista, sendo que ao primeiro acrescentamos a localizacao do cliente (que irá estar relacionado com uma outra classe que falaremos mais adiante) e ao Motorista acrescentou-se o grauCumprimento, classificacao, kms, disponibilidade e matricula.

- Viatura

Nesta classe Viatura definimos como variáveis de instância a velMedKm, precoBaseKm, fiabilidade, matricula, bem como a localização.

Tem como subclasse Moto, CarroLigeiro e Carrinha, sendo que a primeira, além das variáveis de instância que definem a Viatura, tem também capacidade e ocupacao. A Carrinha e o CarroLigeiro têm capacidade e ocupação e uma lista de espera além das variávies que herda da superclasse.

- Viagem

Nesta classe temos os dados necessários para a sua identificação como custo, duracao, distancia. Temos também um inteiro que permite identificar a viagem, além da sua localização, e para isso iremos precisar de uma outra classe (Espaco2D), e uma data que irá ser um LocalDate, usando, para isso, um import respetivo.

- UMeRApp

Esta classe tem um papel fundamental pois é ela que vai permitir que a aplicação ganhe vida. É ela que, através de um simples menu, permite fazer uma reserva ou não de viagens, permite registar um utilizador, iniciar sessão, entre outras.

Assim sendo, esta classe depende de uma classe chamada Menu.

- UMeR

Esta classe é classe-mãe de todo o nosso trabalho, visto que é ela que tem todos os métodos importantes, nomeadamente os requisitos báicos que são pedidos no enunciado.

É composta por três Maps: um para identificar utilizadores através de uma *string*, que corresponde ao email do utilizador por identifica-o inequivocamente, outro para as viatura que são também distinguidas com uma *string*, que corresponde à sua matrícula e, finalmente, um map para as empresas, que são identificadas pelo seu nome.

- Espaco2D

Esta classe permite localizar uma posição num espaço de coordenadas e para isso temos duas variáveis de instância: o float x e o float y.

Menu

É nesta classe que estão os métodos necessários para apresentar o menu, assim como ler a opção que desejamos.

Fizemos esta classe para não tornar a classe UMeRApp mais sofrível de ler, dividindo alguns métodos que nao achássemos necessários para esta mesma classe Menu.

Empresa

A classe Empresa corresponde à agregação de viaturas e de motoristas e, por isso, tem uma lista de motoristas e uma lista de viaturas dessa empresa, assim como o seu nome.

- ListaEspera

A classe ListaEspera é uma interface onde definimos os métodos que vão ter de ser definidos nas classes que implementam esta interface. Essas classes são a classe CarroLigeiro e a classe Carrinha.

A seguir apresentam-se todas as classes de exceção, que permitem tratar de casos que não deveriam acontecer.

- UtilizadorRepetidoException
- ViaturaNaoExistenteException
- EmpresaInexistenteException
- UtilizadorInexistenteException
- MotoristaNaoDisponivelException
- ViaturaNaoEDaEmpresaException
- MotoristaNaoEDaEmpresaException
- ViaturaRepetidaException
- ClienteNaoTemViagemException

- Testes

Esta classe de teste foi implementada devido à necessidade de popular a base de dados para poder testar a aplicação.

Em todas as classes criadas, fizemos evidentemente os sets e gets necessários, bem como os três tipos de construtores abordados e os métodos

clone, equals e toString. A exceção a isto são as classes UMeRApp, ListaEspera, Menu, Testes e todas as exceções por não necessitarem desses métodos.

Na figura abaixo, é possível ver todas as classes desenvolvidas no BlueJ.

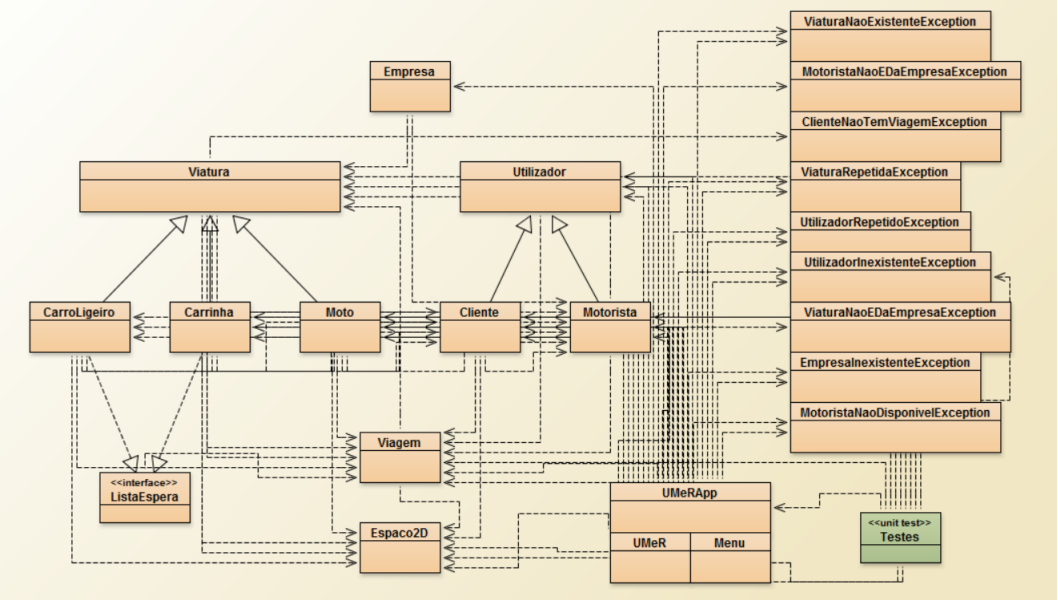

Figura 1: Esquema de classes BlueJ

2.3 Manual da aplicação

Nesta secção, é disponibilizado um pequeno manual para facilitar a utilização desta aplicação. Para iniciar a sua execução, é necessário primeiramente aceder à classe UMeRApp com o botão direito do rato e selecionar a função main como se vê na figura abaixo.

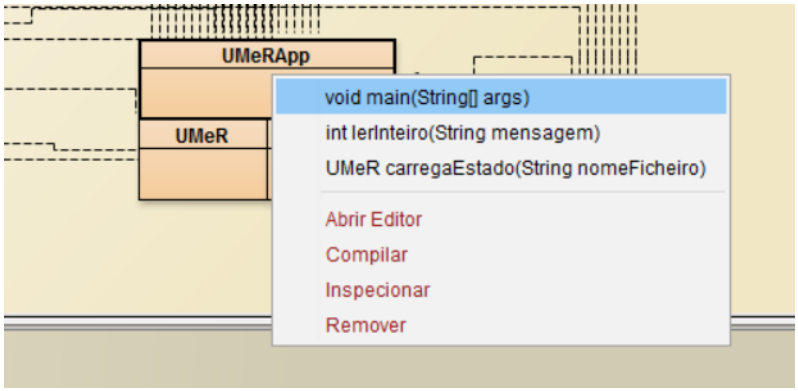

Figura 2: Primeiro passo intermédio para início da aplicação

De seguida, o segundo passo é selecionar apenas a opção OK na janela disponibilizada.

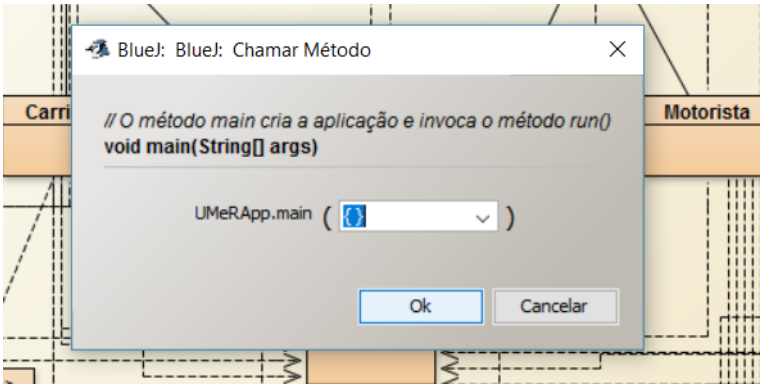

Figura 3: Segundo passo intermédio para início da aplicação

Após os passos anteriores, surge uma janela de terminal do BlueJ com um menu principal que possui várias funcionalidades. Para aceder a estas funcionalidades, escolhe-se o número correspondente a essa opção.

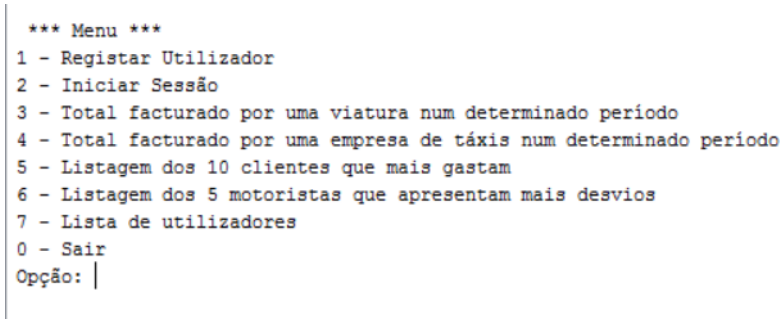

Figura 4: Menu exibido quando a aplicação é iniciada

Dessas opções, podemos destacar a de registar utilizador em que fornecemos isntruções claras durante a execução para melhor guiar alguém menos experiente e ainda a de iniciar sessão, que conforme o utilizador que inicia sessão seja motorista ou cliente, vai apresentar menus diferentes, pois estes têm acesso a funcionalidades diferentes na aplicação. Como se pode visualizar, para iniciar sessão basta fornecer o email e a password e mais funcionalidades serão apresentadas, podendo qualquer uma delas ser também selecionada.

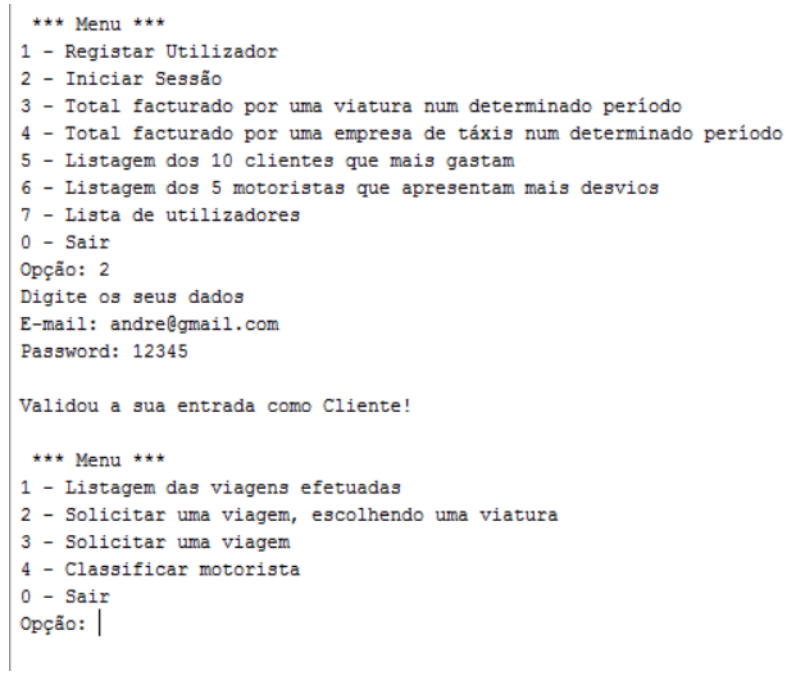

Figura 5: Menu exibido após um cliente iniciar sessão

```
 *** Menu ***
1 - Registar Utilizador
2 - Iniciar Sessão
3 - Total facturado por uma viatura num determinado período
4 - Total facturado por uma empresa de táxis num determinado período
5 - Listagem dos 10 clientes que mais gastam
6 - Listagem dos 5 motoristas que apresentam mais desvios
7 - Lista de utilizadores
0 - Sair
Opção: 2
Digite os seus dados
E-mail: daniel@gmail.com
Password: 123456daniel

Validou a sua entrada como Motorista!

 *** Menu ***
1 - Listagem das viagens efetuadas
2 - Associar viatura
3 - Alterar disponibilidade
0 - Sair
Opção:
```

Figura 6: Menu exibido após um motorista iniciar sessão

2.4 Introdução de novos tipos de viaturas e motoristas

Para a introdução de novos tipos de viaturas poderiam ser criadas novas subclasses da superclasse Viatura, o que possibilitaria, através do mecanismo de herança de classes, a disponibilidade de todas as variáveis de instância e métodos da classe Viatura numa nova subclasse como, por exemplo, Bicicleta. Assim evitar-se-ia a repetição desnecessária de código comum a outros tipos de viaturas já existentes.

A introdução de novos tipos de motoristas poderia ser feita através de um mecanismo semelhante ao que foi descrito acima, criando-se subclasses da superclasse Motorista de modo a apenas ser necessário acrescentar o código que diferencia cada subclasse das restantes e que permita especificar o tipo de Motorista adicionado.

Após isto, faltaria ainda criar a hipótese de utilização destas novas entidades, completando e adaptando as funcionalidades já existentes na aplicação mediante as novidades acrescentadas.

3 Conclusão

Fazendo uma avaliação global da realização deste projeto, deparamo-nos com algumas dificuldades, mas criamos um projeto, na nossa opinião, fluído e bem pensado, passivo de uma boa compreensão por parte de elementos externos à realização do mesmo.

A principal dificuldade com que nos deparamos foi a necessidade de popular a base de dados da aplicação e ainda a implementação das funcionalidades extra relacionadas com as empresas e com as listas de espera de determinadas viaturas que não foram concluídas na sua totalidade.