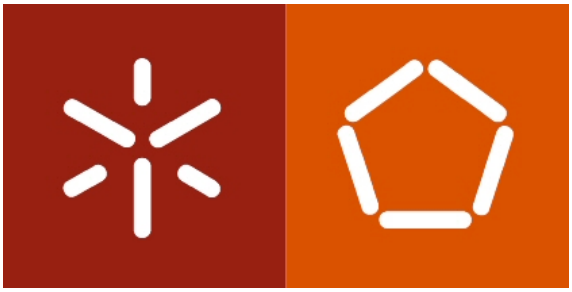
Programação Orientada Aos Objectos 2018/2019

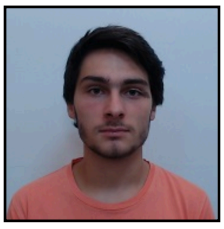
Relatório do Trabalho Prático

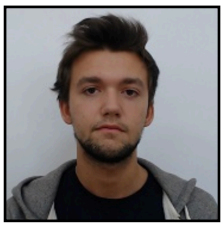


Universidade do Minho - Mestrado Integrado em Engenharia Informática

Grupo

João Abreu - a84802
Hugo Matias - a85370
Jorge Vieira - a84240





Classe Actor

A classe **actor** contém a informação básica dos actores da aplicação (proprietários e clientes).

A informação disponível nesta classe é a seguinte:

- Email
- Nome
- Password
- Morada
- Data de nascimento
- Classificação
- Nif
- Histórico de alugueres

É uma classe abstracta pois não fazia sentido termos a possibilidade de instanciar um novo Actor. É a super classe das classes “Proprietário” e “Cliente”.

Classe Proprietário

A classe **proprietário** é uma das subclasses da classe “Actor”, tendo assim acesso também a toda a informação básica dos actores da aplicação. Juntamente com estes dados também temos acesso aos carros que este proprietário registou na aplicação.

Um proprietário, depois de registado e feito o seu respectivo login na aplicação é capaz de:

- Registar viatura
- Classificar cliente após aluguer
- Ver o histórico de alugueres entre datas
- Sinalizar que um carro está disponível
- Abastecer um carro
- Alterar o preço por km
- Ver quanto custaram as viagens

Classe Cliente

A classe **cliente** é uma das subclasses da classe “Actor”, tendo assim acesso também a toda a informação básica dos actores da aplicação. Juntamente com estes dados também temos acesso às suas coordenadas atuais, para no futuro saber qual o melhor carro para ser alugado.

Um cliente, depois de registado e feito o seu respectivo login na aplicação é capaz de:

- Solicitar um aluguer
- Classificar carro depois do aluguer
- Classificar proprietário depois do aluguer
- Ver o histórico de alugueres entre datas

Classe Carro

A classe **carro** contém a informação básica das viaturas da aplicação.
A informação disponível nesta classe é a seguinte:

- Marca
- Matrícula
- Nif do proprietário
- Velocidade média por Km
- Preço por Km
- Consumo por Km
- Coordenadas
- Classificação
- Disponibilidade
- Lista de alugueres realizados

É uma classe abstracta pois não fazia sentido termos a possibilidade de instanciar um novo Carro e também não sabemos como modificar e obter valores da autonomia do carro (dado que estará nas subclasses) pois esta varia de tipo de combustível de carro para carro. É a super classe das classes “CarroEletrico”, “CarroHibrido” e “CarroGasolina”.

Classe CarroGasolina/CarroEletrico

As classes **CarroGasolina** e **CarroEletrico** são muito semelhantes entre si. As diferenças são apenas no tipo de combustível que usam. Em relação aos métodos não há quaisquer diferenças.

Têm acesso à autonomia atual e autonomia inicial do carro.

Classes CarroHibrido

A classe **CarroHibrido** contém 2 tipos de combustíveis (gasolina e elétrico). Sempre que é criado uma nova instância desta classe, a autonomia passada por parâmetro é separada em 2 e é atribuída cada uma destas metades para os depósitos dos 2 tipos.

Tem acesso à autonomia atual de gasolina e de elétrico e também à autonomia inicial de gasolina e elétrico.

Nota: É guardado nestas classes a autonomia inicial de cada carro pois, no futuro, um aluguer só pode ser realizado se a autonomia atual do carro for superior a 10% da sua inicial.

Classe Aluguer

A classe **aluguer** contém toda a informação relativa ao pedido de um aluguer a partir de um cliente para um proprietário de um carro.

A informação disponível nesta classe é a seguinte:

- Posição inicial antes do aluguer
- Posição destino depois do aluguer
- Preço estimado do aluguer
- Tempo estimado da viagem
- Nif do cliente
- Data da realização do aluguer
- Preferência da escolha do carro do cliente

Classe IO

A classe **IO** é, tal como o nome indica, onde se faz todas as operações de leitura de dados/opções bem como a resposta da aplicação face a esses inputs.

É nesta classe que mostramos ao utilizador as opções do(a):

- Aplicação UMCarroJa
- Proprietário, após o login ter sido efectuado com sucesso
- Cliente, após o login ter sido efectuado com sucesso
- Tipos de alugueres para o cliente escolher
- Top 10 clientes ordenados por kms percorridos/nº de vezes que alugaram

É também aqui que:

- Criamos um novo proprietário/cliente/carro tendo em conta os dados inseridos
- Fazemos os logins do proprietário/cliente
- Escrevemos as operações realizadas na aplicação num ficheiro de logs

Classe UMCarroJa

A classe **UMCarroJa** tem toda a informação da aplicação. Tem 3 coleções de dados capazes de aceder a qualquer informação que queiramos. Usamos Maps para os proprietários, clientes e carros, sendo a key para os proprietários e clientes o seu respectivo NIF e para os carros a sua matrícula (tal como as keys dos maps, estes valores são únicos).

Esta classe também é capaz de guardar toda sua informação num ficheiro e carregá-la mais tarde.

Classe Main

É na classe **main** que tratamos os valores lidos da classe IO. Dependendo da opção escolhida, esta classe é executa a respectiva função da escolha.

Classe Exceptions

Ao longo do desenvolvimento do projecto tivemos a necessidade de criar certas excepções com o objectivo de facilitar o tratamento de dados. As excepções criadas foram:

- ActorNotFoundException —> quando tentamos aceder a um actor inexistente
- CarroNotFoundException —> quando tentamos aceder a um carro inexistente
- WrongPasswordException —> quando o email existe mas a password está mal

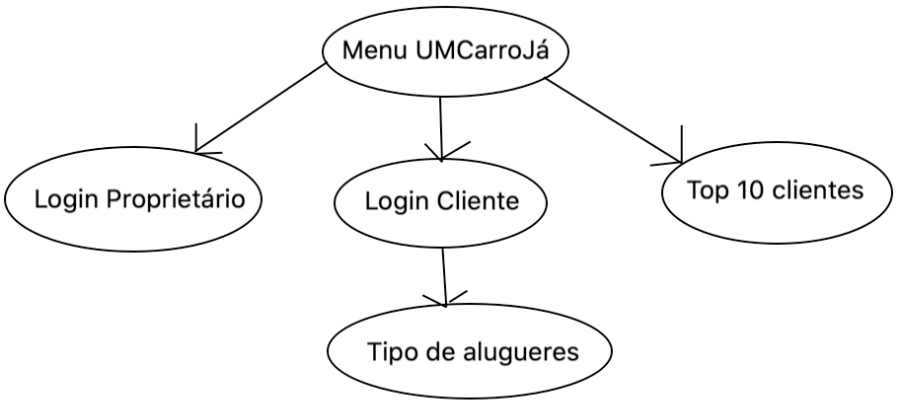
Classe Comparactors

Foram criados 2 tipos de comparactors com o único propósito de fazer os tops 10 clientes, por kms percorridos e por nº de vezes que alugaram.

Para obtermos essa lista de 10 clientes, pegamos nos clientes que estavam na estrutura de dados geral e criamos uma em que tínhamos na mesma todos os clientes mas agora ordenados pela ordem que quiséssemos. Foi aí que os comparactores tiveram utilidade. Depois só precisávamos dos "maiores" 10 elementos para obter os resultados pretendidos, e estes estariam no início ou no fim da lista ordenada (optamos por fazer comparactores que pusessem os 10 "maiores" no início).

Manual de utilização da aplicação

A aplicação começa por mostrar ao utilizador as diversas opções que pode realizar. Como referido na explicação da classe IO, o utilizador pode registar proprietários/clientes/carros, logins nos proprietários/clientes, realizar alugueres, classificar actores ou viaturas etc. No total, existem 5 menus diferentes e a imagem abaixo mostra como aceder a cada um deles. Nota: O menu UMCarroJá é imediatamente mostrado ao user quando o programa arranca.



Como seria possível incluir novos tipos de viaturas na aplicação?

A aplicação desenvolvida é capaz de incluir novos tipos de viaturas, basta-se tornar uma subclasse da classe abstracta “Carro”. Esta contém métodos abstractos que vão obrigar o novo tipo de viatura inserir na sua classe. O resto do projecto seria compatível com a nova inserção de tipo de viatura.

Conclusão

Este trabalho permitiu que os discentes membros deste grupo pusessem em prática os conhecimentos teóricos de JAVA. Houve alguma dificuldade no que tocou a guardar o estado e a decidir o aluguer ,porém foi resolvido com certa rapidez. Foi nutrido conhecimento relativamente a comparators, classes de exceção e interfaces como o Map e o Set.

Arquitectura final do sistema