Universidade do Minho
Escola de Engenharia

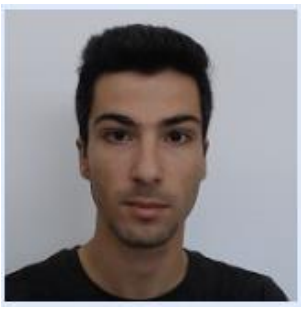
UM Carro já

Trabalho Prático de POO

Grupo 61

a84610 – Gonçalo Almeida

a86788 – Lázaro Pinheiro



a80960 – Rúben Rodrigues

Laboratórios de Informática III
2º Ano, 2º Semestre
2018/2019

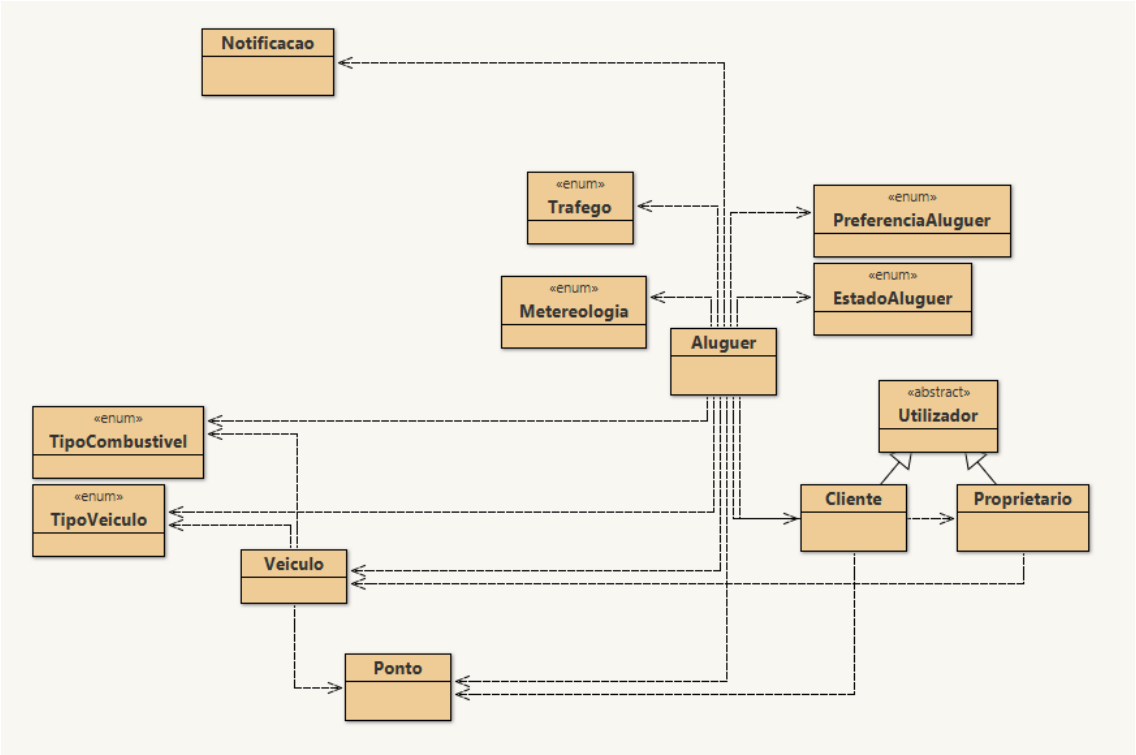
Índice

Introdução

O presente trabalho prático desenvolve-se no âmbito da Unidade Curricular Programação Orientada a Objetos lecionada no 2º semestre do 2º ano do curso de Engenharia Informática.

Este trabalho tem como objetivo criar um serviço de aluguer de veículo particulares pela internet, em que um proprietário pode registar um veículo que pode vir mais tarde a ser alugado por um cliente de acordo com as suas necessidades e a sua localização. Cada proprietário pode ter o número de veículos que assim entender, mas para já só é possível registar carros a combustível, elétricos ou híbridos, mas como será explicado, resolvemos o problema de modo a facilitar a introdução de novos veículos no futuro.

Classes base



Aluguer

A classe aluguer é dividida por três entidades, na primeira fase os clientes podem fazer um pedido de aluguer ao qual indica, através de uma notificação que é enviada ao proprietário do veículo, que o cliente intenção de alugar, o tempo médio de viagem e o veículo pretendido.

A segunda fase consiste na confirmação do aluguer na qual o proprietário pode aceitar ou rejeitar.

Após esta fase o cliente poderá efetuar a viagem e, no caso de o veículo ficar com autonomia baixa, é enviado ao proprietário uma notificação.

Ponto

A classe Ponto implementa um ponto num plano 2D, fornecida nas aulas práticas, em que as coordenadas são duas variáveis do tipo double que juntas criam o ponto (x,y). Esse ponto é usado para saber a localização de um veículo, de um cliente e do destino que o cliente deseja.

Veículo

A classe Veículo foi criada para saber toda a informação relacionada com os veículos, desde o seu tipo até às classificações recebidas nos seus alugueres. É criado com o mínimo de requisitos possível, como marca, matrícula, preço por km, entre outros. Depois é atualizado sempre que usado, ou sempre que o proprietário decide fazer alguma alteração. Sempre que é alugado, é também acrescentada à sua lista de classificações a classificação dada por o utilizador. O carro tem uma autonomia associada, que faz com que o programa saiba se é capaz de realizar o percurso pedido, essa mesma autonomia, se estiver abaixo dos 10% faz com que o veículo não possa ser usado até ser abastecido.
A localização do veículo é atualizada sempre que for usado.

Notificação

A classe Notificação tem um funcionamento semelhante a um email. As suas variáveis de instância são o nif do destinatário, a data de envio, o assunto e o conteúdo. As suas funcionalidades permitem alertar um proprietário de pedidos de alugueres e se um dos seus veículos precisa de ser abastecido. Permite também alertar um cliente se o seu pedido de aluguer foi aceite ou rejeitado.

Utilizador (abstract)

A classe Utilizador é a classe que carateriza um tipo que define como as classes que a herdam se devem comportar. Aqui são criadas as variáveis mutuamente usadas por o Cliente e por o Proprietário, tal como o email, password, nome, nif e morada. Cria assim uma base para trabalharmos com todos os utilizadores do nosso programa.

- ### Cliente

  A classe Cliente é uma subclasse da Utilizador, que segue a sua informação como modulo, sendo que se o utilizador estiver registado, ou se for registar como cliente, o programa necessita saber informações adicionais, como por exemplo a sua localização. Depois mais tarde quando fizer alugueres, vai lhe ser dado um valor de destreza e uma lista de classificações.

- ### Proprietário

  A classe Cliente é uma subclasse da Utilizador, que segue a sua informação como modulo, sendo que se o utilizador estiver registado como proprietário, o programa

sempre que lhe forem feitos alugueres, vai lhe atribuindo uma lista de classificações.

TipoVeículo (enum)

Consiste num conjunto fixo de constantes (static final), como uma lista de valores pré-definidos, sendo neste caso apenas ‘Carro’. Criamos em enum porque facilita no futuro a adição de novos tipos de veículos.

TipoCombustível (enum)

Consiste num conjunto fixo de constantes (static final), como uma lista de valores pré-definidos, sendo neste caso ‘Gasolina’, ‘Elétrico’ e ‘Hibrido’.

PreferenciaAluguer (enum)

Consiste num conjunto fixo de constantes (static final), como uma lista de valores pré-definidos, neste caso ‘MaisPerto’, ‘MaisBarato’, ‘MaisPertoBarato’e ‘Especifico,Autonomia’.

EstadoAluguer (enum)

Consiste num conjunto fixo de constantes (static final), descritos como uma lista de valores pré-definidos, sendo neste caso ‘Espera’, ‘Aceite’, ‘Rejeitado’e ‘Terminado’.

Tráfego (enum)

Consiste num conjunto fixo de constantes (static final), como uma lista de valores pré-definidos, sendo neste caso ‘Fluido(1.05)’ e ’Congestinado(1.6)’. Cada constante tem um valor associado que é uma percentagem que vai afetar o preço e o tempo associado a cada aluguer, que vai ser mais tarde calculado de acordo com o tráfego atual.

Meteorologia (enum)

Consiste num conjunto fixo de constantes (static final), como uma lista de valores pré-definidos, sendo neste caso ‘Neve(1.8)’, ‘Chuva(1.3)’, ‘Nevoeiro(1.4)’, ‘Vento(1.2)’, ‘Nublado(1.1)’ e ‘Sol(1.0)’. Cada constante tem um valor associado que é uma

percentagem que vai afetar o preço e o tempo associado a cada aluguer, que vai ser mais tarde calculado de acordo com a meteorologia atual.

Gestores

GestorVeiculos

A classe gestor de veículos implementa um hashmap cuja chave é a matrícula de um veículo e o conteúdo é o respetivo veículo. Serve como base de dados para todos os veículos registados no sistema.

GestorUtilizadores

A classe gestor de utilizadores implementa um hashmap cuja chave é o nif do utilizador e o valor é o próprio utilizador. Serve como base de dados para todos os utilizadores registados no sistema.

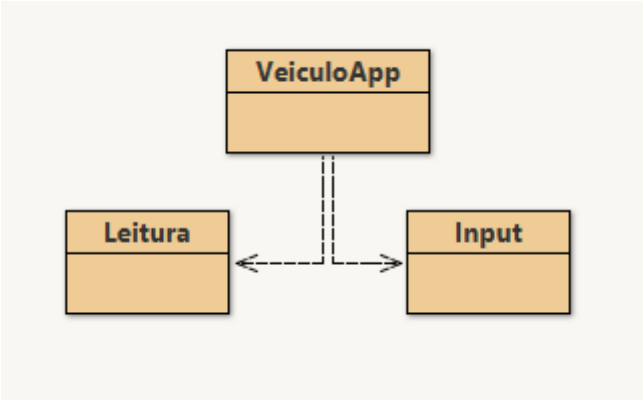
GestorAlugueres

A classe gestor de Alugueres implementa um set de todos os alugueres registados no sistema.

GestorNotificacoes

A classe gestor de notificações implementa um hashmap cuja chave é o nif do utilizador e o valor é a lista de notificações que esse utilizador recebeu. Serve como base de dados para todas as notificações registadas no sistema.

App



VeiculoApp

A classe VeiculoApp implementa todos os menus e faz a ligação necessária com o resto.

Leitura

A classe Leitura tem como variáveis de instância o path do ficheiro de carregamento de dados inicial para os gestores que são variáveis de instância da classe VeiculoApp.

Input

A classe Input descarta possíveis problemas de input e valida, aquando a introdução de elementos, se estes são validos ou não.

Exceções

As seguintes exceções mostram todas as verificações que fazemos em cada funcionalidade da aplicação.

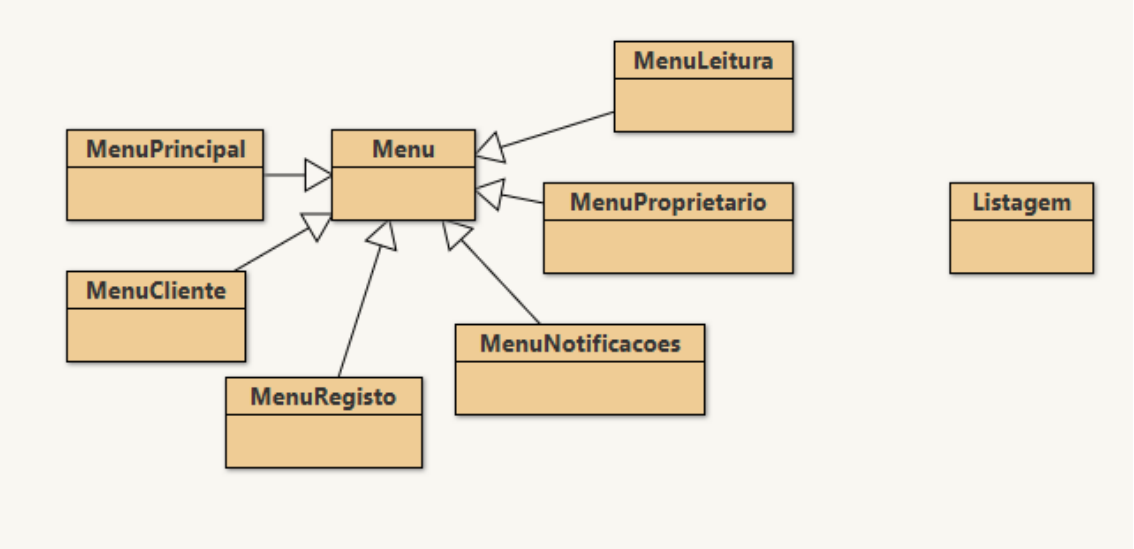
As exceções que criámos foram as seguintes:

- AvaliacaoInvalidaException
- VeiculoNaoEncontradoException
- VeiculoNaoPertenceException
- VeiculoNaoExisteException
- VeiculoJaExisteException
- AluguerJaExisteException
- AluguerNaoExisteException

• AlugueresNaoExistemException
• AluguerNaoEsperaException
• UtilizadorNaoRegistadoException
• UtilizadorJaExisteException
• UtilizadorNaoExisteException
• NotificacaoNaoExisteException
• VeiculoNaoEncontradoException
• GestorVazioException
• CaixaNotificacoesVaziaException
• IOException
• ClassNotFoundException

Funcionalidades da aplicação

Ao iniciar a aplicação, surge o menu com todas as possibilidades, como demonstrado na imagem a baixo, que vão ser explicadas de seguida.



Menu Principal

- #### Carregar estado

  Funcionalidade adjacente que permite carregar de um ficheiro de texto toda a informação presente.

- Criar conta

  Esta é a primeira funcionalidade da aplicação onde se pode escolher se quer criar uma conta de cliente ou de proprietário, lendo todos os valores necessários para a sua inicialização. Caso escolha criar um proprietário é pedido um email, uma password, nome, nif e uma morada. Se escolher criar um cliente é pedido exatamente o mesmo com adição de um Ponto, onde a pessoa se encontra.

- Fazer LogIn

  Esta é a funcionalidade da aplicação onde confirmamos as credenciais do utilizador. É pedido a cada utilizador que seja introduzido o seu email e a sua password, variando o menu apresentado caso se trate de um empresa ou individual.

- Top 10 Clientes

  Aqui é mostrado o top 10 de clientes mais frequentes na aplicação. Sendo possível escolher se quer o top 10 em relação ao número de alugueres por cliente ou o número de quilómetros feitos por cada um.

Menu de Registo

- Criar conta de Cliente

  Esta opção permite criar uma conta de cliente, onde é pedido ao utilizador os dados necessários para o mesmo ser executado. Depois essa conta é guardada e pode ser usada sempre que necessário, sendo depois redirecionado para um menu próprio destinado a clientes.

- Criar conta de Proprietário

  Esta opção permite criar uma conta de proprietário, onde é pedido ao utilizador os dados necessários para o mesmo ser executado. Depois essa conta é guardada e pode ser usada sempre que necessário, sendo depois redirecionado para um menu próprio para proprietários.

Menu de Cliente

- Consultar Caixa Notificações

  Aqui podemos aceder à caixa de notificações do cliente anteriormente logado.

- Aluguer

  Se um cliente necessitar de um novo aluguer, tem esta opção que lhe dá os alugueres disponíveis de acordo com as suas necessidades, sendo possível enviar um pedido de aluguer ao proprietário do veículo.

- Históricos

  Histórico de alugueres feitos pelo cliente, sendo possível escolher se pretende o histórico total ou entre datas dadas pelo utilizador.

Menu de Proprietário

- Consultar Caixa Notificações

  Aqui podemos aceder à caixa de notificações do proprietário anteriormente logado.

- Alugueres

  Esta opção permite ao proprietário ver os alugueres dos seus veículos registados, onde é possível confirmar ou terminar os mesmos.

- Histórico

  Histórico de viaturas que foram alugados, podendo escolher se pretende navegar nesse histórico com as informações totais ou entre determinadas datas. Também é possível saber o total faturado por um certo veiculo registado.

- Definições Veículo

  Por aqui é possível entrar nas definições de veículos, onde pode registar novos veículos ou alterar algo sobre os veículos já registados.

Menu de Leitura

- Sim

  Aceitar leitura.

- Não

  Rejeitar leitura.

Menu de Veículos

- ### Registar Veículo

  Esta opção permite a um proprietário registar um novo veículo.

- ### Alterar o preço por km de um veículo

  Aqui é possível alterar o preço por km de um veículo para ser futuramente alugado.

- ### Abastecer um veículo

  Abastecer um veículo, pois não pode ser usado se tiver com o combustível abaixo dos 10% ou se não for suficiente para efetuar certos percursos.

- ### Visualizar veículos registados

  Lista de veículos já registados por um certo proprietário.

Listagem

Permite navegar por uma lista, de 2 em 2 elementos, sendo esse valor facilmente alterado se necessário. Assim facilita ao utilizador navegar por listas divididas em páginas.

Guardar estado da Aplicação

Durante o correr da aplicação, sempre que selecionada a opção sair no menu principal, os gestores usados como variáveis de instancia da classe VeiculoApp são guardados em ficheiro objeto com o nome do respetivo gestor e a extensão .ser. Na primeira vez que se inicia a aplicação tem que se selecionar a opção ‘carregar estado’ que vai carregar os gestores com o ficheiro de carregamento inicial. Nas seguintes utilizações isto já não é necessário, pois, quando a aplicação é terminada, o estado dos gestores é guardado nesse mesmo ficheiro objeto. Isto implica que o código não deve ser alterado nas utilizações após a primeira. Para que o código pudesse ser alterado teríamos que guardar os estados dos gestores em ficheiro de texto.

Introdução de novos tipos de viaturas

Para facilitar a adição de novos tipos de viaturas no futuro, resolvemos criar um Enum de TipoVeiculo, em que atualmente tem apenas o valor ‘Carro’. Desse modo, para adicionar um novo veiculo podemos apenas criar um novo valor para o tipo de veículo que desejamos, que, posteriormente, será chamada pela class veículo.

Conclusão

Terminada a realização do trabalho prático, chega o momento de refletir de analisar as dificuldades sentidas e os momentos de aprendizagem alcançados.

De forma geral, a realização do trabalho prático não decorreu como esperado, destacando-se a dificuldade sentida na implementação nas leituras e escritas nos ficheiros devido à pouca informação.

Embora com alguns contratempos, geraram-se alternativas de resolução ao problema anteriormente descrito, através da criação de uma aplicação que resolve tudo o que foi pedido.

Durante o desenvolvimento do trabalho fomentou-se um maior conhecimento sobre a programação orientada a objetos, conhecimento esse que só poderia ser reforçado e consolidado com o trabalho árduo enfrentado, ultrapassando todas as limitações e desventuras emergentes.