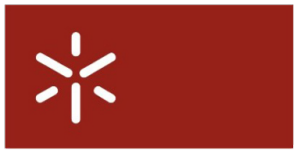

Programação Orientada a Objetos
Relatório

Andreia Alves Guilherme Miranda João Monteiro

1 Introdução

Este projecto, realizado para a cadeira de Programação Orientada a Objetos, tem como objetivo a criação de uma aplicação em java onde os utilizadores possam registar os seus ve´ıculos para depois poderem receber pedidos de transporte de clientes, também utilizadores da aplicação.

A aplicação fará a gestão dos dados de todos os utilizadores e veículuos para facilitar o aluger destes ultimos a quem necessitar de transporte.

A implementação da solução será segundo os principios da Programação Orientada aos Objetos (POO). Neste projeto são explorados mecanismos muito poderosos do JAVA como Hierarquia e Herança e Composiçao de Classes, quanto a POO preocupamo-nos essencialmente com o encapsulamento de dados.

2 Contextualização

Uma empresa pretende criar um serviço de aluguer de veiculo particulares pela internet, muito parecido com o serviço de aluguer de casas Airbnb.

Na aplicação desenvolvida, um proprietário de um automóvel poderá registar o seu veiculo na aplicação UMCarroJá! e este ser alugado por um cliente registado nessa mesma aplicação. Tal como nos serviços Uber e Airbnb após o cliente escolher o veiculo a alugar, o proprietário terá de aceitar (ou não) o aluguer pedido.

Para tal, um histórico de feedback de aluguer de clientes tem ser ser mantido, bem como do automóvel (para permitir aos clientes melhor escolherem o veiculo que desejam). Uma vez que os veículos são particulares, eles encontram-se estacionados na rua, e não concentrados numa agência de aluguer. Assim, a localização (coordenadas GPS) de um veiculo serão importantes para a escolha por parte de um potencial cliente. Por exemplo, um cliente pode desejar escolher o veiculo que está mais perto da sua localização.

Um veiculo tem ainda a indicação do preço que é cobrado por quilometro percorrido, a quantidade de combust´ıvel existente no carro (percentagem de gasolina e/ou bateria), o consumo médio por quilómetro, bem como informação geral do automóvel e do seu proprietário.

3 Motivação e Objetivos

A maior motivação para este trabalho foi a aprendizagem obtida sobre uma das linguagens de programação mais populares em todo o mundo.

Tendo em conta que o objetivo principal da nossa aplicação é o registo e consulta de utilizadores (clientes e proprietários), veículos, pedidos de alugueres e viagens realizadas, o desenvolvimento deste projeto passa por tentar implementar métodos simples e eficientes como também expor de maneira organizada e intuitiva para o utilizador todas as funcionalidades disponíveis e consultas pretendidas.

Para além disto, e como este projeto simula uma interação com utilizadores do mundo real, outro objetivo a cumprir é conseguir pôr em prática as especificações requeridas para esta aplicação.

4 Descrição das classes

A classe Coordenadas contém as variáveis coordX(double) e coordY(double) e é usada para representar uma localização. Os métodos incluidos nesta classe são só os basicos (sets,gets,etc...).

A classe Viagem contém as variáveis inicio(Coordenadas), fim(Coordenadas), nif(int), matricula(String) e preco(double). Está classe contém todas as informações de uma viagem já realisada. Os métodos incluidos nesta classe são só os basicos.

A classe Pedido contém as variáveis loc(Coordenadas), matricula(String), locDest(Coordenadas), data(GregorianCalendar), estado(int), nif(int), combustivel(String). Está classe contém todas as informações de um pedido, incluíndo se já foi tratado ou se aínda está à espera de ser aceite ou recusado. Os métodos incluidos nesta classe são só os basicos.

Optou-se por uma classe abstrata para Veiculo pois, para além de permitir criar um "esqueleto" para suas subclasses (Combust´ıvel, Eletrico e Hibrido) reutilizando código, obriga também que as subclasses implementam vários métodos necessários ao funcionamento da aplicação. Isso faz com que a aplicação seja desde logo extensível no que toca a novas classes de veiculos. As variáveis de instância são o tipo(String), matricula(String), marca(String), velocidade(int), precoBase(double), nif do dono(int), consumo(double), autonomia(int), localizacao(Coordenadas), histórico de viagens (ArrayList de Viagens), todas as classificações recebidas (ArrayList de Integer) e RESRVA (static int).

Para além dos métodos básicos (clone, equals, etc.), a classe também contém os seguintes métodos:

- public boolean viaja(): so viaja se autonomia maior ou igual que 50 km
- public void addClassificacao(int val): adiciona uma classificação entre 0 a 100 a um veiculo
- public double MedClassificacoes(): media das classificaçoes
- public String imprimeHistorial(): imprime o historial de viagens do veiculo no ecrã

Para além dos métodos básicos, as subclasses da classe abstrata Veiculo têm como caracter´ıstica:

- Classe Combustivel: contém apenas a variável int static deposito = 900, considerando a capacidade total do depósito para 900 km
- Classe Eletrico: contém apenas a variável int static bateria = 800, considerando a capacidade total da bateria para 800 km
- Classe Hibrido: contém aa variáveis int static bateria = 800 e int static deposito = 900, considerando a capacidade total da bateria para 800 km e do depósito para 900 km

A classe Utilizador é uma classe abstracta que contém as variáveis e os métodos basicos de qualquér utilizador do sistema. Estas variáveis são o nif(int), nome(String), email(String), password(String), morada(String) e data(GregorianCalendar).

A classe Cliente implementa a classe Utilizador e contém as variáveis adicionais localização(Coordenadas) e listaViagems(ArrayListiViagem¿).

A classe Proprietario implementa a classe Utilizador e contém as variáveis adicionais listaAlugueres(ArrayListiViagem¿), veiculos(ArrayListiVeiculo¿), classificacoes(ArrayListiInteger¿) e listaPedidos(ArrayListiPedido¿). Os métodos incluidos nesta classe são métodos para adcionar e remover elementos das listas, um método para ir buscar os pedidos aínda por tratar e um método para is buscar a média das classificações.

A classe UmCarroJa(Main) é responsável por fazer o carregamento e escrita dos ficheiros.Além disso contem também os menus necessários há navegação do programa.
Os menus são:

- menuCarregarFicheiro: permite escolher o modo de inicialização;
- menuInicial: permite escolher fazer login como cliente ou proprietario;
- menuCliente: permite ao cliente acesso às suas funcionalidades (pesquisa,historico,alterar dados,comentar)
- menuPesquisa: permite o cliente escolher o tipo de pesquisa que pretende fazer;
- menuHistorico: permite ao cliente ver as viagens que já fez, ou os veiculos que já alugou;
- menuAlterar: permite alterar os dados do utilizador;
- menuProprietario: permite ao proprietario colocar um novo veiculo, sinalizar um veiculo como disponivel,abastecer um veiculo, alterar o preço por km e aceitar ou rejeitar pedidos em espera.

5 Estrutura da aplicação desenvolvida e decisões tomadas

5.1 Estrutura da aplicação

Na figura seguinte podemos ver graficamente a estrutura geral do projeto

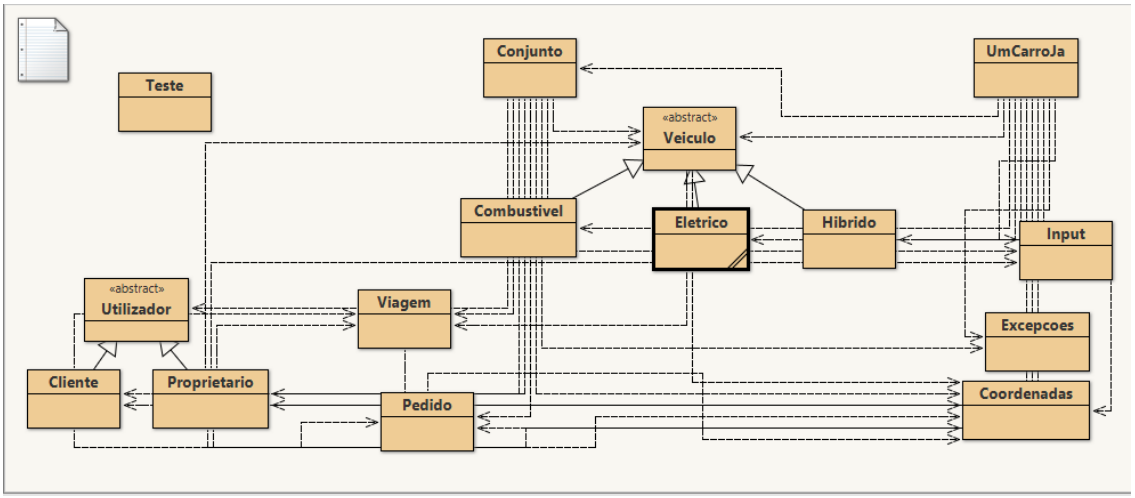

Figura 1: Hierarquia e dependências de classes

5.2 Decisões tomadas

Durante o desenvolvimento do projeto foram tomadas muitas decisões sobre o tipo de dados a representar e que tipos de dados se deveria usar para isto.

- A Classe Input, nada mais é do que a nossa implementação do Scanner. Tomámos essa decisão para assegurar, por exemplo, que quando é necessário um tipo de input (por exemplo inteiro) a aplicação só avança quando for introduzido o tipo esperado(se esperar por um inteiro e o user introduzir um char ele invalida e pede de novo o input). Os métodos da classe utilizados foram:

  - public static String lerString()
  - public static int lerInt()
  - public static double lerDouble()
  - public static float lerFloat()
  - public static boolean lerBoolean()
  - public static short lerShort()
  - public static char lerChar()

- A Classe Excepcoes foi implementada para o tratamento dos vários casos de excepcoes da aplicação. É uma extensão da classe Exception do JAVA
- A classe Conjunto foi implementada para facilitar a manipulacao dos treemaps. Disponibiliza uma classe para guardar todo o conjunto de informação requerida pelo sistema e da aplicação.
- Neste projeto utilizamos treemaps devido à falta de previsão do tamanho dos dados da aplicaçao. Sendo uma aplicação online não se sabe a sua popularidade e o tipo de disponibilidade de espaço dos utilizadores.

6 Discussão sobre a adição de novos tipos de viaturas

Sendo que a classe abstracta Veiculo tem como subclasses as classes Combust´ıvel, Eletrico e Hibrido, adicionar novos tipos de viaturas no sistema equivale a dispô-los em cada subclasse de acordo com o tipo de energia e adicionar mais uma variável de instância à classe Veiculo do tipo String para especificar o tipo de veiculo(exemplo: carro ou trotinete ou prancha).Ou seja, trotinetes eletronicos iriam fazer parte da subclasse Eletrico.

7 Conclusão

Após a conclusão deste projeto, podemos afirmar que adquirimos bastantes conhecimentos no que toca à Programação Orientada a Objetos. Fizemos uso das diversas capacidades da linguagem JAVA, incluindo o polimorfismo que se alcança com a abstração de classes.

Um dos objectivos foi a hierarquia das classes para assim haver reutilização do código e facilitar inserções de novas classes e subclasses.

Com o código feito desta maneira evita-se repetição de código, já que o que é comum a alguns tipos encontra-se na classe principal, por exemplo, a informação comum a todas os veículos encontra-se na classe Veiculo e o que é característico de cada tipo de veículo é especificado em cada subclasse. Também assim se torna mais fácil a leitura.

Para guardar informação utilizou-se coleções do tipo Map's, Set's e Array-List. Esta decisão foi bastante importante pra o grupo, pois uma boa escolha destas coleções permite um dimensionamento bastante superior do programa e tempos inferiores de execução.

Uma etapa realizada com sucesso foi a criação dos métodos que tornaram possível a utilização do programa. Com isso realizado, o programa possuí uma grande variedade de funcionalidades que o utilizador pode tirar proveito.