UM Carro já!

2018/2019

Universidade do Minho

Mestrado Integrado em Engenharia Informática

Programação Orientada a Objetos – Grupo 41

Filipa Santos (A83631), Hugo Cardoso (A85006), João Cunha (A8477)



ÍNDICE

1. INTRODUÇÃO

O projeto de Programação Orientada a Objetos deste ano consiste no desenvolvimento de um serviço de aluguer de veículos particulares, semelhante ao serviço de aluguer de casas *Airbnb*.

Há dois tipos de atores no sistema: clientes e proprietários. Na aplicação final, qualquer ator pode registar-se no sistema, fornecendo os seus dados, e aceder à sua conta através das credenciais.

O proprietário pode registar o(s) seu(s) veículo(s) no sistema, tornando-os disponíveis para aluguer. Dado que são particulares, estes encontram-se estacionados na rua e não concentrados numa agência, e é o cliente que se desloca ao veículo que requisita e não vice-versa.

O cliente escolhe um veículo a partir de uma variedade de opções – um veículo específico, o mais barato, o mais próximo, entre outros – e, caso este tenha autonomia suficiente para realizar a viagem – o cliente indica também o seu destino -, o pedido é enviado ao proprietário.

Durante esta escolha, o cliente tem acesso a toda a informação dos veículos e respetivos proprietários e, do mesmo jeito, estes têm acesso a toda a informação dos clientes durante a confirmação/recusa de um pedido. Todavia, cada perfil de utilizador consegue aceder apenas às informações e funcionalidades respetivas.

Caso o pedido seja aceite, a viagem é realizada e o aluguer registado. O custo da viagem é imputado no fim e a autonomia do automóvel naturalmente vê-se reduzida, bem como as posições do carro e do cliente atualizadas. Há também uma classificação do outro agente por parte de ambos.

O estado inicial da aplicação é carregado de um ficheiro de dados fornecido pelos docentes e, a partir daí, os posteriores estados são também guardados num outro ficheiro, para que seja possível retomar mais tarde a execução.

2. Classes iniciais

A execução deste projeto assenta sobre três classes iniciais que representam os tipos mais básicos do sistema: *Pessoa*, *Veiculo* e *Aluguer*. Sendo o aluguer a única unidade já "final" deste conjunto, é a partir das outras duas que se desenvolvem os agentes e elementos mais complexos do sistema, tendo estas classes apenas as variáveis comuns aos mesmos.

2.1. Classe Pessoa

A classe *Pessoa* possui as variáveis comuns aos dois tipos de agente do sistema: clientes e proprietários. Todos partilham a seguinte informação:

- Email;
- Nome;
- Password;
- Morada;
- Data de nascimento;
- Número de identificação fiscal (que identifica o utilizador);
- Lista de classificações atribuídas ao mesmo;
- Histórico de alugueres realizados;

2.1.1. Classe Cliente

O cliente é a pessoa que solicita e efetua o aluguer de um carro. Além da informação referida anteriormente, possui também:

- Localização (expressa em coordenadas x e y);
- Estado do aluguer (representado por um inteiro).

O cliente pode solicitar apenas um veículo de cada vez. Como tal, num determinado momento, o cliente pode não ter nenhum aluguer pendente, em qual caso é livre de requisitar um carro, pode ter um pedido pendente,

sendo que qualquer pedido adicional será rejeitado enquanto assim for, ou o seu pedido pode ter sido aceite/rejeitado. Estas circunstâncias são todas representadas pela variável associada ao estado do aluguer.

2.1.2. Classe Proprietario

A interação do proprietário com o programa resume-se fundamentalmente ao registo de carros e à aprovação/recusa de pedidos de aluguer, além de questões mais secundárias como abastecer os veículos e alterar, caso queira, o preço por quilómetro da viatura. Possui ainda portanto:

- Lista de carros registados no sistema;
- Lista de alugueres pendentes dos seus carros.

A lista de alugueres acaba então por operar como uma lista de espera.

2.2. Classe Veículo

Esta classe possui todas as informações relativas aos veículos implementados no sistema, bem como os respetivos construtores e métodos de instância:

- Marca;
- Matrícula (que identifica o veículo);
- Número de identificação fiscal do proprietário;
- Velocidade média por km;
- Preço base por km;
- Consumo de combustível/bateria por km;
- Autonomia atual;
- Autonomia máxima;
- Localização (expressa em coordenadas x e y);
- Lista de classificações atribuídas ao veículo;
- Lista de alugueres realizados.

Quanto a estes parâmetros, não há muito a salientar, sendo todos bastante claros. O propriétario usufrui da possibilidade de alterar o preço base por km do seu veículo, e são guardadas duas variáveis diferentes relativas à autonomia de maneira a facilitar o processo de reabastecimento da viatura.

2.2.1. Tipos de veículos

Existem três tipos diferentes de veículos: carros a gasóleo, híbridos e elétricos, cada um com a sua respetiva classe. Apesar de tudo, estes não se diferenciam em nada além do nome, sendo que o processo de consumo/abastecimento de combustível e bateria são idênticos.

2.3. Classe Aluguer

Num aluguer, move-se o cliente e o veículo de uma posição inicial para uma posição final, ficando o veículo de novo disponível nessa localização. O cliente é informado do autonomia do carro e do custo estimado da viagem e calcula-se também o tempo estimado. No fim, o cliente e o propriétario avaliam-se mutuamente, ficando com um registo relativo à viagem guardado nas suas áreas pessoais. Como tal, possui:

- Tempo estimado;
- Posição inicial (do veículo);
- Posição final;
- Preço estimado;
- Matrícula do veículo;
- Número de identificação fiscal do cliente/proprietário;
- Tipo de aluguer (mais barato, mais próximo, etc);
- Data do aluguer.

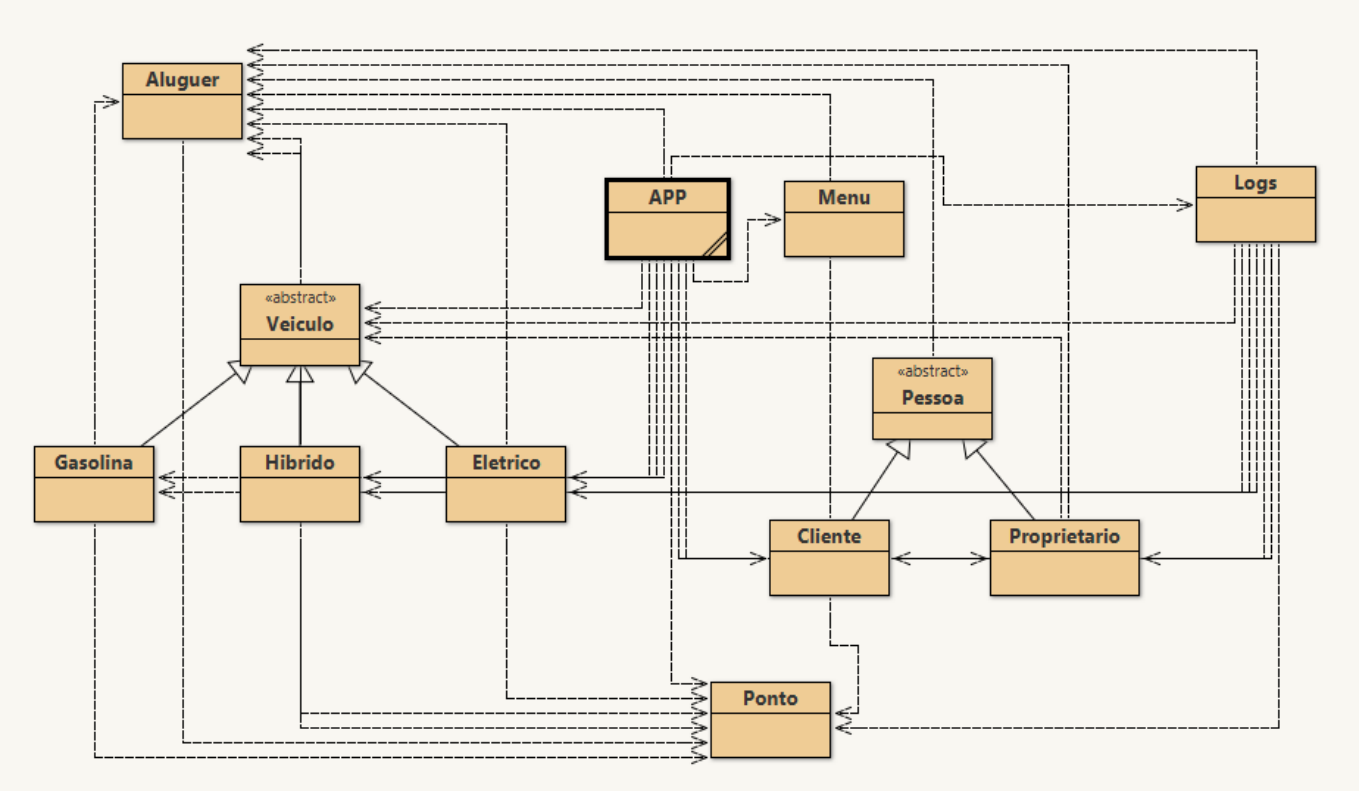
O tempo e o custo estimados são calculado segundo uma constante aleatória que varia entre certos limites e visa simular fatores de aleatoriedade na viagem, nomeadamente condições atmosféricas e outros condicionantes.

Como é o cliente que se desloca até ao veículo no ínicio, é a posição inicial deste que é registada e não vice-versa. Caso se trate do histórico de um cliente, o nif em questão é do proprietário, caso contrário é do cliente.

3. Estruturação principal

As classes explanadas anteriormente definem as estruturas básicas do programa, os dados, bem como todos os métodos necessários para os criar e alterar. Tendo o conteúdo do programa definido, resta apenas criar a estrutura em que estes serão usados, isto é, definir de que maneira o utilizador pode interagir com eles, o que abrange tanto onde este pode criar novos dados, como as operações que pode realizar com eles posteriormente.

Como tal, surgem três novas classes com este propósito: *Menu*, *Logs* e *APP* (aplicação), e o projeto adquire a seguinte arquitetura final:



3.1. Menu

Esta classe gere a apresentação dos menus de escolha ao cliente, bem como a leitura da escolha do mesmo. Consiste em:

- Lista de frases (strings) que constituem cada menu;
- Inteiro que corresponde à opção escolhida pelo utilizador.

Assim, o programa pode criar um menu com qualquer número de opções à escolha do utilizador, bastando fornecer estas mesmo ao construtor desta classe. O novo menu é depois executado, exibindo no ecrã o título e todas as alternativas fornecidas e atribuindo a cada uma um número, além de uma última que permite ao utilizador retroceder e guardar os dados introduzidos até então.

O utilizador digita então o número correspondente à operação que pretende executar e o programa recolhe essa resposta, apenas a aceitando se for válida.

Desta maneira, qualquer menu vai ter o seguinte formato:

```
 *** Menu Principal ***
1 - Iniciar Sessão
2 - Registar
3 - Total faturado por determinado veiculo, entre datas
4 - Listagem dos 10 clientes que mais utilizam a APP
0 - Sair e Guardar
Opção:
```

3.2. Logs

Este módulo trata da gestão dos dados do sistema. Possui três estruturas, *HashMaps*, de proprietários, clientes e veículos, que associam ao número de identificação fiscal/matrícula a correspondente unidade.

Assim, define-se aqui os métodos necessários à leitura do ficheiro fornecido pelos docentes, que possui os dados (gerados automaticamente) para carregamento inicial do sistema.

O programa acede ao ficheiro e lê-o linha por linha, distinguindo o tipo de unidade (proprietário, cliente, carro, aluguer ou classificação) em questão, criando uma variável desse tipo a partir de toda a informação fornecida e adicionando a mesma ao *HashMap* respetivo.

No fim deste processo, o programa possui um extensivo conjunto de dados sobre os quais pode operar. Além disso, são também definidos os métodos necessários para executar as operações possíveis do sistema, nomeadamente:

- Calcular o veículo mais próximo/barato (por tipo);
- Alterar a autonomia, estado de aluguer e posição do veículo;
- Alterar a posição do cliente no final da viagem;
- Adicionar alugueres e classificações aos históricos;
- Alterar o preço base por km de uma viatura;
- Calcular os carros que precisam de abastecimento e abastecê-los.

São ainda definidos métodos para preencher certos requisitos mais específicos do enunciado que não estão relacionados com a interação entre o proprietário e o cliente. Estes consistem em:

- Aceder ao histórico de alugueres no perfil do utilizador;
- Indicar o total faturado por uma viatura num determinado período;
- Determinar os 10 clientes que mais utilizam a aplicação (em número de vezes e em kms percorridos).

3.3. Aplicação

A aplicação trata da interação do utilizador com o programa, consistindo portanto na classe I/O (*intput/output*) do projeto.

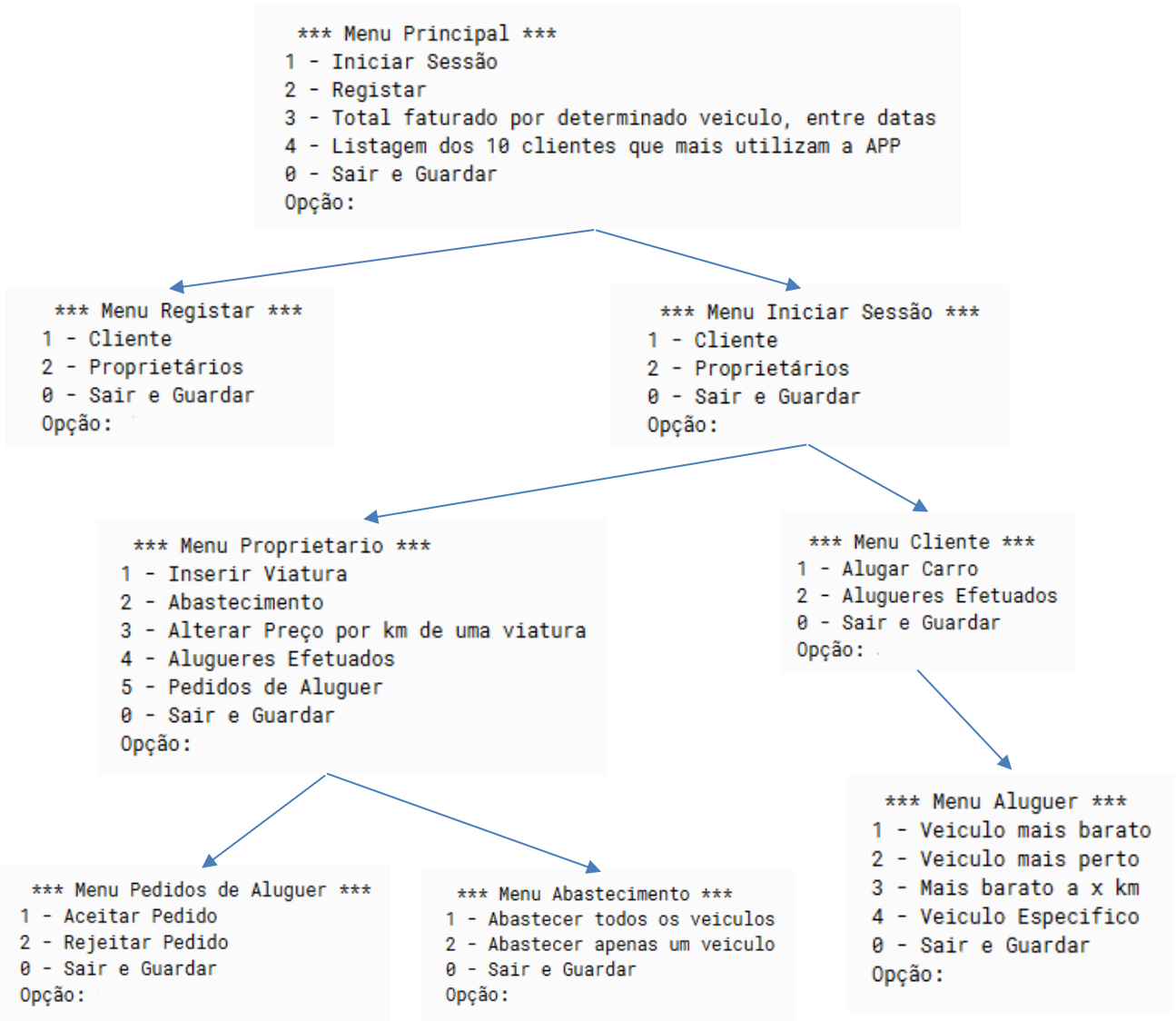
É responsável por guardar e carregar o estado atual do programa em formato binário de um ficheiro, de maneira a ser possível interromper a execução do programa e retomá-la mais tarde.

Cria todos os menus necessários (menu de cliente, de proprietário, de abastecimento, etc) para permitir ao utilizador usufruir do sistema, invocando o método correspondente a cada escolha realizada.

No menu principal, o utilizador pode registar-se no sistema ou aceder à sua conta, caso já o tenha feito, inserindo os seus dados em ambas as opções. Além disso, pode também consultar as características do sistema disponibilizadas.

Uma vez no seu perfil, o cliente/proprietário pode realizar qualquer uma das suas operações correspondentes. É ainda possível retroceder e navegar entre os menus à vontade, sendo que a execução do programa só termina quando se sai do menu principal.

Podemos de seguida observar os menus existentes:



4. Aspetos a melhorar

Não foi possível implementar todas as ferramentas pretendidas devido a uma falta de tempo e recursos, dado que o grupo se encontra numa época complicada de avaliações. Porém, fica aqui um pequeno registo do que se tencionava fazer.

4.1. Fatores de aleatoriedade

Em várias vertentes do projeto é possível implementar fatores de aleatoriedade de maneira a tornar todo o ecossistema mais dinâmico e imprevisível, fazendo dele portanto uma simulação mais realista.

Entende-se por fatores aleatórios, por exemplo, a fiabilidade do carro, a destreza do condutor e as condições metereológicas, isto é, condições que podem variar de situação para situação e que implicam uma flutuação no custo e tempo da viagem.

Neste caso, foi implementada aleatoriedade no cálculo dos mesmos (custo e tempo) mas não foram especificados os motivos destas flutuações. É portanto uma aleatoridade bastante geral.

Para melhorar este aspeto, poder-se-ia introduzir no projeto novas variáveis como o clima, a destreza do condutor e o estado do veículo que influenciariam o tempo e custo da viagem segundo diferentes rácios.

4.2. Exceções

Ao longo do projeto há inúmeras situações em que se benificia do uso de exceções para evitar comportamento não definido da parte do programa. Contudo, devido à falta de tempo, decidiu-se dar prioridade ao acabamento do programa, abrangendo o máximo possível de requisitos do enunciado, tratando a robustez do mesmo como uma questão mais secundária.

Devido a isto, o programa não é o mais robusto e consistente, sendo que em vários sítios depende da introdução correta dos dados ou da concretização dos resultados esperados do programa, caso contrário exibe comportamento não definido.

Contudo, em geral, são abordadas bastantes exceções ao longo do projeto e salvaguardados vários casos excecionais, tanto da parte de introdução de dados como da realização de operações, pelo que, apesar de poder ser mais robusto, não é também muito frágil.

4.3. Conteúdo extra

O grupo teve ainda várias ideias que não eram especificadas no enunciado mas que gostaria de implementar. Entre estas, destacam-se:

- Sistemas de promoções, que consistiriam em selecionar um certo número de veículos e reduzir o seu preço durante um determinado período de tempo, o que alteraria a dinâmica do sistema no que toca aos carros mais procurados e à disponibilidade das viaturas.
- Sistemas de clientes premium, que atribuiriam vantagens a clientes que cumprissem certos requisitos, nomeadamente descontos nos custos das viagens e a deslocação da viatura ao cliente aquando da solicitação em vez do inverso, entre outros.

Tinha-se ainda a intenção de implementar novos tipos de viaturas como é referido no enunciado, por exemplo bicicletas, trotinetes e pranchas de surf elétricas. Apesar de a introdução destas novas entidades no sistema ser fácil, devido à extensibilidade da classe *Veiculo*, julgou-se redundante a mesma se não fosse acompanhada de diferenças mais significativas, pelo que se decidiu não seguir em frente com a ideia (devido à falta de tempo).

5. Conclusão

Em suma, o grupo está satisfeito com o resultado adquirido. Foi um projeto desafiante que acabou por se provar muito mais complexo do que se pensava inicialmente, dado que quanto mais se aprofundava o mesmo, mais variáveis e problemas novos que não se tinha considerado antes surgiam.

O grupo considera este projeto uma experiência valiosa que permitiu não só aprofundar os conhecimentos acerca do paradigma da Programação Orientada a Objetos e da linguagem Java em si, que é uma das linguagens de programação mais usada no mercado, como ter um primeiro contacto com a estruturação de projetos neste ramo da programação, que certamente será muito necessário no futuro.

O grupo termina assim este relatório agradecendo aos docentes tanto pela boa exposição e explicação de conceitos nas aulas, que se provou fulcral para o desenvolvimento do projeto, como pela constante disponibilidade em relação aos alunos ao longo do desenvolvimento do projeto.