Universidade do Minho

Relatório – POO

MIEI-2º Ano- 2º semestre

UM Carro já!

Grupo 6

Catarina Gil
A85266

Margarida Campos
A85166

Tânia Rocha
A85176

ÍNDICE

1. Introdução

Este projeto foi proposto pelos docentes da unidade curricular Programação Orientada aos Objetos e tem como principal objetivo a realização de uma aplicação de serviços de aluguer de veículos.

Neste trabalho aplicamos vários conhecimentos obtidos durante o semestre nesta unidade curricular tais como a linguagem de programação Java, encapsulamento, herança, entre outros.

2. Manual de utilização

```
1:Ler Carregamento inicial
2:Ler ultimo estado
3:Ir para a App
0:Sair e guardar o atual estado
Escolha a opção:
```

Este é o menu principal da aplicação. Para escolher a opção desejada basta clicar na tecla correspondente ao número da opção. Caso escolha 1 é carregado o ficheiro de logs fornecido pelos docentes; a opção 2 corresponde a carregar o ficheiro objeto criado anteriormente; a opção 3 permite entrar na aplicação e, por fim, a escolha de 0 possibilita a saída do programa.

```
******UMCarroJa******
1:Iniciar Sessao
2:Criar Conta
3:Rankings
0:Sair
Escolha a opção:
```

Após escolher a opção número 3, é possível visualizar o menu principal da aplicação (imagem de cima). A partir deste menu poderão ser criadas contas ou iniciadas sessões. O início das sessões é efetuado recorrendo ao e-mail do utilizador e a uma password que lhe corresponde. No caso do ficheiro de logs fornecido, a palavra-passe coincide com o NIF do utilizador.

Ao clicar em “Rankings” são apresentados os 10 clientes que, até ao momento, utilizaram mais a app e os que percorreram um maior número de quilómetros.

Em seguida são apresentados os menus dos clientes e dos proprietários:

```
***MenuCliente***
1:Perfil
2.Alugar Carro
0:Logout
Escolha a opção:
```

```
***MenuProprietario***
1:Perfil
2:Registar novo carro
3:Ver alugueres propostos
4:Alterar preço de um carro
0:Logout
Escolha a opção:
```

Em ambos os menus existe a opção de visualizar o perfil. Este perfil mostra as informações básicas de cada utilizador bem como a sua classificação, o histórico de alugueres até ao momento e ainda, no menu de proprietário, a opção de ver os carros que este possui.

As classificações apresentadas consistem numa junção dos valores do ficheiro de carregamento inicial com valores aleatórios que são criados após cada aluguer. É exibida no final a média destes valores.

No menu do cliente é possível escolher alugar um carro. No momento do aluguer são solicitadas as coordenadas para onde o cliente se quer dirigir e apresentadas as diversas possibilidades de o proceder.

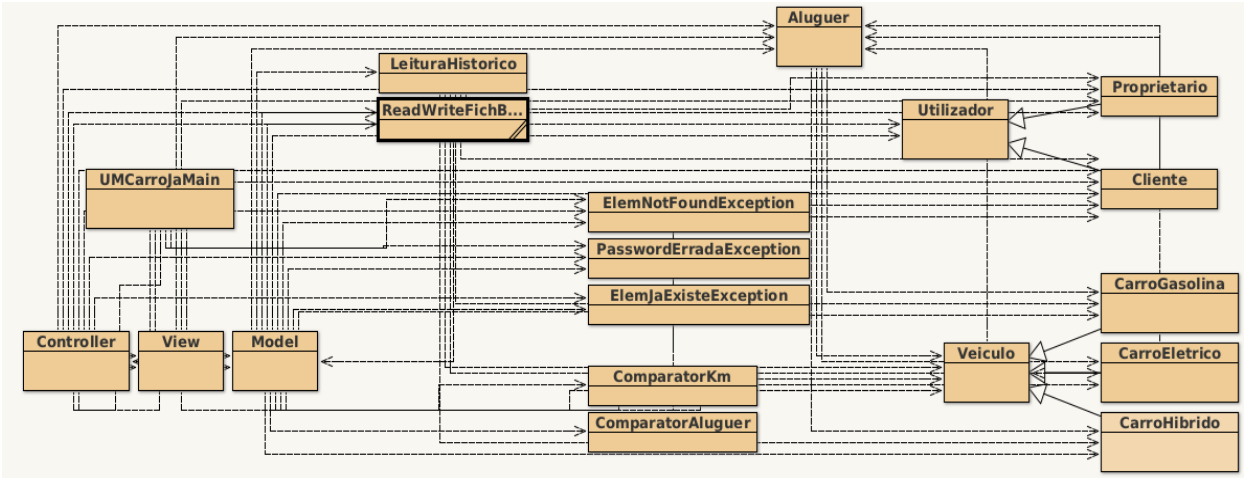
```
******MenuAlugarVeiculo******
Coordenadas x e y para onde vai:
20 -39

*Escolher filtros:*
1:Carro mais barato
2: Carro mais barato dentro de uma determinada distancia
3: Carro mais proximo
4:Escolher tipo de Carro:
5:Carro em especifico
0:Sair
Escolha a opção:
```

Nesta etapa optamos por tentar achar sempre um aluguer para o cliente, ou seja, caso ele escolha o filtro “Carro mais barato” o programa vai achá-lo e caso ele não possua autonomia suficiente para a viagem o programa vai escolher o 2º carro mais barato e assim sucessivamente.
Assim, após a escolha do filtro são mostrados ao cliente o carro e o proprietário capazes de realizar o aluguer proposto assim como o preço, as condições da estrada e a duração da viagem, podendo o cliente prosseguir ou rejeitar a proposta.

As condições da estrada são criadas a partir de um fator aleatório. Existem 3 possíveis condições: estrada molhada, estrada com neve e condições favoráveis.
Caso o cliente pretenda prosseguir com o aluguer terá de esperar que o proprietário aceite a proposta.

O proprietário poderá ver e aceitar ou rejeitar os alugueres propostos carregando na opção 3 do seu menu. Existe também a possibilidade de adicionar novos carros, opção 2, e de alterar o preço por quilómetro, opção 4.

3. Arquitetura das Classes



As duas classes principais são o Utilizador e o Veículo. A partir da classe utilizador especificamos as subclasses Cliente e Proprietário e a partir da classe Veículo especificamos os três tipos de carros: Gasolina, Hibrido e Elétrico.

```
private String email;
private String nif;
private String nome;
private String password;
private String morada;
```

Nas subclasses Proprietário e Cliente apenas foram adicionadas variáveis de instância e métodos necessários para cada um.

Na do Proprietário foram acrescentadas 4 variáveis: uma lista de veículos, que corresponde ao conjunto de veículos de um dado proprietário, dois Sets que armazenam a informação dos alugueres, um dos alugueres propostos e outro dos realizados e por fim uma lista que guarda as classificações dadas ao proprietário.

```
private List<Veiculo> carros;
private Set<Aluguer> aluguerP;
private Set<Aluguer> aluguerF;
private List<Integer> classificacaoP;
```

Na classe Cliente foram criadas as seguintes variáveis: um Point que indica a localização de um cliente, um Set que guarda os alugueres realizados, uma lista que guarda as classificações dadas a um cliente e, finalmente, um double que indica o número de quilómetros percorridos.

```
private Point.Double localizacao;
private Set<Aluguer> aluguer;
private List<Integer> classificacaoC;
private double kms;
```

A classe Veículo tem presente variáveis que se aplicam aos mais variados veículos, podendo esta ser usada como superclasse em muitos outros casos (como trotinetas, autocarros, pranchas e muito mais).

```
private String matricula;
private String marca;
private double velocidade;
private double preco;
private Point.Double localizacao;
private List<Integer> classificacaoV;
private Set<Aluguer> aluguer;
```

Como subclasses desta classe principal temos, portanto, os 3 tipos de carros apresentados anteriormente. As diferenças nestes carros são, nomeadamente, o consumo e autonomia. São estas as variáveis de instância criadas em cada uma das classes.

Houve a necessidade de criar uma classe Aluguer que tem presente todas as variáveis de instância e métodos necessários para a criação de um aluguer. É nesta classe que foram elaborados os métodos correspondentes a cada filtro apresentados no Manual de utilização.

```
private String cliente;
private String proprietario;
private String carro;
private Point.Double fim;
private Point.Double inicio;
private Double dist;
private Double preco;
private Double duracao;
private String tempoM;
private Double dPe;
```

Existem duas classes que permitem a leitura e a escrita em ficheiros. Uma, a LeituraHistorico, lê e guarda a informação do ficheiro de carregamento inicial dos logs. Outra, a ReadWriteFichBinario lê, escreve e guarda tudo em ficheiro objeto. Isto faz com que ao sairmos do programa todos os dados fiquem guardados e não haja perda de informação.

Como auxílio ao desenvolvimento dos rankings dos 10 clientes que mais quilómetros percorreram e dos que mais utilizaram a aplicação foram criadas duas classes de comparadores. Estas apenas contêm um método que ordena os clientes de acordo com o pretendido.

Para além de todas estas classes, foi ainda necessário a elaboração de classes de exceção. O uso destas exceções permite detetar e tratar os erros que possam acontecer no programa. Quando algo de errado se passa na aplicação, em vez de o programa ser encerrado, é enviada uma mensagem ao utilizador com o erro.

Por fim, foi necessária a implementação da arquitetura MVC (Model-View-Controller). Esta arquitetura consiste em dividir a aplicação em três partes que se conectam.

4.Conclusão

Concluindo, neste projeto conseguimos pôr em prática grande parte dos conhecimentos adquiridos nas aulas práticas e teóricas desta unidade curricular.

Este trabalho ajudou-nos a perceber que mecanismos de herança, de encapsulamento, entre outros são bastante importantes no desenvolvimento de aplicações permitindo com que estas fiquem mais eficientes.

Num futuro próximo implementaríamos a possibilidade de um maior número de predefinições para a escolha caracterizada do veículo (como por exemplo escolher o veículo mais rápido).

Para uma maior aproximação à realidade adicionaríamos também um relógio virtual.