UNIVERSIDADE AUTÓNOMA DE LISBOA

ESTAFETA - Gestão de Frotas

Trabalho de Algoritmos e Estruturas de Dados

Licenciatura em Engenharia Informática – Ano Letivo 2018/2019

Diogo Mendes - 30003865

Bruno Silva - 30003696

Turma A

Introdução

O Estafeta tem como objectivo principal desenvolver um programa na Linguagem Python e aplicar conhecimentos adquiridos na unidade curricular sobre o Paradigma Orientado por Objectos. Estes conceitos aplicam-se num contexto de gestão de uma frota que considera viaturas, condutores, clientes e entregas.

O Paradigma Orientado por Objectos (POO) é um paradigma de programação baseado no conceito de "objectos" que podem conter informação, na forma de campos (também conhecidos como atributos) e código, na forma de procedimentos (também conhecidos como métodos). Uma característica dos objectos é a capacidade de aceder e alterar a informação à qual os objectos estão associados. No POO os programas são projetados criando vários objectos que interagem uns com os outros. As Linguagens de POO são diversas, mas os mais populares, como é o caso do Python utilizado neste projecto, são baseados em classes, ou seja, os objectos são instâncias de classes, o que determina o tipo do objecto.

Estruturas de Dados Utilizados

Os dados neste trabalho apresentam-se em:

- Listas – um conjunto de valores numa coleção sequencial em que cada valor está identificado por um índice, os valores dentro de uma lista são chamados dos seus elementos, os valores podem ser de diversos tipos;

- Classes – um modelo programado para criar objectos, providenciando os valores iniciais dos atributos e a implementação de comportamentes (métodos)

- Objectos – uma instância concreta de uma classe onde os objectos podem ser uma combinação de varáveis, funções e estruturas de dados

- Filas – um tipo abstrato de informação em que as entidades são mantidas em ordem e as operações principais são a de *enqueue* (adicionar um elemento no final da fila) e *dequeue* (remover o elemento do início da fila), isto implica uma estrutura First-In-First-Out em que o primeiro elemento a ser adicionado é o primeiro a ser removido.

Descrição Sumária do Código

Este programa está dividido em dois ficheiros python, o ficheiro classes.py e o ficheiro menus.py, em que o primeiro contem todas as declarações de classes necessárias para as funcionalidades desejadas e o ficheiro menus.py todos os outros conjuntos de instruções.

Ficheiro Classes.py:

No ficheiro classes.py começamos por declarar a classe Viatura com os atributos matrícula e capacidade, e com a função modifica_Viatura(), que permite alterar os atributos de qualquer objecto desta classe, declaramos três outras classes, todas hierarquicamente abaixo de Viatura – "Camião", "Automovel" e "Mota" - que são simplesmente subcategorias de Viatura cuja única diferença é a carga máxima que podem transportar, também herdado a função modifica_Viatura().

Em seguida declaramos mais três classes, "Condutor", "Cliente" e "Entrega", que serão utilizados para criar objectos que transportam informações sobre os condutores, os clientes e as entregas respetivamente, sendo que todas estas classes também têm uma função que permite alterar os atributos de um objecto já instanciado.

Finalmente criamos as duas classes finais "Programa", que gera listas necessárias aos algoritmos utilizados para fazer a gestão da frota e "Queue" que utilizamos para criar a estrutura de dados abstrata de uma fila também necessária aos algoritmos utilizados na gestão da forta.

O método __str__ é chamado quando realizamos um print de uma classe, assim se em cada classe o alterarmos explicitamente podemos personalizar a visualização de um objecto em qualquer classe.

Ficheiro Menus.py:

Ao longo de todo este ficheiro realiza-se uma utilização extensiva das keywords try/except do python, keywords que permitem correr um pedaço de código (except) caso a parte principal dê erro (try), o que permite, no caso dos menus, simplesmente recomeçar no menu atual, ao invés de o programa deixar de funcionar, caso haja algum erro durante o processo de escolha.

Neste ficheiro começamos por importar classes (ou seja, o ficheiro classes.py descrito anteriormente) para que este novo ficheiro o possa utilizar, importamos o pickle, um módulo de python utilizado para guardar ficheiros que vamos utilizar para guardar os nossos dados entre utilizações, e definimos uma função separador() que utilizamos ao longo do programa para separar os diversos menus de forma mais clara.

No corpo do programa mais propriamente dito começamos por definir o menu inicial que permite navegar para os submenus veículos, entregas, condutores e clientes, e também enviar as entregas e ver o conteúdo das filas usadas no algoritmo de envio. Os submenus seguem todos a mesma estrutura, contêm uma opção para criar um novo objecto do tipo desejado, uma opção para alterar um objecto existente (utilizando as funções modificar já existentes nas classes), uma opção para ver os objectos desse tipo que já existem e uma opção para retornar ao menu inicial.

A função ver_veículo() é utilizada para para ver as viaturas existentes permitindo ao utilizado escolher qual dos 3 tipos de viatura quer visualizar, a escolha é realizada via teclado, realizada a escolha afixa-se no ecrã a lista de objectos do tipo desejado através de um ciclo for.

A função adiciona_veículo() é utilizada para criar novas viaturas no menu de veículos() do tipo à escolha do utilizador. O utilizador efetua a sua escolha e insere a matrícula da nova viatura e o algoritmo utiliza esta escolha para determinar o segundo atributo da classe, adicionando o novo objecto à lista e fila correspondentes retornando à ao submenu veículos.

Na função adiciona_condutor(), utilizada no menu condutores() pedimos a entrada via teclado dos atributos do novo condutor e depois faz-se recurso à função split, que permite dividir uma string num conjunto de strings (o argumento da função é o caracter que vai ser utilizado como local da divisão, neste caso uma vírgula) e obtemos assim um array de strings. Criamos o novo objecto “Condutor” (o * é um *arg que facilita o código não necessitando de fornecer à função de criação de classe cada argumento explicitamente). Finalmente pede-se a escolha via teclado do tipo de condutor a criar, colocando-o na fila e lista apropriada. Retorna-se ao submenu condutores.

A função adiciona_entregas(), funciona de forma análoga à função anterior mas de forma adaptada ao contexto das entregas.

A função ver_objectos(), utilizada nos submenus condutores, clientes e entregas, é responsável por mostrar os objectos de um desses 3 tipos. Esta função tem um parâmetro (tipo) que é utilizado para a adaptar a cada um dos 3 contextos. Retorna-se ao menu inicial.

As funções modifica_clientes(), modifica_condutores(), modifica_veiculo() e modifica_entrega() funcionam de forma análoga, começa-se por afixar os objectos do tipo respetivo, permite-se a escolha do objecto a alterar por parte do utilizador, recebe-se a entrada dos novos dados e aplica-se a função modifica (que todas as classes têm) para alterar o objecto desejado para os novos atributos.

A função enviar() faz recurso a 3 ciclos while, um para cada fila de entregas. Cada ciclo está em funcionamento enquanto existirem entregas na respetiva fila, em cada iteração verifica-se se existem motas e condutores disponíveis procedendo-se assim ao envio da entrega, caso contrário o ciclo é parado pois não é possível enviar entregas sem veículo ou sem condutor. O envio da entrega começa por tirar um objecto de cada fila

(um veículo, um condutor e uma entrega) para variáveis, calcula a capacidade restante no veículo e em seguida afixa no ecrã uma mensagem informando sobre o envio. Retorna-se ao menu inicial.

A função filas imprime no ecrã o conteúdo de todas as filas utilizadas através de dois ciclos for e retorna ao menu inicial. Esta função existe como um auxílio à avaliação.

Finalmente na main do programa criamos todas as filas necessárias (gerando objectos do tipo fila), criamos uma série de objectos (para auxílio à avaliação) e terminamos com a chamada da função do menu inicial que dá início a todo o programa.

Manual do Utilizador

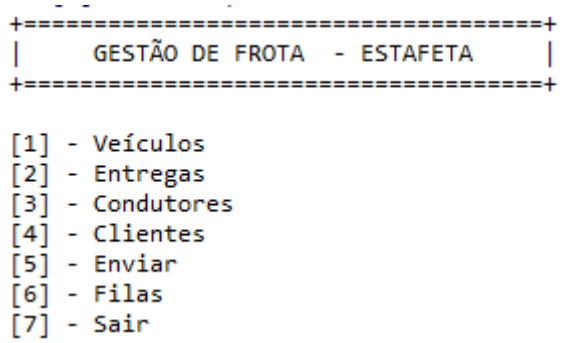

Figura 1 - Menu Inicial

Na figura 1 está o menu inicial do Estafeta, pode selecionar qualquer uma das opções inserindo o número correspondente via teclado. Da primeira à quarta opção pode aceder aos submenus de veículos, entregas, condutores e clientes da frota. Na quinta opção pode enviar todas as entregas por enviar (necessita da existência de veículos e condutores que possam levar essas entregas). Na sexta opção pode ver todos os componentes da frota que estão em fila de espera. Na sétiima opção pode terminar o programa.

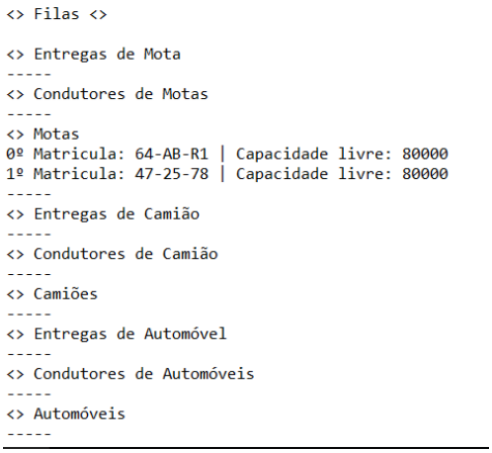

Figura 2 - Exemplo da opção Filas

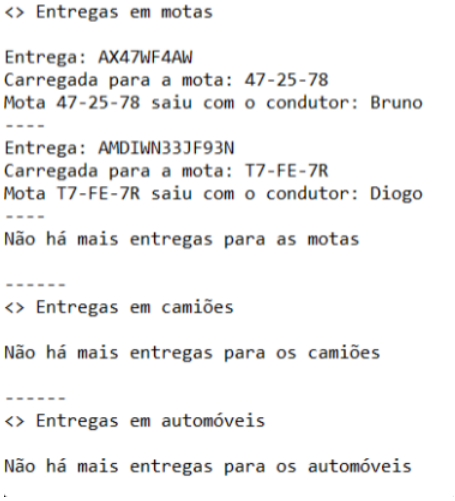

Figura 3 - Exemplo da opção Enviar

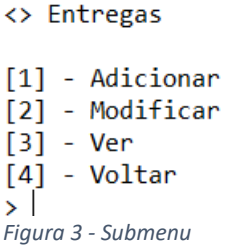

Figura 3 - Submenu Entregas

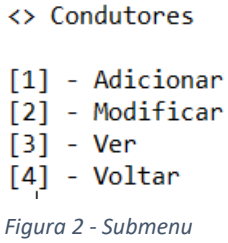

Figura 2 - Submenu Condutores

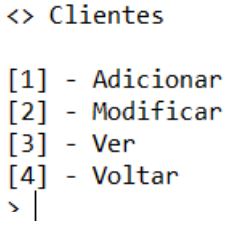

Figura 4 - Submenu Clientes

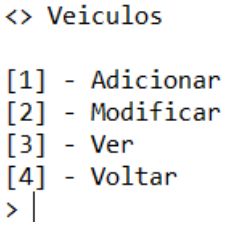

Figura 1 - Submenu Veículos

Como pode observar nas figuras acima os 4 submenus têm uma estrutura similar, da mesma maneira que se interagiu com o menu inicial também aqui se utiliza o teclado para aceder a cada operação:

- A primeira opção permite-lhe adicionar uma entidade nova do tipo desse menu. Em Veículos e Condutores deve primeiro indicar que categoria deseja inserir no sistema e depois adicionar os dados respetivos à nova entidade. Utilize sempre virgulas para separar cada dado;

- A segunda opção permite-lhe selecionar uma entidade já criada e modificá-la segundo as suas necessidades. Só necessita de escolher da lista que lhe é apresentada o que deseja modificar. Se inserir X pode cancelar esta operação;

```
=========================
<> Adicionar veiculo

[1] - Mota
[2] - Camião
[3] - Automóvel
[4] - Voltar
> 1

Matricula: AE-D8-4W
Veiculo com matricula {AE-D8-4W} adicionado
-------------------------
```

Figura 5 - Exemplo de adição de um veículo

- A terceira opção permite-lhe ver o histórico de entidades;
- A quarta opção permite-lhe voltar ao menu inicial.

Conclusão

Concluindo, a aplicação do Paradigma Orientado por Objecto ao contexto da gestão de uma frota foi bem-sucedido e é possível afirmar que todas as funcionalidades requeridas nesta primeira fase foram cumpridas.

O projecto em python foi realizado por ambos, numa fase inicial de design e planeamento, mas o programa em python foi maioritariamente realizado pelo Bruno enquanto que o Relatório foi maioritariamente realizado pelo Diogo.

Bibliografia

- https://en.wikipedia.org/wiki/Object-oriented_programming 05/05/2019
- http://interactivepython.org/courselib/static/thinkcspy/Lists/intro-Lists.html 05/05/19
- https://en.wikipedia.org/wiki/Class_(computer_programming) 05/05/19
- https://en.wikipedia.org/wiki/Object_(computer_science) 05/05/19
- https://en.wikipedia.org/wiki/Queue_(abstract_data_type) 05/05/19