ESCOLA
SUPERIOR
DE MEDIA
ARTES
E DESIGN
POLITÉCNICO
DO PORTO

ALP

PROVA DE AVALIAÇÃO
ATIVIDADE LETIVA

Tecnologias e Sistemas de Informação para a Web			
CURSO			
2022/2023	2022/02/22	16h30	2h
ANO LETIVO	DATA	HORA	DURAÇÃO
Algoritmia e Estruturas de Dados			1º ano
UNIDADE CURRICULAR			ANO
Mário Paulo Teixeira Pinto			Normal - AD
DOCENTE			ÉPOCA

Observações:

- O Teste é individual e de consulta. Poderá haver lugar a uma prova oral tendo em vista esclarecer dúvidas na resolução dos exercícios, com impacto na avaliação do exame
- Resolva os exercícios recorrendo à linguagem Python

Instruções:

1. Guarde os exercícios resolvidos nessa pasta.
2. No final do teste submeta os exercícios resolvidos no moodle, no objeto *SUBMETER TESTE*.

Parte I (40%)

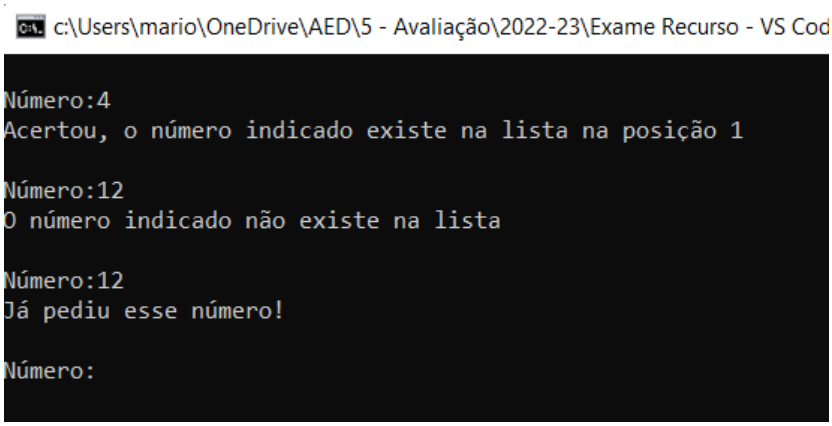
- Crie um ficheiro com a designação ***EX1_numeroAluno.py*** em que **numeroAluno** é o seu número de aluno na ESMAD.
- No início do ficheiro insira o seu número e nome, sob a forma de comentários, como no exemplo abaixo

```
1   # Numero:4029999
2   # Nome: Maria Manuela Maurícia
3
```

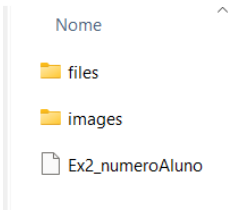
Crie um pequeno programa em ambiente de ConsoleApplication, em que o objetivo é o de gerar uma lista de 10 números aleatórios não repetidos, tendo o utilizador que adivinhar esses números.

Assim, o sue programa deve:

- Criar uma função **preencheLista** que vai preencher uma lista com 10 números, inteiros e aleatórios, entre 1 e 20. A função deve devolver a lista com os 10 números para o corpo do programa
- O seu programa deve ir pedindo sucessivamente ao utilizador que insira um número (entre 1 e 20). Deve contemplar uma estrutura de exceções para validar se o utilizador inseriu um valor numérico, e se este se situa efetivamente entre 1 e 20. Caso contrário, deve surgir uma mensagem de erro ao utilizador ilustrativa do erro em causa.
- Se o número inserido pelo utilizador já foi inserido anteriormente (isto é, se for repetido), deve surgir uma mensagem de erro do género "**já indicou esse número!!**";
- Caso contrário, deve invocar uma função **existeLista** que devolve um valor booleano: **True**, no caso de o número indicado existir efetivamente na lista de números aleatórios; **False**, caso não existe na lista.
- Quando o número existe na lista deve ser impressa a mensagem **"Acertou, o número indicado existe na lista na posição X**" onde X é a posição na lista dos números gerados aleatoriamente . Se não existe na lista, deve surgir a mensagem "**O número indicado não existe**"
- O jogo termina quando o utilizador acertar nos 10 números.



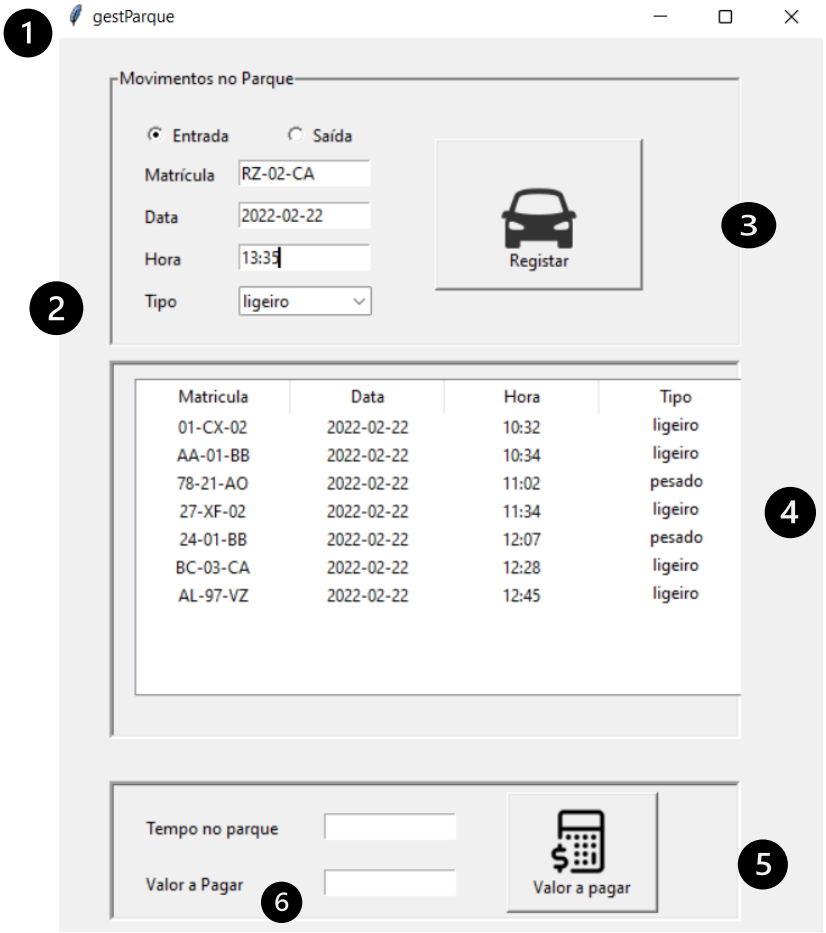
Parte II (60%)



- Descarregue do Moodle o ficheiro **ExameRecurso**.
- Descompacte a pasta. Deve obter a seguir estrutura:
- Altere a designação do ficheiro ***EX2_numeroAluno*** em que **numeroAluno** é o seu número de aluno na ESMAD.
- No início do ficheiro insira o seu número e nome, sob a forma de comentários, como no exemplo abaixo

```
1   # Numero:4029999
2   # Nome: Maria Manuela Maurícia
3
```

Pretende-se desenvolver uma App para gerir um pequeno parque de estacionamento, com uma *interface* semelhante à abaixo apresentada. A generalidade da *interface* encontra-se já implementada no ficheiro .py disponibilizado.

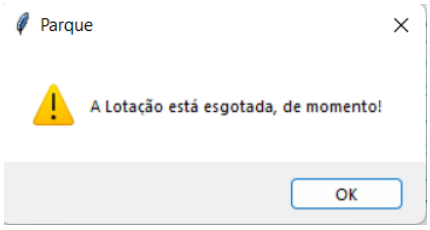


1. No arranque da aplicação deve invocar uma função ***lerOcupacao()***, com o objetivo de ler o conteúdo do ficheiro ***parque.txt*** e o renderizar na TreeView (ocupação inicial do parque).

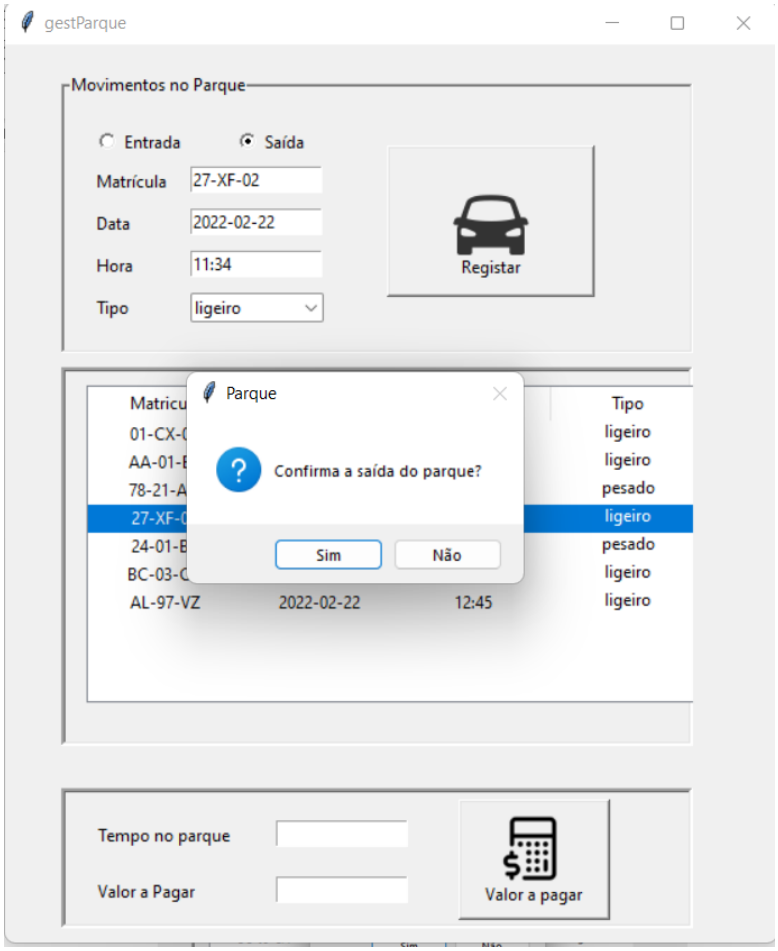
2. A Combobox relativa ao tipo de veículos deve ser preenchida a partir do conteúdo do ficheiro **tipoveiculos.txt.** A Combobox deve apresentar, por defeito, a opção ***ligeiro*** como na imagem acima.

3. O Button **Registar** deve incluir a imagem **car.png**, como se apresenta acima.
   Ao acrescentar a imagem ao Button, defina-o com width=140, height=100

4. Ao clicar no Button **Registar**:
   - Se se tratar de uma entrada (ver Radiobutton ativo) deve acrescentar os dados do carro à Treeview. Contudo, note-se que não podem estar mais de 10 carros no Parque. Caso já esteja lotado deve surgir a mensagem:



- Se se tratar de uma saída (ver Radiobutton ativo), deve selecionar uma linha da Treeview antes de clicar no Button Registar. Ao clicar no Button, surgirá a mensagem:



Caso a resposta seja **Sim**, a linha deve ser removida da Treeview e do ficheiro **parque.txt**.

5. O Button **Valor a pagar** deve incluir a imagem **calcular.png**, como se apresenta acima.
   Ao adicionar a imagem ao Button, defina-o com width=100, height=78

   Quando se trata de uma saída do parque, e depois de confirmada, deverá preencher o tempo no parque em minutos. Note-se que não se pretende que a App faça o cálculo do tempo que o veículo esteve no parque. Neste caso, é o utilizador que indica um valor, em minutos.

6. A caixa de texto (Entry) do valor a pagar não deve ser editável pelo utilizador. Ao clicar no Button **Valor a pagar** deve ser calculado o respetivo valor, com base no seguinte preçário:

Veículos ligeiros:

Primeiros 15 minutos:	1€
]15 min-30 min]	1,20€ a acrescentar ao valor anterior
]30 min - 45 min]	1,50€ a acrescer aos valores anteriores
>45 min	3,70€ + 10 cêntimos por minuto para além dos 45.

Exemplos:

- 12 minutos: valor a pagar = 1€
- 20 minutos: valor a pagar = 1€+1,20€
- 40 minutos: valor a pagar= 1€+1,20€+1,50€
- 50 minutos: valor a pagar = 1€+1,20€+1,50€+ 5*0,10€

Veículos pesados:

- Aplica-se a tabela anterior, acrescida de mais 25% em relação ao preço calculado.

Boa Sorte 😉