# POO (LEI/LCC)

2024/205

Ficha Prática #06

Hierarquia, herança, polimorfismo, interfaces e serialização de informação.

# Conteúdo

# 1 Objectivos

Durante três aulas, o projecto CarRental servirá para abordar vários conceitos leccionados em POO. Do mesmo modo, servirá como auto-avaliação para que os alunos possam fazer um ponto da situação sobre a disciplina.

O objectivo deste projecto é o desenvolvimento, de uma forma incremental, de uma aplicação que combina todos os conceitos leccionados ao longo do semestre. Na primeira fase serão abordados os conceitos de hierarquias de classes, assim como mapeamentos (c.f. API `Map<K,V>` já praticada anteriormente na Ficha 5). A segunda fase abordará os conceitos de Interface, nomeadamente as interfaces `Comparable<T>` e `Comparator<T>`. Na terceira e última fase será abordado o tema da serialização assim como o tratamento de excepções.

# 2 Projeto CarRental

Considere que se pretende desenvolver um sistema para a gestão de uma empresa que possui carros para alugar (classe CarRental). Pretende-se ter uma classe que agrupe a informação sobre os carros, sendo que existem carros de diferentes características que são oferecidos aos clientes. O sistema a desenvolver deverá ser capaz de representar essa informação e fornecer um conjunto de funcionalidades que permita consultá-la e manipulá-la, bem como registar uma viagem num determinado carro.

## 2.1 Fase 1

Um **Carro** é uma conceito que representa a informação que todos conhecemos. A informação que se guarda para cada carro é a seguinte:

- um código de identificação (que poderá ser a matrícula)
- marca do carro
- modelo
- ano de construção
- velocidade média por km
- uma autonomia, que corresponde ao número de kms que o carro pode andar com o combustível que actualmente possui
- kms totais realizados

Actualmente a empresa CarRental tem carros homogéneos pelo que de momento a informação acima apresentada é tudo o que é necessário saber de um carro (previsivelmente esta situação mudará num futuro próximo).

Considere agora que a empresa CarRental tem dois tipos de carros que constituem a sua frota, os carros a combustão, CarroCombustao, e os carros eléctricos, CarroElectrico. Os carros a combustão acrescentam informação sobre o tamanho em litros do seu depósito, o consumo em litros aos 100km e o preço por litro do combustível. Os carros eléctricos acrescentam informação sobre a dimensão da sua bateria (em kWh), o consumo em KWh aos 100km e o preço do KW.

O custo por km dos carros a combustão é dado pela fórmula $(consumoLitros100Km/100) * precoLitro$ e o custo por km dos carros eléctricos é obtido por $(consumoKWh100Km/100) * precoKWh$.

Cada vez que se faz uma viagem é indicado o número de kms a percorrer e em função disso é actualizado o valor da autonomia do carro. Deverá também existir a capacidade de abastecer/carregar completamente o carro (encher o depósito no caso dos carros a combustão ou carregar a bateria a 100% no caso dos carros eléctricos) e fazer reset ao contador parcial de kms.

A empresa sabe também que apesar dos valores anunciados de consumo dos carros o custo real por km é sempre superior e decorrente do desgaste do carro, das condições atmosféricas, entre outros factores. O custo real por km pode ser estimado como sendo o custo por km, calculado pelo carro, acrescido de 15% (obtido de forma empírica através de testes efectuados).

1. Desenhe o diagrama de classe com a arquitectura proposta. Para saber quais os métodos a considerar leia todo o enunciado da questão.

2. Crie as classes `Carro`, `CarroCombustao` e `CarroElectrico` e a classe `CarRental`, e implemente nesta os seguintes métodos:

   (a) Verificar a existência de um carro, dado o seu código.
   ```
   public boolean existeCarro(String cod)
   ```
   (b) Devolver a quantidade de carros existentes na empresa de aluguer.
   ```
   public int quantos()
   ```
   (c) Devolver o número total de carros de uma dada marca.
   ```
   public int quantos(String marca)
   ```
   (d) Devolver a informação de um carro, dado o seu código.
   ```
   public Carro getCarro(String cod)
   ```
   (e) Adicionar um novo carro.
   ```
   public void adiciona(Carro v)
   ```
   (f) Devolver uma lista contendo a cópia de todos os carros no sistema.
   ```
   public List<Carro> getCarros()
   ```
   (g) Adicionar a informação de um conjunto de carros que foram adquiridos e passam agora a fazer parte da empresa.
   ```
   public void adiciona(Set<Carro> vs)
   ```

(h) Efectuar uma viagem de um carro e indicar o número de kms que foram feitos por um cliente.

```
public void registarViagem(String codCarro, int numKms)
```

(i) Carregar ou atestar completamente um veiculo

```
public void atestarCarro(String codCarro)
```

(j) Determinar o carro com o menor custo por km

```
public Carro obterCarroMaisEconomico())
```

(k) Calcular a média de custo por quilómetro de todos os carros da empresa.

```
public double obterMediaCustoPorKm()
```

(l) Calcula o custo real por km de um carro de acordo com a regra enunciada anteriormente.

```
public int custoRealKm(String cod)
```

(m) Determinar o conjunto dos carros que podem ser requisitados para efectuar uma viagem de um determinado número de kms

```
public Set<Carro> carrosComAlcance(int kms)
```

(n) Determinar o conjunto dos carros eléctricos cujo nível de bateria é de pelo menos uma percentagem

```
public Set<CarroElectrico> comBateriaDe(int nivelMinimo)
```

**Auto avaliação 1** Como forma de auto avaliação, propõe-se a implementação da classee `CarRentalList`, com os mesmos requisitos que `CarRental`, mas utilizando como estrutura de dados um `List<Carro>`.

**Auto avaliação 2** Propõe-se também a implementação da classe `CarRentalSet`, com os mesmos requisitos que `CarRental`, mas utilizando como estrutura de dados um `TreeSet<Carro>`.

## 2.2 Fase 2

1. Pretende-se agora poder consultar a informação de forma ordenada por diferentes critérios. Altere a classe `CarRental` para que suporte as seguintes funcionalidades:

   (a) carro com mais kms percorridos. Em caso de empate aquele que tiver a matrícula com menor valor alfabético. `public Carro carroComMaisKms()`

(b) Obter um `Set<CarroElectrico>` ordenado de acordo com a ordem natural dos carros eléctricos (assuma que a ordem natural dos carros eléctricos é dada em primeiro lugar pela ordem crescente de número de kms percorridos e depois pela ordem decrescente do tamanho da sua bateria)

2. Guardar (de forma a que sejam identificáveis por nome) comparadores na classe `CarRental`[1], permitindo assim ter disponíveis diferentes critérios de ordenação.

3. Obter um iterador de `Carro`, ordenado de acordo com o critério indicado:
```
public Iterator<Carro> ordenarCarros(String criterio)
```

4. Obter um Map em que se associa a um valor de autonomia (em kms) a lista dos carros que possuem essa autonomia
```
public Map<Integer, List<Carro>> carrosPorAutonomia()
```

## 2.3 Fase 3

1. Considere agora que queremos ter acesso aos carros da empresa de aluguer, mas apenas de forma a que seja possível apenas perguntar aos objectos qual o número de kms que possuem e qual o custo por km. Como faria para responder a este requisito?

2. Considere agora que existe uma componente de *gamification* e que por cada kms percorrido por um carro se atribuem (inicialmente) pontos na razão de 1 ponto por km nos carros a combustão e 2.5 pontos por kms para os carros eléctricos. Deve

   (a) Desenvolva a interface `PontosPorKms`, que deverá ser implementada por estes tipos de carros. Esta interface deverá garantir as funcionalidades a seguir descritas:
   - definir o total de pontos a atribuir por cada km
   - obter o valor de pontos que se está a atribuir por cada km
   - obter o total de pontos que um determinado carro possui.

   (b) Actualize as classes `CarroCombustao` e `CarroElectrico` de modo a que implementem a interface.

   (c) Acrescente, na classe `CarRental`, um método que devolva os carros eléctricos mas que só deixe consultar os seus pontos.
```
public List<PontosPorKms> verPontos()
```

[1] Recomenda-se estudar a matéria relativa a variáveis e métodos de classe.

3. Um carro híbrido é um carro que tem duas motorizações, um motor a combustão (com um depósito de combustível) e um motor eléctrico (e respectiva bateria). Do ponto de vista conceptual é um conceito que herda as definições dos carros a combustão e dos carros eléctricos. Crie a classe `CarroHibrido` e codifique o método da classe `CarRental`, que devolve os carros híbridos existentes

   ```
   public List<CarroHibrido> getHibridos()
   ```

## 2.4 Fase 4

1. Por forma a melhorar a implementação efectuada até agora de `CarRental`, codifique agora excepções para suportar o tratamento de erros a seguir descritos:

   (a) Carro inexistente numa consulta,

   (b) Adicionar um carro repetido, isto é, com a mesma matrícula,

   (c) Registar uma viagem num carro inexistente.

2. Gravar e carregar a instância de `CarRental` num (e de um) ficheiro de objectos.

3. Finalmente, para completar a implementação do sistema

   (a) Desenvolva a classe `CarRentalApp` que deverá dar acesso às funcionalidades da classe `CarRental` através de um menu em modo texto, utilizando a classe de construção de menús disponibilizada.