

Prática 8 Classes e Interfaces

Exercício 8.1

Considere as seguintes entidades e características representativas de veículos motorizados:

- Motociclo, caracterizado por matrícula, marca, modelo, potência (cv), tipo (desportivo ou estrada).
- Automóvel ligeiro, caracterizado por matrícula, marca, modelo, potência, número do quadro, capacidade da bagageira.
- Táxi, que tem as características de um automóvel ligeiro e, adicionalmente, o número da licença.
- Pesado de mercadorias, caracterizado por matrícula, marca, modelo, potência, número do quadro, peso, carga máxima.
- Pesado de passageiros, caracterizado por matrícula, marca, modelo, potência, número do quadro, peso, número máximo de passageiros.
- Empresa de aluguer de viaturas, caracterizada um nome, código postal, email e por um conjunto de viaturas, que pode incluir qualquer um dos elementos anteriores.

Todas as viaturas devem implementar a seguinte interface:

```
public interface KmPercorridosInterface {  
    void trajeto(int quilometros);  
    int ultimoTrajeto();  
    int distanciaTotal();  
}
```

- a) Analise o problema cuidadosamente e modele as interfaces e classes necessárias, as suas associações (herança, composição) bem como todos os atributos e métodos. Construa a classe principal, a função *main*, e escreva código para testar diversas possibilidades com as entidades anteriores. Por exemplo:
- Crie uma empresa com um ou mais veículos de cada tipo.
 - Teste vários métodos (*setters*, *getters*, *toString*, *equals*, métodos de *KmPercorridosInterface*, ..).
 - Indique qual a viatura com maior quilómetros percorridos.
- b) Crie duas novas entidades, Automóvel ligeiro elétrico e Pesado de passageiros elétrico, derivadas das classes correspondentes, e que implementam a interface ilustrada abaixo. Volte a testar com o programa da alínea anterior, adaptando-o conforme necessário e acrescentando alguns novos veículos destes tipos.

```
public interface VeiculoEletrico {  
    int autonomia(); // devolve autonomia restante  
    void carregar(int percentagem); // simula um carregamento até 'percentagem'  
}
```

- c) (Exercício aberto) Acrescente outras funcionalidades que lhe pareçam interessantes (por exemplo, validar a matrícula, verificar formato de email, ordenar viaturas por potência, etc.).

Exercício 8.2

Considere as seguintes entidades e características representativas de alimentos:

- Carne, tem variedade (vaca, porco, peru, frango, outra), proteínas (double), calorias (double), peso (double).
 - Peixe, tem tipo (congelado ou fresco), proteínas (double), calorias (double), peso (double).
 - Cereal, tem nome (String), proteínas (double), calorias (double), peso (double). É um alimento vegetariano.
 - Legume, tem nome (String), proteínas (double), calorias (double), peso (double). É um alimento vegetariano.
 - Prato, tem um nome (String) e composição (conjunto de alimentos)
 - PratoVegetariano, tem um nome (String) e composição (conjunto de alimentos vegetarianos).
 - PratoDieta, tem um nome (String) e composição (conjunto de alimentos) e limite máximo de calorias (double).
 - Ementa – tem um nome (String), um local (String) e uma lista de pratos.
- a) Analise o problema cuidadosamente e modele as interfaces e classes necessárias, as suas associações (herança, composição) bem como todos os atributos e métodos. Implemente todas as classes necessária, seguindo as seguintes considerações:
- A unidade de calorias e de proteínas é relativa a 100gr.
 - Para cada prato deve ser possível obter informações sobre alimentos, peso total, calorias, proteínas,...
 - Implemente os métodos *hashCode()*, *equals()*, *toString()* em todas as classes.
 - Os pratos devem respeitar a interface *java.lang.Comparable* para permitir comparação com base em calorias.
- b) Teste a implementação com programa A08E02.java disponível no elearning. Verifique se obteve um resultado semelhante ao seguinte (a ordem é variável):

```
A sair .. Prato 'combinado n.1', composto por 0 Ingredientes
  Ingrediente 1 adicionado: Peixe CONGELADO, Proteinas 31.3, calorias 25.3, Peso 200.0
  Ingrediente 2 adicionado: Peixe CONGELADO, Proteinas 31.3, calorias 25.3, Peso 200.0
A sair .. Prato 'combinado n.2', composto por 0 Ingredientes
  Ingrediente 1 adicionado: Peixe CONGELADO, Proteinas 31.3, calorias 25.3, Peso 200.0
  Ingrediente 2 adicionado: Peixe CONGELADO, Proteinas 31.3, calorias 25.3, Peso 200.0
A sair .. Prato 'combinado n.3', composto por 0 Ingredientes - Prato Vegetariano
ERRO: não é possível adicionar Ingrediente 1: Peixe CONGELADO, Proteinas 31.3, calorias 25.3, Peso 200.0
Ingrediente 1 adicionado: Cereal Milho, Proteinas 19.3, calorias 32.4, Peso 110.0
ERRO: não é possível adicionar Ingrediente 2: Peixe CONGELADO, Proteinas 31.3, calorias 25.3, Peso 200.0
ERRO: não é possível adicionar Ingrediente 2: Peixe CONGELADO, Proteinas 31.3, calorias 25.3, Peso 200.0
Ingrediente 2 adicionado: Legume Couve Flor, Proteinas 21.3, calorias 22.4, Peso 150.0
A sair .. Prato 'combinado n.4', composto por 0 Ingredientes - Dieta (0.00 Calorias)
ERRO: não é possível adicionar Ingrediente 1: Carne FRANGO, Proteinas 22.3, calorias 345.3, Peso 300.0
Ingrediente 1 adicionado: Cereal Milho, Proteinas 19.3, calorias 32.4, Peso 110.0
Ingrediente 2 adicionado: Cereal Milho, Proteinas 19.3, calorias 32.4, Peso 110.0
A sair .. Prato 'combinado n.5', composto por 0 Ingredientes - Dieta (0.00 Calorias)
Ingrediente 1 adicionado: Peixe CONGELADO, Proteinas 31.3, calorias 25.3, Peso 200.0
ERRO: não é possível adicionar Ingrediente 2: Peixe CONGELADO, Proteinas 31.3, calorias 25.3, Peso 200.0
ERRO: não é possível adicionar Ingrediente 2: Peixe CONGELADO, Proteinas 31.3, calorias 25.3, Peso 200.0
Ingrediente 2 adicionado: Cereal Milho, Proteinas 19.3, calorias 32.4, Peso 110.0
A sair .. Prato 'combinado n.6', composto por 0 Ingredientes - Prato Vegetariano
ERRO: não é possível adicionar Ingrediente 1: Carne FRANGO, Proteinas 22.3, calorias 345.3, Peso 300.0
Ingrediente 1 adicionado: Legume Couve Flor, Proteinas 21.3, calorias 22.4, Peso 150.0
Ingrediente 2 adicionado: Cereal Milho, Proteinas 19.3, calorias 32.4, Peso 110.0
A sair .. Prato 'combinado n.7', composto por 0 Ingredientes - Prato Vegetariano
ERRO: não é possível adicionar Ingrediente 1: Peixe CONGELADO, Proteinas 31.3, calorias 25.3, Peso 200.0
ERRO: não é possível adicionar Ingrediente 1: Carne FRANGO, Proteinas 22.3, calorias 345.3, Peso 300.0
```

```
Ingrediente 1 adicionado: Cereal Milho, Proteinas 19.3, calorias 32.4, Peso 110.0
Ingrediente 2 adicionado: Legume Couve Flor, Proteinas 21.3, calorias 22.4, Peso 150.0
```

Ementa final

```
-----
Prato 'combinado n.1', composto por 2 Ingredientes, dia Segunda
Prato 'combinado n.2', composto por 2 Ingredientes, dia Terca
Prato 'combinado n.3', composto por 2 Ingredientes - Prato Vegetariano, dia Quarta
Prato 'combinado n.4', composto por 2 Ingredientes - Dieta (71.28 Calorias) , dia Quinta
Prato 'combinado n.5', composto por 2 Ingredientes - Dieta (86.24 Calorias) , dia Sexta
Prato 'combinado n.6', composto por 2 Ingredientes - Prato Vegetariano, dia Sabado
Prato 'combinado n.7', composto por 2 Ingredientes - Prato Vegetariano, dia Domingo
```

c) (Exercício opcional) Construa um programa Ementa que permita as seguintes operações:

- Ingrediente
 - Adicionar Carne
 - Adicionar Peixe
 - Adicionar Cereal
 - Adicionar Legume
- Prato
 - Cria Prato
 - Apaga Prato
 - Selecciona Prato
 - Adiciona Ingrediente
 - Remove Ingrediente
- Ementa
 - Adiciona Prato
 - Remove Prato
 - Imprime Ementa

Exercício 8.3

Crie um programa que simule um carrinho de compras, para isso ele pode ter dois tipos de produtos:

- Produtos Genéricos, com um nome, uma quantidade em stock e um preço;
- Produtos com descontos, que só diferem do anterior porque recebem um valor para o desconto.

Os dois tipos de produtos têm de implementar a seguinte interface:

```
public interface Produto {
    String getNome();
    double getPreco();
    int getQuantidade();
    void adicionarQuantidade(int quantidade);
    void removerQuantidade(int quantidade);
}
```

Vai existir uma classe carrinho de compras que tem de implementar a seguinte interface para gerir a compra:

```
public interface Compra {
    void adicionarProduto (Produto produto, int quantidade);
    void listarProdutos();
    void calcularTotal();
}
```

Teste o seu programa com a “main” disponibilizada.