

## 4 Lab: Modelação com classes (cont.)

### Enquadramento

#### Objetivos de aprendizagem

- Identificar conceitos/classes na descrição de um problema.
- Caracterizar as estruturas de dados de um problema como classes e associações.
- Construir e interpretar diagramas de classes (perspetiva do analista).
- Construir e interpretar diagramas de classes (perspetiva do programador).

#### Preparação

- “[class diagrams](#)”- informação tutorial.

### 4.1 Classes no modelo do domínio

Considere a área das encomendas de comida online. Sugere-se, para o efeito, focar a análise num serviço concreto, possivelmente um que já lhe seja familiar.

Desenvolva um modelo do domínio para o caso de estudo que escolheu. O seu modelo do domínio deve ter **a capacidade expressiva suficiente para permitir captar/memorizar a informação necessária aos processos de encomenda e entrega de comida.**

Explore o seu caso de estudo e procure desenvolver **um mapa completo e representativo da informação necessária.** Procure, por exemplo, em um ou mais sites a informação relevante.

Analise o seu modelo. Certifique-se que a capacidade expressiva do modelo é suficiente para responder (pelo menos) aos seguintes requisitos:

- Os clientes pesquisam online a oferta de menus/opções e compõem o seu pedido.
- A oferta pode envolver diferentes restaurantes parceiros, que é possível pesquisar de forma integrada. (Embora um pedido concreto deva ser confeccionado por um único restaurante.)
- O pedido (encomenda) origina um pagamento e uma entrega que é assegurada por um estafeta.
- Os clientes podem seguir o progresso do seu pedido, desde que foi criado até que se encontre satisfeito.
- Os menus oferecidos pelos restaurantes parceiros mudam; a própria lista de restaurantes parceiros muda.
- Os preços dos menus são alterados com frequência (o que nunca afeta pedidos anteriores).
- O preço dos menus pode mudar de acordo com promoções limitadas no tempo.
- Os responsáveis [da plataforma] consultam a evolução diária das encomendas, quer globalmente, quer por código postal.

### 4.2 Classes no código (por objetos)

Considere a implementação existente (ver:  [Labs](#)/ Lab 04 support/DemoEmentas.zip ) de um projeto em Java que gere pedidos de um restaurante.

Para facilitar, o programa gera uma ementa aleatória quando executado, com alguns pratos adicionados e, depois, simula um pedido, escolhendo dois pratos dessa ementa (DemoClass.java → main() ). O *output* está exemplificado a seguir.

Para explorar esta implementação, considere usar uma ferramenta<sup>1</sup> com destaque de sintaxe, como o [Visual Studio Code](#) com o plug-in “[Extension Pack for Java](#)” instalado.

**Nota:** para resolver o exercício, não é preciso ter um ambiente de desenvolvimento configurado<sup>2</sup>, ou sequer executar o programa (embora possa fazê-lo).

**Tabela 1: output do programa principal, simulando um pedido de comida.**

**A preparar os dados...**

```
A gerar .. Prato [nome=Dieta n.1,0 ingredientes, preco 200.0]
  Ingrediente 1 adicionado: Cereal [nome=Milho; Alimento [proteinas=19.3, calorias=32.4, peso=110.0]]
  Ingrediente 2 adicionado: Peixe [tipo=CONGELADO; Alimento [proteinas=31.3, calorias=25.3, peso=200.0]]
A gerar .. Prato [nome=Combinado n.2,0 ingredientes, preco 100.0]
  Ingrediente 1 adicionado: Peixe [tipo=CONGELADO; Alimento [proteinas=31.3, calorias=25.3, peso=200.0]]
  Ingrediente 2 adicionado: Legume [nome=Couve Flor; Alimento [proteinas=21.3, calorias=22.4, peso=150.0]]
A gerar .. Prato [nome=Vegetariano n.3,0 ingredientes, preco 120.0]
  Ingrediente 1 adicionado: Cereal [nome=Milho; Alimento [proteinas=19.3, calorias=32.4, peso=110.0]]
  Ingrediente 2 adicionado: Cereal [nome=Milho; Alimento [proteinas=19.3, calorias=32.4, peso=110.0]]
A gerar .. Prato [nome=Combinado n.4,0 ingredientes, preco 100.0]
  Ingrediente 1 adicionado: Cereal [nome=Milho; Alimento [proteinas=19.3, calorias=32.4, peso=110.0]]
  Ingrediente 2 adicionado: Cereal [nome=Milho; Alimento [proteinas=19.3, calorias=32.4, peso=110.0]]
```

**Ementa para hoje:** Ementa [nome=Menu Primavera, local=Loja 1, dia 2020-11-22T21:08:45.624777300]

Dieta n.1	200.0
Combinado n.2	100.0
Vegetariano n.3	120.0
Combinado n.4	100.0

**Pedido gerado:**

```
Pedido: Cliente = Joao Pinto
  prato: Prato [nome=Combinado n.2,2 ingredientes, preco 100.0]
  prato: Prato [nome=Combinado n.2,2 ingredientes, preco 100.0]
  datahora=2020-11-22T21:08:45.813778700]
  Custo do Pedido: 200.0
  Calorias do Pedido: 95.4
```

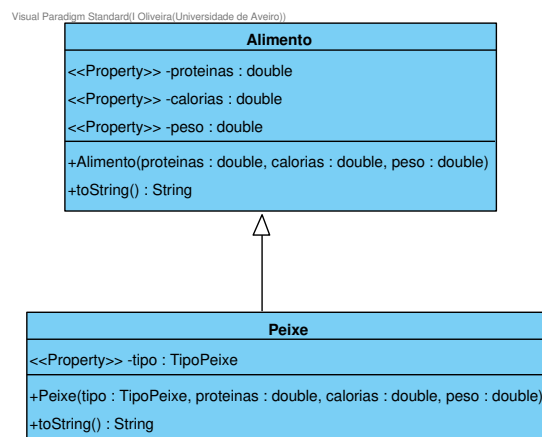
<sup>1</sup> Se tem experiência de desenvolver com outro IDE, também pode usá-lo, e.g.: Eclipse, IntelliJ,...

<sup>2</sup> Querendo instalar o ambiente de desenvolvimento para Java, é necessário instalar um JDK (Java Development Kit), e.g. a versão Temurin 17 → <https://adoptium.net/installation>

## Visualização da estrutura do código

O documento de apoio mostra algumas situações-tipo de código (em Java) e a construção correspondente no modelo (ver: [Labs/Lab 04 support/Java to UML](#)).

- Identifique, na solução dada (pasta **src/ementas/\***, após expandir o Zip), a ocorrência de **classes**. Represente-as num diagrama.
- Verifique os **atributos** associados a cada classe. Represente-os.
- Quando uma classe usa atributos cujo tipo de dados é outra classe do modelo, significa que se estabelece uma associação direcionada. Se o atributo for multivalor (i.e., um *array*, uma lista, uma coleção), a associação pode ser representada como uma agregação. Represente **as associações** que se podem inferir.
- Procure identificar situações de **especialização** (uma classe estende a semântica de uma classe mais geral, relação “*is a*”).
- Procure identificar as **operações** oferecidas pelos objetos de cada classe. Represente-as.



Nota: neste exercício, para simplificar, pode ignorar certas operações, designadamente:

<b>getAtributo()</b> <b>setAtributo( parâmetro)</b>	As operações <i>get/set</i> seguidas do nome de um atributo que pertence à classe são <b>triviais</b> ( <i>getters</i> e <i>setters</i> ) e geralmente não são representadas no modelo (para maior simplicidade).
<b>public</b> <b>NomeDaClasse(</b> parâmetros)	As operações de uma classe cujo nome da operação é igual ao nome da classe chamam-se <i>construtores</i> .  Veja que no exemplo junto os construtores foram incluídos, mas pode omiti-los neste exercício.
<i>toString()</i> <i>equals()</i> <i>compareTo()</i>	Estas operações, podem ou não existir em várias classes e significam sempre o mesmo (têm um propósito predefinido). Por isso mesmo, <b>não são decisivas para entender um modelo</b> e podem ser omitidos neste exercício.  Veja que no exemplo junto os <i>toString()</i> foram incluídos.

## 4.3 Visualização das instâncias (objetos)

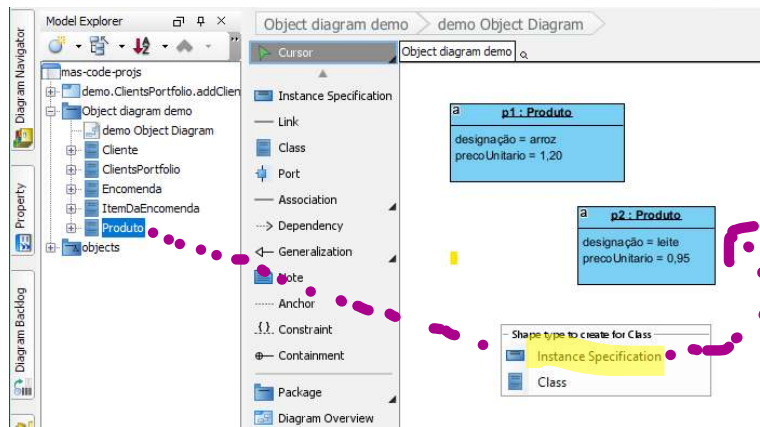
A visualização anterior foca a estrutura das entidades necessárias e responde à pergunta: que tipo de objetos (categorias) estão envolvidos e como se relacionam? Podemos, no entanto, pensar também em termos de objetos (quantas instâncias de cada classe estão envolvidas?).

Considerando a informação que se pode inferir do *output* representado na Tabela 1, podemos ter uma boa ideia de quantos objetos são instanciados de cada tipo e do seu estado (valores dos atributos). Com esta informação<sup>3</sup>, prepare um **diagrama de objetos**. O diagrama pode ser preparado na ferramenta habitual, ou em papel<sup>4</sup>.

Nota 1: no diagrama de objetos, representamos instâncias (*instance specification*) que podemos criar facilmente ao “**arrastar**” a classe pretendida para o diagrama.

<sup>3</sup> Podemos ainda obter informação sobre as instâncias criados analisando o código em si, com especial atenção para o operador **new**, do Java.

<sup>4</sup> É bastante “trabalhoso” criar um diagrama de objetos e definir a informação dos *slots* no Visual Paradigm... É mais fácil fazer “à mão” e digitalizar...



Nota 2: o diagrama de classes e o diagrama de objetos são distintos. Por exemplo, sendo p1 e p2 instâncias da classe Produto, é importante não confundir:

<p>Visual Paradigm Standard (©Universidade de Aveiro)</p> <div data-bbox="349 698 553 840"> <p><b>Produto</b></p> <ul style="list-style-type: none"> <li>-precoUnitario : double</li> <li>-designação : String</li> </ul> </div>	<p>Visual Paradigm Standard (©Universidade de Aveiro)</p> <div data-bbox="751 698 1005 840"> <p><b>p1 : Produto</b></p> <p>designação = arroz precoUnitario = 1,20</p> </div> <div data-bbox="1070 698 1272 840"> <p><b>p2 : Produto</b></p> <p>designação = leite precoUnitario = 0,95</p> </div>
<p><b>Classe</b> Produto. Representada no diagrama de classes.</p>	<p>Algumas <b>instâncias</b> concretas da classe Produto, designadas p1 e p2. Os atributos (=slots) recebem valores concretos. Representa-se no diagrama de objetos.</p>