6 Complementos: modelo do domínio, *round-trip engineering*

Enquadramento

A modelo do domínio, i.e., os conceitos de interesses e seus relacionamentos, tendem a ser um aspeto estruturante e relativamente estável na análise de um problema. Podemos identificar alguns "casos notáveis" na modelação do domínio que, pela sua importância ou frequência, podem ajudar a criar alguns padrões para modelar situações relacionadas.

Objetivos de aprendizagem

— Modelar os conceitos de um domínio, com enfase em alguns "casos notáveis" mais frequentes.

— Aplicar funções de *round-trip engineering*, ligando o modelo em UML à implementação (em Java).

6.1

Considere a área das encomendas de comida online. Para tornar a análise mais realista, selecione um exemplo 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 colocação de encomendas e entrega de comida.**

Confira o seu modelo, e se necessário, reveja-o, para incluir os seguintes requisitos[1]:

a) ***"Order/details"*** → Num pedido/encomenda, para um cliente, são incluídos diversos itens, em quantidades variáveis ["linhas de encomenda"].

b) **Preços praticados numa transação** → Os preços dos produtos mudam com alguma regularidade. No entanto, é sempre necessário saber qual foi o preço praticado para cada item presente encomendas passadas. Para além do preço, a taxa de imposto, que é sempre aplicada numa venda, pode mudar no tempo.

c) **Períodos promocionais** → O preço dos itens disponíveis pode mudar de acordo com promoções limitadas no tempo.

d) **Gestão de "parcerias"/"membros"** → A "frota" de estafetas é estabelecida através da adesão dos interessados à plataforma. Os membros candidatam-se na plataforma; após a análise dos elementos solicitados, deverá haver uma aprovação da candidatura; a adesão pode ser suspensa pela plataforma, ou até cancelada. O estafeta pode pedir para terminar a sua adesão.

e) **Monitorização de estados** → Os clientes podem seguir o progresso do seu pedido, desde que foi criado até que seja satisfeito.

f) **Pagamento** → Os clientes têm várias formas de realizar o pagamento, incluindo com cartão de crédito, MBWay e por cartão de débito (no ato de entrega). Para além do valor e data/hora, é relevante guardar detalhes adicionais (e.g.: id da transação).

6.2

Baseando-se no senso comum, considere uma máquina de estados associada à gestão da relação com os estafetas (alínea *d*, acima).

Modele esse "ciclo de vida" e certifique que "dependura" o novo diagrama na classe Estafeta

[1] Estas situações, exemplificadas para o domínio da encomenda de comida, ocorrem com frequência, com as devidas adaptações.

(ou similar): no Visual Paradigm, **use a opção "Sub-diagram"** para a classe pretendida[2].

6.3

O Visual Paradigm suporta um leque extenso de ferramentas e, nalgumas, é possível usar processos de *forward* e *reverse engineering*, e.g.

- criação de modelos de dados [E-R] e geração da base de dados.
- *round-trip engineering* para projetos em Java.

Neste exercício, vamos explorar este último ponto.

a) Convém ter disponível o ambiente de desenvolvimento para Java, incluindo um IDE (e.g.: Eclipse IDE ou VS Code com o plug-in "Extension Pack for Java").

b) Transferir o projeto já existente, na área dos laboratórios das Práticas (DemoEmentas.zip)

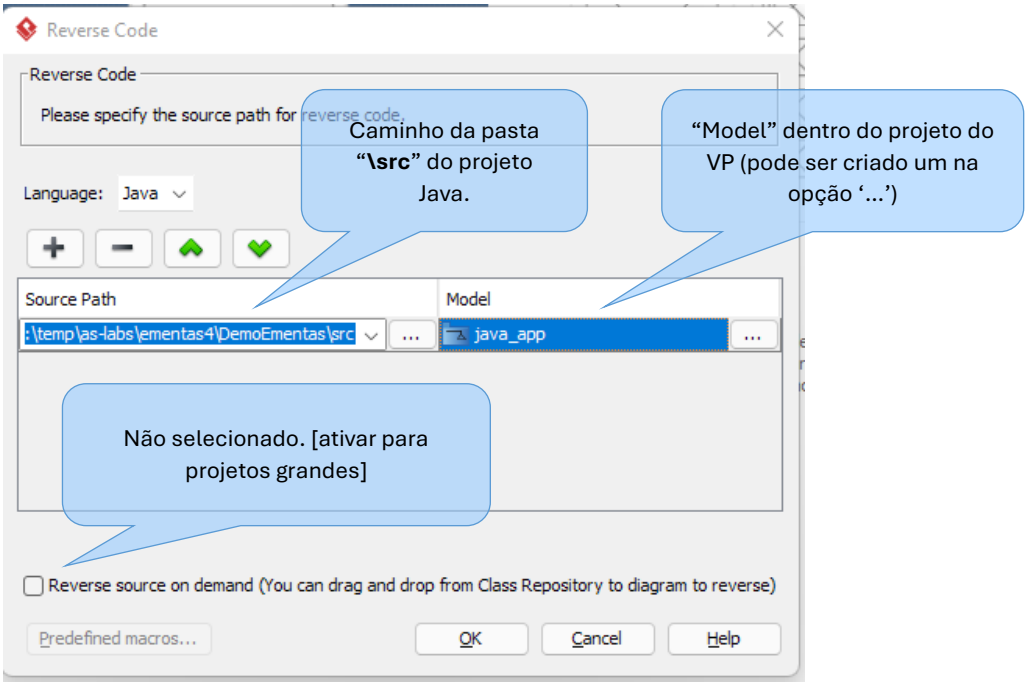
Poderá abrir este projeto e familiarizar-se com a aplicação, executando a classe "DemoMain".

c) Crie um novo projeto no Visual Paradigm (e.g.: AS63.vpp)

d) No Visual Paradigm, selecione a opção Tools → Code → Reverse Java Code. Veja a explicação aqui.

- para evitar usar diretamente a raiz, pode criar um "model" no projeto (e.g.: "java_app").

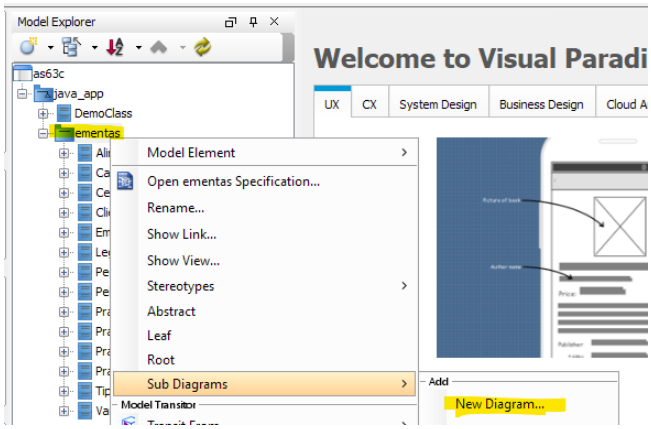
- deixe a opção "Reverse source on-demand" desselecionada.



e) Adicione um diagrama de classes ao projeto. Arraste as classes (que foram obtidas por análise do código e pertence ao *package* "ementas") da árvore do Model Explorer para o diagrama. Explore o "desenho" do projeto, i.e., as classes existentes e seus relacionamentos.

[Exporte o diagrama para as respostas ao exercício.]

[2] Esta situação demonstra que um podemos ter subdiagramas associados a um elemento de modelação, para ampliar/detalhar a sua especificação.



f) Seguindo a estratégia do desenho [das classes] deste projeto, no **Visual Paradigm**:

- crie a(s) classe(s) necessária(s) para representar o PratoOvolactovegetariano (que especializa Prato) e aceita alimentos "OvoLacto"[3] (que especializa Alimento).

- grave o projeto;

- atualize o código fonte com a opção de "geração".

g) Confirme as alterações no IDE (Eclipse,...).

A partir do IDE, adicione (pelo menos) o método "adicionarIngrediente", seguindo o mesmo esquema dos outros "Pratos".

h) No Visual Paradigm, atualize o modelo, para refletir as últimas edições feitas no código. [Exporte o diagrama para as respostas ao exercício.]

[3] Simplificação para referir alimentos como ovos e lactícinios, de origem animal.