CIÊNCIAS DA COMPUTAÇÃO

FACULDADE DE CIÊNCIAS | UNIVERSIDADE AGOSTINHO NETO

PROJECTO DE PROGRAMAÇÃO ORIENTADA À OBJECTOS

Gestor de um *Outlet* de Restaurantes

PERÍODO DE EXECUÇÃO: 02 DE MAIO À 03 DE JUNHO DE 2019

ENQUADRAMENTO

O objetivo do projeto é desenvolver os conceitos de suporte à gestão e utilização de um *outlet* de restaurantes. A aplicação de gestão realiza operações de administração dos alimentos, restaurantes e clientes; permite gerir a publicação das ementas dos restaurantes; e guardar o estado persistentemente. A aplicação de vendas permite que os clientes consultem e adquiram os pratos disponibilizados, e se registem para serem avisados quando ocorrerem alterações do seu interesse. Dependendo do tipo de cliente, podem existir restrições ao tipo de pratos que podem ser adquiridos.

O projeto deve ser concebido de forma a possibilitar futuras extensões ou alterações de funcionalidade com impacto mínimo no código pré-existente. Deverá ser simples (i) gerir vários *outlets*; (ii) adicionar novos tipos de clientes com diferentes restrições à compra de pratos do dia; (iii) adicionar novas políticas de registo de interesses; e (iv) adicionar novas formas de consulta.

No projeto descrito neste documento existe um único *outlet* de restaurantes.

1. Entidades, Propriedades, Funcionalidade

1.1. Entidades

Existem várias entidades na aplicação a desenvolver: (i) restaurantes; (ii) *outlets*; (iii) clientes, (iv) alimentos e (v) pratos do dia.

Cada restaurante disponibiliza um conjunto de pratos do dia (§1.2.2) através de um *outlet* de restaurantes.

Os clientes quando acedem a um *outlet* de restaurantes podem consultar e adquirir os pratos do dia disponíveis, e podem ainda registar-se para serem avisados quando ocorrerem alterações nos pratos do dia disponíveis (e.g., um determinado prato do dia foi disponibilizado a preço mais reduzido ou um novo prato do dia ficou disponível).

1.2. Propriedades

Os identificadores são únicos para cada tipo de entidade. É da responsabilidade do sistema verificar esta restrição. Sempre que for necessário produzir listas, a ordem é determinada pela ordem lexicográfica dos identificadores (exceto indicação específica).

1.2.1. Propriedades dos alimentos

Os alimentos são identificados por um nome único (uma cadeia de carateres).

Os alimentos podem pertencer a um de dois grupos: simples ou agregados. Os alimentos agregados são formados por outros alimentos, simples e/ou agregados, indicando-se ainda a sua respetiva percentagem (um inteiro).

Os alimentos simples podem ainda pertencer a um de três tipos: carne, peixe ou vegetal. Os alimentos que só tenham na sua constituição alimentos do tipo vegetal são denominados vegetarianos.

Cada alimento simples tem a indicação das suas calorias por grama. No caso dos alimentos compostos, as calorias por grama são calculadas com base na sua constituição e de acordo com a fórmula *Calorias(agregado)* = Σ_i(Calorias(i) × per_i), em que i representa cada constituinte e per_i a percentagem do constituinte i no agregado.

1.2.2. Propriedades dos pratos do dia

Um prato do dia está associado a um alimento e é identificado univocamente pelo nome desse alimento. Cada prato tem ainda a indicação da quantidade do alimento que lhe está associado (um inteiro) e um preço de venda (um inteiro). Podem ser aplicados descontos ao preço de venda e cada prato do dia pode ou na˜o estar disponível para venda.

É possível existirem pratos do dia diferentes, identificados pelo mesmo nome, desde que pertençam a restaurantes distintos.

1.2.3. Propriedades dos clientes

Cada cliente tem como identificador único o seu e-mail (cadeia de carateres) e possui ainda um nome (cadeia de carateres). Os clientes podem ser de dois tipos: omnívoros e vegetarianos. Os clientes vegetarianos só podem comprar pratos vegetarianos. Um cliente pode, em qualquer instante, alterar o seu tipo.

Os clientes mantêm um registo do número de encomendas realizadas (um inteiro), o número de pratos do dia encomendados (um inteiro) e o total gasto nessas encomendas (um inteiro).

1.2.4. Propriedades dos restaurantes

Cada restaurante tem como identificador único o seu nome (cadeia de carateres) e um e mail (cadeia de carateres). Os restaurantes mantêm um registo das suas vendas, nomeadamente o número de vendas de cada um dos pratos do dia. Cada restaurante mantém informação dos pratos do dia disponíveis para venda. Uma vez criado, um prato não pode ser removido.

1.2.5. Propriedades das encomendas

Cada cliente tem um carrinho de compras onde ficam registados os pedidos parciais. Quando o cliente decide finalizar uma encomenda, decidindo-se pelo pagamento, a encomenda fica fechada e o carrinho de compras vazio.

1.3. Operações

1.3.1. Operações relativas a clientes

Podem realizar-se as seguintes operações relativas a clientes: (i) visualizar; (ii) registar novos; (iii) alterar o tipo; (iv) inscrever em listas de alerta; (v) desinscrever de listas de alerta; e, (vi) listar mensagens de alerta recebidas.

Os clientes não podem ser removidos e não podem alterar, nem o nome, nem o e-mail, mas podem inscrever-se para serem informados quando ocorrerem alterações no *outlet* de restaurantes que sejam do seu interesse, como, por exemplo, um novo prato ficar disponível. Um cliente só deve ser avisado de pratos que pode comprar: Por exemplo, um cliente vegetariano só deve ser avisado se ficar disponível um novo prato vegetariano.

Estão predefinidos dois tipos de alerta: para reportar (i) reduções de preço e (ii) disponibilização de novos pratos.

1.3.2. Operações relativas a alimentos

Podem realizar-se as seguintes operações relativas a alimentos: (i) visualizar; (ii) registar alimento simples; (iii) registar alimento agregado; (iv) alterar calorias de alimento simples; e, (v) descrever alimento agregado.

Não é possível remover ou alterar propriedades de alimentos (exceptuam-se as calorias dos alimentos simples).

1.3.3. Operações relativas a restaurantes

Podem realizar-se as seguintes operações relativas a restaurantes: (i) visualizar; (ii) registar novos restaurantes; e, (iii) selecionar menu de gestão de um restaurante.

Os restaurantes não podem ser removidos e não se podem alterar as suas propriedades depois de registados.

1.3.4. Operações relativas a um restaurante

Podem realizar-se as seguintes operações relativas a um determinado restaurante: (i) visualizar pratos do dia; (ii) registar novos pratos do dia; (iii) calcular calorias de um prato do dia; (iv) disponibilizar prato do dia; (v) indisponibilizar prato do dia; e, (vi) aplicar um desconto a um prato.

Um restaurante não pode remover pratos do dia.

1.3.5. Operações relativas a *outlets*

Podem realizar-se as seguintes operações relativas a *outlets* (encomendas e vendas): (i) visualizar pratos do dia do outlet de restaurantes; (ii) adicionar prato do dia ao carrinho de compras; e, (iii) pagar.

Cada cliente dispõe de um único carrinho de compras.

1.4. Pesquisas

Deve ser possível efetuar pesquisas sob vários critérios e sobre as diferentes entidades geridas, e.g., **ver todos os clientes**, **ver todos os pratos vegetarianos de um restaurante**, **ver restaurantes sem pratos vegetarianos**, **ver clientes sem compras**, **ver clientes com compras em mais do que um restaurante**.

Deve ser possível introduzir novos métodos de pesquisa com um impacto mínimo na implementação desenvolvida.

2. Interação com o Utilizador

Nas secções seguintes é descrita a funcionalidade máxima das aplicações, descrevendo-se a interação com o utilizador.

As aplicações de gestão e de vendas utilizam a mesma base de código (o núcleo da aplicação), mas definem dois conjuntos de menus independentes: o **menu de gestão** (§3) e o **menu de vendas** (§4). A aplicação de gestão pode ser iniciada sem dados prévios (neste caso, todos os alimentos, pratos do dia, clientes e restaurantes têm de ser criados através do menu de gestão). Os pratos do dia são comercializados através da aplicação de vendas.

As exceções usadas na interação, exceto se indicado, são subclasses de `ao.uan.fc.poo.ui.InvalidOperation`. Estas exceções são lançadas pelos comandos e tratadas pela classe `ao.uan.fc.poo.ui.Menu`. Note-se que, além das exceções descritas, é possível a definição de outras exceções. As novas exceções não devem, no entanto, substituir as fornecidas nos casos descritos por este enunciado.

A apresentação de listas faz-se por ordem crescente do critério de apresentação. Quando a chave é uma cadeia de carateres, a ordem deve ser lexicográfica (`UTF-8`), não havendo distinção entre maiúsculas e minúsculas (§1.2).

Nota importante: Em geral, os comandos pedem toda a informação antes de proceder à validação da mesma (exceto onde indicado).

Em todas as apresentações de dados, tanto na aplicação de gestão, como na de vendas, todos os valores numéricos são apresentados como inteiros.

3. Aplicação de Gestão

Os métodos correspondentes às mensagens de diálogo estão definidos em `rest.textui.manager.Message`.

O menu inicial permite realizar operações de leitura e escrita básicas e, ainda, aceder a submenus que contêm a restante funcionalidade. A lista completa é a seguinte (omite-se a opção **Sair**, uma vez que é implementada automaticamente por todos os menus): **Criar**, **Abrir**, **Guardar**, **Guardar como**... (§3.1), **Gestão de clientes** (§3.2), **Gestão de alimentos** (§3.3), **Gestão de restaurantes** (§3.4) e **Gestão de um Restaurante** (§3.5). As secções abaixo descrevem pormenorizadamente as ações associadas a estas opções.

Sempre que for pedido o identificador único de uma entidade já existente e esse identificador não estiver associado a nenhuma entidade, os comandos devem lançar a exceção `rest.textui.UnknowKeyException`.

Por outro lado, no processo de criação de novas entidades, se o identificador fornecido para a entidade a criar já estiver associado a outra entidade do mesmo tipo, os comandos devem lançar a exceção `rest.textui.DuplicateKeyException`.

3.1. Manipulação de Ficheiros

O estado da aplicação pode ser guardado em ficheiros, para posterior recuperação: utiliza-se a capacidade de serialização do Java (implementações de `java.io.Serializable`). São definidas as operações básicas para manipulação de ficheiros: criação de novo ficheiro ("criar"), abertura de ficheiro existente ("**abrir**"), e salvaguarda de ficheiro aberto ("**guardar**"). Devem ser tratadas todas as exceções relativas à manipulação de ficheiros. Note-se que é sempre possível trabalhar com a aplicação, mesmo que não tenha sido carregado nenhum ficheiro. A funcionalidade de cada operação á a seguinte:

Criar – Criação de novo ficheiro.

Abrir – Abertura e eventual utilização de um ficheiro existente (previamente guardado). O sistema pede o nome do ficheiro a abrir: caso não exista, o sistema limita-se a comunicar o erro (mensagem de ficheiro não encontrado).

Guardar – Salvaguarda das alterações desde a abertura do ficheiro associado à aplicação. Caso não haja nenhum ficheiro associado, deve ser perguntado ao operador o nome do ficheiro a utilizar. Não é executada nenhuma acção se não existirem alterações desde a última salvaguarda.

Guardar como... – Esta opção permite associar/mudar a associação de um ficheiro à aplicação (além de guardar os dados nesse ficheiro).

Apenas é possível trabalhar com um ficheiro de cada vez. Assim, sempre que se abandona um ficheiro com modificações não salvaguardadas, por exemplo, porque se cria outro, deve perguntar-se ao operador se deseja guardar a informação atual:

Se houver alterações, então deve-se perguntar ao operador se deseja guardar o ficheiro antes de sair. Caso a resposta seja afirmativa, então deve-se guardar a informação.

Se não existir nenhum ficheiro associado, então deve-se ainda perguntar ao operador qual o nome a utilizar para guardar a informação.

O comando **Guardar como**... não pergunta ao operador se deseja guardar o ficheiro antes de gravar o novo, pois isso corresponderia a guardar dois ficheiros iguais. Este comando tem exatamente o propósito oposto.

Note-se que parte deste procedimento corresponde à opção **Guardar**. A opção "**sair**", presente em todos os menus, não pede para guardar o estado da aplicação, mesmo que tenha havido alterações.

3.2. Gestão de Clientes

As mensagens de diálogo devem ser definidas em `rest.textui.clients.Message`.

3.2.1. Visualizar todos os clientes

O formato de apresentação de cada cliente é o seguinte:

type | *name* | *email* | *#orders* | *#dotd* | *#spent*

Os valores para o campo *type* são `OMNIVOROUS`, ou `VEGETARIAN`. `#orders`, `#dotd` e `#spent` indicam, respetivamente, o número de encomendas efectuadas, o número de pratos do dia encomendados e o total gasto nessas encomendas.

3.2.2. Registar novo cliente

É pedido o tipo do cliente (devendo ser dada uma resposta como indicado para o campo *type* em §3.2.1). Se o tipo indicado for desconhecido, repete-se a pergunta até se obter uma resposta correta. De seguida, pede o nome do cliente e o endereço de correio eletrónico.

Esta acção associa um identificador único (e-mail) ao cliente criado.

3.2.3. Alterar o tipo de um cliente

É pedido o identificador do cliente e o novo tipo (devendo ser dada uma resposta como indicado para o campo *type* em §3.2.1). Se o tipo indicado for desconhecido, repete-se a pergunta até se obter uma resposta correta.

3.2.4. Inscrever em lista de alerta

É pedido o identificador do cliente e de seguida o tipo do alerta pretendido: `DISCOUNT`, quando pretende ser alertado se ocorrer qualquer redução de preço) ou `NOVELTY`, quando ficar disponível um novo prato do dia).

Nota importante: O identificador do cliente é validado depois de se ler um tipo de alerta válido.

Caso o cliente já esteja inscrito na lista em que se pretende inscrever, não é realizada nenhuma acção.

3.2.5. Desinscrever de lista de alerta

É pedido o identificador do cliente e de seguida o tipo do alerta que já não pretende receber: `DISCOUNT` (quando já não pretende ser alertado se ocorrer uma redução de preço) ou `NOVELTY` (quando já não pretender ser alertado se ficar disponível um novo prato do dia). O identificador do cliente é validado depois de se ler um tipo de alerta válido.

Caso o cliente não esteja inscrito na lista de que pretende sair, não é realizada nenhuma acção.

3.2.6. Listar mensagens de alerta recebidas

É pedido o identificador do cliente. As mensagens são apresentadas cronologicamente, no seguinte formato: `type | message` Quando o campo *type* for `DISCOUNT`, `message` é alertDiscount(); quando *type* for `NOVELTY`, `message` é alertNovelty().

3.3. Gestão de Alimentos

As mensagens de diálogo devem ser definidas em `rest.textui.food.Message`. As exceções a lançar pelos comandos deve ser definidas em `rest.textui.food`.

3.3.1. Visualizar alimentos

O formato de apresentação de cada alimento na lista é o seguinte:

- alimentos simples: `type | name | calories`
- alimentos agregados: `code|name`

Os valores para o campo *type* são `MEAT`, `FISH` e `VEGETABLE`. Os valores para o campo *code* são `VEG` e `NOVEG`.

3.3.2. Registar alimento simples

É pedido o tipo do alimento (devendo ser dada uma resposta como indicado para o campo *type* em §3.3.1). Se o tipo indicado for desconhecido, repete-se a pergunta até se obter uma resposta correta. De seguida, pede-se o nome do alimento e o número de calorias por grama.

Esta acção associa um identificador único (nome) ao alimento simples criado.

3.3.3. Registar alimento agregado

Neste comando, a validação das chaves é imediata e o comando é interrompido quando ocorrem chaves desconhecidas.

Começa por pedir o nome do alimento. De seguida são pedidos os nomes dos alimentos que o constituem e as respetivas percentagens (um inteiro: deve permanecer num ciclo em que pergunta o nome seguido da quantidade até que o somatório das percentagens inseridas seja 100. O comando deve lançar a exceção `PercentageTooHighException` se o somatório das percentagens inseridas for superior a 100.

Esta acção associa um identificador único (nome) ao alimento agregado criado.

3.3.4. Alterar alimento simples

É pedido o identificador do alimento já existente e o novo nu´mero de calorias por grama.

3.3.5. Descrever alimento agregado

É pedido o nome do alimento agregado. Lança a exceção `InvalidKeyException` se o identificador corresponder a um alimento simples.

Se o identificador for válido, primeiro apresenta um cabeçalho idêntico ao indicado em §3.3.1: `code|name`

seguindo-se uma descrição de cada um dos alimentos que o constituem (a ordem de apresentação deve ser idêntica à usada no registo). Os alimentos que entram na sua constituição são descritos por:

- `type|name|percentage` (no caso dos alimentos simples)
- `code|name|percentage` (no caso dos alimentos agregados) em que o valor de *type* e *code* estão indicados em §3.3.1.

Os alimentos agregados são recursivamente descritos (em profundidade) como indicado nesta subsecção.

3.4. Gestão de Restaurantes

Os métodos correspondentes as mensagens de diálogo devem ser definidos em `rest.textui.restaurants.Message`.

3.4.1. Visualizar restaurantes

O formato de apresentação de cada restaurante na lista é o seguinte: `name|#dotd|#spent`

Os campos `#dotd` e `#spent` indicam, respetivamente o número de pratos do dia vendidos e o total recebido nessas vendas.

3.4.2. Registar novo restaurante

É pedido o nome do restaurante e o endereço de correio eletrónico. Esta acção associa um identificador único (`nome`) ao restaurante criado.

3.4.3. Selecionar menu de gestão de um restaurante

É pedido o nome do restaurante e de seguida entra no menu de gestão descrito em 3.5.

3.5. Gestão de um Restaurante

Os métodos correspondentes às mensagens de diálogo devem ser definidas em rest.textui.restaurant.Message. As exceções a lançar pelos comandos devem ser definidas em `rest.textui.restaurant`.

3.5.1. Visualizar pratos do dia

O formato de apresentação dos pratos do dia associados ao restaurante é o seguinte:

`class|name|price|discount|available?`

Os valores para o campo *`class`* são *`VEG`* e *`NOVEG`*. Os valores para o campo *`available?`* são *`AVAILABLE`* e *`UNAVAILABLE`*.

3.5.2. Criar prato do dia

É pedido o identificador de um alimento, uma quantidade e um custo. Quando um prato do dia é criado não fica disponível no *outlet* de restaurantes (§3.5.4) e não tem descontos (§3.5.6).

3.5.3. Calcular calorias de um prato do dia

É pedido o identificador do prato do dia sendo o formato de apresentação o seguinte: `name|calories`

O cálculo das calorias deve ter em conta o valor atual das calorias por grama dos alimentos simples.

3.5.4. Disponibilizar prato do dia no *outlet* de restaurantes

É pedido o identificador do prato do dia a disponibilizar (se o identificador não corresponder a um prato do dia, deve lançar a excepção `InvalidKeyException`). Se o prato do dia já estiver disponível, não é realizada nenhuma acção.

3.5.5. Indisponibilizar prato do dia no *outlet* de restaurantes

É pedido o identificador do prato do dia que deve ficar indisponível (se o identificador não corresponder a um prato do dia, lança-se a exceção `InvalidKeyException`). Se o prato do dia já estiver indisponível, não é realizada nenhuma acção.

3.5.6. Alterar desconto

É pedido o identificador do prato do dia. Lança a exceção `InvalidKeyException` se o identificador não corresponder a um prato disponível. Se o identificador for válido, pede o **desconto** (`inteiro no intervalo [0, 100]`) a aplicar. Caso o valor do desconto esteja fora do intervalo [0, 100], não é realizada nenhuma acção.

4. Aplicação de Vendas

A aplicação de vendas permite usar os dados guardados pela aplicação de gestão. Quando a aplicação de vendas termina, reescreve no mesmo ficheiro uma atualização dos dados que podem vir a ser lidos quer pela aplicação de gestão quer pela aplicação de vendas. Quando se sai da aplicação de vendas são mantidas todas as encomendas não finalizadas. Cada cliente pode ter, no máximo, uma encomenda não finalizada.

A aplicação de vendas começa por solicitar o nome do ficheiro que contém os dados persistentes. Para simplificar, a concretização do sistema pode assumir que o ficheiro indicado existe sempre e que contém dados corretos (contudo, uma boa implementação deveria verificar os problemas de abertura e leitura dos objetos).

Quando a aplicação de vendas termina solicita o nome do ficheiro onde devem ser guardados os dados do *outlet*. Para simplificar, a implementação do sistema pode assumir que o ficheiro indicado não existe (contudo, uma boa implementação deveria verificar os problemas de abertura e escrita dos objetos).

Após o diálogo inicial, é aberto o menu da aplicação de vendas. Os métodos correspondentes às mensagens de diálogo devem ser definidos em `rest.textui.outlet.Message`. As exceções a lançar pelos comandos devem ser definidas em `rest.textui.outlet`. Sempre que for pedido o identificador único de uma entidade já existente e esse identificador não estiver associado a nenhuma entidade, deve-se lançar a exceção `rest.textui.outlet.UnknowKeyException`.

O menu da aplicação permite realizar as seguintes operações (omite-se a opção **Sair**, uma vez que e é implementada automaticamente por todos os menus): Visualizar pratos do dia do *outlet* de restaurantes (§4.1), Adicionar prato a encomenda (§4.2), Finalizar encomenda (§4.3),

O resto desta secção descreve pormenorizadamente as acções associadas a estas opções.

4.1. Visualizar pratos do dia do outlet

O formato de apresentação dos pratos do dia disponíveis no *outlet* é o seguinte:

`class|plateName` Os valores para o campo `class` são `VEG` e `NOVEG`. Só são apresentados pratos do dia que estejam, presentemente, a ser disponibilizados por, pelo menos, um restaurante.

4.2. Adicionar prato a encomenda

É pedido o e-mail do cliente, o nome do prato do dia e o nu´mero de doses a encomendar. Só depois de obter os três dados procede as validações. Se o nome do prato corresponder a um prato não disponível gera a exceção `InvalidKeyException` e se o prato não for adequado (os clientes vegetarianos não podem selecionar pratos não vegetarianos) gera a exceção `NotAdequateException`.

De seguida, lista os restaurantes que podem fornecer o prato do dia pretendido, usando o formato: `restaurantId|plateFinalPrice|foodQuantity`

Em que `plateFinalPrice` representa o preço real que é pago pelo cliente, ou seja, tem em conta o desconto, e `foodQuantity` é a quantidade do alimento, em gramas, do prato do dia.

É então pedido o nome do restaurante que efetua a venda. Se a resposta corresponder a um identificador não apropriado, seleciona o restaurante que apresenta o preço final mais baixo, i.e., considerando o desconto. Se um prato só pode ser fornecido por um restaurante, então é automaticamente selecionado sem listar os restaurantes e sem consultar o utilizador.

4.3. Finalizar encomenda

É pedido o e-mail do cliente, devendo ficar registado nos respetivos restaurantes as parcelas da encomenda relevantes para cada restaurante.

5. Entrega e Penalizações

A não organização das Classes conforme descrito neste enunciado conduz a uma classificação de 0 (zero) valores.

A entrega de código não compilável conduz a uma classificação de 0 (zero) valores, na avaliação da funcionalidade do módulo correspondente.

Não são aceites entregas fora de prazo: `trabalhos não entregues têm uma classificação de 0 (zero) valores`.

A entrega de um projecto pressupõe o compromisso de honra que o trabalho correspondente foi realizado pelos alunos referenciados nos ficheiros submetidos para avaliação. A quebra deste compromisso, ou seja, a tentativa de um grupo se apropriar de trabalho realizado por colegas, tem como consequência a reprovação de todos os alunos envolvidos (incluindo os que possibilitaram a ocorrência).

O projecto deve ser **entregue** até ao dia **03 de Junho de 2019** (12:00 para os Alunos Diúrnos e 20:00 para os Alunos Norturnos), pressupondo a correcta execução de toda a funcionalidade e a apresentação do diagrama UML (classes) que modela o Universo de discurso do Gestor de um *Outlets* de Restaurantes.

BOM TRABALHO!