TÉCNICO LISBOA

Projecto de Programação com Objectos
8 de Outubro de 2020

Esclarecimento de dúvidas:

- Consultar sempre o corpo docente atempadamente: presencialmente ou através do endereço **po-tagus@disciplinas.tecnico.ulisboa.pt**.
- Não utilizar fontes de informação não oficialmente associadas ao corpo docente (podem colocar em causa a aprovação à disciplina).
- Não são aceites justificações para violações destes conselhos: quaisquer consequências nefastas são da responsabilidade do aluno.

Requisitos para desenvolvimento, material de apoio e actualizações do enunciado (ver informação completa na secção **[Projecto]** no Fénix):

- O material de apoio é de uso obrigatório e não pode ser alterado.
- Verificar atempadamente (mínimo de 48 horas antes do final de cada prazo) os requisitos exigidos pelo processo de avaliação.

Processo de avaliação (ver informação completa nas secções **[Projecto]** e **[Método de Avaliação]** no Fénix):

- Datas: **2020/10/23 23:59** (UML); **2020/11/09 23:59** (intercalar); **2020/12/11 23:59** (final); **2020/12/14-2020/12/18** (teste prático).
- Não serão consideradas quaisquer alterações aos ficheiros de apoio disponibilizados: eventuais entregas dessas alterações serão automaticamente substituídas durante a avaliação da funcionalidade do código entregue.
- Trabalhos não entregues no Fénix até final do prazo têm classificação 0 (zero) (não são aceites outras formas de entrega).
- A avaliação do projecto pressupõe o compromisso de honra de que o trabalho foi realizado pelos alunos correspondentes ao grupo de avaliação.
- **Fraudes na execução do projecto terão como resultado a exclusão dos alunos implicados do processo de avaliação**.

O objectivo deste projecto é desenvolver um gestor para uma empresa de distribuição. Assim, o sistema a desenvolver deverá permitir, entre outras acções: registar dados de produtos para venda, registar dados de clientes, registar dados de fornecedores de produtos para venda e fazer pesquisas várias sobre a informação armazenada.

A secção 1 apresenta as entidades do domínio da aplicação a desenvolver. As funcionalidades da aplicação a desenvolver são descritas nas secções 3, 4 e 5. A secção 2 descreve as qualidades que a solução deve oferecer. Neste texto, o tipo **negrito** indica um literal (i.e., é exactamente como apresentado); o símbolo ␣ indica um espaço; e o tipo *itálico* indica uma parte variável (i.e., uma descrição).

1 Entidades do Domínio

Nesta secção descrevem-se as várias entidades que vão ser manipuladas no contexto da aplicação a desenvolver. Existem vários conceitos importantes neste contexto: produtos, clientes, fornecedores, transacções e e tempo. Os produtos, clientes e fornecedores possuem uma chave única (cadeia de caracteres). As transacções (vendas e encomendas) possuem uma chave inteira.

Os conceitos listados não são os únicos possíveis no modelo a realizar por cada grupo e as suas relações (assim como relações com outros conceitos não mencionados) podem depender das escolhas do projecto.

1.1 Produtos

A empresa vende caixas e contentores de vários tipos. Outros tipos de produtos diversos para venda são disponibilizados à empresa por fornecedores. Todos os produtos têm um fornecedor, um preço (um número inteiro positivo), um valor crítico (utilizado na gestão de existências) e o valor das existências.

As caixas são produtos que têm um volume relativamente pequeno e servem para guardar um número relativamente pequeno de itens, por forma a protegê-los durante o seu transporte. As caixas têm um dado nível de serviço: `normal`, `aéreo`, `expresso` e `em mão`.

Os contentores são produtos que têm um volume muito maior que as caixas, permitindo guardar uma maior quantidade de itens. Têm os mesmos 4 níveis de serviço das caixas. Adicionalmente, os contentores têm 4 níveis de qualidade de serviço: `B4`, `C4`, `C5`, `DL`.

A empresa também vende livros. Um livro é caracterizado pelo respectivo título, autor e ISBN (cadeias de caracteres). Poderão ser definidos novos tipos de produtos (e.g. discos ou CDs/DVDs), que terão propriedades específicas. O impacto da introdução dos novos tipos na implementação desenvolvida deve ser mínimo.

1.2 Clientes

Os clientes fazem compras que podem ser pagas mais tarde. Cada cliente tem um nome (cadeia de caracteres) e uma morada (cadeia de caracteres). Associada a cada cliente existe ainda informação relativa às suas compras, assim como o seu estatuto (e.g., cliente "*Elite*-- ver secção 1.6). O estatuto de um cliente tem impacto na sua relação com a empresa Dado um cliente, é possível aceder ao historial das suas transacções.

1.3 Fornecedores

Os fornecedores respondem a encomendas por parte da empresa. Cada fornecedor tem um nome (cadeia de caracteres) e uma morada (cadeia de caracteres) e fornece um determinado conjunto de produtos. Dado um fornecedor, é possível aceder ao seu historial de transacções. É ainda possível activar/desactivar a realização de transacções com um fornecedor. Um fornecedor inactivo não pode realizar transacções.

1.4 Transacções

Existem dois tipos de transacção: vendas e encomendas. As transacções são identificadas por um número (inteiro), atribuído de forma automática pela aplicação. Este identificador começa em 0 (zero), sendo incrementado quando se regista uma nova transacção. A sequência de identificadores é partilhada por todas as transacções (vendas e encomendas).

Uma encomenda está associada a um fornecedor e envolve uma ou mais unidades de um ou mais produtos fornecidos pelo fornecedor em causa. O custo unitário de cada produto corresponde ao custo do produto em causa. A encomenda guarda o seu custo total, para que futuras alterações no preço de um produto não alterem o valor que foi pago pela encomenda. Quando se faz uma encomenda deve considerar-se que a encomenda foi paga imediatamente e que as existências da empresa foram actualizadas considerando os produtos encomendados.

Por seu lado, uma venda envolve uma ou mais unidades de um único produto. Cada venda tem uma data limite de pagamento: primeiro realiza-se a venda e só depois é que se procede ao seu pagamento. O preço a pagar por um cliente depende do tempo que demora a realizar o pagamento. São considerados os seguintes períodos (N é 5 para caixas, 8 para contentores e 3 para livros):

- P_1 - Até N dias antes do limite de pagamento ($data\_limite\_de\_pagamento - data\_actual \geq N$).
- **P2** - Até à data limite ($0 \leq data\_limite\_de\_pagamento - data\_actual < N$).
- **P3** - Até N dias depois da data limite ($1 < data\_actual - data\_limite\_de\_pagamento \leq N$).
- **P4** - Após N dias depois da data limite ($data\_actual - data\_limite\_de\_pagamento > N$).

A empresasabe determinar dois saldos (diferencial entre vendas e compras) distintos. Existe um saldo disponível, correspondente à diferença entre as vendas realmente pagas e as encomendas, e um saldo contabilístico, correspondente à diferença entre o valor contabilístico das vendas (pagas ou não e considerando descontos/penalizações à data da consulta de saldo) e as encomendas.

1.5 Notificações

Quando se regista um novo produto, os clientes devem ser colocados como entidades interessadas em receber notificações sobre eventos a ele associados. Em qualquer momento, um cliente pode activar ou desactivar as notificações relativas a um produto. Os eventos a considerar são os seguintes: (i) quando o produto passa de stock 0 (zero) para outro valor (positivo); (ii) quando um produto fica mais barato. As notificações são compostas pelo identificador do produto e pela descrição da notificação (NEW, para novos produtos; e BARGAIN, para descidas de preços), sendo registadas tanto na empresa, como nos clientes que as recebem.

A entrega de notificações deve ser flexível e prever vários meios de entrega, e.g., correio postal, SMS, email, entre outras. O meio de entrega por omissão corresponde a registar a notificação na aplicação.

1.6 Contabilização de Pontos (Clientes)

Existem três classificações distintas de clientes: *Normal*, *Selection* e *Elite*. A classificação de um cliente tem impacto nas multas, descontos e prazos de pagamento a aplicar no pagamento de uma venda.

Quando um cliente paga uma transacção dentro do prazo acumula um número de pontos correspondente a 10 vezes o valor pago. Não há contabilização de pontos em pagamentos atrasados. A verificação do atraso é realizada quando se realiza o pagamento de uma transacção.

Após a contabilização dos pontos, se o número de pontos for superior a 25000 então o cliente é *Elite*. Se o número de pontos não for superior a 25000 mas for maior do que 2000, então o cliente é *Selection*.

Se um cliente se atrasa no pagamento de uma transacção pode ser despromovido: um cliente *Elite* passa a *Selection* se o pagamento ocorrer com um atraso superior a 15 dias (e perde ainda 75% dos pontos acumulados); um cliente *Selection* passa a *Normal* se o pagamento ocorrer com um atraso de pagamento superior a 2 dias (e perde ainda 90% dos pontos acumulados).

As multas e os descontos a aplicar no pagamento de uma venda dependem do estatuto do cliente e do período em que o pagamento é realizado ($P1, P_2, P_3$ e P_4):

- P_1 - independentemente do estatuto do cliente, a multa é 0 e o desconto é 10%;
- P_2 - para clientes com o estatuto *Normal* a multa e o desconto são ambos 0; para *Selection* a multa é 0 e desconto é 5% até até 2 dias antes da data limite e 0 após; para *Elite* a multa é 0 e o desconto é 10%;
- P_3 - para clientes com o estatuto *Normal* a multa é 5% diários e o desconto é 0%; para *Selection* o desconto é 0 e a multa é 0 até um dia depois da data limite e 2% diários após; para *Elite* a multa é 0 e o desconto é 5%;
- P_4 - para clientes com o estatuto *Normal* a multa é 10% diários e o desconto é 0%; para *Selection* o desconto é 0 e a multa é 5% diários; para *Elite* a multa e o desconto são 0;

1.7 Gestão de Tempo

A data é representada por um número inteiro e tem inicialmente o valor 0 (zero). A data pode começar com outro valor se se recuperar o estado da aplicação a partir de um suporte persistente. Os avanços de data são valores inteiros que representam o número de dias a avançar.

2 Requisitos de Desenho

Devem ser possíveis extensões ou alterações de funcionalidade com impacto mínimo no código já produzido para a aplicação. O objectivo é aumentar a flexibilidade da aplicação relativamente ao suporte de novas funções. Assim, deve ser possível:

- Adicionar novos tipos de produtos;
- Definir novas entidades que desejem ser notificadas da alteração do estado dos produtos;
- Adicionar novos modos de entrega de mensagens (notificações);
- Adicionar novas políticas de recompensa de clientes;
- Adicionar novas formas de consulta.

Embora na especificação actual não seja possível remover entidades, a inclusão desta funcionalidade deve ser prevista, por forma a minimizar o impacto da sua futura inclusão.

3 Funcionalidade da Aplicação

A aplicação permite gerir a informação sobre as entidades do modelo, Possui ainda a capacidade de preservar o seu estado (não é possível manter várias versões do estado da aplicação em simultâneo). Inicialmente, a aplicação apenas tem informação sobre as entidades que foram carregados no arranque da aplicação.

Note-se que não é necessário concretizar a aplicação de raiz . Já é fornecido algum código, nomeadamente, o esqueleto das classes necessárias para concretizar os menus e respectivos comandos a utilizar na aplicação a desenvolver, a classe `woo.app.App`, que representa o ponto de entrada da aplicação a desenvolver, algumas classes do domínio da aplicação e um conjunto de excepções a utilizar durante o desenvolvimento da aplicação. A interface geral do core já está parcialmente concretizada na classe `woo.core.StoreManager` e outras fornecidas (cujos nomes devem ser mantidos), devendo ser adaptadas onde necessário.

A classe `woo.core.StoreManager` deverá ser finalizada por cada grupo e deverá representar a interface geral do domínio da aplicação desenvolvida. Por esta razão, os comandos a desenvolver na camada de interface com o utilizador devem utilizar exclusivamente esta classe para aceder às funcionalidades suportadas pelas classes do domínio da aplicação (a serem desenvolvidas por cada grupo na package `woo.core`). Desta forma será possível concretizar toda a interacção com o utilizador completamente independente da concretização das entidades do domínio. As excepções a criar na camada de serviços (ou seja nos comandos) já estão todas definidas no package `woo.app.exception`. O package `woo.core.exception` contém algumas excepções a utilizar na camada do domínio. Caso seja necessário podem ser definidas novas excepções neste package para representar situações anómalas que possam ocorrer nas entidades do domínio. Finalmente, será ainda necessário concretizar de raiz algumas classes do domínio da aplicação necessárias para suportar a operação da aplicação.

3.1 Serialização

É possível guardar e recuperar o estado actual da aplicação, preservando toda a informação relevante do domínio da aplicação e que foi descrita na secção 1.

4 Interação com o utilizador

Esta secção descreve a **funcionalidade máxima** da interface com o utilizador. Em geral, os comandos pedem toda a informação antes de proceder à sua validação (excepto onde indicado). Todos os menus têm automaticamente a opção **Sair** (fecha o menu).

As operações de pedido e apresentação de informação ao utilizador **devem** realizar-se através dos objectos *form* e *display*, respectivamente, presentes em cada comando. No caso de um comando utilizar mais do que um formulário para interagir com o utilizador então será necessário criar pelo menos mais uma instância de `Form` durante a execução do comando em causa. As mensagens são produzidas pelos métodos das bibliotecas de suporte (**po-uuilib** e **woo-app**). **Não** podem ser definidas novas mensagens. Potenciais omissões devem ser esclarecidas com o corpo docente antes de qualquer concretização.

Por forma a garantir uma independência entre as várias camadas da aplicação não deve haver código de interacção com o utilizador no núcleo da aplicação (*woo.core*). Desta forma, será possível reutilizar o código do núcleo da aplicação (onde é concretizado o domínio da aplicação) com outras concretizações da interface com o utilizador. Para garantir um melhor desenho da aplicação toda alógica de negócio deve estar concretizada no núcleo da aplicação e não na camada de interação com o utilizador. Assim potencia-se a reutilização de código e o código está concretizado nas entidades a que diz respeito.

As excepções usadas no código de interacção com o utilizador para descrever situações de erro, excepto se indicado, são subclasses de `pt.tecnico.po.ui.DialogException` e devem ser lançadas pelos comandos (sendo depois tratadas automaticamente pela classe já existente `pt.tecnico.po.ui.Menu`). Estas excepções já estão definidas no package (fornecido) `woo.app.exception`. Outras excepções não devem substituir as fornecidas nos casos descritos.

Note-se que o programa principal e os comandos e menus, a seguir descritos, já estão parcialmente concretizados no package `woo.app` e nos seus sub-packages (por exemplo, `woo.app.main`). Estas classes são de uso obrigatório e estão disponíveis na secção *Projecto* da página da cadeira.

A apresentação de listas (e.g., Fornecedores, Transacções, etc.) faz-se por ordem crescente da respectiva chave. a ordem pode ser numérica ou lexicográfica (UTF-8), não havendo distinção entre maiúsculas e minúsculas.

Sempre que existe uma interacção com o utilizador, seja para pedir dados seja para apresentar dados, é indicado (caso seja necessário) qual é o método da classe `Message` do package em causa que é responsável por devolver a mensagem a apresentar ao utilizador.

4.1 Menu Principal

As acções deste menu permitem gerir a salvaguarda do estado da aplicação, ver e alterar a data actual e abrir submenus. A lista completa é a seguinte: **Abrir**, **Guardar** , **Mostrar data actual**, **Avançar data actual**, **Gestão de Produtos**, **Gestão de Clientes**, **Gestão de Fornecedores**, **Gestão de Transacções**, **Consultas** e **Mostrar Saldo Global**. As etiquetas das opções deste menu estão definidas na classe `woo.app.main.Label`. Todos os métodos correspondentes às mensagens de diálogo para este menu estão definidos na classe `woo.app.main.Message`.

Os comandos que vão concretizar as funcionalidades deste menu já estão parcialmente concretizados nas várias classes do package `woo.app.main`: `DoOpen`, `DoSave`, `DoDisplayDate`, `DoAdvanceDate`, `DoOpenMenuProducts`, `DoOpenMenuClients`, `DoOpenMenuSuppliers`, `DoOpenMenuTransactions`, `DoOpenMenuLookups` e `DoShowGlobalBalance`.

4.1.1 Salvaguarda do estado actual

Inicialmente, a aplicação está vazia ou tem apenas informação sobre as entidades que foram carregados no arranque via ficheiro textual (ver 5).

O conteúdo da aplicação (que representa o estado relevante actualmente em memória da aplicação) pode ser guardado para posterior recuperação (via serialização Java: `java.io.Serializable`). Na leitura e escrita do estado da aplicação, devem ser tratadas as excepções associadas. A funcionalidade é a seguinte:

Abrir – Carrega os dados de uma sessão anterior a partir de um ficheiro previamente guardado (ficando este ficheiro associado à aplicação, para futuras operações de salvaguarda). Pede-se o nome do ficheiro a abrir (utilizando a cadeia de caracteres devolvida por `openFile()`). Caso ocorra um problema na abertura ou processamento do ficheiro, deve ser lançada a excepção `FileOpenFailedException`. A execução bem sucedida desta opção substitui toda a informação da aplicação.

Guardar – Guarda o estado actual da aplicação (inclui todas as entidades do domínio da aplicação) no ficheiro associado. Se não existir associação, pede-se o nome do ficheiro a utilizar (utilizando a cadeia de caracteres devolvida por `newSaveAs()`), ficando a ele associado.

Note-se que a opção `Abrir` não suporta a leitura de ficheiros de texto (estes apenas são utilizados na inicialização da aplicação). A opção `Sair` **nunca** guarda o estado da aplicação, mesmo que existam alterações.

4.1.2 Mostrar data actual

A data actual do sistema é apresentada através da mensagem `currentDate()`.

4.1.3 Avançar data actual

O número de dias a avançar é pedido utilizando a mensagem devolvida por `requestDaysToAdvance()`. O valor indicado deve ser positivo. Caso contrário, a operação deve lançar a excepção `InvalidDateException`.

4.1.4 Gestão e consulta de dados da aplicação

A gestão e consulta de dados da aplicação são realizadas através das seguintes opções:

Menu de Gestão de Produtos – Abre o menu de gestão de produtos.

Menu de Gestão de Clientes – Abre o menu de gestão de clientes.

Menu de Gestão de Fornecedores – Abre o menu de gestão de fornecedores.

Menu de Gestão de Transacções – Abre o menu de gestão de transacções.

Menu de Consultas – Abre o menu de consultas (pesquisas).

4.1.5 Mostrar saldo global

Esta opção apresenta os valores (inteiros) correspondentes aos saldos disponivel e contabilístico da empresa. Embora internamente o valor dos saldos esteja representado em vírgula flutuante, a apresentação é arredondada ao inteiro mais próximo.

A apresentação da mensagem dos saldos a apresentar ao utilizador faz-se utilizando a mensagem devolvida por `currentBalance()`.

4.2 Menu de Gestão de Produtos

Este menu permite efectuar operações sobre a base de dados de produtos oferecidos pela empresa. A lista completa é a seguinte: **Visualizar todos os produtos**, **Registar caixa**, **Registar contentor**, **Registar livro** e **Alterar preço de produto**.

As etiquetas das opções deste menu estão definidas na entidade `woo.app.products.Label`. Todos os métodos correspondentes às mensagens de diálogo para este menu estão definidos em `woo.app.products.Message`.

Sempre que no contexto de uma operação for pedido o identificador de um produto (utilizando a mensagem devolvida por `requestProductKey()`) e o produto não existir, a operação deve lançar a excepção `UnknownProductKeyException` (excepto no processo de registo). No processo de registo, caso o identificador do produto indicado já exista, a operação deve lançar a excepção `DuplicateProductKeyException`. Na ocorrência de excepções (estas ou outras), as operações não têm efeito. Após o registo com sucesso de um produto, o seu número de existências é zero. Sempre que for pedido o identificador do fornecedor (utilizando a mensagem devolvida por `requestSupplierKey()`) e o fornecedor não existir, a operação lança a excepção `UnknownSupplierKeyException`.

Os comandos deste menu já estão parcialmente concretizados nas classes do package `woo.app.products`: `DoShowAllProducts`, `DoRegisterProductBox`, `DoRegisterProductContainer`, `DoRegisterProductBook` e `DoChangePrice`.

4.2.1 Visualizar todos os produtos

Apresenta todos os produtos disponibilizados pela empresa, um produto por linha. O formato de apresentação de cada tipo de produto é o seguinte:

```
BOX|idProduto|idFornecedor|preço|valor-crítico|stock-actual|tipo-de-serviço
CONTAINER|idProduto|idFornecedor|preço|valor-crítico|stock-actual|tipo-de-serviço|nível-de-serviço
BOOK|idProduto|idFornecedor|preço|valor-crítico|stock-actual|título|autor|isbn
```

4.2.2 Registar caixa

O sistema pede o identificador do produto a criar, o preço (`requestPrice()`), o valor crítico (`requestStockCriticalLevel()`) e o identificador do fornecedor (`requestSupplierKey()`) e o tipo de serviço de transporte associado (`requestServiceType()`).

Se a resposta para o tipo de serviço não for *NORMAL*, *AIR, EXPRESS* ou *PERSONAL*, então esta operação lança a excepção `UnknownServiceTypeException`.

4.2.3 Registar contentor

São pedidas as mesmas informações que para o registo de caixas e verificadas as mesmas condições. Além daquelas informações, é ainda pedida a qualidade de serviço (`requestServiceLevel()`). Se a resposta não for *B4, C4, C5* ou *DL*, então esta operação lança a excepção `UnknownServiceLevelException`.

4.2.4 Registar livro

O sistema pede o identificador único do livro a criar, o título (`requestBookTitle()`), o autor (`requestBookAuthor()`), o ISBN (`requestISBN()`), o preço (`requestPrice()`), o valor crítico (`requestStockCriticalLevel()`) e o identificador do fornecedor (`requestSupplierKey()`).

4.2.5 Alterar preço de produto

O sistema pede o identificador do produto e o novo preço (`requestPrice()`). Tal como mencionado acima, os clientes podem ser notificados quando o preço de um produto varia. Caso ocorra um erro durante a alteração do preço de um produto existente, a operação falha silenciosamente.

4.3 Menu de Gestão de Clientes

Este menu permite efectuar operações sobre a base de dados de clientes. A lista completa é a seguinte: `Mostrar cliente`, `Mostrar todos os clientes`, `Registar cliente`, `Activar/desactivar notificações de um produto`, `Mostrar transacções do cliente`.

As etiquetas das opções deste menu estão definidas na classe `woo.app.clients.Label`. Todos os métodos correspondentes às mensagens de diálogo para este menu estão definidos na classe `woo.app.clients.Message`.

Sempre que for pedido o identificador de um cliente (`requestClientKey()`) e o cliente não existir, é lançada a excepção `UnknownClientKeyException` (excepto no processo de registo). No processo de registo, caso o identificador indicado já exista, deve ser lançada a excepção `DuplicateClientKeyException`.

Os comandos deste menu já estão parcialmente concretizados nas classes do package `woo.app.clients`: `DoShowClient`, `DoShowAllClients`, `DoRegisterClient`, `DoToggleProductNotifications` e `DoShowClientTransactions`.

4.3.1 Mostrar cliente

É pedido o identificador do cliente e apresentada a sua informação. O formato de apresentação é o seguinte (o valor das compras efectuadas refere-se ao momento da compra; o valor das compras pagas refere-se ao valor realmente pago):

```
id|nome|endereço|valor-compras-efectuadas|valor-compras-pagas
```

Após esta linha, são apresentadas as notificações do cliente recebidas através do modo de entrega por omissão utilizando o seguinte formato:

```
tipo-de-notificação|idProduto|preço-do-produto
```

As notificações devem ser apresentadas pela ordem em que foram enviadas pela aplicação. Após esta visualização, considera-se que o cliente fica sem notificações registadas.

4.3.2 Mostrar clientes

Apresenta informações sobre todos clientes. O formato de apresentação de cada obra segue o mesmo formato descrito na secção 4.3.1, mas não se apresentam as notificações dos clientes.

4.3.3 Registar cliente

São pedidos o identificador do cliente, o nome (`requestClientName()`) (uma cadeia de caracteres) e o endereço do cliente (`requestClientAddress()`) (cadeia de caracteres) e regista-se o novo cliente. Quando um cliente é registado, aceita notificações relativas a todos os produtos. Se já existir um cliente com o mesmo identificador, deve ser lançada a excepção `DuplicateClientKeyException`, não se realizando o registo.

4.3.4 Activar/desactivar notificações de um produto

São pedidos o identificador do cliente e o identificador do produto (`requestProductKey()`). Se as notificações relativas ao produto estavam activas, passam a estar inactivas, e vice-versa. É apresentada na saída o resultado da operação utilizando a mensagem devolvida por `notificationsOn()` ou `notificationsOff()`.

4.3.5 Mostrar transacções do cliente

É pedido o identificador do cliente e apresentadas todas as transacções por ele realizadas. O formato de apresentação é como descrito na secção 4.5.1.

4.4 Menu de Gestão de Fornecedores

Este menu apresenta as várias operações disponíveis sobre fornecedores: `Mostrar fornecedores`, `Registar fornecedor`, `Permitir/Inibir transacções`, `Mostrar transacções do fornecedor`.

As etiquetas das opções deste menu estão definidas na classe `woo.app.suppliers.Label`. Todos os métodos correspondentes às mensagens de diálogo para este menu estão definidos na classe `woo.app.suppliers.Message`.

Sempre que é pedido o identificador do fornecedor (`requestSupplierKey()`) e o fornecedor não existir (excepto no processo de registo), a operação em causa lança a excepção `UnknownSupplierKeyException`. No processo de registo, caso o identificador indicado já exista, deve ser lançada a excepção `DuplicateSupplierKeyException`.

Estes comandos já estão parcialmente concretizados nas seguintes classes do package `woo.app.suppliers`: `DoShowSuppliers`, `DoRegisterSupplier`, `DoToggleTransactions` e `DoShowSupplierTransactions`.

4.4.1 Mostrar fornecedores

Apresenta informações sobre todos fornecedores, um fornecedor por linha. O formato de apresentação de um fornecedor é o seguinte;

O formato de apresentação é o seguinte:

```
id|nome|endereço|activo?
```

Os valores para o campo activo? são os devolvidos por `yes()` ou `no()`.

4.4.2 Registar fornecedor

São pedidos o identificador do fornecedor, o nome (`requestSupplierName()`) (uma cadeia de caracteres) e o endereço (`requestSupplierAddress()`) (cadeia de caracteres), registando-se o novo fornecedor. Quando um fornecedor é registado, fica no estado activo. Se já existir um fornecedor com o mesmo identificador, a operação aborta lançando a excepção `DuplicateSupplierKeyException`.

4.4.3 Permitir/Inibir transacções

É pedido o identificador do fornecedor. Se as transacções estavam activas para esse fornecedor, passam a estar inactivas, e vice-versa. É apresentada na saída o resultado da operação: mensagem devolvida por `transactionsOn()` ou `transactionsOff()`.

4.4.4 Mostrar transacções do fornecedor

É pedido o identificador do fornecedor e apresentas todas as transacções (encomendas) por ele realizadas. O formato de apresentação de cada transacção é o descrito na secção 4.5.1

4.5 Menu de Gestão de Transacções

Este menu apresenta as operações relacionadas com transacções. A lista completa é a seguinte: `Visualizar`, `Registar Venda`, `Registar Encomenda` e `Pagar`. As etiquetas das opções deste menu estão definidas na classe `woo.app.transactions.Label`. Todos os métodos correspondentes às mensagens de diálogo utilizadas nas operações deste menu estão definidos na classe `woo.app.transactions.Message`.

Sempre que é pedido o identificador de um fornecedor (`requestSupplierKey()`) e o fornecedor indicado não existir, a operação em causa deve lançar a excepção `UnknownSupplierKeyException`. Sempre que é pedido o identificador de um cliente (`requestClientKey()`), é lançada a excepção `UnknownClientKeyException` se o cliente indicado não existir. Sempre que é pedido o identificador de produto (`requestProductKey()`), é lançada a excepção `UnknownProductKeyException`, se o produto indicado não existir. Sempre que é pedido o identificador da transacção (`requestTransactionKey()`), é lançada a excepção `UnknownTransactionKeyException`, se a transacção indicada não existir.

Estes comandos já estão parcialmente concretizaos nas classes do package `woo.app.transactions`: `DoShowTransaction`, `DoRegisterSaleTransaction`, `DoRegisterOrderTransaction`, e `DoPay`.

4.5.1 Visualizar

O sistema pede o identificador da transacção a visualizar.

A apresentação de uma transacção consiste num cabeçalho (na linha inicial) seguido por uma ou mais linhas, cada uma com a descrição de cada produto incluído na transacção. A apresentação de cada produto segue o formato:

```
idProduto|quantidade
```

O cabeçalho de uma transacção relativa a uma venda a um cliente tem o seguinte formato:

```
id|idCliente|idProduto|quantidade|valor-base|valor-a-pagamento|data-limite|paga?|data-pagamento
```

O campo `valor-a-pagamento` corresponde ao valor que será realmente pago (considerando possíveis multas/descontos). Os valores para o campo `paga?` são as mensagens devolvidas por `yes()` ou `no()`. O campo valor-base é o valor da transacção sem multas/descontos.

Se a transacção corresponder a uma encomenda a um fornecedor, então o cabeçalho tem o seguinte formato:

```
id|idFornecedor|valor-da-encomenda|data-pagamento
```

4.5.2 Registar Venda

Para registar uma venda, é pedido o identificador do cliente, a data limite para o pagamento (`requestPaymentDeadline()`), o identificador do produto a vender e a respectiva quantidade (`requestAmount()`). Se a quantidade for superior às existências actuais, esta operação aborta, lançando a excepção `UnavailableProductException` e a venda não é realizada.

A actualização dos produtos da empresa tem lugar logo após o registo da venda, ou seja, considera-se que a venda é instantânea.

4.5.3 Registar Encomenda

Para registar uma encomenda, é pedido o identificador do fornecedor. De seguida, é pedido o identificador do produto a encomendar e a quantidade a encomendar (`requestAmount()`). Estas duas perguntas são repetidas para outros produtos enquanto a resposta à mensagem devolvida por `requestMore()` (pergunta feita depois de cada quantidade) for afirmativa (leitura de um boleano).

Esta operação lança a excepção `UnauthorizedSupplierException` caso o fornecedor indicado esteja inibido de efectuar transacções. Se o identificador de um produto indicado não for fornecido pelo fornecedor escolhido, então esta operação lança a excepção `WrongSupplierException`. Em ambos os casos a operação não deve alterar o estado da aplicação.

A actualização de existências dos produtos da empresa tem lugar logo após o registo da encomenda, ou seja, considera-se que a encomenda é instantânea. A actualização do saldo da empresa também é assumida como instantânea, i.e., assume-se que a encomenda é paga a pronto.

4.5.4 Pagar

É pedido o identificador da venda a pagar. Se a venda já tiver sido paga, não é realizada nenhuma acção.

4.5.5 Efectuar pesquisa

Assim, considerando as quatro obras no exemplo anterior, uma pesquisa pelo termo **casa** apresentaria a obras com os identificadores 3, 4 e 5:

```
3 - 20 de 23 - Livro - Casa Azul - 15 - Ficção - João Fonseca - 1234567891
4 - 2 de 2 - DVD - Casamento Real - 8 - Ficção - António Fonseca - 200400500
5 - 0 de 4 - Livro - Dicionário - 45 - Referência - Pedro Casanova - 1234567893
```

Caso não sejam encontradas obras, não deve ser produzido qualquer resultado.

4.6 Menu de Consultas

Este menu apresenta as operações relacionadas com consultas. A lista completa é a seguinte: `Mostrar produtos com preço abaixo de limite` e `Mostrar facturas pagas por cliente`.

As etiquetas das opções deste menu estão definidas na classe `woo.app.lookups.Label`. Todos os métodos correspondentes às mensagens de diálogo para este menu estão definidos na classe `woo.app.lookups.Message`.

Sempre que é pedido o identificador do cliente (`requestClientKey()`), é lançada a excepção `UnknownClientKeyException`, se o cliente indicado não existir. Sempre que é pedido o identificador de produto (`requestProductKey()`), é lançada a excepção `UnknownProductKeyException`, se o produto indicado não existir. A apresentação de resultados é como indicado nos casos já descritos de apresentação das várias entidades.

Estes comandos já estão parcialmente implementados nas classes do package `woo.app.lookups`: `DoLookupPaymentsByClient` e `DoLookupProductsUnderTopPrice`.

4.6.1 Mostrar produtos com preço abaixo de limite

Pede-se o valor limite pretendido (requestPriceLimit()) e apresentam-se todos os produtos disponíveis na empresa cujo preço é inferior ao preço indicado. É apresentado um produto por linha.

4.6.2 Mostrar facturas pagas por cliente

Pede-se o identificador do cliente e apresentam-se as transacções do cliente que já estão pagas, uma transacção por linha.

5 Leitura de Dados a Partir de Ficheiros Textuais

Além das opções de manipulação de ficheiros descritas na secção §4.1.1, é possível iniciar a aplicação com um ficheiro de texto especificado pela propriedade Java com o nome `import`. Este ficheiro contém a descrição dos fornecedores, clientes e produtos a carregar no estado inicial da aplicação. Cada linha deste ficheiro textual descreve uma dada entidade a carregar no estado inicial do sistema. Pode assumir que não existem entradas mal-formadas nestes ficheiros. A codificação dos ficheiros a ler é garantidamente `UTF-8`.

As várias entidades têm os formatos descritos abaixo. Assume-se que os títulos não podem conter o carácter — e que o preço é um número inteiro. Cada linha tem uma descrição distinta mas segue um dos seguintes formatos:

```
SUPPLIER|id|nome|endereço
CLIENT|id|nome|endereço
BOX|id|tipo-de-serviço|id-fornecedor|preço|valor-crítico|exemplares
CONTAINER|id|tipo-de-serviço|nível-de-serviço|id-fornecedor|preço|valor-crítico|exemplares
BOOK|id|título|autor|isbn|id-fornecedor|preço|valor-crítico|exemplares
```

As definições de fornecedores e de clientes precedem sempre as dos produtos. Sugere-se a utilização do método `String.split` para o processamento preliminar destas linhas. De seguida descreve-se o conteúdo de um possível ficheiro textual:

```
SUPPLIER|S1|Toshiba|Tokyo, Japan
SUPPLIER|W2|Papelaria Fernandes|Oeiras, Portugal
SUPPLIER|P1|Publicações Europa-América|Lisboa, Portugal
SUPPLIER|P3|O'Reilly|Köln, Germany
CLIENT|R2|Jorge Figueiredo|Lisboa, Portugal
CLIENT|E4|Filomena Figueiredo|Lisboa, Portugal
CLIENT|ER|Abdul Figueiredo|Casablanca, Morocco
CLIENT|O9|Hellen Figueiredo|San Francisco, CA, USA
CLIENT|H2SO4|John Figueiredo|Wellington, New Zealand
```

```
CLIENT|H2O|Rohit␣Figueiredo|New␣Delhi,␣India
BOX|C6H5OH|NORMAL|W2|2|20|100
BOX|H2|AIR|W2|4|20|100
BOX|O3|EXPRESS|W2|8|20|100
BOX|CO2|PERSONAL|W2|16|20|0
CONTAINER|M4|NORMAL|B4|W2|2|20|100
CONTAINER|M2|AIR|C4|W2|4|20|100
CONTAINER|M5|EXPRESS|B4|W2|8|20|100
CONTAINER|M3|PERSONAL|DL|W2|16|20|0
BOOK|B1256|Os␣Lusíadas|Luís␣de␣Camões|1234567890|P1|58|2|5
BOOK|B9854|Head␣First␣Java|Sierra␣&␣Bates|9876543210|P3|75|2|5
BOOK|B1937|How␣to␣fix␣almost␣everything|Satoshi␣Yamada|1928374650|S1|5|2|5
```

Note-se que o programa **nunca** produz ficheiros com este formato.

6 Execução dos Programas e Testes Automáticos

Usando os ficheiros `test.import`, `test.in` e `test.out`, é possível verificar automaticamente o resultado correcto do programa. Note-se que pode ser necessária a definição apropriada da variável de ambiente `CLASSPATH` (ou da opção equivalente `-cp` do comando `java`), para localizar as classes do programa, incluindo a que contém o método correspondente ao ponto de entrada da aplicação (`woo.app.App.main`. As propriedades são tratadas automaticamente pelo código de apoio.

```
java -Dimport=test.import -Din=test.in -Dout=test.outhyp woo.app.App
```

Assumindo que aqueles ficheiros estão no directório onde é dado o comando de execução, o programa produz o ficheiro de saída `test.outhyp`. Em caso de sucesso, os ficheiros das saídas esperada (`test.out`) e obtida (`test.outhyp`) devem ser iguais. A comparação pode ser feita com o comando:

```
diff -b test.out test.outhyp
```

Este comando não deve produzir qualquer resultado quando os ficheiros são iguais. Note-se, contudo, que este teste não garante o correcto funcionamento do código desenvolvido, apenas verifica alguns aspectos da sua funcionalidade.

7 Notas de Concretização

Tal como indicado neste documento, algumas classes fornecidas como material de apoio, são de uso obrigatório e **não** podem ser alteradas: as várias classes `Message`, `Label` e de excepção presentes em diferentes subpackages de `woo.app`. Outras dessas classes são de uso obrigatório e têm de ser alteradas: os diferentes comandos presentes em subpackages de `woo.app` e algumas classes do domínio da aplicação já parcialmente concretizadas em `woo.core`.

A serialização Java usa as classes da package `java.io`, em particular, a interface`java.io.Serializable` e as classes de leitura `java.io.ObjectInputStream` e escrita `java.io.ObjectOutputStream` (entre outras).