Contents

ÉPOCA NORMAL

O objectivo do projecto é desenvolver uma aplicação de gestão do inventário de um entreposto de recursos naturais e seus derivados (formados por um ou mais produtos ou seus derivados). O entreposto concede prémios de fidelização aos bons parceiros (que lhe compram e vendem produtos), baseando-se no volume de negócio (compras e vendas). A funcionalidade

da aplicação inclui, entre outras acções, manipular dados de produtos para negociar, registar/manipular dados de parceiros, registar/manipular transacções de compra e venda e outras e fazer pesquisas várias sobre a informação armazenada.

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.

Entreposto, Produtos, Parceiros e Transacções

Os produtos e os parceiros possuem uma chave única (cadeia de caracteres, não havendo distinção entre maiúsculas e minúsculas). As transacções possuem uma chave única inteira gerida automaticamente.

O saldo inicial do entreposto é 0 (zero).

Propriedades e Funcionalidade dos Produtos

Um produto pode ser simples ou derivado. Um produto derivado tem uma receita de fabrico. A receita corresponde à discriminação dos identificadores dos componentes (simples ou derivados) que formam o produto e respectivas quantidades. Não é possível a definição de receitas cujos componentes não sejam previamente conhecidos.

Os produtos, tanto simples, como derivados, são comprados, vendidos, ou fabricados (no caso dos produtos derivados) em lotes. Cada lote tem um fornecedor (o parceiro a quem é comprado o produto), o número de unidades disponíveis do produto no lote e o preço de cada unidade. Podem existir vários lotes do mesmo produto, com o mesmo fornecedor, com preços iguais ou diferentes. O preço de um produto é sempre um número em vírgula flutuante não negativo. Quando um lote é esgotado, é removido do registo de dados.

O preço dos produtos é definido pelo parceiro no acto da compra (pelo entreposto). Este processo é uniforme para todos os produtos, excepto nas operações indicadas (ver a seguir e abaixo).

O entreposto pode realizar operações de agregação e de desagregação. Uma operação de agregação consiste em criar um produto derivado a partir das existências dos seus componentes. O preço dos produtos derivados, resultantes de agregações, é calculado com base nos preços dos componentes, de acordo com a receita, tendo um agravamento multiplicativo (número positivo em vírgula flutuante), definido pela receita, relativo ao valor acrescentado associado à combinação. Por exemplo: a água é feita de hidrogénio (2x) e de oxigénio (1x). O seu preço é: $P_{H_2O} = (1 + \alpha) \times (2 \times P_H + P_O)$ (onde P_x são preços e α é o factor de agravamento, e.g. 0.1, correspondente a um aumento de 10% sobre o valor dos componentes). A reserva de recursos para construção de determinada quantidade de produto derivado contempla a totalidade da construção (e.g., não é possível respeitar um pedido de 10 unidades de água se apenas estiverem disponíveis 8 unidades de oxigénio). Em geral, operações de consumo de existências começam sempre pelas existências de menor preço, consumindo um ou mais lotes, total ou parcialmente, no processo.

É possível desagregar um produto derivado e recuperar os seus componentes. Não há perdas de produtos nesta operação, mas pode haver perda de valor: o valor do produto derivado pode ser superior à soma dos valores dos preços dos seus componentes. O parceiro que pede a operação de desagregação paga o diferencial de valor, caso seja positivo, sendo registada uma transacção de desagregação (com o valor do diferencial, mesmo que negativo).

Propriedades e Funcionalidade dos Parceiros

Cada parceiro tem um nome (cadeia de caracteres) e uma morada (cadeia de caracteres). Associada a cada parceiro existe ainda informação relativa às suas transacções.

Um parceiro tem ainda um estatuto (e.g., **Elite**, etc- -- ver abaixo), ao qual está associada contabilização de pontos. O estatuto tem impacto na relação do parceiro com o entreposto.

Dado um parceiro, é possível aceder ao historial das suas transacções (ver a seguir).

Propriedades e Funcionalidade das Transacções

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 todos os tipos de transacções. Todas as transacções têm uma data de pagamento.

O entreposto realiza com os parceiros transacções de compra (do entreposto aos parceiros), venda (do entreposto aos parceiros) e desagregações (pelo entreposto a pedido dos parceiros). As desagregações: podem ser vistas como compras e vendas combinadas, para efeitos de progressão do estatuto dos parceiros. Uma desagregação corresponde a uma venda do produto a desagregar e à compra dos componentes que resultam da desagregação.

As compras são sempre pagas instantaneamente. As vendas podem ser pagas pelos parceiros mais tarde (excepto no contexto de uma desagregação: nesse caso, são pagas imediatamente).

Compras (do entreposto a parceiros)

Uma compra está associada a um parceiro e envolve uma ou mais unidades de um produto fornecido pelo parceiro em causa. O custo unitário do produto é definido pelo parceiro no momento da transacção. Quando se faz uma compra deve considerar-se que a compra é paga imediatamente e que as existências do entreposto são actualizadas considerando os produtos comprados. É mantida informação sobre cada produto, o parceiro que o forneceu e o preço que foi pago.

Vendas (do entreposto a parceiros)

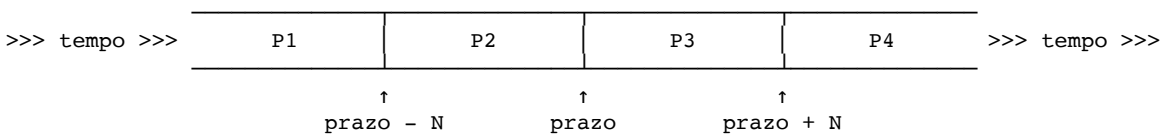
Uma venda está associada a um parceiro e envolve uma ou mais unidades de um único produto. As unidades podem ser provenientes de vários lotes e ter preços diferentes, dependendo dos parceiros que forneceram o produto. Cada venda tem uma data limite de pagamento: primeiro realiza-se a venda e só depois é que se procede ao seu pagamento.

Se o produto a adquirir for derivado e não existir em quantidade suficiente no entreposto, a quantidade em falta pode ser fabricada a partir de outros produtos, de acordo com a receita correspondente ao produto em venda. O preço a pagar é o definido pela reserva dos componentes individuais (com a selecção do preço mais baixo, etc.) e o correspondente factor de valor acrescentado (ver acima). Note-se que, na venda de múltiplas unidades de um produto derivado, o processo de construção deve ser repetido, pelo que cada uma das unidades do produto derivado pode ter custo diferente das anteriores. Caso falte algum componente da receita, então a operação de agregação deve ser abortada indicando o primeiro produto (pela ordem da receita) que viola a condição de disponibilidade.

O preço a pagar por um parceiro (numa operação de venda por parte do entreposto) depende ainda do tempo que demora a realizar o pagamento. São considerados os seguintes períodos (**N** é 5 para produtos simples, 3 para produtos derivados):

- **P1** - até **N** dias antes do limite de pagamento (**data_limite_de_pagamento – data_actual ≥ N**).
- **P2** - até à data limite (**0 ≤ data_limite_de_pagamento – data_actual < N**).
- **P3** - até **N** dias depois da data limite (**0 < data_actual – data_limite_de_pagamento ≤ N**).
- **P4** - após **N** dias depois da data limite (**data_actual – data_limite_de_pagamento > N**).

Intervalos ao longo do tempo:



Note-se que estes períodos são arbitrários e não correspondem a qualquer conceito especial de tempo, podendo ser redefinidos ou mesmo eliminados em revisões do processo de venda.

Desagregações de produtos (pelo entreposto a pedido de parceiros)

Uma desagregação é uma transacção associada a um parceiro e envolve um único produto. Um parceiro pode pedir uma desagregação para permitir outras operações sobre os componentes do produto a desagregar. Se o produto a desagregar for simples, a acção não tem qualquer efeito.

As desagregações são operações sobre o estado do entreposto por parte de um parceiro: a desagregação remove uma quantidade de produto derivado das existências do entreposto (como se fosse uma venda) e introduz nas existências do entreposto os seus componentes (como se fosse uma compra ao parceiro envolvido). O preço de inserção é o menor preço nos lotes disponíveis para esse produto. Se não existirem lotes disponíveis, o preço é o maior preço associado a esse produto no histórico de operações do entreposto. O parceiro paga ao entreposto a diferença (positiva) entre os valores inicial (o valor da quantidade de produto não desagregado) e final (o valor das quantidades de componentes obtidas com base na desagregação).

Notificações

Quando o entreposto recebe um novo produto, os parceiros devem ser colocados como entidades interessadas em receber notificações sobre eventos a ele associados. Em qualquer momento, um parceiro 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 aparece um lote mais barato de um produto. As notificações são compostas pelo identificador do produto e pela descrição da notificação: **NEW**, para novas existências de produtos, mas não quando se regista um novo produto; e **BARGAIN**, para disponibilidade de preços mais baixos. As notificações são registadas nos parceiros 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.

Contabilização de Pontos (Parceiros)

As multas e os descontos aplicam-se apenas no pagamento de transacções de venda.

Os valores pagos (instantaneamente) nas transacções de desagregação também dão direito à contabilização de pontos (apenas quando é positivo).

Existem três classificações distintas de parceiros: **Nomal**, **Selection** e **Elite**. A classificação de um parceiro tem impacto nas multas e descontos a aplicar no pagamento de uma venda.

Quando um parceiro paga uma venda 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 venda.

Os parceiros passam ao nível **Selection** se acumularem mais de 2000 pontos. Os parceiros passam ao nível **Elite** se acumularem mais de 25000 pontos.

Se um parceiro se atrasa no pagamento da venda, é despromovido: um parceiro **Elite** passa a **Selection** se o pagamento ocorrer com um atraso de pagamento superior a 15 dias (perde 75% dos pontos acumulados); um parceiro **Selection** passa a **Normal** se o pagamento ocorrer com um atraso de pagamento superior a 2 dias (perde 90% dos pontos acumulados). Se um parceiro **Normal** se atrasa num pagamento, perde os pontos que tem.

As multas e os descontos dependem do estatuto do parceiro e dos prazos associados à venda e ao estatuto.

	P1		**P2**		**P3**		**P4**	
	Multa	Desconto	Multa	Desconto	Multa	Desconto	Multa	Desconto
Normal	0	10%	0	0	5% diários	0	10% diários	0
Selection	0	10%	0	≥ 2 dias antes da data limite: 5%; depois, sem desconto	> 1 dia depois da data limite: 2% diários (0, caso contrário)	0	5% diários	0
Elite	0	10%	0	10%	0	5%	0	0

Data

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 do entreposto a partir de um suporte persistente.

Os avanços de data são valores inteiros positivos que representam o número de dias.

Funcionalidade da aplicação

A aplicação permite manter 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).

A base de dados com os conceitos pré-definidos é carregada no início da aplicação.

É possível saber os saldos do entreposto (diferencial entre vendas e compras). Existe um saldo disponível, correspondente à diferença entre as vendas realmente pagas e as compras, 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 compras. Note-se que as desagregações são consideradas combinações de compras e vendas, pelo que também influenciam o saldo do entreposto.

Deve ser possível efectuar pesquisas sujeitas a vários critérios e sobre as diferentes entidades geridas pelo entreposto.

 Note-se que não é necessário/desejável implementar de raiz a aplicação: já existem classes que representam e definem a interface geral da funcionalidade do *core* da aplicação, tal como é visível pelos comandos da aplicação.

 A interface geral do *core* já está parcialmente implementada na classe **ggc.WharehouseManager** e outras fornecidas (cujos nomes devem ser mantidos), devendo ser adaptadas onde necessário. É ainda necessário criar e implementar as restantes classes que suportam a operação da aplicação.

Serialização

É possível guardar e recuperar o estado actual da aplicação, preservando toda a informação relevante, descrita acima.

Funcionalidade Associada a Entidades do Domínio

A seguinte funcionalidade sobre produtos deve ser suportada pela aplicação: (i) visualizar produtos e lotes de produtos; (ii) visualizar lotes específicos.

A seguinte funcionalidade sobre parceiros deve ser suportada pela aplicação: (i) visualizar um ou mais parceiros; (ii) registar um novo parceiro; (iii) activar/desactivar notificações relativas a produtos; (iv) consultar o histórico de transacções realizadas por um parceiro.

A seguinte funcionalidade sobre transacções deve ser suportada: (i) visualizar uma transacção; (ii) registar uma nova compra; (ii) registar uma nova venda; (iii) registar uma desagregação; (iv) receber pagamento de uma venda.

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:

- 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 parceiros;
- 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.

Interacção com o utilizador

Descreve-se nesta secção a **funcionalidade máxima** da interface com o utilizador. Em geral, os comandos pedem toda a informação antes de procederem 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. As mensagens são produzidas pelos métodos das bibliotecas de suporte (**po-uilib** e **ggc-app**). As mensagens não podem ser usadas no núcleo da aplicação (**ggc-core**). Além disso, não podem ser definidas novas. Potenciais omissões devem ser esclarecidas antes de qualquer implementação.

A apresentação de valores monetários (preços, saldos, etc.) é sempre feita com arredondamento ao inteiro mais próximo.

As excepções usadas na interacção (subclasses de **pt.tecnico.uilib.menus.CommandException**), excepto se indicado, são lançadas pelos comandos (subclasses de **pt.tecnico.uilib.menus.Command**) e tratadas pelos menus (instâncias de subclasses de **pt.tecnico.uilib.menus.Menu**). Outras excepções não devem substituir as fornecidas nos casos descritos.

A apresentação de listas de entidades do domínio (parceiros, transacções, etc.) faz-se por ordem crescente da respectiva chave: dependendo dos casos, a ordem pode ser numérica ou lexicográfica (UTF-8), não havendo distinção entre maiúsculas e minúsculas.

Note-se que o programa principal e os comandos e menus, a seguir descritos, já estão parcialmente implementados nas *packages* **ggc.app**, **ggc.app.main**, **ggc.app.products**, **ggc.app.partners**, **ggc.app.transactions**, **ggc.app.lookups**. Estas classes são de uso obrigatório e estão disponíveis no CVS (módulo **ggc-app**).

Menu Principal

As acções deste menu permitem gerir a salvaguarda do estado da aplicação, abrir submenus e aceder a alguma informação global. A lista completa é a seguinte: Abrir, Guardar, Mostrar data actual, Avançar data actual, Gestão de Produtos, Gestão de Parceiros, Gestão de Transacções, Consultas, Mostrar Saldo Global.

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

Estes comandos já estão implementados nas classes da *package* **ggc.app.main** (disponível no CVS), respectivamente: **DoOpenFile**, **DoSaveFile**, **DoDisplayDate**, **DoAdvanceDate**, **DoShowGlobalBalance**.

Salvaguarda do estado actual da aplicação

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

O conteúdo da aplicação (toda a informação actualmente em memória) 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 (**Prompt.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 no ficheiro associado. Se não existir associação, pede-se o nome do ficheiro a utilizar, ficando a ele associado. Esta interacção realiza-se através do método **Prompt.newSaveAs()**. Não é executada nenhuma acção se não existirem alterações desde a última salvaguarda.

Note-se que a opção **Abrir** não permite a leitura de ficheiros de texto (estes apenas são utilizados na inicialização da aplicação).

A opção **Sair** nunca implica a salvaguarda do estado da aplicação, mesmo que existam alterações.

Mostrar data actual

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

Avançar data actual

O número de dias a avançar é pedido através de **Prompt.daysToAdvance()**. Deve ser lançada a excepção **InvalidDateException** se o número de dias a avançar for inválido.

Gestão e consulta de dados da aplicação

- **Menu de Gestão de Produtos** -- Abre o menu de gestão de produtos.
- **Menu de Gestão de Parceiros** -- Abre o menu de gestão de parceiros.
- **Menu de Gestão de Transacções** -- Abre o menu de gestão de transacções.
- **Menu de Consultas** -- Abre o menu de consultas (pesquisas).

Mostrar saldo global

Esta opção apresenta os valores correspondentes aos saldos disponível e contabilístico do entreposto. Embora internamente o valor dos saldos estejam representados em vírgula flutuante, a apresentação é arredondada ao inteiro mais próximo.

A apresentação faz-se através da mensagem **Message.currentBalance()**.

Menu de Gestão de Produtos

Este menu permite efectuar operações sobre a base de dados de produtos. A lista completa é a seguinte: Visualizar todos os produtos, Visualizar todos os produtos disponíveis, Visualizar os lotes fornecidos por um dado parceiro, Visualizar os lotes de um dado produto.

Todos os métodos correspondentes às mensagens de diálogo para este menu estão definidos na classe **ggc.app.products.Message**.

Sempre que é pedido o identificador de um parceiro (**Prompt.partnerKey()**) e o parceiro não existir, é lançada a excepção **UnknownPartnerKeyException**. Sempre que é pedido o identificador de um produto (**Prompt.productKey()**) e o produto não existir, é lançada a excepção **UnknownProductKeyException**. Na ocorrência de excepções (estas ou outras), as operações não têm efeito.

Estes comandos já estão implementados nas classes da *package* **ggc.app.products** (disponível no CVS), respectivamente: **DoShowAllProducts**, **DoShowAvailableBatches**, **DoShowBatchesByPartner**, **DoShowBatchesByProduct**.

Visualizar todos os produtos

O formato de apresentação de cada produto é o seguinte (cada linha indica um produto):

Produtos simples:

```
idProduto|preço-máximo|stock-actual-total
```

Exemplo:

```
HIDROGÉNIO|200|5000
OXIGÉNIO|1200|2500
```

Produtos derivados (é apresentado o produto e a lista de componentes e respectivas quantidades da sua receita):

```
idProduto|preço-máximo|stock-actual-total|agravamento|componente-1:quantidade-1#...#componente-n:quantidade-n
```

Exemplo:

```
ÁGUA|5000|800|0.1|HIDROGÉNIO:2#OXIGÉNIO:1
```

Em ambos os casos, *stock-actual-total* pode ser 0 (zero).

Visualizar todos os lotes

O formato de apresentação de cada lote de produto é o seguinte (cada linha indica um lote):

Produtos simples e derivados:

```
idProduto|idParceiro|preço|stock-actual
```

Exemplo:

```
ÁGUA|EPAL|1150|800
HIDROGÉNIO|Cryostuffs|110|5000
OXIGÉNIO|Cryostuffs|650|2500
OXIGÉNIO|Cryostuffs|650|5000
OXIGÉNIO|Cryostuffs|820|2500
OXIGÉNIO|EPAL|820|2500
```

Note-se que podem existir vários lotes para o mesmo produto. Neste caso, a ordenação é, primeiro, pelo identificador do produto e, depois, pelo identificador do parceiro. Se um parceiro fornecer vários lotes do mesmo produto, apresentam-se por ordem crescente de preço e existências.

Visualizar os lotes fornecidos por um dado parceiro

É pedido o identificador do parceiro e são apresentados os lotes por ele fornecidos. A ordenação é como indicada para a listagem de todos os produtos disponíveis (lotes).

Visualizar os lotes de um dado produto

É pedido o identificador do produto e são apresentados os lotes conhecidos para esse produto. A ordenação é como indicada para a listagem de todos os produtos.

Menu de Gestão de Parceiros

Este menu permite efectuar operações sobre a base de dados de parceiros. A lista completa é a seguinte: Mostrar parceiro, Mostrar todos os parceiros, Registar parceiro, Activar/desactivar notificações de um produto, Mostrar transacções de compra com parceiro, Mostrar transacções de venda (e desagregação) com parceiro.

Todos os métodos correspondentes às mensagens de diálogo para este menu estão definidos na classe **ggc.app.partners.Message**.

Sempre que for pedido o identificador de um parceiro (**Prompt.partnerKey()**) e o parceiro não existir, é lançada a excepção **UnknownPartnerKeyException** (excepto no processo de registo). No processo de registo, caso o identificador indicado já exista, deve ser lançada a excepção **DuplicatePartnerKeyException**.

Estes comandos já estão implementados nas classes da *package* **ggc.app.partners** (disponível no CVS), respectivamente: **DoShowPartner**, **DoShowAllPartners**, **DoRegisterPartner**, **DoToggleProductNotifications**, **DoShowPartnerAcquisitions**, **DoShowPartnerSales**.

Mostrar parceiro

É pedido o identificador do parceiro e apresentada a sua informação. O formato de apresentação é o seguinte (o valor das vendas efectuadas -- não inclui desagregações -- refere-se ao momento da venda; o valor das vendas pagas refere-se ao valor realmente pago): Note-se que compras e vendas são ponto de vista do entreposto e não do parceiro. Os valores dos pontos e dos preços são apresentados como inteiros (valores arredondados ao inteiro mais próximo).

```
id|nome|endereço|estatuto|pontos|valor-compras|valor-vendas-efectuadas|valor-vendas-pagas
```

O estatuto corresponde a **NORMAL**, **SELECTION**, **ELITE**, conforme o caso.

Após esta linha, são apresentadas as notificações do parceiro (modo de entrega por omissão), pela ordem em que foram enviadas pela aplicação.

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

Após esta visualização, considera-se que o parceiro fica sem notificações registadas.

Mostrar todos os parceiros

O formato de apresentação é como para parceiros individuais (opção anterior), mas não se apresentam as notificações dos parceiros.

Registar parceiro

São pedidos o identificador do parceiro, o nome (**Prompt.partnerName()**) (cadeia de caracteres) e o endereço do parceiro (**Prompt.partnerAddress()**) (cadeia de caracteres) e regista-se o novo parceiro. Quando um parceiro é registado, aceita notificações relativas a todos os produtos.

Se já existir um parceiro com o mesmo identificador, deve ser lançada a excepção **DuplicatePartnerKeyException**, não se realizando o registo.

Activar/desactivar notificações de um produto

É pedido o identificador do produto (**Prompt.partnerKey()**) e o identificador do produto (**Prompt.productKey()**). Se as notificações relativas ao produto estavam activas para o parceiro em causa, passam a estar inactivas, e vice-versa.

Se o produto indicado não existir, é lançada a excepção **UnknownProductKeyException**.

Mostrar transacções de compra com parceiro

É pedido o identificador do parceiro e apresentadas todas as compras com ele realizadas. O formato de apresentação é como descrito abaixo (visualização de transacções).

Mostrar transacções de venda (e desagregação) com parceiro

É pedido o identificador do parceiro e apresentadas todas as vendas (e desagregações) com ele realizadas. O formato de apresentação é como descrito abaixo (visualização de transacções).

Menu de Gestão de Transacções

Este menu apresenta as operações relacionadas com transacções. A lista completa é a seguinte: Visualizar, Registar Desagregação, Registar Venda, Registar Compra, Receber Pagamento de Venda.

Todos os métodos correspondentes às mensagens de diálogo para este menu estão definidos na classe **ggc.app.transactions.Message**.

Sempre que é pedido o identificador do parceiro (**Prompt.partnerKey()**), é lançada a excepção **UnknownPartnerKeyException**, se o parceiro indicado não existir. Sempre que é pedido o identificador de produto (**Prompt.productKey()**), é lançada a excepção **UnknownProductKeyException**, se o produto indicado não existir. Sempre que é pedido o identificador da transacção (**Prompt.transactionKey()**), é lançada a excepção **UnknownTransactionKeyException**, se a transacção indicada não existir.

Estes comandos já estão implementados nas classes da *package* **ggc.app.transactions** (disponível no CVS), respectivamente: **DoShowTransaction**, **DoRegisterBreakdownTransaction**, **DoRegisterSaleTransaction**, **DoRegisterAcquisitionTransaction**, **DoReceivePayment**.

Visualizar transacção

O sistema pede o identificador da transacção a visualizar.

Nas apresentações, o campo *valor-base* é o valor da transacção sem multas/descontos.

Se a transacção for respeitante a uma venda a um parceiro, apresenta-se com o seguinte formato:

```
VENDA|id|idParceiro|idProduto|quantidade|valor-base|valor-a-pagamento|data-limite|data-pagamento
```

O campo *valor-a-pagamento* corresponde ao valor que será realmente pago (considerando possíveis multas/descontos à data da visualização). A data de pagamento (e o separador correspondente) só é apresentada se a venda tiver sido paga.

Se a transacção corresponder a uma compra a um parceiro, apresenta-se com o seguinte formato:

```
COMPRA|id|idParceiro|idProduto|quantidade|valor-pago|data-pagamento
```

Se a transacção for uma desagregação, então deve ser apresentada, primeiro a "venda" do produto derivado, seguida da "compra" dos componentes obtidos na desagregação. O formato para N componentes é o seguinte:

```
DESAGREGAÇÃO|id|idPa|idPr|quantidade|vbase|vpag|data|idC1:q1:v1#...#idCN:qN:vN
```

O valor base de uma desagregação corresponde à diferença entre a compra e a venda. O valor pago corresponde ao valor realmente pago pelo parceito (é um valor não negativo, podendo ser 0 caso o valor base seja negativo. Nesta linha, **idPa** é o identificador do parceiro; **idPr** é o identificador do produto; **vbase** é o valor real da transacção (diferencial); **vpag** é o valor a pagar (não negativo); **data** é a data da transacção; **idCx**, **qx**, **vx**, são (respectivamente) o identificador, a quantidade e o valor do componente **x** (não confundir com valor unitário). Cada componente é separado do seguinte por **#**.

Registar Desagregação (pelo entreposto a pedido de um parceiro)

Para registar uma desagregação, é pedido o identificador do parceiro que pede a operação e o identificador do produto a desagregar e a respectiva quantidade (**Prompt.amount()**). Se a quantidade for superior às existências actuais, deve ser lançada a excepção **UnavailableProductException** (não se realiza a desagregação). Se o produto a desagregar não for derivado, a acção não tem efeito.

A actualização dos produtos do entreposto, bem como o respectivo valor, tem lugar logo após o registo da desagregação, ou seja, considera-se que a operação é instantânea.

Esta transacção regista o produto de origem e os lotes dos resultantes (associados ao parceiro que pede a operação), bem como os respectivos valores.

Registar Venda (do entreposto a um parceiro)

Para registar uma venda, é pedido o identificador do parceiro, a data limite para o pagamento (**Prompt.paymentDeadline()**), o identificador do produto a vender e a respectiva quantidade (**Prompt.amount()**). Se a quantidade for superior às existências actuais, deve ser lançada a excepção **UnavailableProductException** (não se realiza a venda).

A determinação das existências começa com o preço mais baixo e vai prosseguindo por todos os parceiros que tenham o produto disponível. O preço a pagar é determinado pelo preço das várias fracções.

Se a venda corresponder a um produto derivado e não existir produto suficiente nos lotes disponíveis, o produto pode ser fabricado a partir de componentes de outros lotes. As existências dos componentes têm de ser suficientes para satisfazer a totalidade das necessidades de produto derivado a fabricar. O preço é calculado como descrito acima.

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

Registar Compra (do entreposto a um parceiro)

É pedido o identificador do parceiro a quem se realiza a compra, o identificador do produto que se está a comprar, o preço do produto (**Prompt.price()**) e a respectiva quantidade (**Prompt.amount()**). O produto comprado mantém informação sobre o processo de aquisição: em particular, se vários parceiros fornecerem o mesmo produto, podem fazê-lo com preços diferentes. O entreposto poderá, então, vender um dado produto com diferentes preços.

Se o identificador do produto for desconhecido, então é perguntado ao parceiro se quer introduzir uma receita para um produto derivado (**Prompt.addRecipe()**). Se a resposta for negativa, trata-se de um produto simples. Caso contrário, é pedido o número de componentes da receita (**Prompt.numberOfComponents()**), o valor do agravamento (**Prompt.alpha()**) e, em ciclo, os identificadores (**Prompt.productkey()**) e quantidades (**Prompt.amount()**) dos vários componentes.

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

Receber Pagamento de Venda (do entreposto a um parceiro)

Apenas é permitido o pagamento de vendas a parceiros. Tentativas de pagamento de transacções que não correspondam a vendas não produzem nenhum resultado.

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

Menu de Consultas

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

Sempre que é pedido o identificador do parceiro (**Prompt.partnerKey()**), é lançada a excepção **UnknownPartnerKeyException**, se o parceiro indicado não existir. Sempre que é pedido o identificador de produto (**Prompt.productKey()**), 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 da *package* **ggc.app.lookups** (disponível no CVS), respectivamente: **DoLookupProductBatchesUnderGivenPrice**, **DoLookupPaymentsByPartner**.

Mostrar lotes de produtos com preço abaixo de limite

Pede-se o valor limite pretendido (**Prompt.priceLimit()**) e apresentam-se todos os lotes cujo preço é inferior ao preço indicado. É apresentado um lote por linha.

Mostrar transacções pagas por parceiro

Pede-se o identificador do parceiro e apresentam-se as transacções do parceiro que já estão pagas (uma transacção por linha).

Leitura de Dados a Partir de Ficheiros Textuais

Além das opções de manipulação de ficheiros descritas no menu principal, é possível iniciar a aplicação com um ficheiro de texto especificado pela propriedade Java **import**.

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 (sugere-se a utilização do método **String.split** para o processamento preliminar destas linhas). Não existem entradas mal-formadas.

Cada linha tem uma descrição distinta, mas que segue os seguintes formatos.

```
PARTNER|id|nome|endereço
```

Os lotes têm as seguintes definições (respectivamente, para produtos simples e derivados). Produtos derivados (é apresentado o produto e a lista de componentes e respectivas quantidades da sua receita):

```
BATCH_S|idProduto|idParceiro|preço|stock-actual
BATCH_M|idProduto|idParceiro|preço|stock-actual|agravamento|componente-1:quantidade-1#...#componente-n:quantidade-n
```

As definições de parceiros precedem sempre as dos lotes.

Um exemplo de conteúdo do ficheiro inicial é como se segue:

Exemplo de ficheiro de entrada textual **[Expand]**

A codificação dos ficheiros a ler é garantidamente UTF-8.

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

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 é necessária a definição apropriada da variável **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 (**ggc.app.App.main**). As propriedades são tratadas automaticamente pelo código de apoio.

```
java -Dimport=test.import -Din=test.in -Dout=test.outhyp ggc.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 verificando alguns aspectos da sua funcionalidade.

Notas de Implementação

Tal como indicado acima, algumas classes fornecidas como material de apoio, são de uso obrigatório e não podem ser alteradas. Outras dessas classes são de uso obrigatório e têm de ser alteradas.

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).

Categories: **Ensino** **PO** **Projecto de PO**