# Programação com Objectos/Projecto de Programação com Objectos/Enunciado do Projecto de 2022-2023

*From Wiki**3*

< Programação com Objectos | Projecto de Programação com Objectos

**AVISOS - Avaliação em Época Normal** **[Collapse]**

Esclarecimento de dúvidas:

- Consultar sempre o corpo docente atempadamente: presencialmente ou através do endereço oficial da disciplina [1].
- 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 em Projecto de Programação com Objectos):

- 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 desenvolvimento.

Processo de avaliação (ver informação completa em Avaliação do Projecto):

- Datas: **2022/10/04 12:00** (inicial); **2022/10/21 12:00** (intercalar); **2022/11/04 12:00** (final); **2022/11/04 (early bird) 2021/11/07 (normal)** (teste prático).
- **Todas as entregas são cruciais para o bom desenvolvimento do projecto, sendo obrigatórias: a não realização de uma entrega implica a exclusão da avaliação do projecto e, por consequência, da avaliação da disciplina.**
- Verificar atempadamente (até 48 horas antes do final de cada prazo) os requisitos exigidos pelo processo de avaliação, incluindo a capacidade de acesso ao repositório CVS.
- **Apenas se consideram para avaliação os projectos existentes no repositório CVS oficial.**
- Trabalhos não presentes no repositório no final do prazo têm classificação 0 (zero) (não são aceites outras formas de entrega). Não são admitidas justificações para atrasos em sincronizações do repositório. A indisponibilidade temporária do repositório ou de outros materiais, desde que inferior a 24 horas, não justifica atrasos na submissão de um trabalho.
- A avaliação do projecto pressupõe o compromisso de honra de que o trabalho correspondente 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.**

**Material de Uso Obrigatório** **[Collapse]**

As bibliotecas **po-uilib** e o conteúdo inicial do CVS são de **uso obrigatório**:

- po-uilib (classes de base) po-uilib-202209081626.tar.bz2 (não pode ser alterada) - javadoc
- prr-core (classes do "core") (via CVS) (deve ser completada -- os nomes das classes fornecidas não podem ser alterados)
- prr-app (classes de interacção) (via CVS) (deve ser completada -- os nomes das classes fornecidas não podem ser alterados)

A máquina virtual, fornecida para desenvolvimento do projecto, já contém todo o material de apoio.

**Uso Obrigatório: Repositório CVS**

**Apenas se consideram para avaliação os projectos existentes no repositório CVS oficial**.
Trabalhos não presentes no repositório no final do prazo têm classificação 0 (zero) (não são aceites outras formas de entrega). Não são admitidas justificações para atrasos em sincronizações do repositório. A indisponibilidade temporária do repositório, desde que inferior a 24 horas, não justifica atrasos na submissão de um trabalho.

Contents

**ÉPOCA NORMAL**

O objectivo do projecto e desenvolver uma aplicação de gestão de uma rede de terminais de comunicação, denominada por **prr**. Genericamente, o programa permite o registo, gestão e consulta de clientes, terminais e comunicações.

# Clientes, terminais, comunicações, planos tarifários

Os clientes, terminais e comunicações possuem chaves únicas, cadeias de caracteres para os clientes e para os terminais e inteiros para as comunicações.

A noção de saldo é definida como a diferença entre os valores dos pagamentos efectuados e das dívidas por pagar. O saldo pode ser calculado globalmente (considerando todos os clientes), por cliente (considerando todos os terminais do cliente) ou por terminal (considerando todas as comunicações do terminal, pagas e por pagar).

## Propriedades e funcionalidade dos clientes

Cada cliente, para além da chave única, tem ainda o nome (cadeia de caracteres) e o número de identificação fiscal (inteiro). A cada cliente podem estar associados vários terminais.

O cliente mantém informação sobre os pagamentos efectuados (sobre comunicações passadas) e valores em dívida (comunicações cujo valor ainda não foi pago).

Existem três tipos de clientes: **Normal** (situação inicial, após o registo -- no entanto, ver a situação da leitura de dados textuais), **Gold** e **Platinum**. O tipo de cliente influencia o custo das comunicações que efectua (ver planos tarifários). O tipo do cliente evolui nas seguintes condições:

| Antes | Depois | Condição |
|---|---|---|
| Normal | Gold | O saldo do cliente (após realizar um pagamento) é superior a 500 créditos. |
| Normal | Platinum | (não é possível) |
| Gold | Normal | O saldo do cliente (após realizar uma comunicação) é negativo. |
| Gold | Platinum | O cliente realizou 5 comunicações de vídeo consecutivas e não tem saldo negativo. A contabilização da 5ª comunicação ainda considera que o cliente é do tipo Gold. |
| Platinum | Gold | O cliente realizou 2 comunicações de texto consecutivas e não tem saldo negativo. A contabilização da 2ª comunicação ainda considera que o cliente é do tipo Platinum. |
| Platinum | Normal | O saldo do cliente (após realizar uma comunicação) é negativo. |

## Propriedades e funcionalidade dos terminais

Cada terminal é identificado por uma cadeia de caracteres numérica (exactamente 6 dígitos) e está associado a um único cliente.

Os terminais podem realizar três tipos de comunicação: texto, voz e vídeo. As comunicações realizadas pelo terminal são contabilizadas de acordo com o tarifário associado ao cliente. O terminal tem contabilidade própria, sendo sempre possível saber os valores dos pagamentos efectuados e dos valores devidos.

Existem, pelo menos, dois tipos de terminal: básicos e sofisticados. Os terminais básicos só conseguem realizar comunicações de texto e de voz, não podendo nem iniciar nem receber comunicações de vídeo. Os terminais sofisticados podem realizar todos os tipos de comunicação.

Cada terminal tem uma lista de amigos (inicialmente vazia). Um terminal não pode ser amigo de si próprio.

Um terminal recém-criado fica no estado de espera (*idle*); tem os valores de pagamentos e dívidas ambos a zero. Existem outros estados, definidos a seguir.

### Estados dos terminais

Cada terminal pode estar em espera, em silêncio, ocupado ou desligado.

- Espera -- situação normal sem actividade (*idle*);

- Silêncio -- tal como em Espera, pode iniciar-se qualquer tipo de comunicação suportada pelo terminal, mas só podem ser recebidas comunicações de texto;
- Ocupado -- não podendo iniciar-se comunicações, mas podem ser recebidas mensagens de texto;
- Desligado -- não podem ser iniciadas ou recebidas comunicações.

### Transições entre estados de terminais

Um terminal pode chegar aos vários estados nas seguintes condições (outras transições não são possíveis):

- Espera -- de desligado (ao ligar em espera); de silêncio (ir para espera); de ocupado (final de comunicação);
- Silêncio -- de desligado (ao ligar em silêncio); de espera (colocar em silêncio); de ocupado (final de comunicação);
- Ocupado -- de espera ou de silêncio (início de comunicação);
- Desligado -- de espera ou de silêncio (ao desligar).

### Notificações

Apenas são passíveis de notificação os clientes que tentaram comunicação com um terminal e a comunicação não foi possível nessa altura. Quando uma comunicação não se efectua, regista-se a tentativa de contacto, para que, assim que seja possível a realização do contacto pretendido, se enviarem notificações aos terminais de origem. O registo da tentativa de contactos só tem lugar quando o cliente do terminal de origem tem activa a recepção de contactos falhados no instante em que se tentou efectuar a comunicação.

São geradas notificações e é possível avisar um cliente nas seguintes circunstâncias:

1. Um terminal desligado é colocado em silêncio (*off-to-silent*): notifica-se disponibilidade para receber comunicações de texto.
2. Um terminal desligado ou em silêncio é colocado em espera (*off-to-idle* ou *silent-to-idle*): notifica-se disponibilidade para receber comunicações (qualquer suportada).
3. Um terminal deixa de estar ocupado (*busy-to-idle*): notifica-se disponibilidade para receber comunicações (qualquer suportada).

## Propriedades e funcionalidade das comunicações

Cada comunicação tem um identificador único (número inteiro, no contexto de todos os clientes). A primeira tem como identificador “1”, sendo os identificadores subsequentes obtidos por incremento unitário do mais recente identificador utilizado. A comunicação contém ainda informação sobre os terminais de origem e de destino e o estado da comunicação: em curso ou terminada.

As comunicações de texto têm ainda a mensagem enviada. As comunicações interactivas (vídeo e voz) possuem informação sobre a duração da comunicação. O custo de uma comunicação depende do comprimento da mensagem de texto ou da duração das comunicações interactivas. O custo depende ainda do plano tarifário associado a cada cliente (calculado no final da comunicação). Todos os cálculos envolvendo os custos das comunicações devem ser realizados sem arredondamentos.

Um terminal não pode estabelecer uma comunicação interactiva consigo próprio.

## Planos tarifários

Cada plano tarifário define os custos para cada tipo de comunicação, baseado no nível do cliente, no tipo de comunicação, entre outras características. Os planos tarifários têm um nome único no contexto da rede de terminais a que estão associados. A rede de terminais pode oferecer vários planos tarifários mas em cada momento um cliente apenas tem um plano tarifário. A rede de terminais oferece pelo menos o plano tarifário designado como **base**. Este plano tarifário é o plano atribuído inicialmente a todos os clientes. O custos das comunicações é medido em créditos.

O custo (medido em créditos) de uma comunicação de texto com **N** caracteres no plano tarifário base está representado na tabela seguinte:

| | **Normal** | **Gold** | **Platinum** |
|---|---|---|---|
| **N** < 50 caracteres | 10 | 10 | 0 |
| 50 caracteres <= **N** < 100 caracteres | 16 | 10 | 4 |

| | | | |
|---|---|---|---|
| **N** >= 100 caracteres | 2 x **N** | 2 x **N** | 4 |

Quando é efectuada uma comunicação de voz ou de vídeo, o custo no plano tarifário base é proporcional ao tempo de conversação e, quando se comunica com um terminal amigo, é aplicado um desconto de 50%. O custo, em créditos por minuto, é o seguinte para terminais não amigos:

| | **Normal** | **Gold** | **Platinum** |
|---|---|---|---|
| Comunicação de voz | 20 | 10 | 10 |
| Comunicação de vídeo | 30 | 20 | 10 |

O custo de uma comunicação deve ser calculado quando a comunicação termina e guardado, por forma a garantir que o custo não é afectado por mudanças futuras dos planos tarifários.

## Entrega de notificações

Os clientes podem activar a recepção de notificações sobre eventos associados a terminais em algumas circunstâncias. Em qualquer momento, um cliente pode activar ou desactivar essas notificações. A entrega de notificações deve ser flexível e deve 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.

As notificações contêm informação acerca da sua natureza e do terminal a que dizem respeito. Um dado evento apenas produz uma notificação por cliente (o conjunto de clientes a notificar é limpo após o envio da notificação).

# 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).

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

Uma base de dados textual com conceitos pré-definidos pode ser carregada no início da aplicação.

Note-se que não é necessário nem 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 **prr.Manager** 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.

# 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 novos tipos de clientes; definir novos tipos de comunicação; definir novos planos tarifários; definir novas formas de pesquisa; permitir a gestão de várias redes de terminais.

Embora na especificação actual não seja possível remover algumas 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 **prr-app**). As mensagens não podem ser usadas no núcleo da aplicação (**prr-core**). Além disso, não podem ser definidas novas. Potenciais omissões devem ser esclarecidas antes de qualquer implementação.

A apresentação de valores monetários é sempre feita com arredondamento ao inteiro mais próximo, mas a representação interna não deve ser arredondada.

A apresentação de listas de entidades do domínio (clientes, 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.

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.

Note-se que o programa principal e os comandos e menus, a seguir descritos, já estão parcialmente implementados nas *packages* **prr.app**, **prr.app.clients**, **prr.app.lookups**, **prr.app.main**, **prr.app.terminal**, **prr.app.terminals**. Estas classes são de uso obrigatório e estão disponíveis no CVS (módulo **prr-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, Gestão de clientes, Gestão de terminais, Consultas, Mostrar informação global sobre pagamentos e dívidas.

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

Estes comandos já estão implementados nas classes da *package* **prr.app.main** (disponível no CVS), respectivamente: **DoOpenFile**, **DoSaveFile**, **DoShowGlobalPaymentsAndDebts**.

## Salvaguarda do estado actual da aplicação

Inicialmente, a aplicação está vazia ou tem apenas informação sobre as entidades que foram carregadas 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 (para operações de salvaguarda subsequemtes). 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 podem ser utilizados no início da aplicação).

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

## Gestão e consulta de dados da aplicação

- **Menu de Gestão de Clientes** -- Abre o menu de gestão de clientes.
- **Menu de Gestão de Terminais** -- Abre o menu de gestão de terminais.

- **Menu de Consultas** -- Abre o menu de consultas (pesquisas).

### Mostrar informação global sobre pagamentos e dívidas

Esta opção apresenta os valores globais correspondentes a pagamentos e dívidas (soma dos valores parciais para todos os clientes registados), através da mensagem **Message.globalPaymentsAndDebts()**.

## Menu de gestão de clientes

Este menu permite efectuar operações sobre a base de dados de clientes. A lista completa é a seguinte: Visualizar cliente, Visualizar todos os clientes, Registar cliente, Activar recepção de contactos falhados, Desactivar recepção de contactos falhados, Mostrar informação sobre pagamentos e dívidas de cliente.

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

Sempre que é pedido o identificador de um cliente (**Prompt.key()**) e o identificador não existir (excepto no processo de registo), é lançada a excepção **UnknownClientKeyException**. Na ocorrência de excepções, as operações não têm efeito.

Estes comandos já estão implementados nas classes da *package* **prr.app.clients** (disponível no CVS), respectivamente: **DoShowClient**, **DoShowAllClients**, **DoRegisterClient**, **DoEnableClientNotifications**, **DoDisableClientNotifications**, **DoShowClientPaymentsAndDebts**.

### Visualizar cliente

É pedido o identificador do cliente e apresentada a sua informação. O formato de apresentação de cada cliente é como se indica de seguida. Os valores para *type* são **NORMAL**, **GOLD**, ou **PLATINUM**. Os valores para *notifications* são **YES** ou **NO** (notificações activas/inactivas). O campo *terminals* representa o número de terminais associados ao cliente, *payments* é o valor dos pagamentos do cliente e *debts* é o valor das dívidas do cliente. Se o cliente não tiver terminais, os valores de *terminals*, *payments* e *debts* são 0 (zero).

```
CLIENT|key|name|taxId|type|notifications|terminals|payments|debts
```

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

```
tipo-de-notificação|idTerminal
```

O tipo de notificação é um de **O2S** (off-to-silent), **O2I** (off-to-idle), **B2I** (busy-to-idle) ou **S2I** (silent-to-idle), tal como descritos acima. Após esta visualização, considera-se que o cliente fica sem notificações registadas.

### Visualizar todos os clientes

O formato de apresentação é como para clientes individuais (opção anterior), mas não se apresentam as notificações dos clientes (nem se limpam as listas correspondentes).

### Registar cliente

O sistema pede o identificador que ficará associado ao cliente (identificador único}. De seguida, pede o nome do cliente (**Prompt.name()**) e o número de identificação fiscal (**Prompt.taxId()**). Após o registo, o cliente fica no estado **Normal** e o registo de contactos falhados fica activo.

Caso o identificador indicado já exista, deve ser lançada a excepção **DuplicateClientKeyException**, não se realizando o registo.

### Activar recepção de contactos falhados

É pedido o identificador do cliente. Se o registo de contactos falhados já estava activo, o cliente não é alterado e é apresentada a mensagem **clientNotificationsAlreadyEnabled()**.

### Desactivar recepção de contactos falhados

É pedido o identificador do cliente. Se o registo de contactos falhados já estava inactivo, o cliente não é alterado e é apresentada a mensagem **clientNotificationsAlreadyDisabled()**.

### Mostrar informação sobre pagamentos e dívidas de cliente

O sistema pede o identificador do cliente, apresentando os valores dos seus pagamentos e dívidas (**Message.clientPaymentsAndDebts()**).

## Menu de Gestão de Terminais

Este menu permite efectuar operações sobre a base de dados de terminais. A lista completa é a seguinte: Mostrar todos os terminais, Registar terminal, Menu da consola de um terminal.

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

Sempre que for pedido o identificador de um terminal (**Prompt.terminalKey()**) e o terminal não existir, é lançada a excepção **UnknownTerminalKeyException** (excepto no processo de registo). Sempre que é pedido o identificador de um cliente (**Prompt.clientKey()**) e o identificador não existir, é lançada a excepção **UnknownClientKeyException**. Na ocorrência de excepções (estas ou outras), as operações não têm efeito.

Estes comandos já estão implementados nas classes da *package* **prr.app.terminals** (disponível no CVS), respectivamente: **DoShowAllTerminals**, **DoRegisterTerminal**, **DoOpenMenuTerminalConsole**.

### Mostrar todos os terminais

O formato de apresentação de cada terminal é um dos seguintes:

```
terminalType|terminalId|clientId|terminalStatus|balance-paid|balance-debts|friend1,...,friend
terminalType|terminalId|clientId|terminalStatus|balance-paid|balance-debts
```

Os valores para o campo *terminalType* são **BASIC** (terminais básicos) ou **FANCY** (terminais sofisticados). Os valores para o campo *terminalStatus* são **IDLE**, **OFF**, **SILENCE**, **BUSY**. Os valores *friend1*, ..., *friendN* são os identificadores dos terminais amigos e são apresentados por ordem crescente desses identificadores. O segundo formato é para o caso de um terminal que não tem amigos. Note-se que os valores dos pagamentos efectuados e os dos valores em dívida são apresentados separadamente.

### Registar terminal

É pedido o número identificador do terminal. De seguida, o sistema pede o tipo de terminal (**Prompt.terminalType()**). A resposta deve ser **BASIC** (terminal básico) ou **FANCY** (terminal sofisticado). Se a resposta não corresponder a nenhum dos dois valores, a pergunta é repetida até se obter uma resposta válida. Finalmente, é pedido o identificador do cliente que ficará associado ao terminal.

Caso o identificador do terminal seja inválido, lança-se a excepção **InvalidTerminalKeyException**, não se realizando o registo. Caso o identificador indicado já exista, deve ser lançada a excepção **DuplicateTerminalKeyException**, não se realizando o registo.

### Menu da consola de um terminal

É pedido o número identificador de um terminal, sendo aberto o correspondente menu da sua consola.

## Menu de Consultas

Este menu apresenta as operações relacionadas com consultas. A lista completa é a seguinte: Mostrar todas as comunicações, Mostrar comunicações feitas por um cliente, Mostrar comunicações recebidas por um cliente, Mostrar clientes sem dívidas, Mostrar clientes com dívidas, Mostrar terminais sem actividade, Mostrar terminais com saldo positivo.

Sempre que é pedido o identificador do cliente (**Prompt.clientKey()**), é lançada a excepção **UnknownClientKeyException** se o cliente indicado não existir. Sempre que é pedido o identificador de terminal (**Prompt.terminalKey()**), é lançada a excepção **UnknownTerminalKeyException** se o terminal indicado não existir.

A apresentação de resultados é como se indica nos casos já descritos de apresentação das várias entidades. Sempre que for feita uma consulta e nenhuma entidade satisfizer as condições associadas ao pedido, nada deve ser apresentado.

Estes comandos já estão parcialmente implementados nas classes da *package* **prr.app.lookups** (disponível no CVS), respectivamente: **DoShowAllCommunications**, **DoShowCommunicationsFromClient**, **DoShowCommunicationsToClient**, **DoShowClientsWithoutDebts**, **DoShowClientsWithDebts**, **DoShowUnusedTerminals**, **DoShowTerminalsWithPositiveBalance**.

### Mostrar todas as comunicações

O formato de apresentação é o seguinte:

```
type|idCommunication|idSender|idReceiver|units|price|status
```

Os possíveis valores para o campo *type* são **VOICE**, **TEXT** ou **VIDEO**. Os possíveis valores para o campo *status* são **ONGOING** (comunicação em curso) ou **FINISHED** (comunicação terminada). O valor de *units* corresponde às unidades de contabilização (caracteres ou minutos). Caso a comunicação esteja em curso, os valores de **units** e de **price** são ambos zero.

### Mostrar comunicações feitas por um cliente

É pedido o identificador do cliente, sendo apresentadas as comunicações iniciadas pelos seus terminais. O formato de apresentação é o descrito acima.

### Mostrar comunicações recebidas por um cliente

É pedido o identificador do cliente, sendo apresentadas as comunicações recebidas pelos seus terminais. O formato de apresentação é o descrito acima.

### Mostrar clientes sem dívidas

São apresentados os clientes sem dívidas. Utiliza-se o formato de apresentação de um cliente descrito anteriormente.

### Mostrar clientes com dívidas

São apresentados os clientes por ordem decrescente do valor das respectivas dívidas (valores superiores a zero). Se as dívidas tiverem o mesmo valor, apresentam-se os clientes por ordem crescente do seu identificador. Utiliza-se o formato de apresentação de um cliente descrito anteriormente.

### Mostrar terminais sem actividade

São apresentados os terminais que ainda não efectuaram nem receberam qualquer comunicação. O formato de apresentação é o descrito acima.

### Mostrar terminais com saldo positivo

São apresentados os terminais que que têm um valor de pagamentos estritamente superior ao valor das dívidas. O formato de apresentação é o descrito acima.

## Menu da consola de um terminal

Este menu apresenta as operações relacionadas com a consola de um terminal, i.e., o seu uso e administração. A lista completa é a seguinte: Ligar terminal, Desligar terminal, Silenciar terminal, Adicionar amigo, Retirar amigo, Efectuar pagamento, Mostrar informação sobre pagamentos e dívidas, Enviar comunicação de texto, Iniciar comunicação interactiva, Terminar comunicação interactiva, Mostrar comunicações em curso.

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

Sempre que é pedido o identificador do cliente (**Prompt.clientKey()**), é lançada a excepção **UnknownClientKeyException**, se o cliente indicado não existir. Sempre que é pedido o identificador do terminal (**Prompt.terminalKey()**), é lançada a excepção **UnknownTerminalKeyException**, se o terminal seleccionado não existir.

Estes comandos já estão implementados nas classes da *package* **prr.app.terminal** (disponível no CVS), respectivamente: **DoTurnOnTerminal**, **DoOffTurnTerminal**, **DoSilenceTerminal**, **DoAddFriend**, **DoRemoveFriend**, **DoPerformPayment**,

**DoShowTerminalPaymentsAndDebts**, **DoSendTextCommunication**, **DoStartInteractiveCommunication**, **DoEndInteractiveCommunication**, **DoShowOngoingCommunications**.

## Colocar terminal em espera

Coloca o terminal em espera. Se o terminal já estiver em espera, é apresentada a mensagem **alreadyIdle()**. Se não for possível colocar o terminal em espera, o comando não tem efeito.

## Desligar terminal

Desliga o terminal. Se o terminal já estiver desligado, é apresentada a mensagem **alreadyOff()** e o comando não realiza outras acções. Se não for possível desligar o terminal, o comando não tem efeito.

## Silenciar terminal

Coloca o terminal em silêncio. Se o terminal já estiver em silêncio, é apresentada a mensagem **alreadySilent()**. Se não for possível colocar o terminal em silêncio, o comando não tem efeito.

## Adicionar amigo

É pedido o identificador do terminal a adicionar à lista de amigos. Se o terminal indicado já fizer parte da lista de amigos, a operação termina sem alterações.

## Retirar amigo

É pedido o identificador do terminal a retirar da lista de amigos. Se o terminal indicado não fizer parte da lista de amigos, a operação termina sem alterações.

## Efectuar pagamento

É pedido o identificador da comunicação a pagar (**Prompt.commKey()**). A comunicação tem de pertencer ao terminal actual. Caso contrário, é apresentada a mensagem **foreignCommunication()**.

## Mostrar informação sobre pagamentos e dívidas

São apresentados os valores dos pagamentos e das dívidas do terminal (**terminalPaymentsAndDebts()**).

## Enviar comunicação de texto

Esta operação está disponível para um terminal que não esteja desligado ou em comunicação. Permite enviar uma comunicação de texto para outro terminal.

É pedido o número do terminal de destino e o corpo da mensagem (**Prompt.textMessage()**).

## Iniciar comunicação interactiva

Esta operação está disponível caso o terminal selecionado não esteja desligado ou em comunicação. Permite estabelecer uma comunicação interactiva entre o terminal selecionado com outro terminal.

É pedido o número do terminal de destino e o tipo de comunicação (**Prompt.commType()**). A resposta deve ser uma das seguintes opções (cadeia de caracteres): **VIDEO**, **VOICE**. Se a resposta não corresponder a nenhum destes valores, a pergunta é repetida até se obter uma resposta válida.

Quando o terminal de destino está desligado, é apresentada a mensagem **destinationIsOff()**. Quando o terminal de destino está ocupado, é apresentada a mensagem **destinationIsBusy()**. Quando o terminal de destino está em silêncio, é apresentada a mensagem **destinationIsSilent()**.

Se se tenta iniciar uma comunicação não suportada pelo terminal de origem ou pelo terminal de destino, devem ser apresentadas as **unsupportedAtOrigin()** ou **unsupportedAtDestination()**, respectivamente.

## Terminar comunicação interactiva

Esta operação está disponível para um terminal que iniciou uma comunicação interactiva enquanto a comunicação está em curso. O terminal de destino não pode interromper a comunicação, pelo que o comando não está disponível. O comando permite terminar a comunicação e registar a sua duração.

É pedida a duração da comunicação (**Prompt.duration()**, em segundos). Após o término da comunicação, é apresentado o seu custo, através da mensagem **communicationCost()**.

### Mostrar comunicação em curso

É apresentada a comunicação em curso (de acordo com o formato indicado acima).

Se não houver nenhuma comunicação em curso, é apresentada a mensagem **noOngoingCommunication()**.

## 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:

```
CLIENT|id|nome|taxId
terminal-type|idTerminal|idClient|state
FRIENDS|idTerminal|idTerminal1,...,idTerminalN
```

As definições de clientes precedem sempre as dos terminais. As ligações entre amigos estão sempre após a definição das restantes entidades. Para os terminais, *terminal-type* é **BASIC** ou **FANCY**.

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

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

```
CLIENT|cli001|Manuel Pinheiro|103443
CLIENT|cli002|Pedro Pinheiro|103447
CLIENT|Cli201|Ludgero Oliveira|103440
CLIENT|cli Es|Maria Eucalipto|103441
CLIENT|01|Oliveira Preto|103547
CLIENT|cli003|Pedro Oliveira|103449
BASIC|969001|cli001|ON
BASIC|969003|cli002|ON
FANCY|969002|cli002|SILENCE
FANCY|969007|cli Es|ON
BASIC|969008|cli003|OFF
BASIC|969009|cli003|OFF
BASIC|969010|cli003|ON
FANCY|969006|cli003|ON
FANCY|969005|cli003|ON
BASIC|969004|cli003|ON
FRIENDS|969001|969008,969009,969004
FRIENDS|969004|969001
FRIENDS|969003|969008
```

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 (**prr.app.App.main**). As propriedades são tratadas automaticamente pelo código de apoio.

```
	java -Dimport=test.import -Din=test.in -Dout=test.outhyp prr.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**