

Programação com Objectos/Projecto de Programação com Objectos/Enunciado do Projecto de 2018-2019

< *Programação com Objectos* | *Projecto de Programação com Objectos*

AVISOS – Avaliação em Época Normal	[Expandir]
Material de Uso Obrigatório	[Expandir]
Contents [hide]	
<ul style="list-style-type: none">1 Conceitos e Relações do Modelo <ul style="list-style-type: none">1.1 Universidade, Cursos, Disciplinas 1.2 Pessoas 1.3 Projectos 1.4 Inquéritos 1.5 Notificações 2 Requisitos de Desenho <ul style="list-style-type: none">2.1 Salvaguarda do estado actual da aplicação 2.2 Gestão e consulta de dados da aplicação 3.2 Gestão de actividades dos alunos 3.3 Gestão de actividades dos delegados 3.4 Gestão de actividades dos professores 3.5 Serialização 4 Interação com o utilizador <ul style="list-style-type: none">4.1 Início da aplicação: estabelecimento da identidade do utilizador 4.2 Menu Principal <ul style="list-style-type: none">4.2.1 Salvaguarda do estado actual da aplicação 4.2.2 Gestão e consulta de dados da aplicação 4.3 Portal Pessoal <ul style="list-style-type: none">4.3.1 Mostrar pessoa 4.3.2 Alterar número de telefone 4.3.3 Mostrar pessoas 4.3.4 Procurar pessoa 4.4 Portal do Docente <ul style="list-style-type: none">4.4.1 Criar projecto 4.4.2 Fechar projecto 4.4.3 Ver submissões de um projecto 4.4.4 Ver alunos de disciplina leccionada 4.4.5 Ver resultados de um inquérito 4.5 Portal do Estudante <ul style="list-style-type: none">4.5.1 Entregar projecto 4.5.2 Preencher inquérito 4.5.3 Ver resultados de inquérito 4.6 Portal do Delegado <ul style="list-style-type: none">4.6.1 Criar inquérito 4.6.2 Cancelar inquérito 4.6.3 Abrir inquérito 4.6.4 Fechar inquérito 4.6.5 Finalizar inquérito 4.6.6 Mostrar inquéritos de uma disciplina 5 Leitura de Dados a Partir de Ficheiros Textuais <ul style="list-style-type: none">5.1 Exemplo de ficheiro a importar 6 Execução dos Programas e Testes Automáticos 7 Notas de Implementação	

O objectivo do projecto é criar uma aplicação que gere parte da actividade dos cursos de uma universidade. Em particular, a aplicação faz a gestão de inquéritos aos projectos realizados pelos alunos nas várias disciplinas do curso.

Neste texto, o tipo **negrito** indica um literal (i.e., é exactamente como apresentado); o símbolo indica um espaço; e o tipo *ítilco* indica uma parte variável (i.e., uma descrição).

Conceitos e Relações do Modelo

Existem vários conceitos importantes neste contexto: universidade, curso, aluno, professor, funcionário, projecto, inquérito.

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

Universidade, Cursos, Disciplinas

Uma universidade tem vários cursos, tendo ainda funcionários, docentes e alunos. Os alunos estão necessária e exclusivamente associados a um curso. Os docentes podem estar associados a vários cursos. Os funcionários não têm nenhuma relação especial com os cursos.

Cada curso tem um nome (único no contexto da universidade) e é constituído por várias disciplinas. Cada curso tem ainda um conjunto de alunos que podem inscreverem-se nas várias disciplinas do curso. Cada aluno pode inscrever-se, no máximo, em 6 disciplinas.

Cada disciplina tem um nome e apenas pertence a um curso. O nome da disciplina é o identificador único dentro do curso respectivo. Cada disciplina é leccionada por um ou mais docentes (embora possa não ter nenhum associado em algum momento) e tem o ou mais alunos inscritos. Os alunos inscritos numa disciplina têm que pertencer ao curso da disciplina. Existe uma capacidade máxima relativamente aos alunos que se podem inscrever numa disciplina. Um docente pode estar associado a várias disciplinas.

Pessoas

Os utilizadores da aplicação correspondem aos vários tipos de pessoa que existem na universidade: alunos, docentes e funcionários. Alunos, docentes e funcionários têm nome (cadeia de caracteres), número de telefone (cadeia de caracteres) e um identificador único (íntero com 6 dígitos). Este identificador é atribuído automaticamente (de forma incremental) e começa em 100000. Um aluno pode ainda ser delegado do curso em que o aluno está inscrito. O número máximo de delegados de um curso é 7. Em qualquer instante, um aluno pode tornar-se delegado ou deixar de ser delegado.

Projectos

Os docentes podem criar projectos associados às disciplinas que leccionam. Cada projecto tem um nome e descrição e pode estar aberto (aceita entregas) ou fechado (não aceita entregas). O nome é único no contexto da disciplina. Quando o projecto é criado fica automaticamente aberto.

Um aluno pode submeter a sua versão do projecto desde que o projecto esteja aberto e o aluno esteja inscrito na disciplina do projecto. Os alunos podem realizar várias submissões no mesmo projecto, sendo preservada apenas a última. Por razões de simplificação, a submissão é representada por uma cadeia de caracteres introduzida pelo aluno.

Inquéritos

A cada projecto não fechado pode ser associado, no máximo, um inquérito.

O inquérito recolhe informação sobre o número de horas gastas na execução do projecto (número inteiro) e um comentário livre por parte do aluno que responde (cadeia de caracteres).

Um inquérito pode estar em três vários estados: criado, aberto, fechado e finalizado. O comportamento do inquérito depende desta situação. A mudança de situação é explícita, excepto quando o projecto associado é fechado; neste caso, o inquérito passa automaticamente de criado para aberto. Esta é a única mudança implícita; as outras mudanças são explícitas e da exclusiva responsabilidade dos delegados.

É possível realizar várias operações sobre um inquérito: cancelar, abrir, fechar, finalizar, submeter resposta e obter resultados.

Cancelar um inquérito criado ou aberto (e sem respostas) corresponde a remover o inquérito. Se o inquérito estiver aberto, mas já tiver pelo menos uma resposta, então não pode ser removido e deve ser assinalado um erro. Cancelar um inquérito fechado corresponde a abri-lo novamente. Finalmente, cancelar um inquérito finalizado é impossível, pelo que deve ser assinalado um erro.

Um inquérito criado é aberto quando o projecto que lhe está associado é fechado. Um inquérito fechado pode ser reaberto. Caso se tente abrir um inquérito que não obedeça a estas restrições deve ser assinalado um erro.

Um inquérito aberto pode ser fechado. Se estiver fechado, a operação de fecho não tem efeito. Em qualquer outra situação deve dar um erro. Finalmente, um delegado pode finalizar um inquérito que esteja fechado. Se o inquérito já estiver finalizado, a operação não tem efeito. Nas restantes situações, fechar um inquérito corresponde a um erro.

Um inquérito fechado pode ser finalizado, impedindo futuras alterações. Finalizar um inquérito previamente finalizado não tem qualquer efeito. Finalizar inquéritos noutras situações corresponde a um erro.

Só os alunos que entregaram o projecto (ou seja, realizaram pelo menos uma submissão) podem responder ao inquérito. Cada aluno só pode responder uma vez (respostas subsequentes são ignoradas). As respostas devem ser anónimas, ou seja, não deve guardada informação que permita saber qual a resposta de um dado aluno a um inquérito.

Os alunos podem submeter uma resposta a um inquérito aberto. Em qualquer outra situação, esta operação deve assinalar um erro. Alunos, docentes e delegados podem obter o resultado de um inquérito, de acordo com a sua relação com o inquérito.

Note-se que as operações indicadas anteriormente estão disponíveis apenas para pessoas que estejam associadas à disciplina à qual pertence o projecto com o inquérito em causa.

Notificações

Deve existir um mecanismo de mensagens que permita avisar utilizadores quando os inquéritos associados a uma dada disciplina ficam em determinadas situações:

- Quando um inquérito de um projecto de uma disciplina abre, quer-se enviar uma mensagem a todos os alunos, delegados e docentes da disciplina.
- Quando um inquérito de um projecto de uma disciplina finaliza, quer-se enviar uma mensagem a todos os alunos, delegados e docentes da disciplina.

Nota: dado que um aluno pode também ser delegado, o mecanismo de envio de mensagens deve tratar este como se fosse uma única entidade. Assim, um aluno de uma disciplina que seja também delegado deve receber apenas uma mensagem e não duas cada vez que o sistema tem que enviar uma mensagem.

A apresentação das mensagens enviadas para um dado utilizador deve ser feita quando o utilizador faz login no sistema (ver abaixo). Após o login ter sido realizado com sucesso devem ser apresentadas todas as mensagens que tenham sido enviadas para o utilizador em causa desde a última vez que se registou no sistema. Após esta visualização, considera-se que o utilizador fica sem mensagens. As mensagens devem ser apresentadas pela mesma ordem em que foram enviadas pelo sistema.

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:


- A determinação da descrição de um utilizador deve estar realizada de forma a evitar a duplicação de código comum, facilitando assim a introdução de novas funcionalidades.
- A concretização da entidade inquérito deve ser feita por forma a aumentar a legibilidade do código relacionado com esta entidade por forma a que caso se queira alterar a lógica de cada situação em que o inquérito possa estar, isso possa ser feito facilmente.
- O mecanismo de envio das mensagens deve ser suficientemente flexível para suportar outras entidades que queiram também receber as mensagens. Deve ainda permitir que os destinatários das mensagens possam dizer se querem ou não ser notificados relativamente a cada disciplina. Por exemplo, um aluno de uma disciplina pode dizer que já não quer receber mensagens relativas a abrir e finalizar um inquérito associado a disciplina.


Funcionalidade da aplicação

A aplicação permite manter informação sobre as entidades do modelo, permitindo, em particular, gerir projectos e inquéritos. 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 *carregado no início da aplicação*. Não é possível adicionar ou remover pessoas durante a execução da aplicação. Não é possível a um utilizador alterar a informação pessoal de outros utilizadores.

A aplicação tem comportamento que depende do utilizador, pelo que deve permitir o estabelecimento de uma identidade inicial (login). Um utilizador que tenha estabelecido a sua identidade com sucesso é dito como estando registado. Em cada momento, a aplicação tem, no máximo, um utilizador registado.

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

 A interface geral do core já está *parcialmente implementada* na classe `sth.SchoolManager` 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.

Gestão de informação sobre pessoas

Um utilizador registado pode ver todos os utilizadores ou só um dado utilizador, procurar um utilizador e actualizar o seu número de telefone.

Gestão de actividades dos alunos

Um utilizador registado que seja aluno pode realizar as seguintes tarefas: entregar um projecto, preencher um inquérito e ver resultados de um inquérito.

Gestão de actividades dos delegados

Um utilizador registado que seja delegado tem à sua disposição as seguinte funcionalidade: criar inquéritos, apagar inquéritos, abrir inquéritos, fechar inquéritos, finalizar inquéritos e ver os resultados de inquéritos.

Gestão de actividades dos professores

Um utilizador registado que seja um docente pode realizar as seguintes actividades: criar projectos, fechar projectos, ver as submissões de um projecto, ver os alunos de uma disciplina leccionada e ver resultados de inquéritos.

Serialização

É possível guardar e recuperar o estado actual da aplicação, preservando toda a informação relacionada com a universidade e que foi descrita *acima*.


Interação 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 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. As mensagens são produzidas pelos métodos das *bibliotecas de suporte* (`po-utility` e `sth-app`). As mensagens não podem ser usadas no núcleo da aplicação (**sth-core**). Além disso, não podem ser definidas novas.

Potenciais omissões devem ser esclarecidas antes de qualquer implementação.

As excepções usadas na interação, excepto se indicado, são subclasses de `pt.tecnico.po.ui.DialogException`, são lançadas pelos comandos e tratadas por `pt.tecnico.po.ui.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* `sth.app`, `sth.app.main`, `sth.app.person`, `sth.app.teaching`, `sth.app.student` e `sth.app.representative`. Estas classes são de uso obrigatório e estão disponíveis no CVS (módulo `sth-app`).

Início da aplicação: estabelecimento da identidade do utilizador


Antes de iniciar o menu principal, a aplicação deve primeiro registar o utilizador. Assim, pede-se o identificador do utilizador a registar (`requestPersonId()`). Se o identificador não corresponder a um utilizador conhecido, a aplicação termina. Caso contrário, o menu principal é aberto.

Note-se que nesta fase, não existem nunca notificações para apresentar, já que não é possível recuperar aqui nenhum estado anterior.

Menu Principal

As acções deste menu permitem gerir a salvaguarda do estado da aplicação e abrir submenus. A lista completa é a seguinte: *Abrir*, *Guardar* e *Portal Pessoal*, *Portal do Docente*, *Portal do Estudante* e *Portal do Delegado*. Inicialmente, a aplicação apenas tem informação sobre as pessoas, cursos e disciplinas que foram carregados no arranque. Note-se que nem todas as opções do menu estão disponíveis para todos os utilizadores (ver abaixo).

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

 Estes comandos *já estão implementados nas classes da package* `sth.app.main` (disponível no CVS), respectivamente: `DoOpen`, `DoSave`, `DoOpenPersonMenu`, `DoOpenTeachingMenu`, `DoOpenStudentMenu`, `DoOpenRepresentativeMenu`.

Salvaguarda do estado actual da aplicação

O conteúdo da aplicação (toda a informação devida pela universidade 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 (`openFile()`). Caso o ficheiro não exista, é apresentada a mensagem `fileNotFound()`. Esta opção substitui toda a informação da aplicação. Note-se que quando esta operação é executada, o login actual deve ser revalidado (i.e., o identificador do utilizador deve ser verificado contra o novo conteúdo). Caso o identificador do utilizador actual não exista na informação que está no ficheiro a carregar, a operação falha e deve ser lançada a excepção `NoSuchPersonException`, não sendo substituída a informação actual da aplicação. Caso o utilizador exista, são apresentadas as mensagens enviadas para esse utilizador, de acordo com o formato seguinte.

Formato de mensagens para o utilizador	[Expandir]
Exemplo de mensagens para o utilizador	[Expandir]
<ul style="list-style-type: none">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 interação realiza-se através do método <code>newSaveAs()</code>. 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 guarda o estado da aplicação, mesmo que existam alterações.

Gestão e consulta de dados da aplicação

- Portal Pessoal** – Abre o menu de consulta de informações sobre pessoas.
- Portal do Docente** – Abre o menu de gestão de actividades dos professores (esta opção apenas está disponível para professores).
- Portal do Estudante** – Abre o menu de gestão de actividades dos alunos (esta opção apenas está disponível para alunos).
- Portal do Delegado** – Abre o menu de gestão de actividades dos delegados (esta opção apenas está disponível para delegados).

Portal Pessoal

Este menu permite efectuar operações sobre a base de dados de pessoal da universidade, estando disponível para todos os utilizadores. A lista completa é a seguinte: *Mostrar pessoa*, *Alterar número de telefone*, *Mostrar pessoas*, *Procurar pessoas*.

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

Sempre que for pedido o identificador de uma pessoa (`requestPersonId()`) e a pessoa não existir, é lançada a excepção `NoSuchPersonException`.

 Estes comandos *já estão implementados nas classes da package* `sth.app.person` (disponível no CVS), respectivamente: `DoShowPerson`, `DoChangePhoneNumber`, `DoShowAllPersons`, `DoSearchPerson`.

Mostrar pessoa

Apresenta as informações sobre a pessoa especificada pela identificação fornecida pelo processo de login, de acordo com o seguinte formato (e variações descritas abaixo).

Formato de apresentação (cabeçalho)	[Expandir]
Exemplo de apresentação (aluno)	[Expandir]

Dependendo do tipo de pessoa, *TIPO* pode tomar os valores `FUNCIONARIO`, `DOCENTE`, `ALUNO` e `DELEGADO` (para alunos que são delegados). Se os utilizadores forem do tipo `DOCENTE`, então a linha de identificação devem seguir-se às linhas com os nomes dos cursos e das disciplinas que lecciona nesses cursos, ordenados por ordem alfabética do nome do curso e da disciplina (dentro do curso). Se os utilizadores forem do tipo `ALUNO` ou `DELEGADO`, então a linha de identificação devem seguir-se às linhas com os nomes das disciplinas em que está inscrito, ordenados por ordem alfabética do nome da disciplina.

Exemplo de apresentação (professor)	[Expandir]
Exemplo de apresentação (aluno)	[Expandir]

Alterar número de telefone

Pede o novo número de telefone (`requestPhoneNumber()`). De seguida, altera o número de telefone da pessoa registada e apresenta as informações da pessoa (tal como descrito em *Mostrar pessoa*).

Mostrar pessoas

Apresenta informações sobre todas as pessoas e contactos.

A lista é ordenada pelo identificador da pessoa e o formato é o descrito em *Mostrar pessoa*.

Procurar pessoa


Pede o nome da pessoa a procurar (`requestPersonName()`). A operação apresenta a lista de pessoas (ordenadas por ordem alfabética do nome) cujo nome satisfaz a procura (o nome da pessoa deve conter a cadeia de caracteres em questão). O formato de apresentação é o descrito em *Mostrar pessoa*.

Portal do Docente

Este menu apresenta as operações disponíveis para docentes, estando apenas disponível para estes utilizadores. A lista completa é a seguinte: *Criar projecto*, *Fechar projecto*, *Ver submissões de um projecto*, *Ver alunos de disciplina leccionada*, *Ver resultados de um inquérito*.

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

Sempre que é pedido o nome da disciplina (`requestDisciplineName()`), é lançada a excepção `NoSuchDisciplineException`, se a disciplina indicada não existir ou se o docente não leccionar a disciplina em causa.

 Estes comandos *já estão implementados nas classes da package* `sth.app.teaching` (disponível no CVS), respectivamente: `DoCreateProject`, `DoCloseProject`, `DoShowProjectSubmissions`, `DoShowDisciplineStudents`, `DoRequestProjectException`.

Criar projecto

É pedido o nome da disciplina e o nome do projecto no contexto dessa disciplina (`requestProjectName()`). É lançada a excepção `DuplicateProjectException`, se o projecto já existir. Em caso de identificação bem sucedida, o projecto é criado.

Fechar projecto

É pedido o nome da disciplina e o nome do projecto no contexto dessa disciplina (`requestProjectName()`). É lançada a excepção `NoSuchProjectException`, se o projecto não existir. Em caso de identificação bem sucedida, o projecto é fechado.

Se o projecto já estiver fechado, o comando não executa nenhuma acção.

Ver submissões de um projecto

É pedido o nome da disciplina e o nome do projecto no contexto dessa disciplina (`requestProjectName()`). É lançada a excepção `NoSuchProjectException`, se o projecto não existir. Em caso de identificação bem sucedida, as submissões do projecto são apresentadas, ordenadas pelo número de aluno que realizou a submissão, e com o seguinte formato.

Formato de apresentação	[Expandir]
Exemplo de apresentação	[Expandir]

Este comando pode operar, tanto sobre projectos abertos, como sobre projectos fechados.

Ver alunos de disciplina leccionada

É pedido o nome da disciplina. Em caso de identificação bem sucedida, é apresentada a lista de alunos, ordenada pelo identificador de aluno, no formato definido em *Mostrar pessoa*.

Ver resultados de um inquérito

É pedido o nome da disciplina e o nome do projecto no contexto dessa disciplina (`requestProjectName()`). É lançada a excepção `NoSuchProjectException`, se o projecto não existir. e `NoSurveyException`, se o inquérito não tiver sido criado para o projecto em causa. Em caso de identificação bem sucedida, os resultados do inquérito são apresentados de acordo com o formato seguinte.


Formato de apresentação para inquéritos criados (por abrir)	[Expandir]
Formato de apresentação para inquéritos abertos	[Expandir]
Formato de apresentação para inquéritos fechados	[Expandir]
Formato de apresentação para inquéritos finalizados	[Expandir]
Exemplo de apresentação	[Expandir]

Portal do Estudante

Este menu apresenta as operações disponíveis para alunos, estando apenas disponível para estes utilizadores. A lista completa é a seguinte: *Entregar projecto*, *Preencher inquérito*, *Ver resultados de inquérito*.

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

Sempre que é pedido o nome da disciplina (`requestDisciplineName()`), é lançada a excepção `NoSuchDisciplineException`, se a disciplina indicada não existir ou se não for do curso do delegado. Sempre que é pedido o nome do projecto no contexto dessa disciplina (`requestProjectName()`), é lançada a excepção `NoSuchProjectException`, se o projecto não existir.

 Estes comandos *já estão parcialmente implementados nas classes da package* `sth.app.student` (disponível no CVS), respectivamente: `DoDeliverProject`, `DoAnswerSurvey`, `DoShowSurveyResults`.

Entregar projecto

É pedido o nome da disciplina, o nome do projecto no contexto dessa disciplina e o texto relativo à entrega (`requestDeliveryMessage()`). É lançada a excepção `NoSuchProjectException`, se o projecto não estiver aberto.

Em caso de identificação bem sucedida, a submissão do projecto é registada.

Preencher inquérito

É pedido o nome da disciplina e o nome do projecto no contexto dessa disciplina. De seguida, é pedido o número de horas gasto a realizar o projecto (`requestProjectHours()`) e o comentário ao projecto (`requestComment()`), procedendo-se ao registo da resposta ao inquérito.

É lançada a excepção `NoSuchProjectException`, se o aluno não tiver feito nenhuma submissão, e `NoSurveyException`, se o inquérito não tiver sido criado para o projecto em causa ou não estiver aberto.

Ver resultados de inquérito

É pedido o nome da disciplina e o nome do projecto no contexto dessa disciplina. É lançada a excepção `NoSuchProjectException`, se o aluno não tiver feito nenhuma submissão, e `NoSurveyException`, se o inquérito não tiver sido criado para o projecto em causa. Em caso de identificação bem sucedida, os resultados do inquérito são apresentados de acordo com o formato seguinte.

Para os inquéritos criados (por abrir), abertos e fechados, o formato de apresentação é como para os docentes.

Formato de apresentação para inquéritos finalizados	[Expandir]
Exemplo de apresentação	[Expandir]

Caso uma disciplina não tenha projectos ou projectos com inquéritos, não deve ser apresentada nenhuma saída.

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.io.tmpdir`.

Exemplo de ficheiro a importar

Cada utilizador tem uma descrição distinta mas que segue o seguinte formato geral.

<pre>TIPO identificador telefone nome Dependendo do tipo de pessoa, TIPO pode tomar os valores FUNCIONARIO, DOCENTE, ALUNO e DELEGADO (para alunos que são delegados). A seguir à linha da descrição, podem seguir-se outras (iniciadas pelo carácter #) que apresentem informação de disciplina para um utilizador em causa. A informação adicional para os alunos (delegados ou não) e para os docentes é uma lista de cursos e disciplinas (a utilizar por linha): para os alunos, são as disciplinas que frequentam; para os docentes, são as disciplinas que leccionam. Para os funcionários, não há informação adicional a processar. # Nome do curso Nome da disciplina Exemplo de ficheiro de entrada textual</pre>	[Expandir]
---	-------------------

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

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 (`sth.app.App.main`). As propriedades são tratadas automaticamente pelo código de apoio.

<pre>java -Dimport=test.import -Din=test.in -Dout=test.outyhy sth.app.App Assumindo que aqueles ficheiros estão no directório onde é dado o comando de execução, o programa produz o ficheiro de saída test.outyhy. Em caso de sucesso, os ficheiros das saídas esperada (test.out) e obida (test.outyhy) devem ser iguais. A comparação pode ser feita com o comando: diff -b test.out test.outyhy</pre>	[Expandir]
---	-------------------

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