TÉCNICO LISBOA

Projecto de Programação com Objectos
6 de Outubro de 2018

Esclarecimento de dúvidas:

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

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

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

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

- Datas: **2018/10/19 18:00** (UML); **2018/11/16 23:59** (intercalar); **2018/12/07 23:59** (final); **2018/12/17-2018/12/20** (teste prático).
- Os diagramas de classe UML são entregues exclusivamente em papel (impressos ou manuscritos) na portaria do Tagus. Diagramas ilegíveis ou sem identificação completa serão sumariamente ignorados. É obrigatório a identificação do grupo e turno de laboratório (dia, hora e sala).
- **Apenas se consideram para avaliação os projetos submetidos no Fénix.** As classes criadas de acordo com as especificações fornecidas devem ser empacotadas num arquivo de nome `proj.jar` (que deverá apenas conter os ficheiros `.java` do código realizado guardados nos *packages* correctos). O ficheiro `proj.jar` deve ser entregue para avaliação através da ligação presente no Fénix. É possível realizar múltiplas entregas do projecto até à data limite, mas apenas será avaliada a última entrega.
- Não serão consideradas quaisquer alterações aos ficheiros de apoio disponibilizados: eventuais entregas dessas alterações serão automaticamente substituídas durante a avaliação da funcionalidade do código entregue.
- Trabalhos não presentes no Fénix no final do prazo têm classificação 0 (zero) (não são aceites outras formas de entrega).
- A avaliação do projecto pressupõe o compromisso de honra de que o trabalho foi realizado pelos alunos correspondentes ao grupo de avaliação.
- **Fraudes na execução do projecto terão como resultado a exclusão dos alunos implicados do processo de avaliação**.

O objectivo 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.

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

1 Entidades do Domínio

Nesta secção descrevem-se as várias entidades que vão ser manipuladas no contexto da aplicação a desenvolver. Existem vários conceitos importantes neste contexto: universidade, curso, aluno, professor, funcionário, projecto e 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.

1.1 Universidade, Cursos e Disciplinas

Uma universidade tem um nome e pode ter vários cursos, tendo ainda funcionários, docentes e alunos. Os alunos estão exclusivamente associados a um curso. Os funcionários e os docentes podem estar associados a vários 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 se podem inscrever 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 e tem 0 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.

1.2 Pessoas

O sistema a desenvolver tem de lidar com diferentes tipos de utilizadores: alunos, docentes e funcionários. Alunos, docentes e funcionários têm um nome, número de telefone e são identificados por um identificador único (inteiro 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.

1.3 Projectos

Os docentes podem criar projectos associados às disciplinas que leccionam. Quando um projecto é criado fica automaticamente aberto. Cada projecto tem um nome e descrição. O nome deve ser único no contexto da disciplina.

Um aluno pode submeter a sua versão do projecto desde que o projecto ainda 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 submissão. Por razões de simplificação, a submissão é representada por uma cadeia de caracteres introduzida pelo aluno.

Um projecto pode estar aberto ou fechado. Só um docente da disciplina é que pode criar e fechar projectos.

1.4 Inquéritos

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

Um inquérito pode estar em várias situações: criado, aberto, fechado e finalizado. O comportamento do inquérito dependa da situação em que ele esteja. A mudança de situação é explícita, excepto quando o projecto associado é fechado. Neste caso, o inquérito do projecto passa automaticamente da situação de criado para aberto. Esta é a única mudança implícita, as outras mudanças são explícitas e da exclusiva responsabilidade de um delegado. O delegado tem que pertencer ao mesmo curso da disciplina no contexto do qual o inquérito está a ser realizado.

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 do sistema. Se o inquérito estiver aberto mas já tiver pelo menos uma resposta, então não pode ser removido e deve dar erro. Cancelar um inquérito fechado corresponde a abri-lo outra vez. Finalmente, cancelar um inquérito finalizado é impossível pelo que deverá dar erro.

Um inquérito criado é aberto quando o projecto que lhe está associado fica fechado. Caso se tente abrir um inquérito que não obedeça a estas duas restrições deve ser dado 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.

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) é que podem responder ao inquérito. Cada aluno só pode responder uma única vez. As respostas devem ser anónimas, ou seja, o sistema não deve guardar informação que permita saber qual a resposta de um dado aluno a um inquérito.

Os alunos inscritos a uma disciplina podem submeter uma resposta a um inquérito aberto associado a um projecto da disciplina em causa.. Em qualquer outra situação, esta operação deverá dar um erro. Alunos, docentes e delegados podem obter o resultado de um inquérito que esteja finalizado. Nas restantes situações deve apenas ser indicado o estado do inquérito. Note-se que os alunos devem estar inscritos na disciplina associada ao inquérito (via projecto), os docentes devem leccionar a disciplina (via projecto) em causa e os delegados devem pertencer ao mesmo curso da disciplina. Caso isto não aconteça, então deve ser assinalado um erro.

1.5 Notificações

Deve existir um mecanismo de mensagens que permita avisar determinados 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. 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 enviadas pelo sistema.

2 Requisitos de Desenho

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

- A determinação da descrição de um utilizador deve estar realizada de forma a evitar a duplicação de código comum, facilitando assim a introdução de novos tipos de utilizadores. Ver mais detalhes na secção 4.3.1.
- A concretização da entidade inquérito deve ser feita por forma 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 à disciplina.

3 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 carregada no início da aplicação. Não é possível adicionar ou remover pessoas durante a execução da aplicação.

Note-se que não é necessário concretizar de raiz a aplicação. Já é fornecido algum código, nomeadamente, o esqueleto das classes necessárias para concretizar os menus e respectivos comandos a utilizar na aplicação a desenvolver, a classe `sth.app.App`, que representa o ponto de entrada da aplicação a desenvolver, algumas classes do domínio da aplicação e um conjunto de excepções a utilizar durante o desenvolvimento da aplicação. A interface geral do core já está parcialmente concretizada na classe `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. A classe `sth.SchoolManager` deverá ser finalizada por cada grupo e deverá representar a interface geral do domínio da aplicação desenvolvida. Por esta razão, os comandos a desenvolver na camada de interface com o utilizador devem utilizar exclusivamente esta classe para aceder às funcionalidades suportadas pelas classes do domínio da aplicação (a serem desenvolvidas por cada grupo na package `sth.core`). Desta forma será possível concretizar toda a interacção com o utilizador completamente independente da concretização das entidades do domínio.

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.

3.1 Consulta de informação sobre pessoas

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

3.2 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.

3.3 Gestão de actividades dos delegados

Um utilizador registado que seja delegado tem à sua disposição as seguintes tarefas: criar um inquérito, apagar um inquérito, abrir um inquérito, fechar um inquérito, finalizar um inquérito, mostrar os inquéritos de uma disciplina.

3.4 Gestão de actividades dos professores

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

3.5 Serialização

É possível guardar e recuperar o estado actual da aplicação, preservando toda a informação relacionada com a universidade e que foi descrita na secção 1.

4 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-uuilib** e **sth-app**). **Não** podem ser definidas novas mensagens. Potenciais omissões devem ser esclarecidas com o corpo docente antes de qualquer concretização.

Não deve haver código de interacção com o utilizador no núcleo da aplicação (*sth.core*). Desta forma, será possível reutilizar o código do núcleo da aplicação (onde é concretizado o domínio da aplicação) com outras concretizações da interface com o utilizador.

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

Note-se que o programa principal e os comandos e menus, a seguir descritos, já estão parcialmente concretizados nos 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 na secção *Projecto* da página da cadeira.

4.1 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 utilizando a mensagem devolvida por `sth.app.main.requestPersonId()`. Se o identificador não corresponder a um utilizador conhecido, a aplicação termina. Caso contrário, são apresentadas as mensagens enviadas para o utilizador. O menu principal é aberto logo de seguida.

De seguida apresenta-se um possível exemplo de apresentação das mensagens de um utilizador após ter feito o registo:

```
Pode␣preencher␣inquérito␣do␣projecto␣Gatos␣Simples␣da␣disciplina␣Programação␣com␣Objectos
Resultados␣do␣inquérito␣do␣projecto␣Gatos␣Complexos␣da␣disciplina␣Programação␣com␣Objectos
Resultados␣do␣inquérito␣do␣projecto␣1°␣Projecto␣da␣disciplina␣Fundamentos␣da␣Programação
```

Após apresentar as mensagens, inicia-se o menu principal tendo em conta o tipo do utilizador registado.

4.2 Menu Principal

As acções deste menu permitem gerir a salvaguarda do estado da aplicação e abrir submenus. A lista completa é a seguinte: **Reiniciar**, **Abrir**, **Guardar** , **Portal Pessoa**, **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 (como descrito mais à frente). 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`.

Os comandos que vão concretizar as funcionalidades deste menu já estão parcialmente concretizados nas várias classes do package `sth.app.main`, respectivamente: `DoOpen`, `DoSave`, `DoOpenPersonelMenu`, `DoOpenTeachingMenu`, `DoOpenStudentMenu` e `DoOpenRepresentativeMenu`.

4.2.1 Salvaguarda do estado actual

O conteúdo da aplicação (inclui todos as entidades descritas no domínio da aplicação, e.g. cursos, disciplinas, docentes, etc, atualmente 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 (ficando associado à aplicação). A informação a carregar compreende todas as entidades do domínio da aplicação. A mensagem (cadeia de caracteres) a utilizar para pedir o nome do ficheiro a abrir é a devolvida pelo método `openFile()`. Caso o ficheiro não exista é apresentada a mensagem `fileNotFound()`.

Guardar – Guarda o estado actual da aplicação (inclui todas as entidades do domínio da aplicação) no ficheiro associado. Se não existir associação, pede-se o nome do ficheiro a utilizar, ficando a ele associado. Esta interacção realiza-se através do método `newSaveAs()`.

A opção `Sair` **nunca** guarda o estado da aplicação, mesmo que existam alterações.

4.2.2 Gestão e consulta de dados da aplicação

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

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 professors).

Portal do Estudante – Abre o menu de gestão de actividades dos estudantes (esta opção apenas está disponível para estudantes).

Portal do Delegado – Abre o menu de gestão de actividades dos delegados (esta opção apenas está disponível para delegados).

4.3 Portal Pessoal

Este menu permite efectuar consultas sobre a base de dados de pessoal da universidade. A lista completa é a seguinte: **Mostrar pessoas**, **Mostrar pessoa**, **Alterar número de telefone** e **Procurar pessoa**.

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 (utilizando a mensagem devolvida por `requestPersonId()`) e o serviço não existir, deve ser lançada a excepção `NoSuchPersonException`.

Os comandos deste menu já estão parcialmente concretizados nas classes do package `sth.app.person`: `DoShowAllPersons`, `DoShowPerson`, `DoChangePhoneNumber` e `DoSearchPerson`. De seguida descreve-se com algum detalhe a funcionalidade que cada comando deve concretizar.

4.3.1 Mostrar pessoa

Apresenta a informação do utilizador registado de acordo com o seguinte formato (e variações descritas abaixo). O formato de apresentação do cabeçalho da informação a apresentar é o seguinte:

```
TIPO|identificador|telefone|nome
```

Dependendo do tipo de pessoa, `TIPO pode tomar os valores FUNCIONÁRIO, DOCENTE, ALUNO e DELEGADO (para alunos que são delegados). Se o utilizador for um docente`, então à linha de identificação devem seguir-se as 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 o utilizador for um alunos ou delegados, então à linha de identificação devem seguir-se as linhas com os nomes das disciplinas em que está inscrito, ordenados por ordem alfabética do nome da disciplina.
Exemplo de apresentação (professor)

```
DOCENTE|100050|987␣654␣321|Maria␣João
*␣Informática␣-␣Análise␣e␣Síntese␣de␣Algoritmos
*␣Matemática␣-␣Algebra

Exemplo␣de␣apresentação␣(aluno)
ALUNO|101090|123␣456␣789|João␣Maria
*␣Informática␣-␣Análise␣e␣Síntese␣de␣Algoritmos
*␣Informática␣-␣Inteligência␣Artificial
*␣Informática␣-␣Sistemas␣Operativos
*␣Informática␣-␣Fundamentos␣da␣Programação
*␣Informática␣-␣Programação␣com␣Objectos
```

Para um utilizador que seja funcionário apenas é apresentado a linha referente ao cabeçalho.

4.3.2 Mostrar pessoas

Apresenta informações sobre todas as pessoas conhecidas pelo sistema. A lista é ordenada pelo identificador da pessoa e a informação de cada pessoa é apresentada de acordo com o formato descrito na secção §4.3.1.

4.3.3 Alterar número de telefone

Pede o novo número de telefone (utilizando a cadeia de caracteres devolvido por `requestPhoneId()`). De seguida altera o número de telefone da pessoa registada e apresenta as informações da pessoa utilizando o formato descrito na secção §4.3.1.

4.3.4 Procurar pessoa

Pede uma cadeia de caracteres ao utilizador (`requestPersonName()`) e de seguida apresenta as informações de todos as pessoas conhecidas pelo sistema cujo nome contém a cadeia de caracteres inserida pelo utilizador. As pessoas são apresentadas por ordem alfabética do nome (crescente) utilizando o formato descrito na secção §4.3.1.

4.4 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** e **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`. Os comandos que concretizam estas operações já estão parcialmente concretizados nas classes do package `sth.app.teaching`: `DoCreateProject`, `DoCloseProject`, `DoShowProjectSubmissions`, `DoShowDisciplineStudents` e `DoShowSurveyResults`.

Considere que sempre que for pedido o nome de um projecto ou disciplina deve ser utilizada a mensagem devolvida por `requestProjectName()`, `requestDisciplineName()`, respectivamente. Se o projecto não existir ou o docente não leccionar uma disciplina com o nome indicado deve ser lançada a excepção `NoSuchProjectException` ou `NoSuchDisciplineException`, respectivamente. Considere ainda que quando um docente tentar realizar uma operação sobre uma disciplina que não lecciona deve ser lançada a excepção `NoSuchDisciplineException`.

4.4.1 Criar projecto

Cria um projecto numa disciplina leccionada pelo docente registado. Para tal, primeiro pede-se o nome da disciplina e o nome do projecto no contexto dessa disciplina (`requestProjectNameId()`). De seguida cria-se o projecto, caso seja possível.

Além das excepções indicadas em 4.4, a execução deste comando `DuplicateProjectException`, se o projecto já existir.

4.4.2 Fechar projecto

Fecha um projecto de uma disciplina leccionada pelo docente registado. Para tal, primeiro pede-se o nome da disciplina e do projecto e de seguida fecha o projecto, caso seja possível.

A execução deste comando pode lançar as excepções indicadas na secção 4.4.

4.4.3 Ver submissões de um projecto

Mostra as submissões já efectuadas pelos alunos de uma disciplina leccionada pelo docente registado. Para tal, primeiro pede-se o nome do curso e do projecto e de seguida, caso seja possível, apresentam-se as submissões do projecto indicado, ordenadas por ordem crescente do número de aluno que realizou a submissão, utilizando o seguinte formato:

```
Nome␣da␣Disciplina␣-␣Nome␣do␣Projecto
*␣Identificador␣do␣1º␣aluno␣-␣submissão␣do␣1º␣aluno
...
*␣Identificador␣do␣Nº␣aluno␣-␣submissão␣do␣Nº␣aluno
```

Por exemplo, considere que a disciplina *Programação com Objectos* tem o projecto com o nome *Gatos Simples* e que apenas houve três submissões para este projecto. Considere ainda que as cadeias de caracteres associadas a cada submissão são `Gato.java`, `Cat.java` e `Tigre.ajva`, respectivamente. Neste caso, caso fosse este o projecto escolhido para ver as submissões, esta opção deveria apresentar o seguinte:

```
Programação␣com␣Objectos␣-␣Gatos␣Simples
*␣0234␣-␣Gato.java
*␣6789␣-␣Cat.java
*␣7912␣-␣Tigre.java
```

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

A execução deste comando pode lançar as excepções indicadas na secção 4.4.

4.4.4 Ver alunos de disciplina leccionada

Mostra os alunos inscritos numa disciplina leccionada pelo docente registado. Para tal, primeiro pede-se o nome da disciplina e depois, caso seja possível, é apresentada a lista de alunos, ordenada pelo identificador de aluno. Cada aluno é apresentado utilizando o formato definido na secção §4.3.1.

A execução deste comando pode lançar as excepções indicadas na secção 4.4.

4.4.5 Ver resultados de um inquérito

Mostra os resultados de um inquérito de um projecto uma disciplina leccionada pelo docente. Para tal, primeiro pede-se o nome da disciplina e do projecto e de seguida, caso seja possível, apresentam-se os resultados do inquérito associado ao projecto indicado.

Além das excepções indicadas na secção 4.4, a execução deste comando deverá lançar a excepção `NoSurveyException` caso o projecto indicado não tenha nenhum inquérito associado.

Os inquéritos criados, abertos e fechados devem usar o formato seguinte:

```
Nome␣da␣disciplina␣-␣Nome␣do␣projecto␣(CASO)
```

onde caso é substituído por `por abrir`, `aberto` ou `fechado` dependendo se o inquérito está na situação de criado, aberto ou fechado, respectivamente. No caso dos projectos finalizados, o formato a utilizar é o seguinte:

```
Nome␣da␣disciplina␣-␣Nome␣do␣projecto
␣*␣Número␣de␣submissões:␣Número␣de␣submissões
␣*␣Número␣de␣respostas:␣Número␣de␣respostas
␣*␣Tempos␣(mínimo,␣médio,␣máximo):␣MIN,␣MED,␣MAX
```

em que `MIN`, `MED` e `MAX` representam, respectivamente, o tempo mínimo de resolução do projecto, o tempo médio de resolução do projecto e o tempo máximo de resolução do projecto

De seguida apresenta-se um possível resultado de executar este comando três vezes para três projectos distintos:

```
Programação␣com␣Objectos␣-␣Coisas␣Estranhas␣(aberto)
Programação␣com␣Objectos␣-␣Gatos␣Simples
␣*␣Número␣de␣submissões:␣30
␣*␣Número␣de␣respostas:␣20
␣*␣Tempos␣de␣resolução␣(horas)␣(mínimo,␣médio,␣máximo):␣10,␣16,␣20
Programação␣com␣Objectos␣-␣Gatos␣Sofisticados␣(fechado)
```

Note-se que em cada execução do comando, apenas deve ser apresentado um inquérito (o do projecto especificado).

4.5 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** e **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`. Os comandos que concretizam as operações deste menu já estão parcialmente implementados nas classes da package `sth.app.passenger`, respectivamente: `DoDeliverProject`, `DoAnswerSurvey` e `DoShowSurveyResults`.

Considere que sempre que for pedido o nome de um projecto ou disciplina deve ser utilizada a mensagem devolvida por `requestProjectName()` ou `requestDisciplineName()`, respectivamente. Se o projecto não existir deve ser lançada a excepção `NoSuchProjectException`. Se o aluno registado não tiver escrito a nenhuma disciplina com o nome indicado então deve ser lançada a excepção `NoSuchDisciplineException`.

4.5.1 Entregar projecto

Realiza a submissão de um projecto de uma disciplina a que o aluno registado está inscrito. Primeiro pede o nome da disciplina e do projecto a entregar e o texto relativo à entrega (`requestDeliveryMessage()`). De seguida, caso seja possível, regista a submissão do aluno no projecto indicado.

Além das excepções indicadas na secção 4.5, a execução deste comando deve lançar a excepção `NoSuchProjectException` se o projecto existir mas não estiver aberto

4.5.2 Preencher inquérito

Preenche o inquérito de um projecto de uma disciplina a que o aluno registado está inscrito. Assim , primeiro pede o nome da disciplina e do projecto com um inquérito a preencher. Pede ainda o número de horas gasto a realizar o projecto (`requestProjectHours()`) e o comentário ao projecto (`requestComment()`). Em caso de identificação bem sucedida, a resposta ao inquérito é registada.

Além das excepções indicadas em 4.5, pode ainda serem lançadas as excepções `NoSurveyException`, caso o inquérito não esteja aberto ou não exista, ou `NoSuchProjectException` , caso o aluno não tenha realizado qualquer submissão no projecto em causa.

4.5.3 Ver resultados de inquérito

Mostra os resultados de um inquérito de uma disciplina do aluno registado. Para tal, primeiro pede-se o nome da disciplina e de um projecto dessa disciplina. De seguida, caso seja possível, apresentam-se os resultados do inquérito associado ao projecto indicado.

Além das excepções indicadas em 4.5, a execução deste comando pode ainda lançar a excepção `NoSurveyException`, caso o projecto indicado não tenha nenhum inquérito indicado.

Os formato de apresentação de um inquérito é o mesmo indicado na secção 4.4.5 para os inquéritos que estejam criados, abertos ou fechados. Para os inquéritos finalizados deve-se utilizar o formato:

```
Nome␣da␣disciplina␣-␣Nome␣do␣projecto
␣*␣Número␣de␣respostas:␣Número␣de␣respostas
␣*␣Tempo␣médio␣(horas):␣Tempo␣médio
```

De seguida apresenta-se um possível resultado da execução deste comando para um projecto finalizado:

```
Programação␣com␣Objectos␣-␣Gatos␣Simples
␣*␣Número␣de␣respostas:␣20
␣*␣Tempo␣médio␣(horas):␣16
```

4.6 Portal do Delegado

Este menu apresenta as operações disponíveis para os delegados, estando apenas disponível para estes utilizadores. A lista completa é a seguinte: **Criar inquérito**, **Apagar inquérito**, **Abrir inquérito**, **Fechar inquérito**, **Finalizar inquérito** e **Mostrar inquéritos de uma disciplina**. As etiquetas das opções deste menu estão definidas na classe `sth.app.representative.Label`. A classe `sth.app.representative.Message` define todos os métodos correspondentes às mensagens de diálogo para este menu. Os comandos que concretizam as operações deste menu já estão parcialmente implementados nas classes da package `sth.app.representative`: `DoCreateSurvey`, `DoRemoveSurvey`, `DoOpenSurvey`, `DoCloseSurvey`, `DoFinishSurvey` e `DoShowSurveys`.

Considere que sempre que for pedido o nome de um projecto ou disciplina deve ser utilizada a mensagem devolvida por `requestProjectName()` ou `requestDisciplineName()`, respectivamente. Se o projecto não existir deve ser lançada a excepção `NoSuchProjectException`. Se o curso do delegado registado não tiver nenhuma disciplina com o nome indicado, então deve ser lançada a excepção `NoSuchDisciplineException`.

4.6.1 Criar inquérito

Cria um inquérito para um projecto de uma disciplina. Pede o nome da disciplina e do projecto no contexto dos quais deve ser criado o inquérito. De seguida tenta criar o inquérito para o projecto indicado. Além das excepções indicadas em 4.6, caso já exista um inquérito associado ao projecto, então deverá ser lançada a excepção `DuplicateSurveyException`.

4.6.2 Cancelar inquérito

Cancela o projecto indicado. Primeiro pede o nome da disciplina e do projecto e de seguida cancela o inquérito associado ao projecto (com a semântica indicada na secção 1.4. Além das excepções indicadas em 4.6, deverá ser lançada a excepção:

- `NoSurveyException` se o projecto não tiver nenhum inquérito.
- `NonEmptySurveyException` se o inquérito estiver aberto e já tiver pelo menos uma resposta.
- `SurveyFinishedException` se o inquérito estiver finalizado.

4.6.3 Abrir inquérito

É pedido o nome da disciplina e do projecto e de seguida tenta-se abrir o inquérito, caso seja possível. Além das excepções indicadas em 4.6, se o projecto não tiver um inquérito associado deve ser lançada a excepção `NoSurveyException`. Caso exista e não seja possível abri-lo, então deve ser lançada a excepção `OpeningSurveyException`.

4.6.4 Fechar inquérito

É pedido o nome da disciplina e do projecto e de seguida tenta-se fechar o inquérito, caso seja possível. Além das excepções indicadas em 4.6, e o projecto não tiver um inquérito associado deve ser lançada a excepção `NoSurveyException`. Caso exista e não seja possível fechá-lo, então deve ser lançada a excepção `ClosingSurveyException`.

4.6.5 Finalizar inquérito

É pedido o nome da disciplina e do projecto e de seguida tenta-se finalizar o inquérito, caso seja possível. Além das excepções indicadas em 4.6, se o projecto não tiver um inquérito associado deve ser lançada a excepção `NoSurveyException`. Caso exista e não seja possível finalizá-lo, então deve ser lançada a excepção `FinishingSurveyException`.

4.6.6 Mostrar inquéritos de uma disciplina

É pedido o nome da disciplina. Caso o delegado possa aceder à disciplina, então os inquéritos realizados para os projectos desta disciplina são apresentados de acordo com o formato seguinte, ordenados por ordem alfabética de nome de projecto: Os formato de apresentação de um inquérito é o mesmo indicado na secção 4.4.5 para os inquéritos que estejam criados, abertos ou fechados. Para os inquéritos finalizados deve-se utilizar o formato:

```
Nome␣da␣disciplina␣-␣Nome␣do␣projecto␣-␣Número␣de␣respostas␣respostas␣-␣Tempo␣médio␣de␣execução␣horas
```

De seguida apresenta-se um possível resultado da execução deste comando:

```
Programação␣com␣Objectos␣-␣Coisas␣Estranhas␣(aberto)
Programação␣com␣Objectos␣-␣Gatos␣Simples␣-␣20␣respostas␣-␣16␣horas
Programação␣com␣Objectos␣-␣Gatos␣Sofisticados␣-␣40␣respostas␣-␣20␣horas
Programação␣com␣Objectos␣-␣Mais␣Gatos␣-␣10␣respostas␣-␣2␣horas
Programação␣com␣Objectos␣-␣Zap␣to␣it!␣(fechado)
```

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

A execução desta operação pode lançar uma das excepções indicadas na secção 4.6.

5 Leitura de Dados a Partir de Ficheiros Textuais

Além das opções de manipulação de ficheiros descritas na secção §4.2.1, é possível iniciar a aplicação com um ficheiro de texto especificado pela propriedade Java com o nome `import`. Este ficheiro contém a descrição de cursos, disciplinas e pessoas da universidade a carregar no estado inicial da aplicação.

5.0.7 Exemplo de ficheiro a importar

Cada utilizador é descrito no ficheiro com o seguinte formato:

```
TIPO|identificador|telefone|nome
```

Dependendo do tipo de utilizador, `TIPO` pode tomar os valores `FUNCIONÁRIO`, `DOCENTE`, `ALUNO` e `DELEGADO` (para alunos que são delegados). A seguir a cada linha de descrição, podem seguir-se outras (iniciadas pelo carácter `#`) que acrescentam informação adicional para o utilizador em causa. A informação adicional para os alunos (delegados ou não) e para os docentes é uma lista de cursos e disciplinas (uma 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. De seguida descreve-se m exemplo de um ficheiro a importar:

```
FUNCIONÁRIO|100001|123789456|Maria␣José
DELEGADO|100010|123456789|João␣Maria
#␣Informática|Fundamentos␣da␣Programação
#␣Informática|Algoritmos␣e␣Estruturas␣de␣Dados
#␣Informática|Programação␣com␣Objectos
#␣Informática|Análise␣e␣Síntese␣de␣Algoritmos
DELEGADO|100011|123456789|João␣Manuel
#␣Informática|Fundamentois␣da␣Programação
#␣Informática|Algoritmos␣e␣Estruturas␣de␣Dados
#␣Informática|Programação␣com␣Objectos
#␣Informática|Análise␣e␣Síntese␣de␣Algoritmos
DELEGADO|100012|123456789|João␣Miguel
#␣Matemática|Álgebra␣Linear
#␣Matemática|Probabilidades␣e␣Estatística
#␣Matemática|Cálculo␣I
#␣Matemática|Métodos␣Numéricos
ALUNO|100013|123456789|José␣Manuel
#␣Matemática|Álgebra␣Linear
#␣Matemática|Probabilidades␣e␣Estatística
#␣Matemática|Cálculo␣I
#␣Matemática|Métodos␣Numéricos
DOCENTE|100100|987654321|Ana␣Maria
#␣Informática|Análise␣e␣Síntese␣de␣Algoritmos
#␣Matemática|Álgebra␣Linear
FUNCIONÁRIO|101010|789456123|José␣Carlos
```

Pode assumir que não existem entradas mal-formadas nestes ficheiros. A codificação dos ficheiros a ler é garantidamente `UTF-8`.

6 Execução dos Programas e Testes Automáticos

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

```
java -Dimport=test.import -Din=test.in -Dout=test.outhyp 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.outhyp`. Em caso de sucesso, os ficheiros das saídas esperada (`test.out`) e obtida (`test.outhyp`) devem ser iguais. A comparação pode ser feita com o comando:

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

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

7 Notas de Concretização

Tal como indicado neste documento, algumas classes fornecidas como material de apoio, são de uso obrigatório e não podem ser alteradas. 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).