TÉCNICO LISBOA

Projecto de Programação com Objectos
18 de Outubro de 2015

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 desenvolvimento.

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

- Datas: **2015/10/23 12:00** (UML); **2015/11/16 23:59** (intercalar); **2015/12/01 23:59** (final); **2015/12/01–2015/12/15** (teste prático).
- Os diagramas UML são entregues exclusivamente em papel (impressos ou manuscritos) na portaria do Tagus. Diagramas ilegíveis serão sumariamente ignorados.
- **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 em 2015/2016**.

O objectivo do projecto é criar uma aplicação que permite ver e alterar um documento textual. A aplicação mantém o nome do documento onde deve ser guardado o documento em disco. A aplicação apenas edita um documento de cada vez, embora possa trocar de documento.

Neste texto, o tipo `fixo` indica um literal; o símbolo ␣ indica um espaço; e o tipo *itálico* indica uma parte variável.

1 Estrutura e Formatação de um Documento

Um documento tem uma estrutura interna, correspondente à organização do seu conteúdo, sendo possível alterar este conteúdo.

Os documentos têm um título, um ou mais autores, e um conjunto de parágrafos e secções. A dimensão de um documento é o numero de caracteres presentes no documento. Assim, é necessário ter em conta todos os parágrafos e os títulos de todas as secções do documento.

Cada autor tem um nome e um contacto (email).

As secções têm ainda um título, uma sequência de parágrafos e uma sequência de subsecções (ambas potencialmente vazias). Os parágrafos e as subsecções são identificados implicitamente em cada secção pelo seu número de ordem (ambos com início em 0). Cada parágrafo (sequências de frases relacionadas contextualmente) ou secção pode ter um identificador único no contexto do documento (cadeia de caracteres).

É possível adicionar e remover secções e parágrafos a/de um documento. É possível alterar o conteúdo de um parágrafo.

2 Interação com o Utilizador: Editor de Documentos

Descreve-se abaixo 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 leitura e escrita **devem** realizar-se exclusivamente através de duas classes: `pt.utl.ist.po.ui.Form` e `pt.utl.ist.po.ui.Display`. As mensagens a apresentar durante a execução da aplicação são produzidas pelos métodos das bibliotecas de suporte (**po-uilib** e **edt-support**). Não deve haver código de interacção com o utilizador no núcleo da aplicação.

Desta forma,será possível reutilizar o código do núcleo da aplicação com outras concretizações da interface com o utilizador. Além disso, não podem ser definidas novas mensagens. Potenciais omissões devem ser esclarecidas antes de qualquer concretização.

As várias situações de erro desta aplicação devem ser representadas por excepções distintas. As excepções usadas na interacção com o utilizador devem ser subclasses de `pt.utl.ist.po.ui.InvalidOperation` por forma a poderem ser tratadas pela *framework* de interacção com o utilizador **po-uilib**: A classe `pt.utl.ist.po.ui.Menu` já trata automaticamente as excepções que ocorram durante a execução dos comandos e portanto estas não devem ser tratadas pelos vários comandos a desenvolver. Por razões de simplificação do código a desenvolver, estas excepções podem ser lançadas directamente pelo núcleo da aplicação caso isso facilite o código a produzir. As excepções devem ser definidas no *package* `edt.textui.exception`.

2.1 Menu Principal

As acções do menu, listadas em `edt.textui.main.MenuEntry`, permitem gerir a salvaguarda do estado da aplicação (§2.1.1), assim como operar sobre o documento actual: **Listar meta-informação**, **Adicionar autor**, **Listar secções**, **Mostrar elemento de texto**, e **Editar** (§2.2). A classe `edt.textui.main.Message` define os métodos para geração das mensagens de diálogo. Inicialmente, a aplicação tem um documento vazio e anónimo.

2.1.1 Salvaguarda do Documento Actual

O conteúdo do documento 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:

Criar – Cria um novo documento anónimo não associado a nenhum ficheiro.

Abrir – Carrega um documento anteriormente salvaguardado, ficando o documento carregado associado ao ficheiro nomeado: pede-se o nome do ficheiro a abrir (`openFile()`). Caso o ficheiro não exista é apresentada a mensagem `fileNotFound()`.

Guardar – Guarda o documento actual no ficheiro associado. Se não existir associação, pede-se o nome do ficheiro a utilizar, ficando a ele associada. Esta interacção realiza-se através do método `newSaveAs()`. Não é executada nenhuma acção se não existirem alterações desde a última salvaguarda.

Apenas existe um documento no editor (explicitamente criado ou aberto). A opção "Sair" **nunca** guarda o estado da aplicação, mesmo que existam alterações.

Estes três comandos já estão parcialmente implementados nas classes do package **edt.textui.main** (disponível no ficheiro *template.jar*): **NewDocument**, **OpenDocument** e **SaveDocument**.

2.1.2 Listar meta-informação

É apresentada a seguinte informação: título do documento, autores do documento (listados por **ordem alfabética**), número de secções de topo do documento, dimensão do documento e número de identificadores únicos do documento. Cada linha é gerada por uma mensagem de `edt.textui.main.Message`.

`Título:␣`*título do documento*
`Autor:␣`*nome do autor 1 / email do autor 1*
...
`Autor:␣`*nome do autor N / email do autor N*
`Secções␣de␣topo:␣`*número de secções de topo*
`Dimensão␣do␣documento␣(bytes):␣`*dimensão do documento (soma da dimensão dos títulos e dos parágrafos)*
`Identificadores␣únicos:␣`*número de identificadores*

Este comando já está parcialmente implementado na classe do package **edt.textui.main** (disponível no ficheiro *template.jar*): **ShowMetadata**.

2.1.3 Adicionar autor

É pedido o nome (`requestAuthorName()`) e o email (`requestEmail()`) do novo autor. Se o nome já estiver associado ao documento, apresenta a mensagem `duplicateAuthor()` e não executa nenhuma acção.

Este comando já está parcialmente implementado na classe do package **edt.textui.main** (disponível no ficheiro *template.jar*): **AddAuthor**.

2.1.4 Listar secções de topo

É apresentada a lista de títulos de todas as secções de topo do documento (um título por linha): cada título entre chavetas é precedido pelo seu identificador único entre parênteses rectos (ou só os parânteses, caso o identificador não esteja definido). O título do documento deve ser apresentado antes da lista de secções, tal como os títulos das secções (entre chavetas). Se algum título for vazio, apresentam-se apenas as chavetas. Note-se que apesar de o documento ter uma estrutura semelhante à das secções, não possui identificador, pelo que o seu título não é precedido de nenhuma marca identificação (nem sequer parênteses rectos vazios). Exemplo:

```
{Título␣do␣documento}
[x123]␣{Isto␣é␣um␣título␣de␣uma␣secção}
[]␣{Isto␣é␣outro␣título␣de␣outra␣secção}
[w83]␣{}
[]␣{}
```

Este comando já está parcialmente implementado na classe do package **edt.textui.main** (disponível no ficheiro *template.jar*): **ListTopSections**.

2.1.5 Mostrar elemento de texto

Permite apresentar o conteúdo do elemento de texto indicado pelo utilizador (através da especificação do identificador único `requestElementId()`). Se o elemento de texto não existir, é apresentada a mensagem `noSuchTextElement()` e a acção termina. Caso contrário, é apresentado o conteúdo. Se o elemento de texto for um parágrafo, apresenta-se simplesmente o seu texto. Se for uma secção, apresenta-se tal como especificado em §2.2.3.

Este comando já está parcialmente implementado na classe do package **edt.textui.main** (disponível no ficheiro *template.jar*): **ShowTextElement**.

2.1.6 Editar

Abre o menu de edição (alteração) do documento e do seu conteúdo, estabelecendo como secção actual o próprio documento.

Este comando já está parcialmente implementado na classe do package **edt.textui.main** (disponível no ficheiro *template.jar*): **EditSection**.

2.2 Menu de Edição

Este menu permite efectuar operações sobre um documento. A lista completa é a seguinte: **Alterar título**, **Listar secções**, **Mostrar conteúdo**, **Ir para secção**, **Inserir secção**, **Nomear secção**, **Remover secção**, **Inserir parágrafo**, **Nomear parágrafo**, **Alterar parágrafo** e **Remover parágrafo**.

As etiquetas das opções deste menu estão definidas na classe `edt.textui.section.MenuEntry`. Todos os métodos correspondentes às mensagens de diálogo para este menu estão definidos na classe `edt.textui.section.Message`.

Em todos os casos, só podem ser indicados secções e parágrafos filhos da secção actual. O documento é estruturalmente semelhante a uma secção, sendo considerado a secção raiz (na primeira abertura deste menu).

2.2.1 Alterar título

Permite alterar o título da secção actual (ou do próprio documento). Para tal, pede-se o novo título (`requestSectionTitle()`).

Este comando já está parcialmente implementado na classe do package **edt.textui.section** (disponível no ficheiro *template.jar*): **ChangeTitle**.

2.2.2 Listar secções

É apresentada a lista de títulos de todas as subsecções da secção actual (recursivamente): cada título entre chavetas é precedido pelo seu identificador único entre parênteses rectos (ou só os parânteses, caso não haja um identificador único atribuído à subsecção). Se algum título for vazio, apresentam-se apenas as chavetas.

Este comando já está parcialmente implementado na classe do package **edt.textui.section** (disponível no ficheiro *template.jar*): **ListSections**.

2.2.3 Mostrar conteúdo

Apresenta todo o conteúdo da secção actual: título (utilizando o formato descrito em §2.1.4), seguido do texto de cada parágrafo da secção e do conteúdo de cada subsecção (apresentação recursiva). Os parágrafos e subsecções de uma secção são apresentados de acordo com a ordem local definida dentro da secção.

Este comando já está parcialmente implementado na classe do package **edt.textui.section** (disponível no ficheiro *template.jar*): **ShowSection**.

2.2.4 Ir para secção

Permite seleccionar a subsecção indicada pelo utilizador (através da especificação do identificador local `requestSectionId()`). Se a secção indicada não existir, continua na secção actual, sendo apresentada a mensagem `noSuchSection()`. Caso contrário, é apresentada a mensagem `newActiveSection()` e o menu de edição de secções é aberto para a secção seleccionada.

Este comando já está parcialmente implementado na classe do package **edt.textui.section** (disponível no ficheiro *template.jar*): **SelectSection**.

2.2.5 Inserir secção

Permite inserir uma subsecção antes de outra de referência (indicada pelo identificador local `requestSectionId()`). É pedido o título (`requestSectionTitle()`) da nova secção. Se a subsecção de referência não existir, insere a nova subsecção no final da secção actual.

Este comando já está parcialmente implementado na classe do package **edt.textui.section** (disponível no ficheiro *template.jar*): **InsertSection**.

2.2.6 Nomear secção

Permite atribuir um identificador único (no contexto do documento) a uma secção. É pedido o identificador local (inteiro) da subsecção a identificar (`requestSectionId()`) e o identificador único a atrubuir (`requestUniqueId()`) (cadeia de caracteres). Se a secção não existir é apresentada a mensagem `noSuchSection()` e a acção termina. Se o identificador já existir, passa a designar a nova secção (a ligação anterior é destruída). Se a secção já tiver um identificador, a mensagem de aviso `sectionNameChanged()` é apresentada e é atribuído o novo identificador único à secção.

Este comando já está parcialmente implementado na classe do package **edt.textui.section** (disponível no ficheiro *template.jar*): **IndexSection**.

2.2.7 Remover secção

Permite remover a secção indicada pelo utilizador (`requestSectionId()`). Se a secção indicada não existir, não realiza nenhuma acção.

Este comando já está parcialmente implementado na classe do package **edt.textui.section** (disponível no ficheiro *template.jar*): **RemoveSection**.

2.2.8 Inserir parágrafo

Permite inserir um novo parágrafo antes de outro parágrafo da secção actual. É pedido o número do parágrafo de referência (`requestParagraphId()`) e o texto do novo parágrafo (`requestParagraphContent()`). Se o parágrafo de referência não existir, insere o novo parágrafo no final da sequência de parágrafos da secção.

Este comando já está parcialmente implementado na classe do package **edt.textui.section** (disponível no ficheiro *template.jar*): **InsertParagraph**.

2.2.9 Nomear parágrafo

Permite atribuir um identificador único (no contexto do documento) a um parágrafo. É pedido o número do parágrafo a identificar (`requestParagraphId()`) e o identificador único a atribuir (`requestUniqueId()`) (cadeia de caracteres). Se o parágrafo não existir é apresentada a mensagem `noSuchParagraph()` e a acção termina. Se o identificador já existir, passa a designar o novo parágrafo (a ligação anterior é destruída). Se o parágrafo já tiver um identificador, a mensagem de aviso `paragraphNameChanged()` é apresentada e é atribuído o novo identificador único ao parágrafo.

Este comando já está parcialmente implementado na classe do package **edt.textui.section** (disponível no ficheiro *template.jar*): **IndexParagraph**.

2.2.10 Alterar parágrafo

Permite alterar o texto do parágrafo indicado. É pedido o número do parágrafo a alterar (`requestParagraphId()`) e o texto do novo parágrafo (`requestParagraphContent()`). Se o parágrafo indicado não existir, apresenta a mensagem `noSuchParagraph()` e não realiza nenhuma acção.

Este comando já está parcialmente implementado na classe do package **edt.textui.section** (disponível no ficheiro *template.jar*): **ChangeParagraph**.

2.2.11 Remover Parágrafo

Permite remover o parágrafo indicado pelo utilizador (`requestParagraphId()`). Se o parágrafo indicado não existir, apresenta a mensagem `noSuchParagraph()` e não realiza nenhuma acção.

Este comando já está parcialmente implementado na classe do package **edt.textui.section** (disponível no ficheiro *template.jar*): **RemoveParagraph**.

3 Leitura de um Documento a Partir de Ficheiro

Além das opções descritas em §2.1.1, é possível iniciar a aplicação com um ficheiro de texto especificado pela propriedade Java **import** (apresentada abaixo; este exemplo está no ficheiro `test.import`). Estes dados são apenas uma forma cómoda de inicialização e **nunca** são produzidos pela aplicação (nem mesmo para salvaguardar o estado para execuções futuras). Quando se especifica a propriedade, o sistema de ficheiros é povoado com as entidades do ficheiro indicado (uma por linha).

Assume-se que não há entradas mal-formadas (nestes ficheiros, as secções têm sempre identificador; os parágrafos não são identificados). Sugere-se a utilização do método `String.split`, para dividir uma cadeia de caracteres em campos.

```
Objects␣in␣Space
Obi-Wan␣Kenobi/ob1@tatooine.geocities.com|Master␣Yoda/yoda@dagobah.net|Luke␣Skywalker/luke@jedi.org
SECTION|lfd|A␣Walk␣in␣the␣Desert
PARAGRAPH|These␣aren't␣the␣droids␣you're␣looking␣for.
SECTION|wm|Patience␣You␣Must␣Have
PARAGRAPH|Do␣or␣do␣not.
PARAGRAPH|There␣is␣no␣try.
SECTION|dvls|Drama␣in␣the␣Clouds
PARAGRAPH|Obi-Wan␣never␣told␣you␣what␣happened␣to␣your␣father.
PARAGRAPH|He␣told␣me␣enough!␣He␣told␣me␣YOU␣killed␣him.
PARAGRAPH|NOOOOOOOOOOOOOOOOOO...
```

O editor nunca produz ficheiros neste formato, nem os pode ler a partir de nenhum menu.

4 Considerações sobre Flexibilidade e Eficiência

Devem ser possíveis extensões ou alterações de funcionalidade com impacto mínimo no código já produzido para o editor de texto. O objectivo é aumentar a flexibilidade da aplicação relativamente ao suporte de novas funções. Em particular, a solução encontrada para visualizar o conteúdo do documento deve ser suficientemente flexível por forma a que se possa visualizar o conteúdo de um documento noutro formato (por exemplo XML) sem que isso implique alterações no código "core" da aplicação.

5 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 (`edt.textui.Editor.main`). As propriedades são tratadas automaticamente pelo código de apoio.

```
java -Dimport=test.import -Din=test.in -Dout=test.outhyp edt.textui.Editor
```

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.