TÉCNICO LISBOA

Projecto de Programação com Objectos
8 de Outubro de 2013

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: **2013/10/25 12:00** (UML); **2013/11/15 12:00** (intercalar); **2013/12/03 12:00** (final); **2013/12/03–2013/12/13** (teste prático).
- Os diagramas UML são entregues exclusivamente em papel (impressos ou manuscritos) na portaria do Tagus ou em mão a um docente de PO. 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` (apenas os ficheiros `.java`). O ficheiro `proj.jar` deve ser entregue para avaliação através da ligação apresentada no Fénix. São possíveis múltiplas entregas até à data limite, mas será avaliada apenas a última versão.
- 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 Fenix no final do prazo têm classificação 0 (zero) (não são aceites outras formas de entrega).
- A avaliação do projeto pressupõe o compromisso de honra de que o trabalho correspondente foi realizado pelos alunos correspondentes ao grupo de avaliação. **Fraudes na execução do projeto terão como resultado a exclusão dos alunos implicados do processo de avaliação em 2013/2014**.

O objectivo do projecto é desenvolver uma folha de cálculo.

1 Estrutura e Funcionalidade da Folha de Cálculo

1.1 Endereçamento: Células, Intervalos, Gamas

O número de linhas e de colunas da folha é fixo. Os endereços (posições na folha: linha e coluna) têm início em 1 (um).

Uma célula é definida com base na sua posição na folha: *CÉLULA* `::=` *LINHA*`;`*COLUNA*
Exemplos: `1;2` (linha 1, coluna 2); ou `23;4` (linha 23, coluna 4).

Um intervalo (fechado) é definido entre duas células da mesma linha ou da mesma coluna: *INTERVALO* `::=` *CÉLULA*`:`*CÉLULA*
Exemplos: `1;2:1;20` (linha 1, entre as colunas 2 e 20); ou `23;4:57;4` (coluna 4, entre as linhas 23 e 57).

Utiliza-se o termo "gama" para especificar indiscriminadamente uma célula única ou um intervalo de células.

1.2 Conteúdo: Literais, Referências, Funções

Por omissão, as células estão vazias (sem conteúdo). Os conteúdos admissíveis são: literais (números inteiros), referências, funções. As referências indicam-se com o símbolo "=" seguido do endereço da célula referida (§1.1). As funções indicam-se com o símbolo "=", o nome da função e a lista (possivelmente vazia) de argumentos (separados por vírgulas). Estão pré-definidas:

1. Funções binárias, cujos argumentos podem ser referências a células ou valores literais: ADD (adição), SUB (subtracção), MUL (multiplicação), DIV (divisão inteira). Exemplos: `ADD(2;3,1)`, `SUB(6;2,22;1)`, `MUL(1,2)`, `DIV(1,5;2)`.
2. Funções aplicadas a um intervalo de células: AVG (média com divisão inteira), PRD (piatório). Exemplos: `AVG(1;2:1;19)`, `PRD(2;33:5;33)`.

Considera-se que não existem dependências circulares (directas ou indirectas), entre funções e células referidas pelos seus argumentos. O valor a apresentar para referências inválidas ou funções com argumentos inválidos (referem células vazias), é `#VALUE`.

1.3 Operações Sobre Células

É possível inserir, apagar e mostrar conteúdos (§1.2). É ainda possível copiar conteúdos entre células.

1.4 Pesquisas

É possível pesquisar o conteúdo das células sob diferentes aspectos: (i) valores resultantes de avaliação; (ii) nomes de funções.

2 Considerações sobre Flexibilidade e Eficiência

Devem ser possíveis extensões ou alterações de funcionalidade com impacto mínimo no código já produzido: em particular, deve ser simples definir novas funções e novas pesquisas sobre células. O objectivo é aumentar a flexibilidade da aplicação relativamente ao suporte de novas funções.

Deve ser possível representar a estrutura de uma folha de cálculo utilizando diferentes abordagens sem que isso tenha impacto no restante código da aplicação. O objectivo deste requisito será permitir optimizar o espaço ocupado em memória para guardar o conteúdo de uma folha de cálculo. Por exemplo, no caso de folhas de cálculo de dimensões pequenas pode-se ter uma estrutura que guarda todas as células independentemente de estarem vazias ou não. Esta representação é ineficiente para situações com folhas de cálculo de grandes dimensões e em que haja muitas células vazias. Neste caso será melhor utilizar uma abordagem do tipo matiz esparsa, em que apenas se guardam as células que estão preenchidas. A solução a desenvolver não necessita de concretizar todas as situações (apenas necessita de concretizar o primeiro caso) mas deve ser suficientemente flexível para suportar novas representações.

Uma função depende de células cujo conteúdo pode ser alterado em qualquer instante. Por forma a reduzir o custo de execução do programa sempre que uma célula é alterada, o número de vezes que cada fórmula é calculada deve ser minimizado. Este requisito deve ser considerado apenas depois de tudo estar implementado.

3 Interação com o Utilizador

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 de dados relacionadas com a interação com o utilizador **têm** de ser realizadas através dos objetos `pt.utl.ist.po.ui.Form` e `pt.utl.ist.po.ui.Display`, utilizando-se as mensagens descritas (todas as mensagens são produzidas através de chamadas a métodos). Uma lista completa das classes disponibilizadas pode ser obtida nas bibliotecas **po-uilib** e **calc** (material de apoio). A interacção com o utilizador, via operações de escrita ou leitura, apenas deve ser realizadas no código relacionado com a interface com o utilizador (e nunca no núcleo da aplicação). Não devem ser definidas novas mensagens textuais. Casos potencialmente omissos devem ser esclarecidos com o corpo docente antes de qualquer concretização.

As excepções usadas na interacção, excepto se indicado, são subclasses de `pt.utl.ist.po.ui.InvalidOperation`, são lançadas pelos comandos e tratadas por `pt.utl.ist.po.ui.Menu`. Note-se que, além das exceções descritas, é possível a definição de outras. As novas exceções não devem, no entanto, substituir as fornecidas nos casos descritos por este enunciado.

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

3.1 Endereçamento de Células

Para o pedido de uma gama, utiliza-se a mensagem `calc.textui.edit.Message.addressRequest()`. A excepção `calc.textui.edit.InvalidCellRange` é lançada se forem especificados endereços inválidos.

3.2 Menu Principal

As acções do menu, listadas em `calc.textui.main.MenuEntry`, permitem manipular ficheiros (§3.2.1), editar células (§3.3) e efectuar consultas (§3.4). Os métodos para as mensagens de diálogo estão definidos em `calc.textui.main.Message`.

Inicialmente, a aplicação não tem nenhuma folha. Nesta situação, apenas são apresentadas as opções **Criar** e **Abrir**, pois as restantes necessitam uma folha. As opções irrelevantes nesta situação devem ser omitidas.

3.2.1 Manipulação de Ficheiros

O estado da aplicação pode ser guardado em ficheiros, para posterior recuperação (serialização Java: `java.io.Serializable`). Na manipulação de ficheiros, devem ser tratadas as excepções associadas. A funcionalidade de cada operação é a seguinte:

Criar – Permite criar de uma nova folha vazia: pedem-se as dimensões da nova folha, devendo ser utilizadas as mensagens `linesRequest()` e `columnsRequest()`. Esta folha não fica associada a nenhum ficheiro (é anónima).

Abrir – Carrega uma nova folha a partir de um ficheiro previamente guardado, ficando associada a esse ficheiro. Pede-se o nome do ficheiro a abrir (`openFile()`): caso não exista, é apresentada a mensagem `fileNotFound()`.

Guardar – Guarda a folha no ficheiro associado. Se a folha for anónima, 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.

Guardar como... – Esta opção permite associar um (novo) ficheiro à folha, guardando nele o estado da folha. Esta interacção utiliza o método `saveAs()`. É ignorada a associação actual a um ficheiro, mesmo que existam alterações.

Apenas existe uma folha na aplicação. Quando se abandona uma folha com modificações não guardadas (porque se cria ou abre outra), deve perguntar-se se se quer guardar a informação actual antes de a abandonar, através da mensagem `saveBeforeExit()` (a resposta é obtida invocando `readBoolean()`). A opção "Sair" **nunca** guarda a folha, mesmo que existam alterações.

3.2.2 Outras Opções

Além das operações de manipulação de ficheiros e das opções de visualização básicas, existem ainda no menu principal as opções **Menu de Edição** e **Menu de Consultas**. Estas opções só estão disponíveis quando existir uma folha activa válida.

3.3 Menu de Edição

O menu de edição permite visualizar e alterar o conteúdo das células da folha activa. A lista completa é a seguinte: **Visualizar**, **Inserir**, **Apagar**, **Copiar**, **Cortar**, **Colar** e **Visualizar *cut buffer***. As secções abaixo descrevem estas opções.

As opções deste menu estão definidas na classe `calc.textui.edit.MenuEntry`. Todos os métodos correspondentes às mensagens de diálogo para as acções do menu estão definidos na classe `calc.textui.edit.Message`.

3.3.1 Visualizar

Permite visualizar a gama especificada (§3.1), conforme as indicações das tabelas. A primeira tabela descreve a apresentação de uma célula para as diferentes situações. A primeira linha é para células vazias, a segunda para células com um literal e a terceira para células que referem fórmulas ou referências. A segunda tabela descreve a apresentação dos dois tipos de intervalos.

Permite visualizar a gama especificada (§3.1), conforme as indicações das tabelas. Se o conteúdo não for um literal, deve ser apresentado o seu valor e a sua expressão (referências ou fórmulas). Não deve ser apresentadado qualquer conteúdo para células vazias (primeiro exemplo da tabela abaixo).

Célula: Conteúdos
linha`;`*coluna*`|`
linha`;`*coluna*`|`*literal*
linha`;`*coluna*`|`*valor=expressão*

Intervalo Vertical	**Intervalo Horizontal**
linha1`;`*coluna*`|`*conteúdo*	*linha*`;`*coluna1*`|`*conteúdo*
linha2`;`*coluna*`|`*conteúdo*	*linha*`;`*coluna2*`|`*conteúdo*
linha3`;`*coluna*`|`*conteúdo*	*linha*`;`*coluna3*`|`*conteúdo*

3.3.2 Inserir

É especificada a gama (§3.1). É especificado o conteúdo a inserir (§1.2), através da mensagem `contentsRequest()`. O conteúdo é inserido em todas as células da gama. Por simplicidade, não são inseridos nomes de funções inexistentes.

3.3.3 Copiar

É especificada a gama (§3.1) cujo conteúdo deve ser copiado para o *cut buffer*. Os conteúdos copiados são independentes da folha (i.e., alterações aos originais não são propagadas para o *cut buffer*).

3.3.4 Apagar

É especificada a gama (§3.1) cujo conteúdo deve ser apagado.

3.3.5 Cortar

Esta acção corresponde à execução sequencial das acções descritas em §3.3.3 e §3.3.4.

3.3.6 Colar

Insere o conteúdo do *cut buffer* na gama especificada (§3.1). Se o *cut buffer* estiver vazio, não é realizada qualquer operação.

Se a gama for uma única célula, todo o *cut buffer* deve ser inserido a partir da célula especificada, até ser atingido o limite da folha de cálculo. Caso contrário, se a dimensão do *cut buffer* for diferente da da gama de destino, não é inserido nenhum valor.

3.3.7 Visualizar *cut buffer*

Permite visualizar o conteúdo do *cut buffer*. O formato de apresentação é o descrito em §3.3.1 (a primeira célula do *cut buffer* começa sempre em 1;1).

3.4 Menu de Consultas

Permite efectuar pesquisas sobre as células da folha activa, produzindo listas de células. As células resultantes são apresentadas pela ordem em que aparecem na tabela (§3.3.1). As entradas do menu estão definidas em `calc.textui.search.MenuEntry`. As mensagens estão definidas em `calc.textui.search.Message`.

Entrada do menu	**Acção**
Procurar Valor	Pede-se o valor a procurar (mensagem `searchValue()`) e apresentam-se os resultados.
Procurar Função	Pede-se o nome da função a procurar (mensagem `searchFunction()`) e apresentam-se os resultados.

O termo de pesquisa deve ser comparado com o valor avaliado de cada célula. Assim, considerando as expressões iniciais das células indicadas de seguida, uma pesquisa pelo valor `5` encontraria as células `1;3` e `2;1`.

```
1;1|4
1;2|1
1;3|5
2;1|5=ADD(1;1,1;2)
2;2|
2;3|
```

Sempre que for feita uma consulta e nenhuma entidade satisfizer as condições associadas ao pedido, nada deve ser apresentado.

4 Inicialização por Ficheiro de Dados Textuais

Por omissão, quando a aplicação começa, não existe nenhuma folha activa. Além das opções descritas em §3.2.1, é possível iniciar a aplicação utilizado um ficheiro de texto especificado pela propriedade Java **import** (apresentada abaixo; este exemplo está presente no ficheiro `test.import`). Os dados textuais 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, é criada uma folha activa anónima que representa o conteúdo do ficheiro indicado.

No processamento destes dados, assume-se que não há entradas mal formadas (células não referidas são vazias). Sugere-se a utilização do método `String.split`, para dividir uma cadeia de caracteres em campos.

As duas primeiras linhas definem o número de linhas e de colunas da folha de cálculo. As restantes linhas contêm sempre o formato `linha;coluna|conteúdo` (o campo `conteúdo` está descrito em §1.2).

No seguinte exemplo, o conteúdo do ficheiro inicial (`test.import`), à esquerda, correspondente à folha apresentada à direita.

```
linhas=4
colunas=3
3;3|=ADD(3;1,3;2)
4;1|
1;1|5
1;2|49
2;1|25
2;2|43
2;3|=ADD(2;2,5)
3;1|10
3;2|=1;1
1;3|=ADD(2,5)
4;3|=AVG(1;3:3;3)
4;2|
```

	1	2	3
1	5	49	=ADD(2,5)
2	25	43	=ADD(2;2,5)
3	10	=1;1	=ADD(3;1,3;2)
4			=AVG(1;3:3;3)

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 (método `calc.textui.Calc.main`). A propriedade `import` é acedida através de `System.getProperty("import");` (ver manual da linguagem para mais informação). As outras propriedades são tratadas automaticamente pelo código de apoio.

```
java -Dimport=test.import -Din=test.in -Dout=test.outhyp calc.textui.Calc
```

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.