

Programação com Objectos/Projecto de Programação com Objectos/Enunciado do Projecto de 2023-2024

< Programação com Objectos | Projecto de Programação com Objectos

AVISOS - Avaliação em Época Normal

[Expand]

Material de Uso Obrigatório

[Expand]

Contents (hide)

- 1 Propriedades e funcionalidade da folha de cálculo
 - 1.1 Endereçamento: Células, Intervalos, Gamas
 - 1.2 Conteúdo: Literais, Referências, Funções
- 2 Funcionalidade da aplicação
 - 2.1 Operações Sobre Células
 - 2.2 Pesquisas
 - 2.3 Cut Buffer
 - 2.3.1 Copiar
 - 2.3.2 Cortar
 - 2.3.3 Colar
 - 2.4 Serialização
 - 2.5 Utilizadores
 - 3 Requisitos de Desenho
 - 4 Interação com o Utilizador
 - 4.1 Endereçamento de Células
 - 4.2 Menu Principal
 - 4.2.1 Salvaguarda do estado actual da aplicação
 - 4.2.2 Gestão e consulta de dados da aplicação
 - 4.3 Menu de Edição
 - 4.3.1 Visualizar
 - 4.3.2 Inserir
 - 4.3.3 Copiar
 - 4.3.4 Apagar
 - 4.3.5 Cortar
 - 4.3.6 Colar
 - 4.3.7 Visualizar cut buffer
 - 4.4 Menu de Consultas
 - 4.4.1 Procurar Valor
 - 4.4.2 Procurar Função
 - 5 Inicialização por Ficheiro de Dados Textuais
 - 6 Execução dos Programas e Testes Automáticos

ÉPOCA NORMAL

O objectivo do projecto é desenvolver uma folha de cálculo.

Propriedades e funcionalidade da folha de cálculo

A folha de cálculo é capaz de manipular números inteiros, cadeias de caracteres, referências entre células e funções sobre células. Além disso, permite também operações de corte, cópia e inserção (cut/copy/paste).

Endereçamento: Células, Intervalos, Gamas

O número de linhas e de colunas da folha é fixo, sendo definidos na altura da criação. 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**

Um intervalo de endereçamento (são sempre fechados) é definido entre duas células da mesma linha ou da mesma coluna: **INTERVALO** ::= **CÉLULA**:**CÉLULA** (as duas células pode ser a mesma).

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

Exemplo:

- 1;2 (linha 1, coluna 2); ou 23;4 (linha 23, coluna 4)
- 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)

Conteúdo: Literais, Referências, Funções

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

- Funções binárias sobre valores inteiros, cujos argumentos podem ser referências a células ou valores literais: **ADD** (adição), **SUB** (subtração), **MUL** (multiplicação), **DIV** (divisão inteira). Todos os argumentos têm de ser passíveis de produzir valores inteiros.
- Funções aplicáveis a um intervalo de células:
 - AVERAGE** (média de todos os valores do intervalo; a divisão é inteira), **PRODUCT** (produto de todos os valores do intervalo). Todos os valores do intervalo têm de ser inteiros (não podem conter outros valores).
 - CONCAT** (aceita um intervalo de células que podem ou não conter cadeias de caracteres e retorna a concatenação das cadeias de caracteres, ignorando todos os outros valores). Se o intervalo tiver apenas uma célula, produz o valor de entrada. Se não existir no intervalo nenhuma cadeia de caracteres, produz a cadeia vazia.
 - COALESCE** (aceita um intervalo de células que podem ou não conter cadeias de caracteres e retorna o primeiro valor que for uma cadeia de caracteres). Se o intervalo não contiver nenhuma cadeia de caracteres, produz a cadeia vazia.

Considera-se que não existem dependências circulares, nem directas, nem indirectas, entre uma função e as células referidas pelos seus argumentos. Funções com argumentos inválidos (e.g., referem gamas com células vazias), têm valor inválido (apresentado como **#VALUE**), excepto nos casos especiais indicados.

Exemplos:


- Inteiros (números com um sinal opcional no início) -**25, 48**
- Cadeias de caracteres (começam sempre com plica): **'string**
- Cadeias de caracteres (vazia): **'** (apenas uma plica)
- Referências: =1;2
- Funções: =**ADD**(2;3;1), =**SUB**(6;2;2;1), =**MUL**(1;2), =**DIV**(1;5;2), =**AVERAGE**(1;2;1;19), =**PRODUCT**(2;33;5;33), =**CONCAT**(1;1;2;1;17), =**COALESCE**(1;1;2;1;17)


Funcionalidade da aplicação

A aplicação permite manter informação sobre as entidades do modelo. Possui ainda a capacidade de preservar o seu estado (não é possível manter várias versões do estado da aplicação em simultâneo).

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

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

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

 **A interface geral do core já está parcialmente implementada na classe `xxl.Calculator` 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.**

Operações Sobre Células

É possível inserir, apagar e mostrar conteúdos. É ainda possível copiar conteúdos entre células.

Pesquisas

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

Cut Buffer

Existe um cut buffer por folha da aplicação e que permite as operações habituais (copiar, cortar, colar) entre células de uma dada folha.

As operações têm a semântica descrita a seguir.

Copiar

O conteúdo da origem é copiado para o cut buffer e torna-se independente da fonte, i.e., alterações aos objectos originais não são propagadas para os que estão no cut buffer. O conteúdo anterior do cut buffer é destruído.

Cortar

Esta operação funciona como a operação de cópia, mas o conteúdo original é destruído.

Colar

Esta operação insere o conteúdo do cut buffer numa gama da folha. 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 insere nenhum valor.

O conteúdo do cut buffer não é alterado pela operação. Os objectos inseridos no destino são independentes dos que estão no cut buffer.

Serialização

É possível guardar e recuperar o estado actual da aplicação, preservando toda a informação relevante, descrita acima.

Utilizadores

A aplicação deve suportar o conceito de utilizador. Cada utilizador tem um identificador único na aplicação (o seu nome) e detém um conjunto de folhas de cálculo. Devido a necessidades futuras ainda não completamente especificadas, deve existir uma relação bidireccional de muitos para muitos entre utilizadores e folhas de cálculo.

A aplicação tem sempre um utilizador activo. Cada vez que se cria uma folha de cálculo na aplicação, ela deve ser associada ao utilizador activo. O utilizador com o nome **root** deve existir sempre. Por omissão, o utilizador activo é o utilizador com o nome **root**.

Requisitos de Desenho

Devem ser possíveis extensões ou alterações de funcionalidade com impacto mínimo no código 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.

A estrutura de armazenamento dos conteúdos da folha deve ser flexível, por forma a permitir a representação eficiente de folhas de diferentes dimensões. Assim, deve ser possível usar diferentes estratégias de representação do conteúdo, sem impacto no resto da aplicação. O objectivo deste requisito é otimizar o espaço ocupado em memória para guardar o conteúdo de uma folha de cálculo. Não é necessário concretizar para todas as situações, bastando ser suficientemente flexível para permitir novas implementações.

A determinação do valor de uma função que envolve uma gama pode ter um custo relativamente alto, caso envolva um número grande células. Caso nenhuma das células envolvidas tenha sido alterada desde a última vez que o valor da função foi determinado, então calcular o valor de novo é um desperdício. Pretende-se obter uma solução que optimize o número de vezes que uma função que envolva uma gama necessita de calcular o seu valor. Este requisito deve ser considerado apenas depois de tudo estar implementado e apenas vai ser tido em conta na entrega final do projecto.

A solução encontrada para otimizar o número de vezes que uma função tem que ser calculada deve ser extensível por forma a permitir que seja possível especificar outros tipos de funcionalidade que devem ocorrer quando o valor de uma célula muda de valor. Por exemplo, deveria ser possível concretizar o seguinte requisito sem que isso implicasse modificar as classes existentes (não é necessário implementar este exemplo): enviar um email a um dado utilizador cada vez que a célula em causa muda de valor. A sua solução deve ser flexível por forma a poder suportar diferentes tipos de entidades interessadas em modificações de uma célula sem que isso implique alterar o código já realizado.

A aplicação já deve estar preparada para suportar vários utilizadores, devendo já ter o código (ao nível da camada de domínio apenas) para a inserção de novos utilizadores.


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 procederem à sua validação (excepto onde indicado). Todos os menus têm automaticamente a opção **Sair** (fecha o menu).

As operações de pedido e apresentação de informação ao utilizador **devem** realizar-se através dos objectos *form* e *display*, respectivamente, presentes em cada comando. As mensagens são produzidas pelos métodos das bibliotecas de suporte (**po-uilib** e **xxl-app**). As mensagens não podem ser usadas no núcleo da aplicação (**xxl-core**). Além disso, não podem ser definidas novas. Potenciais omissões devem ser esclarecidas antes de qualquer implementação.

De um modo geral, sempre que no contexto de uma operação com o utilizador aconteça alguma excepção, então a operação não deve ter qualquer efeito no estado da aplicação, excepto se indicado em contrário na operação em causa.

As excepções usadas na interação (subclasses de **pt.tecnico.uilib.menus.CommandException**), excepto se indicado, são lançadas pelos comandos (subclasses de **pt.tecnico.uilib.menus.Command**) e tratadas pelos menus (instâncias de subclasses de **pt.tecnico.uilib.menus.Menu**). Outras excepções não devem substituir as fornecidas nos casos descritos.

 **Note-se que o programa principal e os comandos e menus, a seguir descritos, já estão parcialmente implementados nas packages `xxl.app`, `xxl.app.main`, `xxl.app.edit`, `xxl.app.search`. Estas classes são de uso obrigatório e estão disponíveis no GIT (módulo **xxl-app**).**


Endereçamento de Células

Para o pedido de uma gama, utiliza-se a mensagem **PromptAddress()**. A excepção **xxl.app.edit.InvalidCellRangeException** é lançada se forem especificados endereços inválidos (excedem o limite da folha de cálculo ou representam gamas de células que não são intervalos verticais ou horizontais).

Menu Principal

As acções deste menu permitem gerir a salvaguarda do estado da aplicação, abrir submenus e aceder a alguma informação global. A lista completa é a seguinte: Criar, Abrir, Guardar, Menu de Edição, Menu de Consultas.

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

 **Estes comandos já estão implementados nas classes da package `xxl.app.main` (disponível no GIT), respectivamente: `DoNew`, `DoOpen`, `DoSave`, `DoMenuOpenEdit`, `DoMenuOpenSearch`.**

Salvaguarda do estado actual da aplicação

Inicialmente, a aplicação não tem nenhuma folha, excepto quando usada a propriedade **import** (ver abaixo) para pré-carregar dados iniciais. Na situação em que a aplicação começa sem nenhuma folha, apenas são apresentadas as opções Criar e Abrir, pois as restantes necessitam da existência de uma folha. As opções irrelevantes nesta situação devem ser omitidas. O conteúdo da aplicação (toda a informação actualmente em memória) pode ser guardado para posterior recuperação (via serialização Java: **java.io.Serializable**). Na leitura e escrita do estado da aplicação, devem ser tratadas as excepções associadas. A funcionalidade é a seguinte:

- Criar** – Cria uma nova folha vazia: pedem-se as dimensões da nova folha, devendo ser utilizadas as mensagens **Prompt.Lines()** e **Prompt.columns()**. Esta folha não fica associada a nenhum ficheiro (é anónima).
- Abrir** – Carrega os dados de uma sessão anterior a partir de um ficheiro previamente guardado (ficando este ficheiro associado à aplicação, para futuras operações de salvaguarda). Pede-se o nome do ficheiro a abrir (**Prompt.OpenFile()**). Caso ocorra um problema na abertura ou processamento do ficheiro, deve ser lançada a excepção **FileOpenFailedException**. A execução bem-sucedida desta opção substitui toda a informação da aplicação.
- Guardar** – Guarda o estado actual da aplicação no ficheiro associado. Se não existir associação, pede-se o nome do ficheiro a utilizar, ficando a ele associado (para operações de salvaguarda subsequentes). Esta interação realiza-se através do método **Prompt.newSaveAs()**. Não é executada nenhuma acção se não existirem alterações desde a última salvaguarda.

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 de **Prompt.SaveBeforeExit()** (a resposta é obtida invocando **readBoolean()** ou de **Form.confirm()**).

Note-se que a opção **Abrir** não permite a leitura de ficheiros de texto (estes apenas podem ser utilizados no início da aplicação).

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

Gestão e consulta de dados da aplicação

Além das operações de manipulação de ficheiros, existem ainda no menu principal as seguintes opções.

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

Estas opções só estão disponíveis quando existir uma folha activa válida.

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, Copiar, Apagar, Cortar, Colar e Visualizar cut buffer. As secções abaixo descrevem estas opções.

As etiquetas das opções deste menu estão definidas na classe **xxl.app.edit.Label**. Todos os métodos correspondentes às mensagens de diálogo para este menu estão definidos em **xxl.app.edit.Prompt** e **xxl.app.edit.Message**.

 **Estes comandos já estão implementados nas classes da package `xxl.app.edit` (disponível no GIT), respectivamente: `DoShow`, `Dolnsert`, `DoCopy`, `DoDelete`, `DoCut`, `DoPaste`, `DoShowCutBuffer`.**

Visualizar

Permite visualizar a gama especificada (Endereçamento), de acordo com os formatos abaixo. Se o conteúdo não for um literal, deve ser apresentado o seu valor e a sua expressão. Não deve ser apresentado qualquer conteúdo para células vazias.

Formatos (valor representa o valor final da célula, inteiro ou cadeia de caracteres, depois de efectuadas todas as avaliações):

- Célula vazia:

1;1ha:coluna()
- Célula com um valor (literal):

1;1ha:coluna|valor
- Célula com um valor (calculado a partir de expressão: referência ou função):

1;1ha:coluna|valor=expressão
- Intervalo vertical (*conteúdo* é um dos três casos iniciais):

1;1ha:1:coluna|conteúdo
1;1ha:2:coluna|conteúdo
1;1ha:3:coluna|conteúdo
- Intervalo horizontal (*conteúdo* é um dos três casos iniciais):

1;1ha:coluna1|conteúdo
1;1ha:coluna2|conteúdo
1;1ha:coluna3|conteúdo

Inserir

É pedida a gama (Endereçamento). É especificado o conteúdo a inserir (Conteúdo), através da mensagem **Prompt.contents()**. O conteúdo é inserido em todas as células da gama. Se for inserido um nome de função inexistente, é lançada a excepção **xxl.app.edit.UnknownFunctionException**.

Copiar

É pedida a gama (Endereçamento) cujo conteúdo deve ser copiado para o cut buffer.

Apagar

É pedida a gama (Endereçamento) cujo conteúdo deve ser apagado.

Cortar

É pedida a gama (Endereçamento). Esta acção corresponde à execução sequencial das acções descritas em Copiar e Apagar.

Colar

É pedida a gama (Endereçamento). Insere o conteúdo do cut buffer na gama especificada.

Visualizar cut buffer

Permite visualizar o conteúdo do cut buffer. O formato de apresentação é o descrito em Visualizar.

Menu de Consultas

O menu de consultas permite efectuar pesquisas sobre as células da folha activa, produzindo listas de células. Sempre que for feita uma consulta e nenhuma entidade satisfizer as condições associadas ao pedido, nada deve ser apresentado. A lista completa das consultas é a seguinte: Procurar Valor, Procurar Função. As secções abaixo descrevem estas opções.

As etiquetas das opções deste menu estão definidas na classe **xxl.app.search.Label**. Todos os métodos correspondentes às mensagens de diálogo para este menu estão definidos em **xxl.app.search.Prompt** e **xxl.app.search.Message**.

 Estes comandos já estão implementados nas classes da package `xxl.app.search` (disponível no GIT), respectivamente: `DoShowValues`, `DoShowFunctions`.

Procurar Valor

Pede-se o valor a procurar (**Prompt.searchValue()**) e apresentam-se os resultados.

O termo de pesquisa deve ser comparado com o valor avaliado de cada célula. As células que obedecem ao critério de procura devem ser apresentadas por ordem ascendente de linha e coluna e utilizando o formato especificado em Visualizar. Assim, considere-se o exemplo:

```
1;1|14
1;2|1
1;3|5
2;1|5=ADD(1;1,1;2)
2;2|
2;3|
```

Neste exemplo, uma pesquisa pelo valor 5 encontraria as células 1;3 e 2;1.

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

Procurar Função

Pede-se o segmento a procurar num nome de função (**Prompt.searchFunction()**) e apresentam-se os resultados ordenados (são apresentados como especificado em Visualizar), de forma ascendente, por nome de função, linha e coluna (por esta ordem).

Inicialização por Ficheiro de Dados Textuais

Por omissão, quando a aplicação começa, não existe nenhuma folha activa. No entanto, além das opções de manipulação de ficheiros descritas no menu principal, é possível iniciar externamente a aplicação com um ficheiro de texto especificado pela propriedade Java **import**. Quando se especifica esta propriedade, é criada uma folha activa anónima que representa o conteúdo do ficheiro indicado.

No processamento destes dados, assume-se que não existem entradas mal-formadas. Assume-se que células não explicitamente referidas estão vazias (que também podem ser explicitamente definidas). 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 Conteúdo).

No seguinte exemplo, o conteúdo do ficheiro inicial (**test.import**) correspondente à folha apresentada a seguir.

Exemplo de ficheiro de entrada textual				[Expand]
1	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			=AVERAGE(1,3,3,3)	

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

 Note-se que o programa nunca produz ficheiros com este formato.

Execução dos Programas e Testes Automáticos

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

```
java -Dimport=test.import -Din=test.in -Dout=test.out.thypp xxl.app.App
```

Assumindo que aqueles ficheiros estão no directório onde é dado o comando de execução, o programa produz o ficheiro de saída **test.out.thypp**. Em caso de sucesso, os ficheiros das saídas esperada (**test.out**) e obtida (**test.out.thypp**) devem ser iguais. A comparação pode ser feita com o comando:

```
diff -b test.out test.out.thypp
```

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 desenvolvendo, apenas verificando alguns aspectos da sua funcionalidade.

Retrieved from "https://web.tecnico.ulisboa.pt/~david.matos/wp/t/index.php?title=Programação_com_Objectos/Projecto_de_Programação_com_Objectos/Enunciado_do_Projecto_de_2023-2024&oldid=16084"

Categories: Ensino PO Projecto de PO