Programação Centrada em Objectos (PCO)
Programação Por Objectos (PPO)

Projecto de Programação III (até 22 de Dezembro de 2015)

1 Introdução

O projecto envolve uma aplicação chamada YCel, para folhas de cálculo rudimentares. Folhas de cálculo YCel permitem a introdução de texto e valores, bem como o uso de tradicionais operadores sobre células (soma, média, etc), e ainda a formatação de algumas propriedades do estilo das células como o tipo ou a cor da letra.

São disponibilizados 2 projectos Eclipse: um para desenvolvimento e outro com uma demonstração da aplicação. Em ambos os projectos pode encontrar exemplos de folhas de cálculo para teste (pasta `resources/examples`). A aplicação é ilustrada na Figura 1 e poderá executá-la a partir do projecto de desenvolvimento (`ycel.Program`) ou de demonstração (`ycel.Demo`).

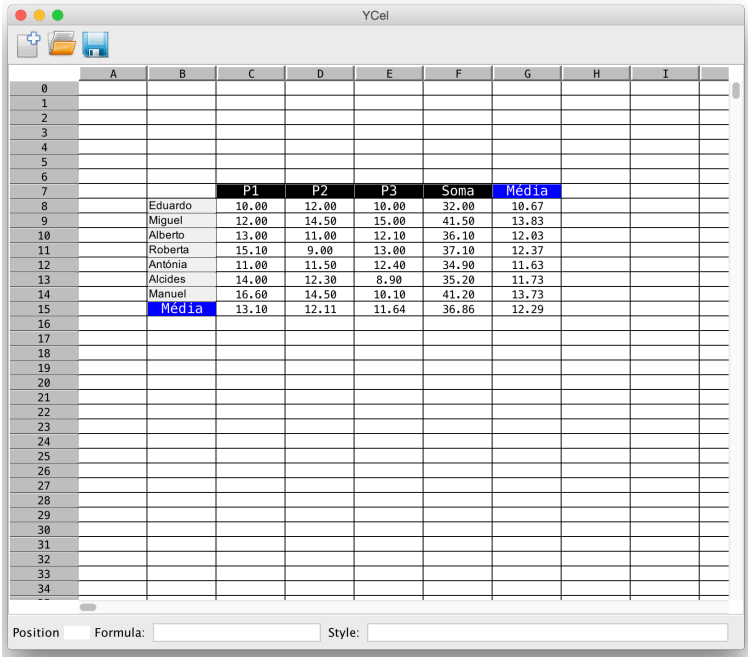

Figura 1: A aplicação YCel

1.1 Sumário de tarefas

Para a realização do projecto terá de implementar o seguinte conjunto de classes:

1. Classe `Document` para a representação de uma folha de cálculo;
2. Classe `CellPosition` para a posição de uma célula;
3. Classes para os vários tipos de conteúdos de células, implementando os interfaces `CellContent` ou `NumberContent`, com excepção de algumas já dadas;
4. Classe `DocumentIO` para leitura / gravação de uma folha de cálculo de / para um ficheiro.

A perspectiva geral e detalhes sobre o formato de ficheiro YCel são dados na secção 2. Os detalhes sobre a implementação das classes são expressos pela **documentação Javadoc já fornecida para todas as classes, que deve ser considerada parte do enunciado**.

1.2 Critérios de avaliação

1. O seu código deverá funcionar correctamente e usar classes/objectos de forma adequada.
2. Neste projecto não precisa de escrever comentários Javadoc.
3. O seu código deverá ser organizado e fácil de ler código com um estilo de programação "limpo" e coerente, ex. em termos de indentação, uso adequado de nomes para variáveis, campos, etc. Veja informação no guião *Uma questão de estilo – Elementos de estilo Java*, V. Vasconcelos, DI/FCUL.
4. O código de todos os grupos será analisado por uma ferramenta de detecção automática de similaridade entre trabalhos. Em caso de cópia, os trabalhos dos grupos envolvidos serão anulados.

2 Descrição do trabalho

2.1 Organização geral

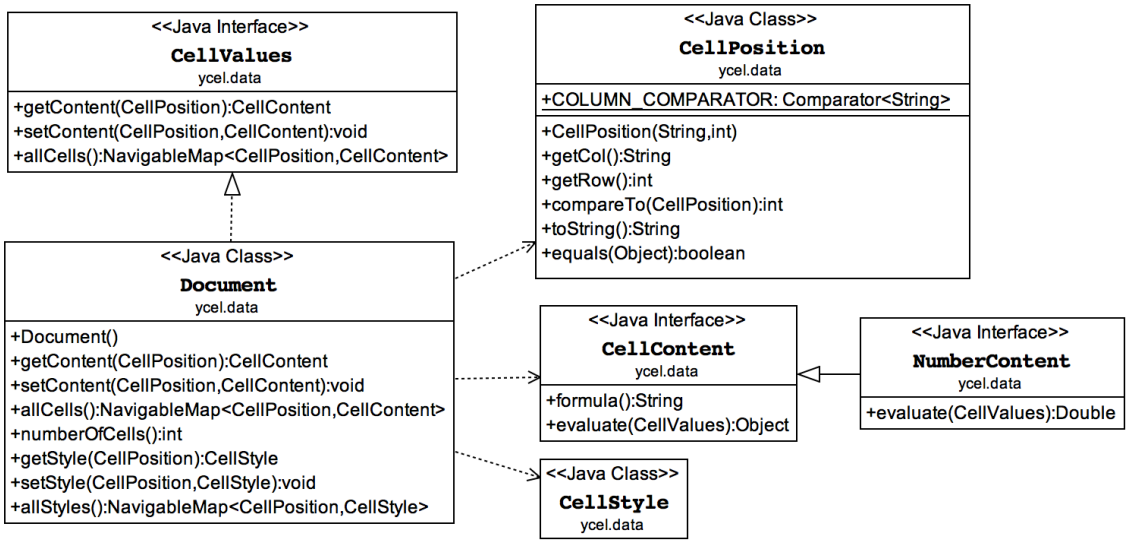

Figura 2: Classes e interfaces para representação de uma folha de cálculo

Como ilustrado na Figura 2, a classe `Document` implementa o interface `CellValues`, que define operações abstractas respeitantes a uma associação entre posições e conteúdos de células, Uma posição na folha de cálculo, `CellPosition`, é definida por uma coluna de tipo `String` e uma linha de tipo `int`. Por sua vez, o interface `CellContent` é o tipo de dados para conteúdo de uma célula, através dos seguintes métodos abstractos:

- `String formula()`, com o propósito de devolver o a fórmula associado a uma célula;
- `Object evaluate(CellValues cells)`: com o propósito de avaliar a célula e retornar um resultado;

O interface `NumberContent` estende `CellContent` apenas com o propósito de redefinir `evaluate()` com tipo de retorno `Double`, isto é, conteúdos que implementam `NumberContent` devolvem um número (`Double`) ao serem avaliadas. Finalmente, a classe `CellStyle` representa estilos para a formatação das células.

2.2 Conteúdos de células

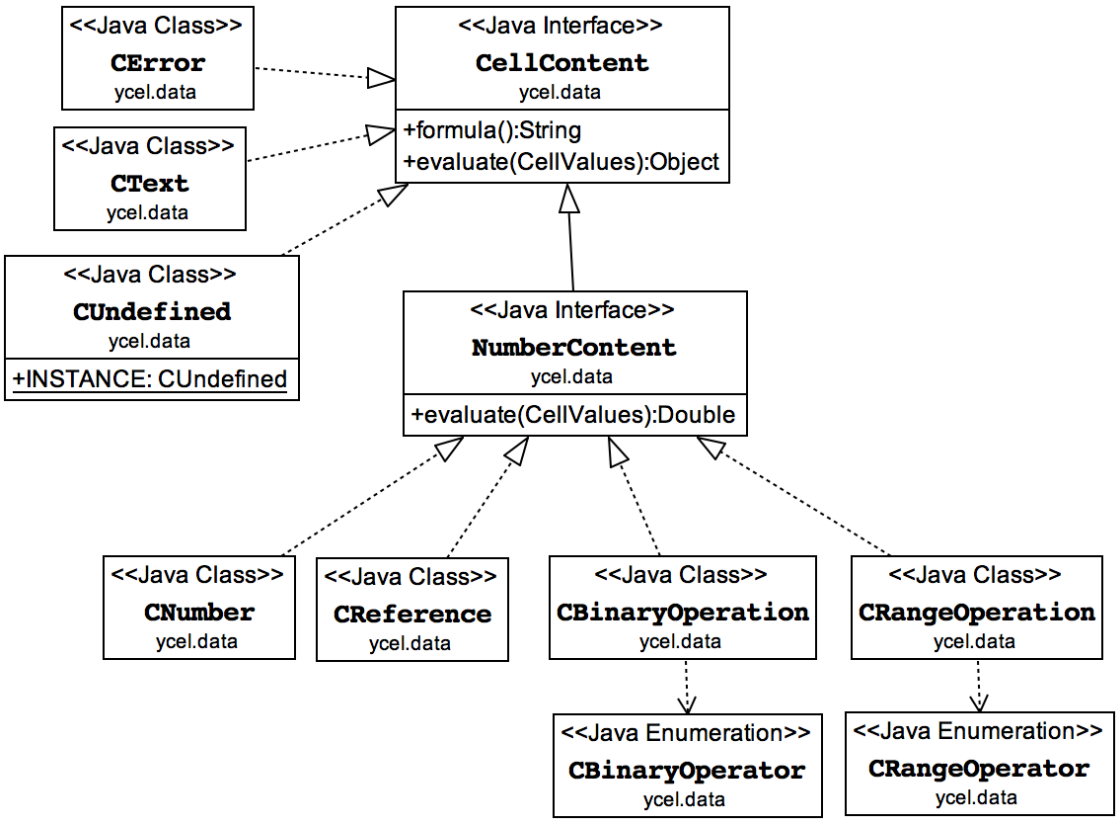

Figura 3: Tipos de células

Uma série de classes implementam `CellContent` e `NumberContent`, ilustradas no diagrama UML da Figura 3. **Leia com atenção a documentação Javadoc para os detalhes de funcionamento.** A funcionalidade pretendida é sumariamente a seguinte:

`UndefinedCell`	Célula não preenchida na folha de cálculo. **A implementação já é dada.**
`ErrorCell`	Célula mal preenchida na folha de cálculo.**A implementação já é dada.**
`TextCell`	Representa texto.
`NumberCell`	Representa um número.
`CellReference`	Representa uma referência a outra célula.
`BinaryOperation`	Operação binária (ex. soma, multiplicação) sobre dois valores. O operador usado pela operação é dado pela enumeração `CRangeOperator`, que deverá também completar.
`RangeOperation`	Operação sobre uma gama (ex. soma, média) de células. O operador usado pela operação é dado pela enumeração `CRangeOperator`, que deverá também completar.

2.3 Formato de texto `.ycel.txt`

Dá-se abaixo um exemplo de um exemplo de ficheiro YCel e do resultado esperado de visualização. Como ilustrado, os ficheiros têm 2 secções: a de conteúdos e a de estilos. Cada uma destas seções é iniciada por um inteiro que assinala quantas posições têm valores definidos (para o conteúdo ou para o estilo). Para mais detalhes consulte a documentação Javadoc e veja outros exemplos na pasta `resources/examples`.

Nota: As mudanças de linha não são significativas e desaconselha-se portanto a leitura do ficheiro linha-a-linha. Veja a documentação de `DocumentIO.load()` sobre o uso de métodos da classe utilitária `CellParser` que facilitam boa parte do trabalho.

Ficheiro `varios_tipos.ycel.txt`

```
14
A1 'Exemplo
A2 'CText
A3 'CNumber
A4 'CReference
A5 'BinaryOp
A6 'RangeOp
A7 'CError
B1 'Valor
B2 'A_B_C
B3 2.5
B4 #B3
B5 + * #B4 6.5 1.0
B6 SUM B3 B5
B7 1.0nao_faz_sentido_#C4
5
A1 dp=2;align=CENTER;bg=000000;fg=ffffff;font=ArialMT-16;
B1 dp=2;align=CENTER;bg=000000;fg=ffffff;font=ArialMT-16;
B3 dp=1;align=CENTER;bg=ffffff;fg=000000;font=Monospaced-12;
B4 dp=3;align=CENTER;bg=ffffff;fg=000000;font=Monospaced-12;
B5 dp=2;align=CENTER;bg=ffffff;fg=000000;font=Monospaced-12;
```

	A	B
0		
1	Exemplo	Valor
2	CText	A B C
3	CNumber	2.5
4	CReference	2.500
5	BinaryOp	17.25
6	RangeOp	22.25
7	CError	Error: ...

Figura 4: Visualização para `varios_tipos.ycel.txt`

2.4 Iniciando o trabalho ...

1. Comece por dar corpo a `CellPosition`, `Document` e `CNumber` até a aplicação permitir a edição e visualização correcta de números.

2. Codifique de seguida `CText`, `CReference`, e `CBinaryOperation/Operator`.

3. Programe e teste a leitura/gravação de documentos e correspondente implementação de conteúdos de células progressivamente usando os exemplos disponíveis.