Relatório Técnico do Projecto

CleanSheets

Grupo 2

Elemento
1100554 – Bruno Cunha

1 Introdução

1.1 Apresentação

O problema proposto na 3ª iteração - na área linguagem Core - consiste em capacitar a aplicação de reconhecer novas instruções, especificamente a nova fórmula Eval, e também a utilização de variáveis temporárias.

1.2 Requisitos

Permitir escrever fórmulas nos mais diversos contextos, especificamente a nova fórmula Eval e a utilização de variáveis temporárias.

1.3 Objectivos

O objectivo principal é que a aplicação seja capaz de interpretar uma nova função – Eval – e também suporte a utilização de variáveis temporárias numa sequência de fórmulas.

1.4 Dificuldades

A principal dificuldade foi o suporte das variáveis temporárias, pois envolve alterações no compilador da gramática. Permitir a função Eval também foi algo "complicado" pois envolveu a alteração de operadores primários, nomeadamente o da adição – "+".

1.5 Estrutura do Relatório Técnico

1-Introdução
2- Enquadramento
3-Análise
4-Concepção
5-Conclusão
6-Bibliografia

2 Enquadramento

2.1 Descrição dos Requisitos

Permitir escrever fórmulas nos mais diversos contextos, especificamente a nova fórmula Eval e a utilização de variáveis temporárias:

O programa deverá reconhecer uma nova fórmula, e também deverá ser capaz de utilizar variáveis temporárias.

2.2 Enquadramento do Projecto

Os requisitos do projecto, nesta iteração, são o reconhecimento das variáveis temporárias e da função Eval, que serão inseridas na gramática Portuguesa. É algo novo à aplicação, e que será exclusivo da gramática Portuguesa, criada desde a 1ª iteração.

2.3 Funcionalidades

A funcionalidade requisitada para "melhorar" a aplicação foi o reconhecimento de variáveis temporárias e também o suporte de uma nova função.

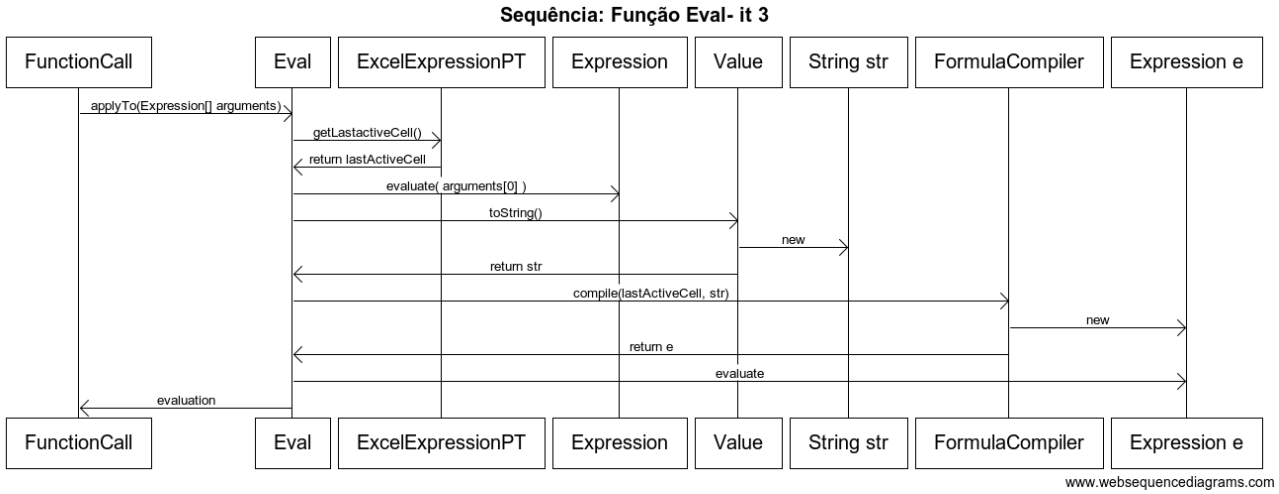
3 Análise

3.1 Requisitos Funcionais

Diagrama Sequência:

(Inclui apenas os casos de uso desta iteração)

***3.1.1** Fórmula "Eval"*



3.1.2 Variáveis temporárias

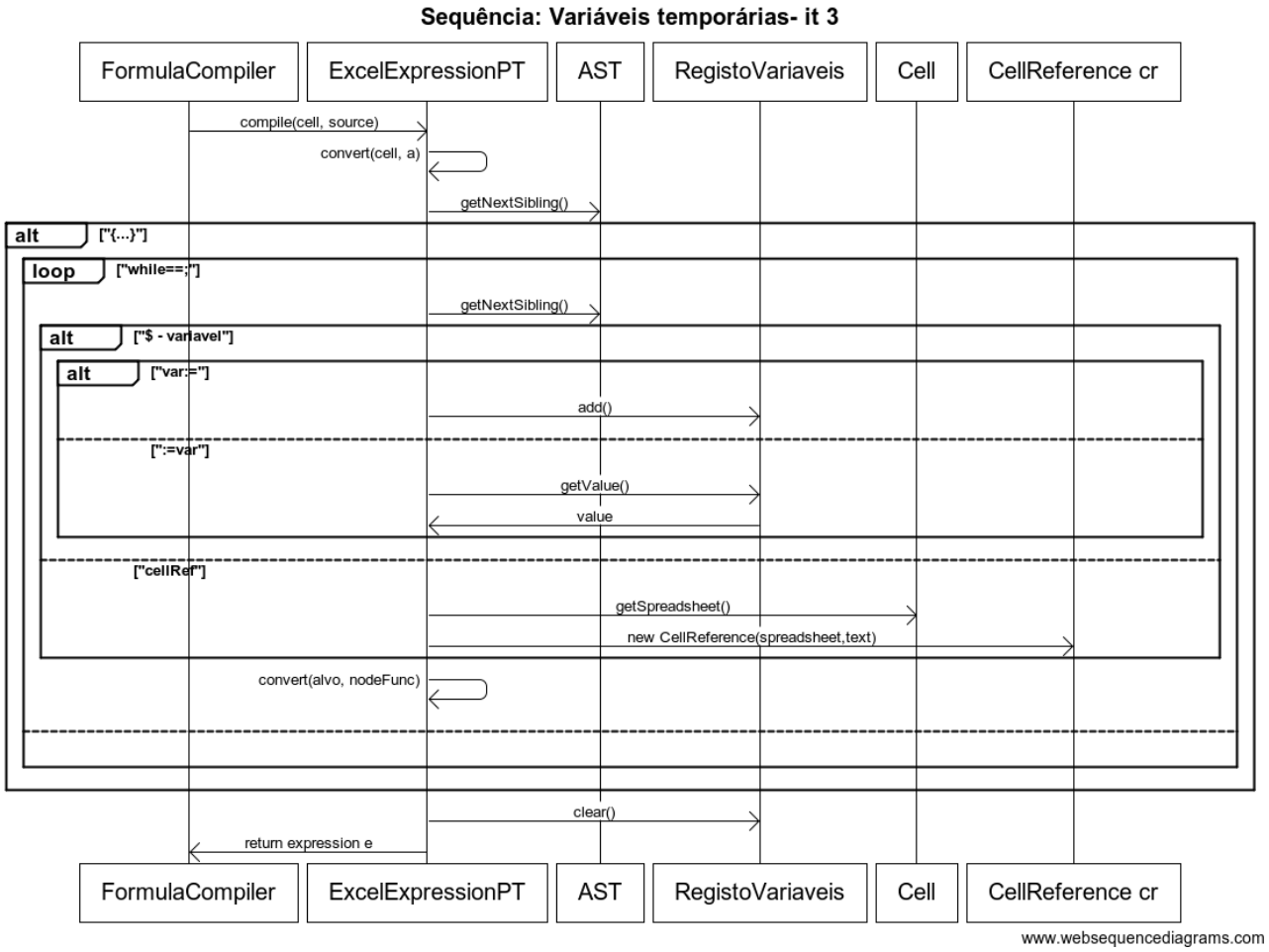


Diagrama Casos de Uso:

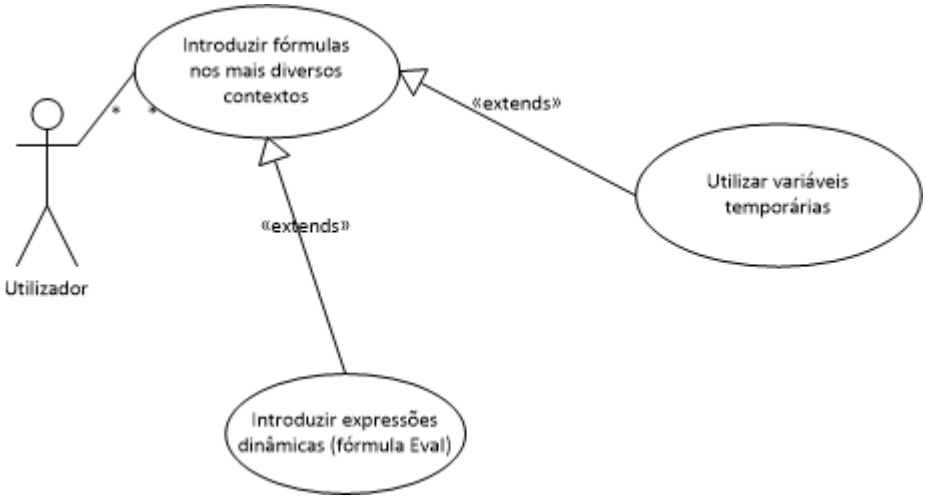
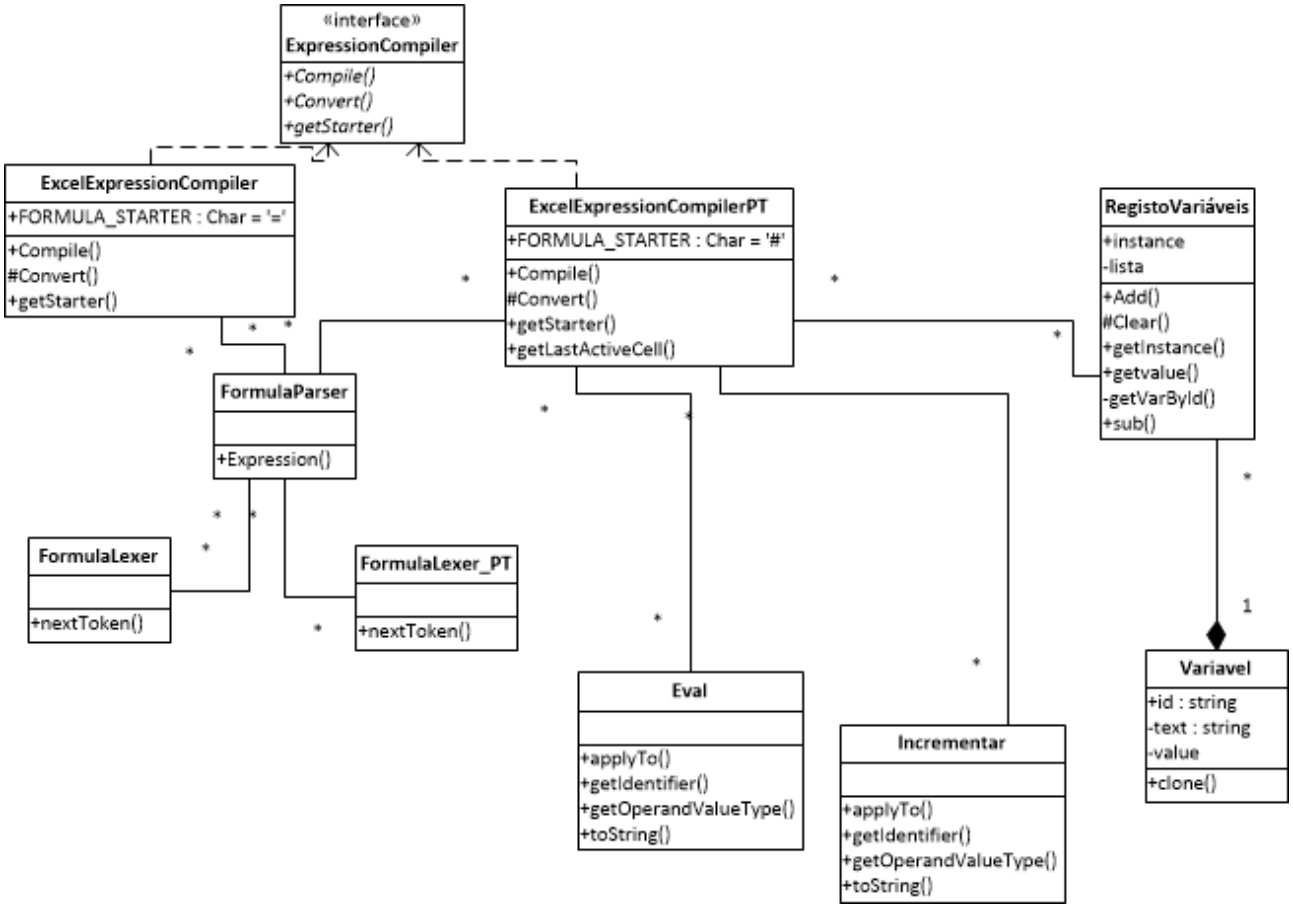


Diagrama Classes:



3.2 Requisitos Não Funcionais

- A solução deverá ser multiplataforma, incluindo, no mínimo, suporte para um sistema operativo da família Windows e para uma distribuição de GNU/Linux;
- A solução também não deverá depender em particular de nenhum SGBD Relacional

3.3 Contextualização

Permitir escrever fórmulas nos mais diversos contextos, especificamente a nova fórmula Eval e a utilização de variáveis temporárias.

3.4 Planeamento

Todo o planeamento pode ser consultado no ficheiro integrado no repositório.

4 Concepção

4.1 Especificação

Ver ponto 3 – análise.

4.2 Aspectos técnicos da solução

Foi utilizada uma interface comum às gramáticas. É utilizado o ANTLR para gerar *parsers*. Neste caso o ANTLR está também a gerar árvore sintáctica resultante do "parsing" (AST). A AST é percorrida (método "convert") e os seus nodos são convertidos em instâncias de "Expression" que são usadas para executar a expressão. Para analisar as expressões relativas à iteração 2/3, são analisados os "nodes" gerados pelo sistema ANTLR, ou seja, é analisada a árvore léxica gerada pela expressão introduzida.
Quanto ao reconhecimento das variáveis, estas são tratadas quando a árvore léxica é gerada, após a introdução da sequência de expressões.
A função Eval utiliza o ExpressionCompiler para analisar o argumento recebido. Foi também alterado o operador "+" da gramática PT para permitir argumentos do tipo ("a"+b1), concatenando a "String".
Foi implementada uma Thread para a análise das sequências de expressões, como foi pedido.

4.3 Testes

4.3.1 Testes unitários

Devido à grande extensão dos testes unitários, apenas são incluídos alguns exemplos dos mesmos. No entanto, os ficheiros completos encontram-se no repositório.

```
@Test
   public void testEval() throws Exception {
      System.out.println("Teste Eval");
      String source = "#eval(\"Media(2;4;6)\")";
      Cell cell = s.getCell(1, 1);
      Formula f = FormulaCompiler.getInstance().compile(cell, source);
      assertEquals(true, f.evaluate().toDouble()==4);
   }
```

```
@Test
public void testSequencia() throws Exception {
    System.out.println("Teste se e Sequencia");
    String source = "#{a2:=se(2>1;20;30);a3:=a2}";
    Cell cell = s.getCell(1, 1);
    Formula f = FormulaCompiler.getInstance().compile(cell, source);
    assertEquals(true, f.evaluate().toDouble()==20);
}

@Test
public void testVariavel() throws Exception {
    System.out.println("Teste variaveis");
    String source = "#{$temp1:=20;a3:=soma($temp1;10)}";
    Cell cell = s.getCell(1, 1);
    Formula f = FormulaCompiler.getInstance().compile(cell, source);
    assertEquals(true, f.evaluate().toDouble()==30);
}
```

4.3.2 Casos de teste

Semana 3
(11 de Junho a 17 de Junho)

Nome do caso de teste: *Core – Fórmula Eval*
Casos de uso relacionados:

Objectivo	*O utilizador deverá conseguir inserir uma chamada à função "Eval"*
Pré-requisitos	
Dados de teste	*Expressão (s) inserida na barra de edição.*
Passos	1. *Inserir a expressão com a fórmula Eval* 2. *Verificar o resultado*
Notas e Questões	

Resultados

#Execução	Dados	Resultados	Passou?	Observaçõ es
#1	#eval("soma(1;2)")	3	Sim, valor correcto.	
#2	#{a1:=4;b4:=10; c1:= eval(b+"a1")}	A1=4 B4=10 C1=10	Sim, valor correcto.	
#3	#a1:=soma("AD "+1)	A1=AD1	Sim, valor correcto.	

Nome do caso de teste: *Core – variáveis temporárias*
Casos de uso relacionados:

Objectivo	*O utilizador deverá utilizar variáveis temporárias nas sequências de expressões.*
Pré-requisitos	
Dados de teste	*Expressão (s) inserida na barra de edição.*
Passos	1. *Inserir a expressão* 2. *Verificar o resultado*
Notas e Questões	

Resultados

#Execução	Dados	Resultados	Passou?	Observaçõ es
#1	#{$temp1:=4; A2:=$aux1}	A2= VARIAVEL NÃO DEFINIDA	nao, variável com nome errado.	
#2	#{$temp1:=4; A2:=$temp1}	A2= 4	Sim, valor correcto.	
#3	#{$temp1:=2; D2:=10;c4:=eval (“soma((“d”+$temp1),10)”}	C4= 20	Sim, valor correcto.	

5 Implementação

- Existem dependências de 3 ficheiros de configuração, referentes às 2 gramáticas e aos seus compiladores de fórmulas. São também utilizados os scripts para criar “dinamicamente” os “lexers” das 2 gramáticas.

6 Conclusão

Esta iteração foi concluída com sucesso, tendo sido cumpridos todos os requisitos. Ficou totalmente funcional o reconhecimento, não só da nova função, mas também das variáveis temporárias. Foi também implementada uma nova thread para analisar as sequências.

7 Bibliografia:

http://www.antlr.org/

http://csheets.sourceforge.net/api/