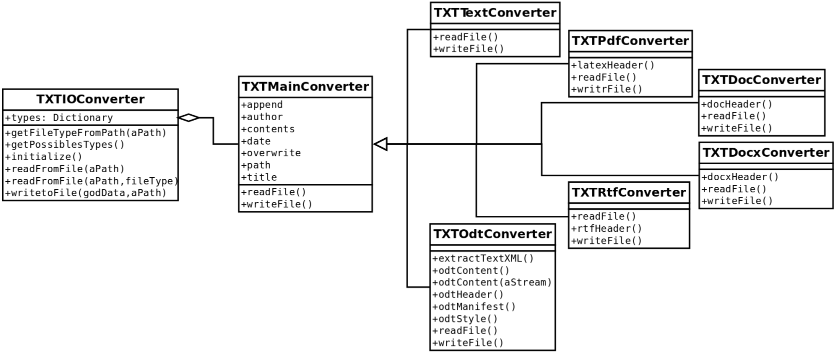
# GODTextIO

André M. Vale     Evandro F. Giovanini

Departamento de Ciências da Computação
Universidade de São Paulo

25 de junho de 2015

- **Objetivo**: importar e exportar dados de GODData para arquivos PDF, RTF, DOC, etc
- **Escopo inicial**:
  - AbiWord atual incompatível com o projeto
  - Sem suporte para doc e docx
- **Metodologia adotada**:
  - Substituir o AbiWord por unoconv (Libreoffice)
  - Implementação dos formatos DOC e DOCX



**TXTIOConverter**

+types: Dictionary

+getFileTypeFromPath(aPath)
+getPossiblesTypes()
+initialize()
+readFromFile(aPath)
+readFromFile(aPath,fileType)
+writetoFile(godData,aPath)

**TXTMainConverter**

+append
+author
+contents
+date
+overwrite
+path
+title

+readFile()
+writeFile()

**TXTTextConverter**

+readFile()
+writeFile()

**TXTPdfConverter**

+latexHeader()
+readFile()
+writrFile()

**TXTDocConverter**

+docHeader()
+readFile()
+writeFile()

**TXTDocxConverter**

+docxHeader()
+readFile()
+writeFile()

**TXTRtfConverter**

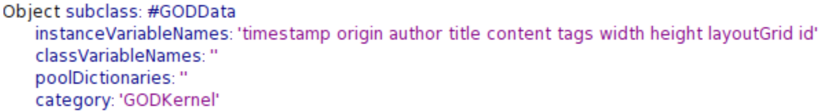
+readFile()
+rtfHeader()
+writeFile()

**TXTOdtConverter**

+extractTextXML()
+odtContent()
+odtContent(aStream)
+odtHeader()
+odtManifest()
+odtStyle()
+readFile()
+writeFile()

Diagrama de Classes

# GODData



```
Object subclass: #GODData
    instanceVariableNames: 'timestamp origin author title content tags width height layoutGrid id'
    classVariableNames: ''
    poolDictionaries: ''
    category: 'GODKernel'
```

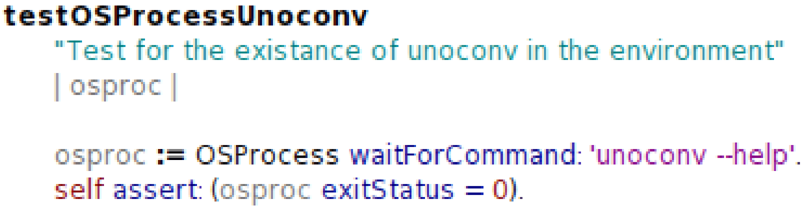
# Testes

testOSProcessPdflatex
testOSProcessUnoconv

testReadDocFromFileReturn
testWriteDocToExistentFileOverwrite
testWriteDocToFileAppend
testWriteDocToFileBasic
testWriteDocToFileOverwrite
testWriteDocToNonexistentFileAppend

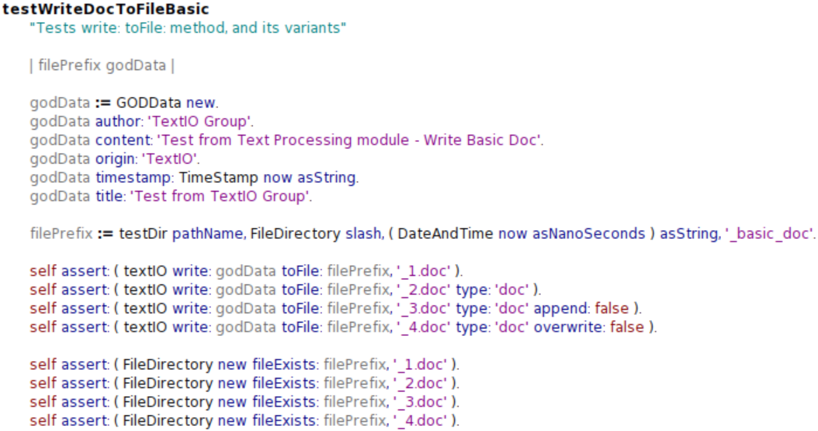
testReadDocxFromFileReturn
testWriteDocxToExistentFileOverwrite
testWriteDocxToFileAppend
testWriteDocxToFileBasic
testWriteDocxToFileOverwrite
testWriteDocxToNonexistentFileAppend

# testOSProcessUnoconv



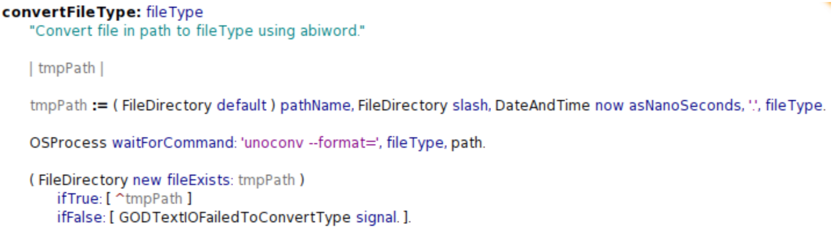
```
testOSProcessUnoconv
    "Test for the existance of unoconv in the environment"
    | osproc |

    osproc := OSProcess waitForCommand: 'unoconv --help'.
    self assert: (osproc exitStatus = 0).
```

# testWriteDocToFileBasic



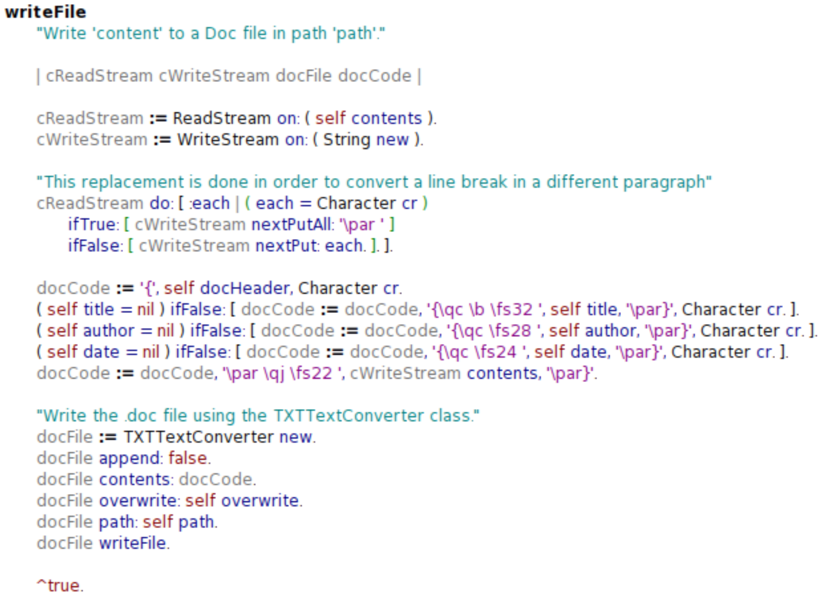
```
testWriteDocToFileBasic
    "Tests write: toFile: method, and its variants"

    | filePrefix godData |

    godData := GODData new.
    godData author: 'TextIO Group'.
    godData content: 'Test from Text Processing module - Write Basic Doc'.
    godData origin: 'TextIO'.
    godData timestamp: TimeStamp now asString.
    godData title: 'Test from TextIO Group'.

    filePrefix := testDir pathName, FileDirectory slash, ( DateAndTime now asNanoSeconds ) asString, '_basic_doc'.

    self assert: ( textIO write: godData toFile: filePrefix, '_1.doc' ).
    self assert: ( textIO write: godData toFile: filePrefix, '_2.doc' type: 'doc' ).
    self assert: ( textIO write: godData toFile: filePrefix, '_3.doc' type: 'doc' append: false ).
    self assert: ( textIO write: godData toFile: filePrefix, '_4.doc' type: 'doc' overwrite: false ).

    self assert: ( FileDirectory new fileExists: filePrefix, '_1.doc' ).
    self assert: ( FileDirectory new fileExists: filePrefix, '_2.doc' ).
    self assert: ( FileDirectory new fileExists: filePrefix, '_3.doc' ).
    self assert: ( FileDirectory new fileExists: filePrefix, '_4.doc' ).
```

# Chamada ao unoconv



```
convertFileType: fileType
    "Convert file in path to fileType using abiword."

    | tmpPath |

    tmpPath := ( FileDirectory default ) pathName, FileDirectory slash, DateAndTime now asNanoSeconds, '.', fileType.

    OSProcess waitForCommand: 'unoconv --format=', fileType, path.

    ( FileDirectory new fileExists: tmpPath )
        ifTrue: [ ^tmpPath ]
        ifFalse: [ GODTextIOFailedToConvertType signal. ].
```

# Escrevendo um arquivo DOC



```
writeFile
    "Write 'content' to a Doc file in path 'path'."

    | cReadStream cWriteStream docFile docCode |

    cReadStream := ReadStream on: ( self contents ).
    cWriteStream := WriteStream on: ( String new ).

    "This replacement is done in order to convert a line break in a different paragraph"
    cReadStream do: [ :each | ( each = Character cr )
        ifTrue: [ cWriteStream nextPutAll: '\par ' ]
        ifFalse: [ cWriteStream nextPut: each. ]. ].

    docCode := '{', self docHeader, Character cr.
    ( self title = nil ) ifFalse: [ docCode := docCode, '{\qc \b \fs32 ', self title, '\par}', Character cr. ].
    ( self author = nil ) ifFalse: [ docCode := docCode, '{\qc \fs28 ', self author, '\par}', Character cr. ].
    ( self date = nil ) ifFalse: [ docCode := docCode, '{\qc \fs24 ', self date, '\par}', Character cr. ].
    docCode := docCode, '\par \qj \fs22 ', cWriteStream contents, '\par}'.

    "Write the .doc file using the TXTTextConverter class."
    docFile := TXTTextConverter new.
    docFile append: false.
    docFile contents: docCode.
    docFile overwrite: self overwrite.
    docFile path: self path.
    docFile writeFile.

    ^true.
```