FACULTÉ DES SCIENCES ET TECHNOLOGIES
Département Informatique

Université de Lille

Présentation de projet

Thème :

AVL et Artefact

Réalisés par :

BOUNEHAR Lynda

GHOMARI Mehdi

Année Universitaire : 2023/2024

Plan de travail

1. Artefact
2. AVL
3. Techniques
4. Tests
5. Outils et documentation
6. Conclusion

Artefact

https://book.huihoo.com/smalltalk/pharo/enterpr

Generate PDF Documents with A

Olivier Auverlot and Guillaume Larchevêque with Johan Fabry

The Adobe PDF format is probably one of the most widespread electronic documen text and graphics. If you receive a bill, follow a purchase on a web site, downloa programmers that need to provide any such reporting functionality, supporting this fo

In Pharo, Artefact is an innovative framework that supports the design and generatio

1. Overview of Artefact

Artefact is a PDF framework whose design was guided by the goals of efficiency document is an object containing a collection of other objects, each corresponding to the possibility to be reused in the same document but also across documents. Obje with advanced behavior and a special appearance, e.g. that display data in a table o

Artefact contains default elements such as paragraphs or tables that allow to quickl order in which you position them in the document does not affect their appearance. (a piece of blue text will be followed by another piece of blue text in the absence of attribute is not defined in the element, Artefact then uses a style sheet that is set at t

This autonomy of elements and style management is a strong feature of Artefact. It r

1.1. Concepts, Key Aspects and Limits

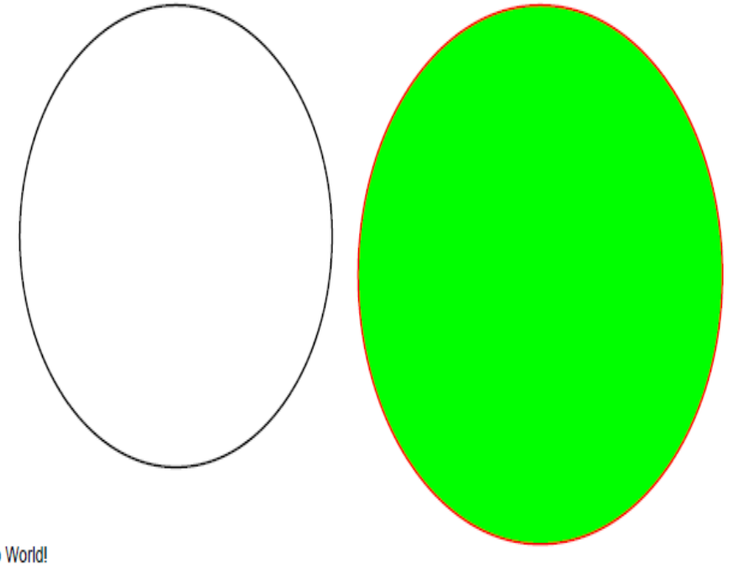
After more than a year of development, the concepts used in Artefact are conside limitations.

- Artefact has a simple architecture that facilitates scalability and new features.

Test Artefact

Do it | Publish | Bindings | Versions | Pages

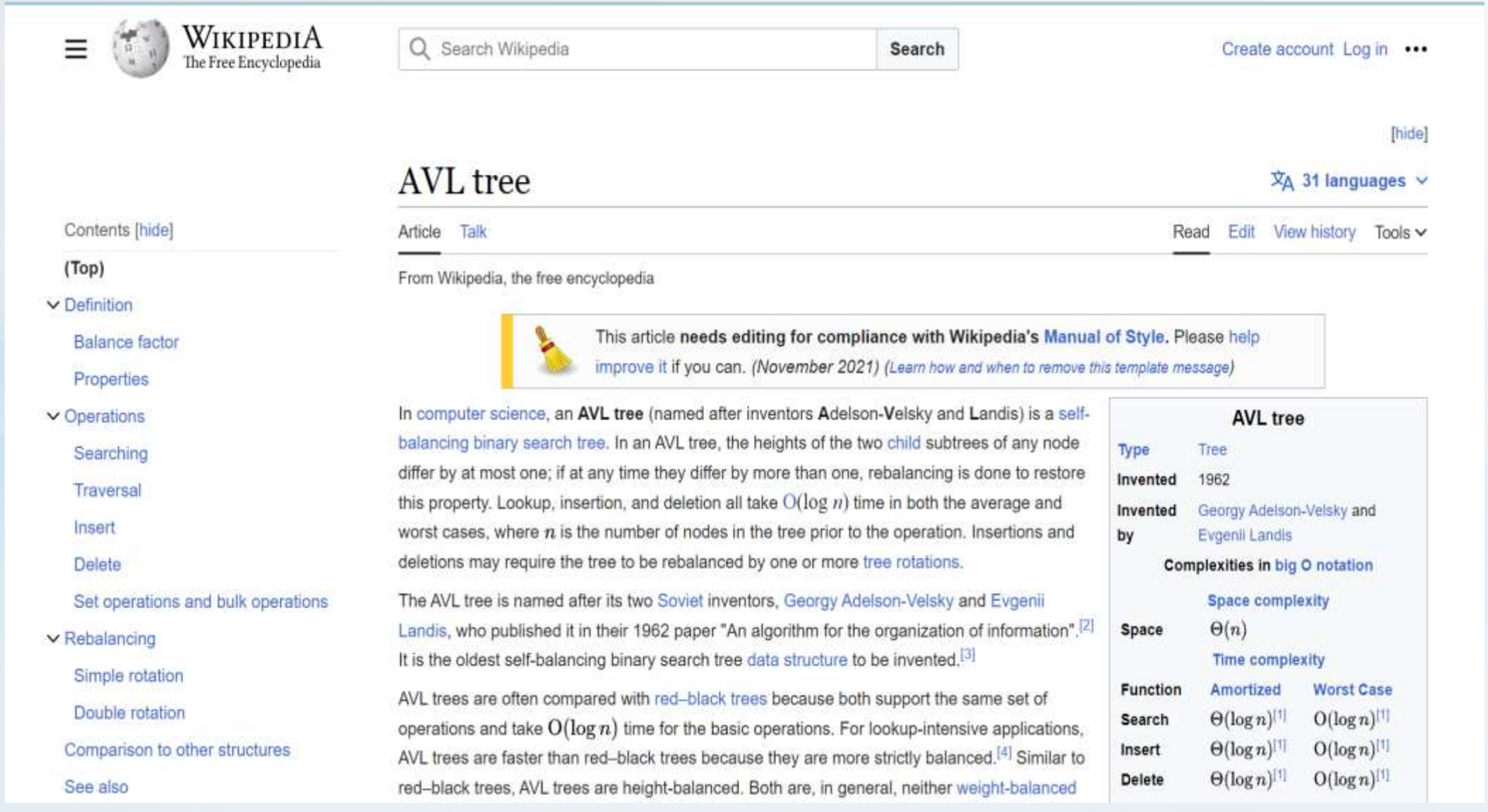
```
| pdfDocument fileStream outputPath aPage|
outputPath := 'C:\Users\GHO\Desktop\test_pdf\'.
pdfDocument := PDFDocument new.
aPage := PDFPage new.
aPage add: (PDFTextElement new text: 'Hello World!'; from: 10mm @ 90mm).
aPage add: (PDFCircleElement center: 50 mm @ 50 mm radius: 30 mm).
aPage add: ((PDFCircleElement from: 90 mm @ 20 mm to: 150 mm @ 90 mm)
fillColor: (PDFColor r: 0 g: 255 b: 0); drawColor: (PDFColor r: 255 g: 0 b:
0)).
aPage add: (PDFDataTableElement new     data: #(       #('Smith' 'Peter'
'peter.smith@mail.org')       #('Jones' 'Mickael' 'mickael.jones@epr.com')
        #('washington' 'robert' 'robert.washington@blif.com')     );
      textColor:(PDFColor r: 0 g: 0 b: 255);     fillColor: (PDFColor new
setGreyLevel: 230);      font: (PDFTimesFont new fontSize: 6 pt);
      dotted: (PDFDotted new length: 0.2mm; space: 0.2mm);      from: 10 mm
@ 100 mm;      dimension: 190 mm @ 50 mm   ).
pdfDocument add: aPage.
fileStream := (outputPath, 'test2.pdf')  asFileReference binaryWriteStream.
pdfDocument exportTo: fileStream. fileStream close.
```



Hello World!

Smith	Peter	peter.smith@mail.org
Jones	Mickael	mickael.jones@epr.com
washington	robert	robert.washington@blif.com

AVL (A delson- V elsky et Landis)



WIKIPEDIA
The Free Encyclopedia

Search Wikipedia | Search

Create account Log in •••

[hide]

AVL tree

31 languages

Article Talk | Read Edit View history Tools

From Wikipedia, the free encyclopedia

Contents [hide]

(Top)
Definition
Balance factor
Properties
Operations
Searching
Traversal
Insert
Delete
Set operations and bulk operations
Rebalancing
Simple rotation
Double rotation
Comparison to other structures
See also

This article **needs editing for compliance with Wikipedia's Manual of Style.** Please help improve it if you can. *(November 2021)* *(Learn how and when to remove this template message)*

In computer science, an **AVL tree** (named after inventors **A**delson-**V**elsky and **L**andis) is a self-balancing binary search tree. In an AVL tree, the heights of the two child subtrees of any node differ by at most one; if at any time they differ by more than one, rebalancing is done to restore this property. Lookup, insertion, and deletion all take $O(\log n)$ time in both the average and worst cases, where n is the number of nodes in the tree prior to the operation. Insertions and deletions may require the tree to be rebalanced by one or more tree rotations.

The AVL tree is named after its two Soviet inventors, Georgy Adelson-Velsky and Evgenii Landis, who published it in their 1962 paper "An algorithm for the organization of information".[2] It is the oldest self-balancing binary search tree data structure to be invented.[3]

AVL trees are often compared with red–black trees because both support the same set of operations and take $O(\log n)$ time for the basic operations. For lookup-intensive applications, AVL trees are faster than red–black trees because they are more strictly balanced.[4] Similar to red–black trees, AVL trees are height-balanced. Both are, in general, neither weight-balanced

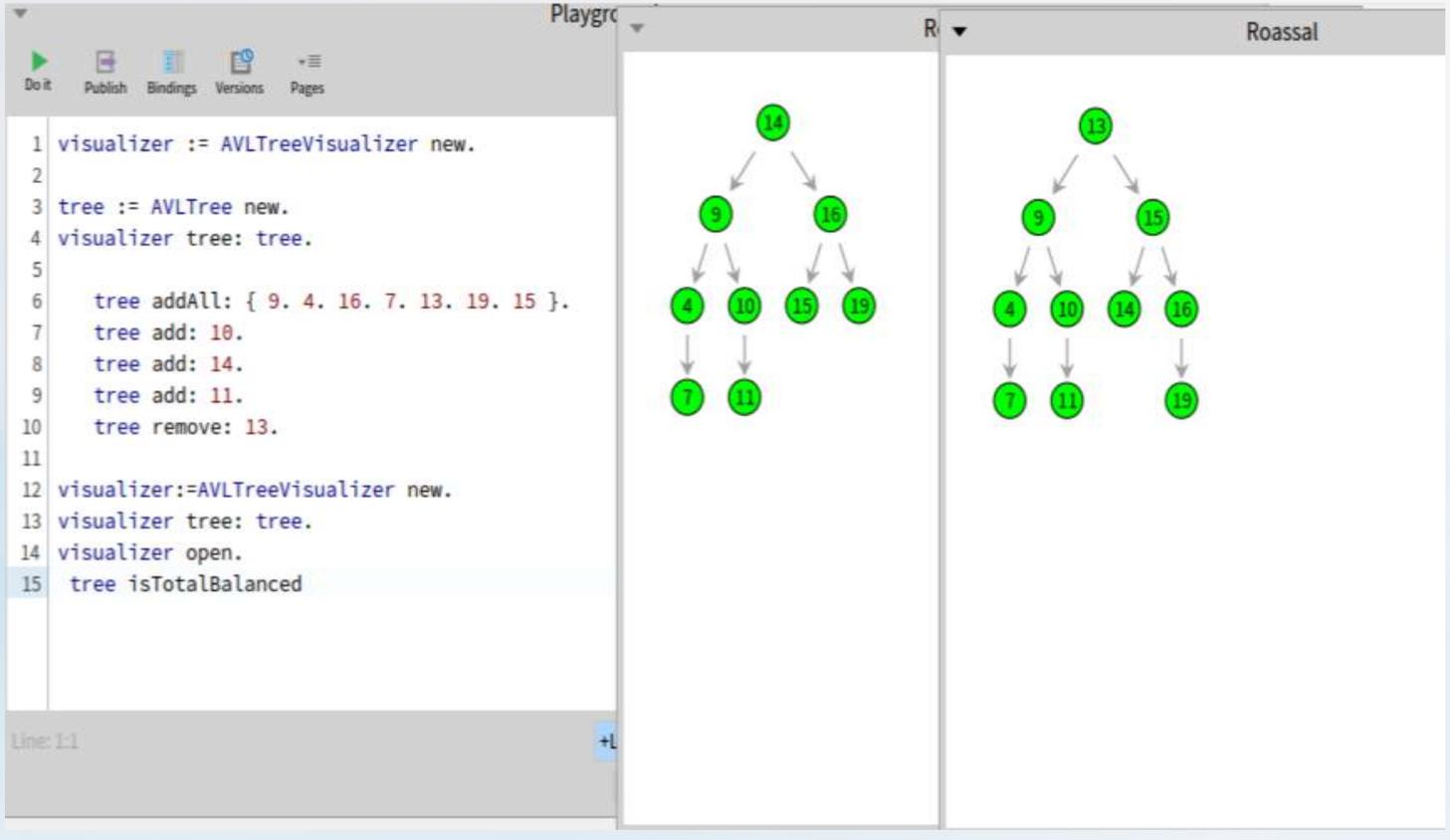
AVL tree

Type	Tree
Invented	1962
Invented by	Georgy Adelson-Velsky and Evgenii Landis

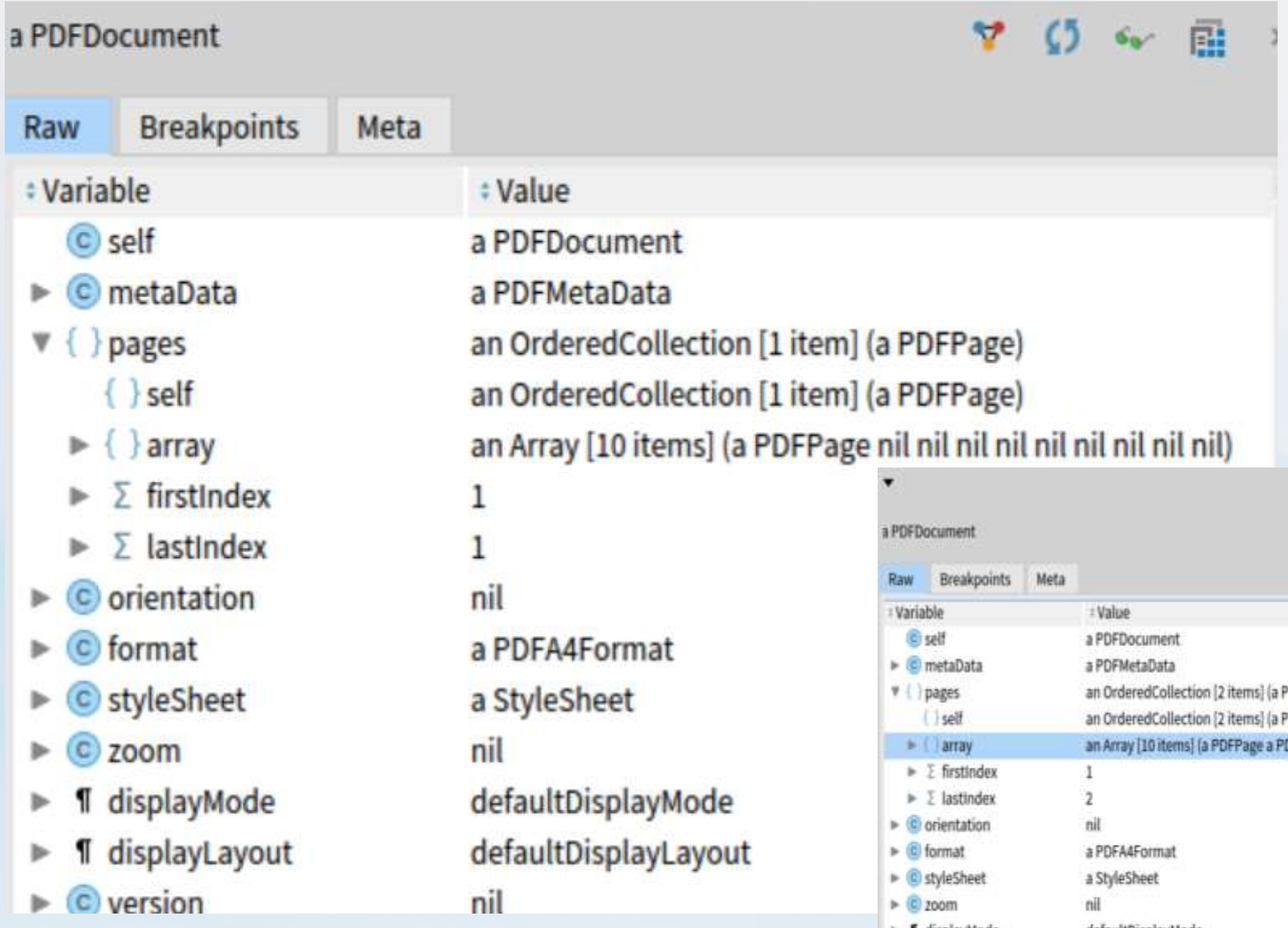
Complexities in big O notation

	Space complexity	
Space	$\Theta(n)$	
	Time complexity	
Function	Amortized	Worst Case
Search	$\Theta(\log n)$[1]	$O(\log n)$[1]
Insert	$\Theta(\log n)$[1]	$O(\log n)$[1]
Delete	$\Theta(\log n)$[1]	$O(\log n)$[1]

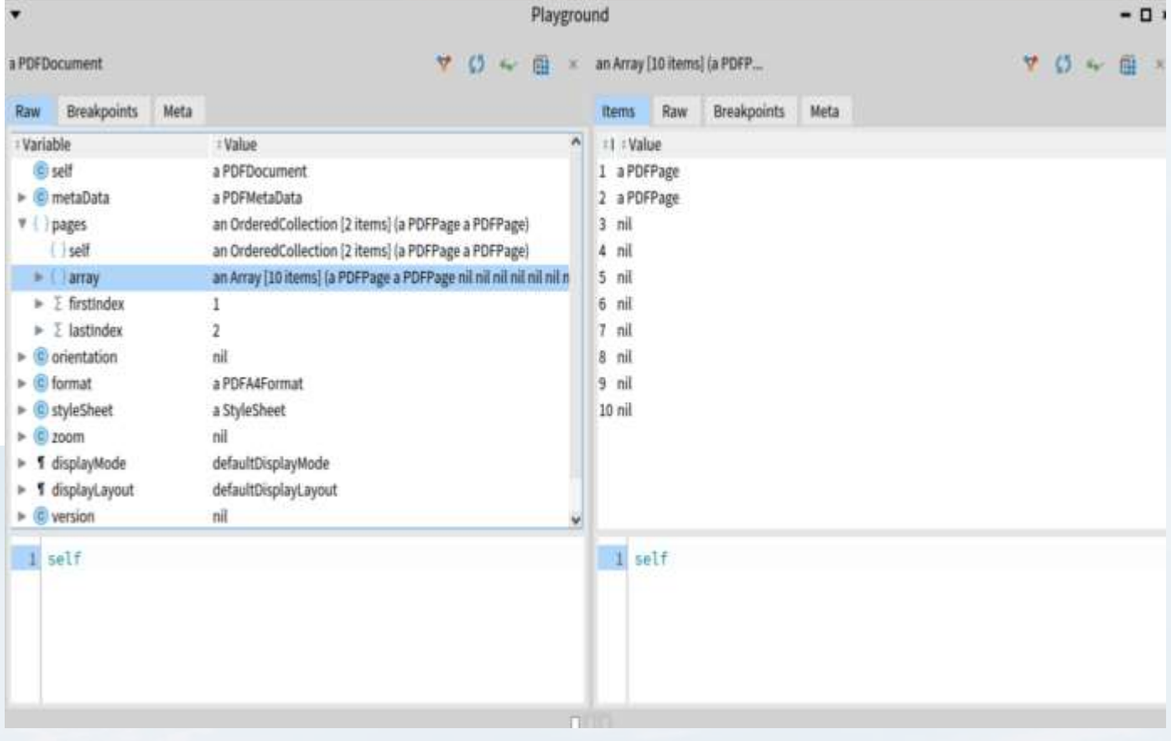
TEST AVL


Playgro
Do it
Publish
Bindings
Versions
Pages
visualizer := AVLTreeVisualizer new.
tree := AVLTree new.
visualizer tree: tree.
tree addAll: { 9. 4. 16. 7. 13. 19. 15 }.
tree add: 10.
tree add: 14.
tree add: 11.
tree remove: 13.
visualizer:=AVLTreeVisualizer new.
visualizer tree: tree.
visualizer open.
tree isTotalBalanced
Line: 1:1
Roassal
14
9
16
4
10
15
19
7
11
13
9
15
4
10
14
16
7
11
19

Inspecteur Artefact



|pdfDocument|
pdfDocument := PDFDocument new.



OrderedCollection

```
| myOrderedCollection |

"Créer une nouvelle OrderedCollection"
myOrderedCollection := OrderedCollection new.

"Ajouter 12 éléments à la collection"
1 to: 12 do: [:i |
    myOrderedCollection add: i.
].
myOrderedCollection
```

Items | Raw | Breakpoints | Meta

Variable	Value
self	an OrderedCollection [12 items] (1 2 3 4 5 6 7 8 9 10 11 12)
array	an Array [20 items] (1 2 3 4 5 6 7 8 9 10 11 12 nil nil nil nil nil nil nil nil)
firstIndex	1
lastIndex	12

comment artefact envoie notre contenu au fichier ? Il envoie quoi ? Et comment?

```
fileStream := (outputPath, 'test2.pdf')  asFileReference binaryWriteStream.
pdfDocument exportTo: fileStream.
fileStream close.
```

Implementors of exportTo: [1]

Implementors of exportTo: [1]

PDFDocument	exportTo:	generate	Artefact-Core

Filter...

Current image | Flat | Hier. | Browse | Class refs. | Implementors | Senders

exportTo:

```
exportTo: aStream
    "Generate a PDFDataObject from document"

    ^ PDFStreamPrinter new
      printModel: (PDFGenerator new generatePDFCompleteModelFrom: self)
      compression: self compression
      on: aStream
```

PDFStreamPrinter

Senders of streamContent [2]

PDFStreamPrinter	printElementCodeSegment:	print	Artefact-Core
PDFStreamPrinter	printPDFDataStream:	print	Artefact-Core

Filter...

Current image | Flat | Hier. | Browse | Class refs. | Implementors | Senders

printPDFDataSt ×

```
printPDFDataStream: aPDFDataStream

   | endPosition startPosition streamData |

   characterStream
      nextPutAll: 'stream';
      lf.
   startPosition := stream position.

   self streamContent: (WriteStream on:  String new).
   aPDFDataStream codeSegment printWith: self.
   streamData := self streamContent contents.
   (self compression) ifTrue: [ streamData := (self compressWithGZip: streamData) asByteArray  ].

   stream nextPutAll: streamData.
   characterStream lf.

   endPosition := stream position.
   self size at: aPDFDataStream put: endPosition - startPosition.
   characterStream nextPutAll: 'endstream'
```

Inspecteur du PDFStreamPrinter

```
outputStream := (outputPath, 'test3.pdf') asFileReference binaryWriteStream.
pdfStreamPrinter :=pdfDocument  exportTo: outputStream.
pdfStreamPrinter .
```

a PDFStreamPrinter

Raw | Breakpoints | Meta

Variable	Value
self	a PDFStreamPrinter
stream	a ZnBufferedWriteStream
positions	a Dictionary [12 items] (1 0 obj << /Type /Pages /Kids [2 0 R 5 0 R] /Count 2 /Me
size	a Dictionary [2 items] (a PDFDataStream->323 a PDFDataStream->121)
streamContent	a WriteStream
compression	true
characterStream	a ZnCharacterWriteStream

Flux de données (Stream)

```
startPosition
      |
      v
 +------------------->  +----------------> +----------------->
 | 'stream' + lf   |   | Content (PDF)   |  | 'endstream'   |
 +------------------->  +----------------> +----------------->

                          stream                  endPosition
```

<u>Utilisation des streams</u>

- Écriture de fichiers
- Communication réseau

Comment AVL balance un arbre ?

```
balanceZ: z y: y x: x
    | a b c |
    c := z value.
    b := y value.
    a := x value.
    (y key and: [ x key ]) ifTrue: [
        ^ self rrrotationZ: c y: b x: a ].
    (y key not and: [ x key not ]) ifTrue: [
        ^ self llrotationZ: c y: b x: a ].
    (y key not and: [ x key ]) ifTrue: [
        ^ self lrrotationZ: c y: b x: a ].
    "(y key and: [ x key not ])"
    ^ self rlrotationZ: c y: b x: a.
    "self notYetImplemented."
```

A quelle moment AVL balance arbre ?

Senders of balanceZ:y:x: [3]

AVLNode	checkPath:	private	AVL-Tree
AVLNode	checkRemovingPath:	private	AVL-Tree
RSRNode	checkPath:	private	Roassal3

Filter...

Current image | Flat | Hier. | Browse | Class refs. | Implementors | Senders

checkPath:

```
checkPath: aCollection
   aCollection size < 3 ifTrue: [ ^ self ].
   (1 to: aCollection size - 2) reverseDo: [ :index |
      | assoc |
      assoc := aCollection at: index.
      assoc value isBalanced ifFalse: [ | z y x |
         z := aCollection at: index.
         y := aCollection at: index + 1.
         x := aCollection at: index + 2.
         ^ self balanceZ: z y: y x: x ] ]
```

Ajout et suppression dans un arbre

Senders of balanceZ:y:x: [3] | Senders of checkPath: [3]

AVLNode	addChild:	adding	AVL-Tree
AVLTreeVisualizer	addFindingPath:	events	AVL-Tree-Inspector
RSRNode	addChild:	adding	Roassal3

Filter...

Current image | Flat | Hier. | Browse | Class refs. | Implementors | Senders

addChild:

```
addChild: newObject
    | path |
    path := OrderedCollection with: nil -> self.
    self add: newObject path: path.
    self checkPath: path.
    ^ self
```

Senders of checkRemovingPath: [1]

Senders of balanceZ:y:x: [3] | Senders of checkPath: [3] | Senders of checkRemovingPath: [1]

AVLTree	remove:ifAbsent:	removing	AVL-Tree

Filter...

Current image | Flat | Hier. | Browse | Class refs. | Implementors | Senders

remove:ifAbsen

```
remove: oldObject ifAbsent: anExceptionBlock
    | toRemove path |
    path := OrderedCollection new.
    toRemove := root remove: oldObject path: path.
    toRemove ifNil: [ ^ anExceptionBlock value ].

    toRemove == root ifTrue: [
        root := root successor: path.
        root ifNil: [ root := AVLNilNode new ] ].
    root checkRemovingPath: path.


    ^ toRemove contents
```

Mutation testing pour stream

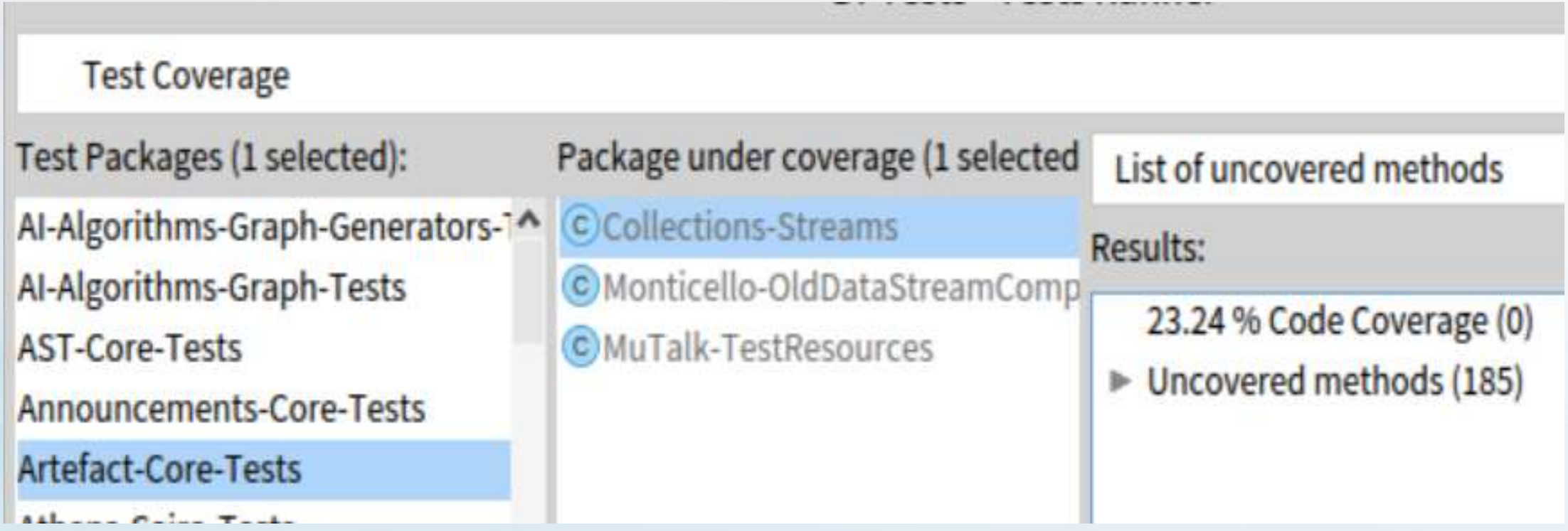
```
testCases :=  { PDFStreamPrinterTest }.
classesToMutate := { PDFStreamPrinter }.

analysis := MutationTestingAnalysis
    testCasesFrom: testCases
    mutating: classesToMutate
    using: MutantOperator contents
    with:
AllTestsMethodsRunningMutantEvaluationStrategy
new.

analysis run.
analysis generalResult mutationScore.
```

Integer | Raw | Breakpoints | Meta

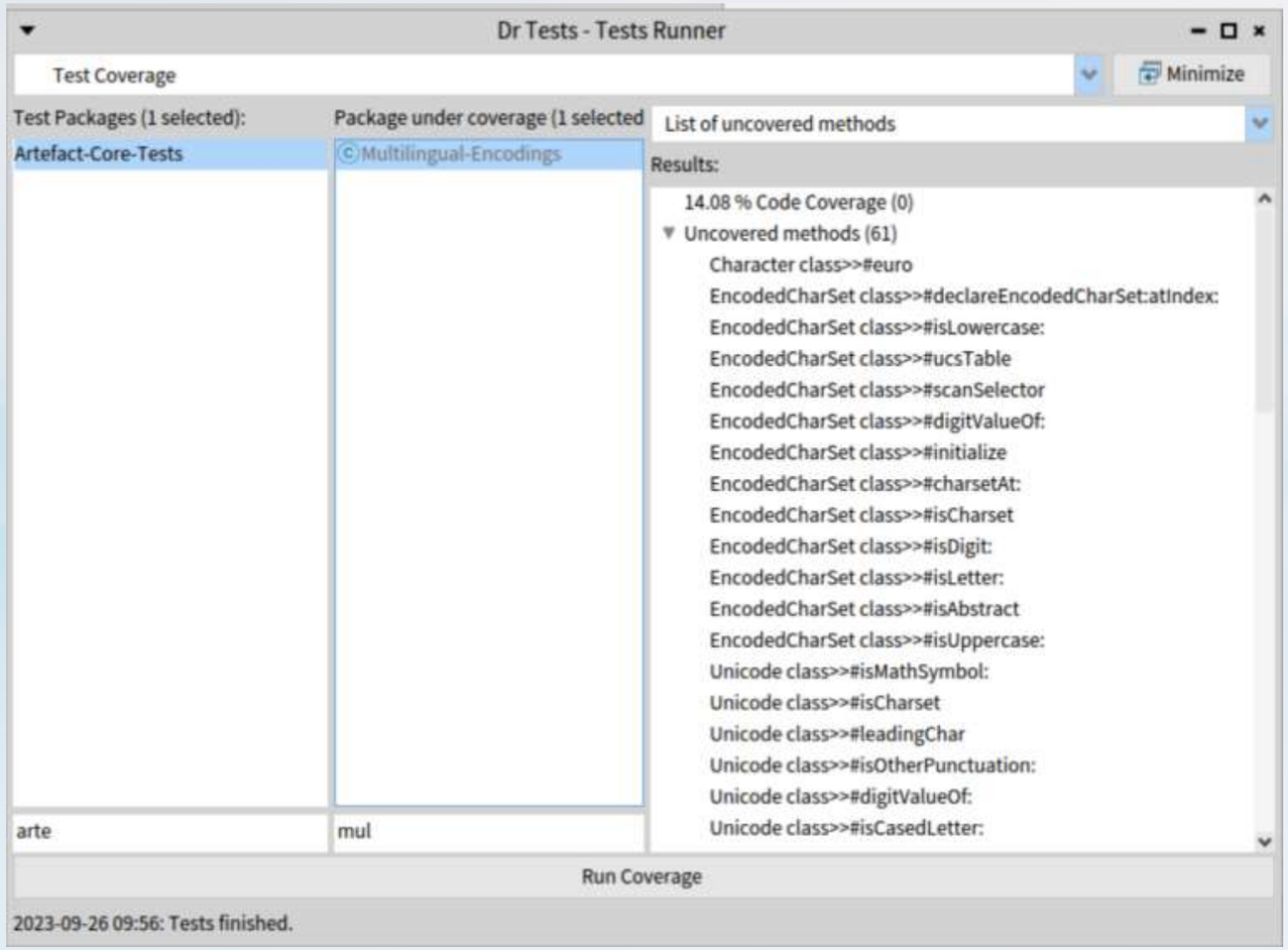
key	value
self	24
decimal	24
hex	18
octal	30
binary	11000
character	

```
1 self
```

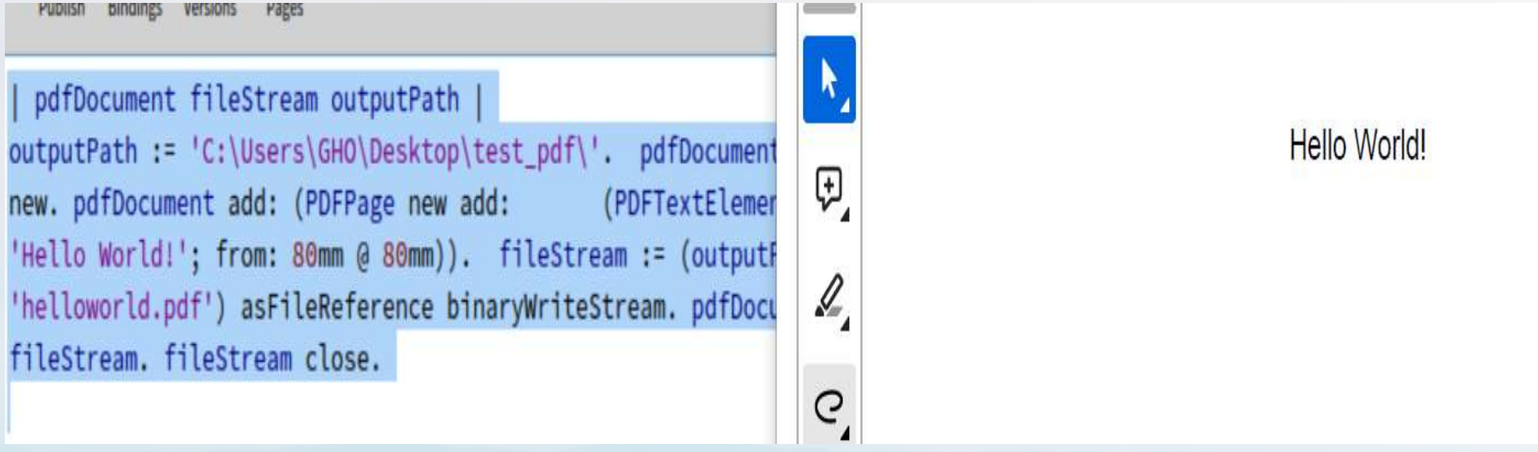
Couverture de test pour stream

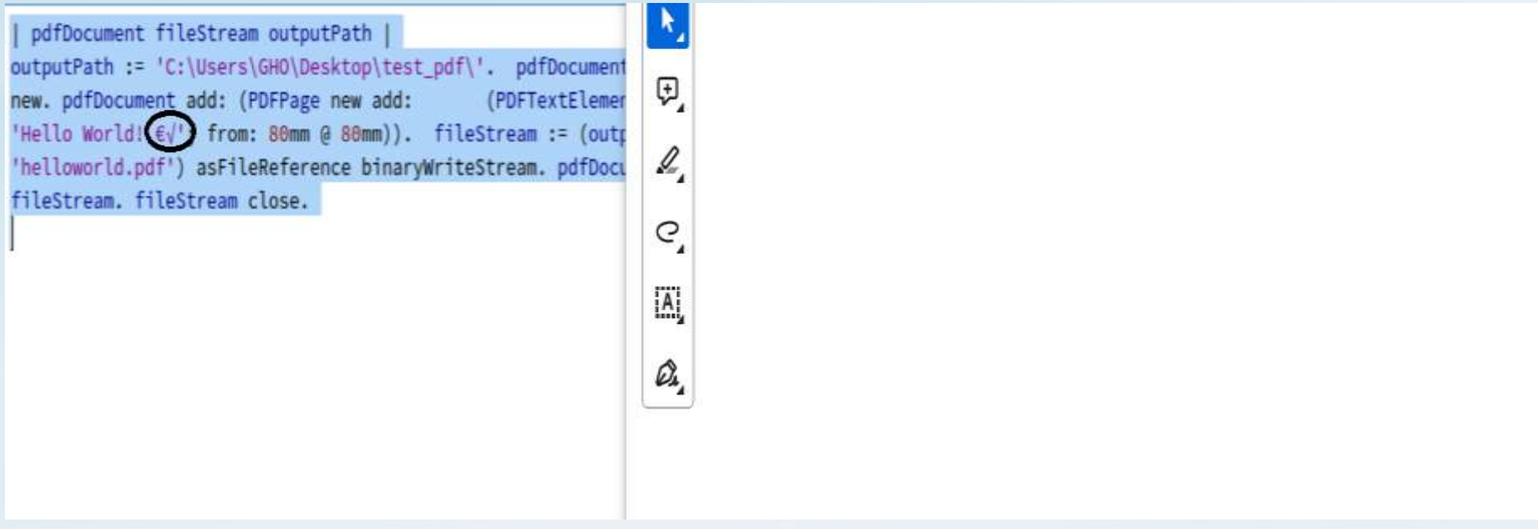


Test Coverage pour multilingual-Encoding



Test contre exemple





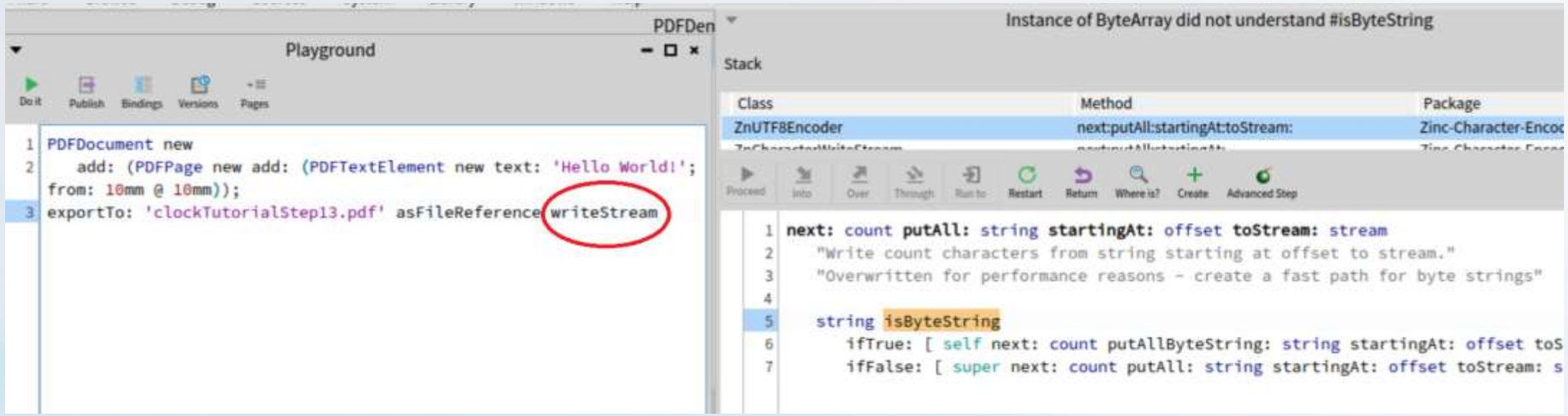
Outils et documentations pour Artefact

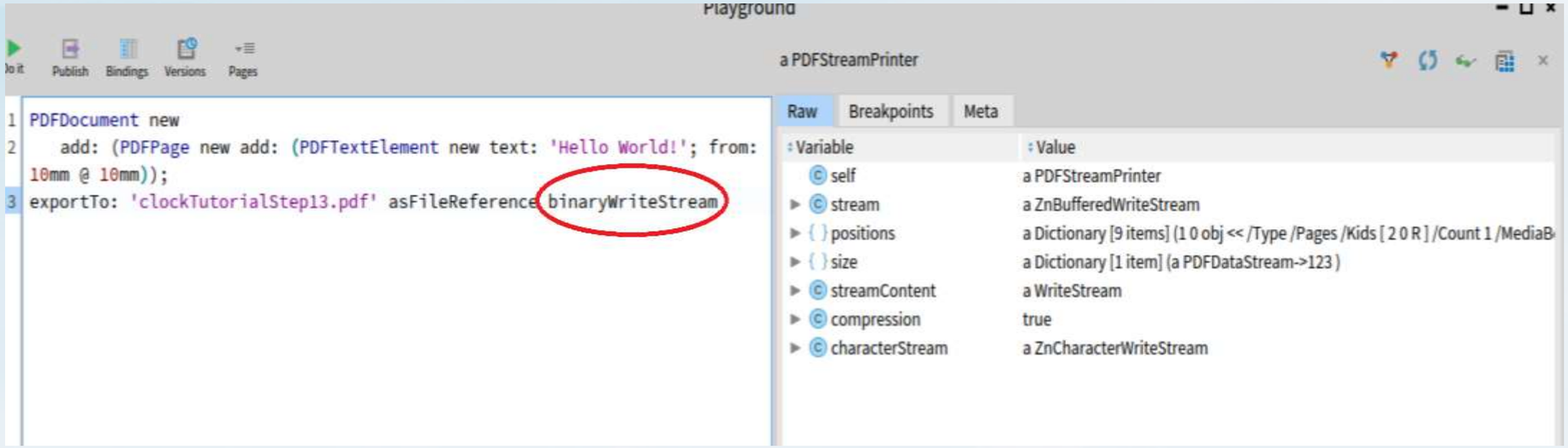
- Documentation sur GITHUB.
- Aide des amis en classe.
- Tests et commentaires très utiles.

Class: PDFPage

```
A PDFPage is a page of a PDFDocument. A page is composed by a collection of PDFElements.

Instance Variables
document:    <Object>
elements:    <Object>
format:      <Object>
margins:     <Object>
```

Critique


Playground
Do it
Publish
Bindings
Versions
Pages
PDFDocument new
add: (PDFPage new add: (PDFTextElement new text: 'Hello World!'; from: 10mm @ 10mm));
exportTo: 'clockTutorialStep13.pdf' asFileReference writeStream
Instance of ByteArray did not understand #isByteString
Stack
Class
Method
Package
ZnUTF8Encoder
next:putAll:startingAt:toStream:
Proceed
Into
Over
Through
Run to
Restart
Return
Where is?
Create
Advanced Step
next: count putAll: string startingAt: offset toStream: stream
"Write count characters from string starting at offset to stream."
"Overwritten for performance reasons - create a fast path for byte strings"
string isByteString
ifTrue: [self next: count putAllByteString: string startingAt: offset toS
ifFalse: [super next: count putAll: string startingAt: offset toStream: s


Playground
Publish
Bindings
Versions
Pages
PDFDocument new
add: (PDFPage new add: (PDFTextElement new text: 'Hello World!'; from: 10mm @ 10mm));
exportTo: 'clockTutorialStep13.pdf' asFileReference binaryWriteStream
a PDFStreamPrinter
Raw
Breakpoints
Meta
Variable
Value
self
a PDFStreamPrinter
stream
a ZnBufferedWriteStream
positions
a Dictionary [9 items] (1 0 obj << /Type /Pages /Kids [2 0 R] /Count 1 /MediaB
size
a Dictionary [1 item] (a PDFDataStream->123)
streamContent
a WriteStream
compression
true
characterStream
a ZnCharacterWriteStream

Outils et documentations pour AVL

- Utilisation de Wikipédia.
- Les tutoriels YouTube.
- Tests utiles

Conclusion

- Importance de se fixer un but sur lequel partir quand on connais pas le code.
- Importance des mutations testing et couverture du code.
- Les streams sont une structure de données importantes .
- L'équilibrage d’un arbre AVL est assuré par des rotations simples et doubles.
- Application futures de ce qu’on a appris.