

PRÉSENTATION

AVL ET ARTEFACT

Présenté par :

- Skander Ayadi
- Mohamed Edem Ghariani
- Abdellatif KEBRAOUI
- Abdelali Rhofir

Plan

ARBRE AVL

1.PRÉSENTATION DU PROJET

2.IMPLEMENTATION ET UTILISATION

3.TESTS

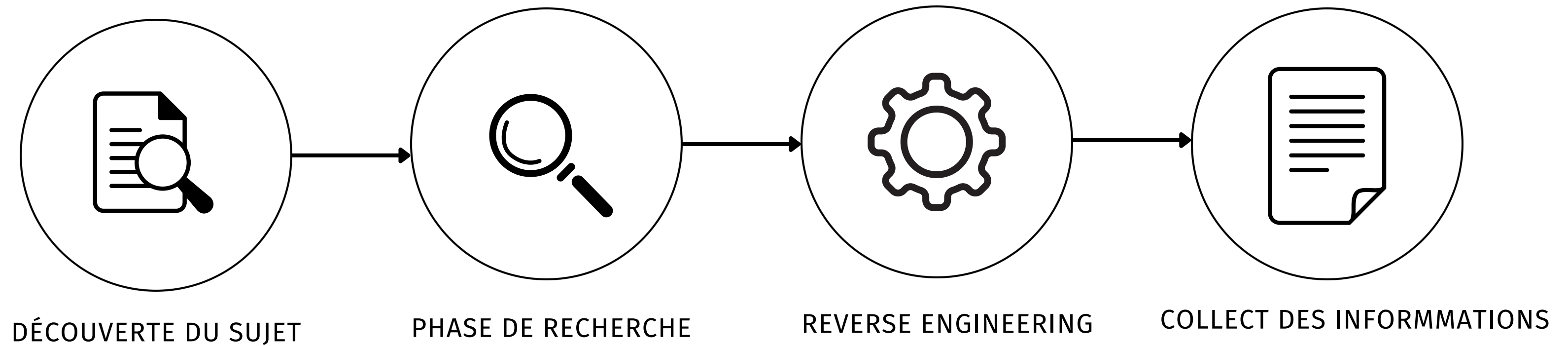
ARTEFACT

1.PRÉSENTATION DU PROJET

2.IMPLEMENTATION ET FONCTIONNALITÉ

3.TESTS

CONCLUSION



- Le projet implémente un arbre AVL en pharo
- Offre plusieurs fonctionnalités sur un arbre AVL
- Classes clés comme : AVLNode / AVLNilNode
- Structure de données : Classe AVLTree est une sous classe de Collection



- Classes et méthodes non commentées
- Pas d'exemples expliquant les fonctionnalités du projet
- Pas de starting points dans le README du projet

Arbre AVL

IMPLEMENTATIONS ET TESTS

Playground

an AVLTree [9 items] (5 5 1...

Do it Publish Bindings Versions Pages

```
1 |myTree myTreeVis |
2
3 myTree := AVLTree new.
4
5 myTree add: 5.
6 myTree add: 10.
7 myTree add: 20.
8 myTree add: 30.
9 myTree add: 35.
10 myTree add: 40.
11 myTree add: 50.
12 myTree add: 60.
13 myTree add: 5.
14
15
16 myTree.
17
18
19
```

Items AVL Raw Breakpoints Meta

```
graph TD
    30((30)) --> 10((10))
    30 --> 40((40))
    10 --> 5_1((5))
    10 --> 20((20))
    40 --> 35((35))
    40 --> 50((50))
    5_1 --> 5_2((5))
    50 --> 60((60))
```

1 self

Line: 19:1

- Insertion des noeuds
- Insertion d'un duplicata

Arbre AVL

IMPLEMENTATION ET TESTS

Playground

an AVLTree [8 items] (5 10 ...)

Do it Publish Bindings Versions Pages

```
1  
2 myTree remove: 5 ifAbsent: [ Transcript show:  
3   'Error' printString ].  
4  
5 myTree
```

Items AVL Raw Breakpoints Meta

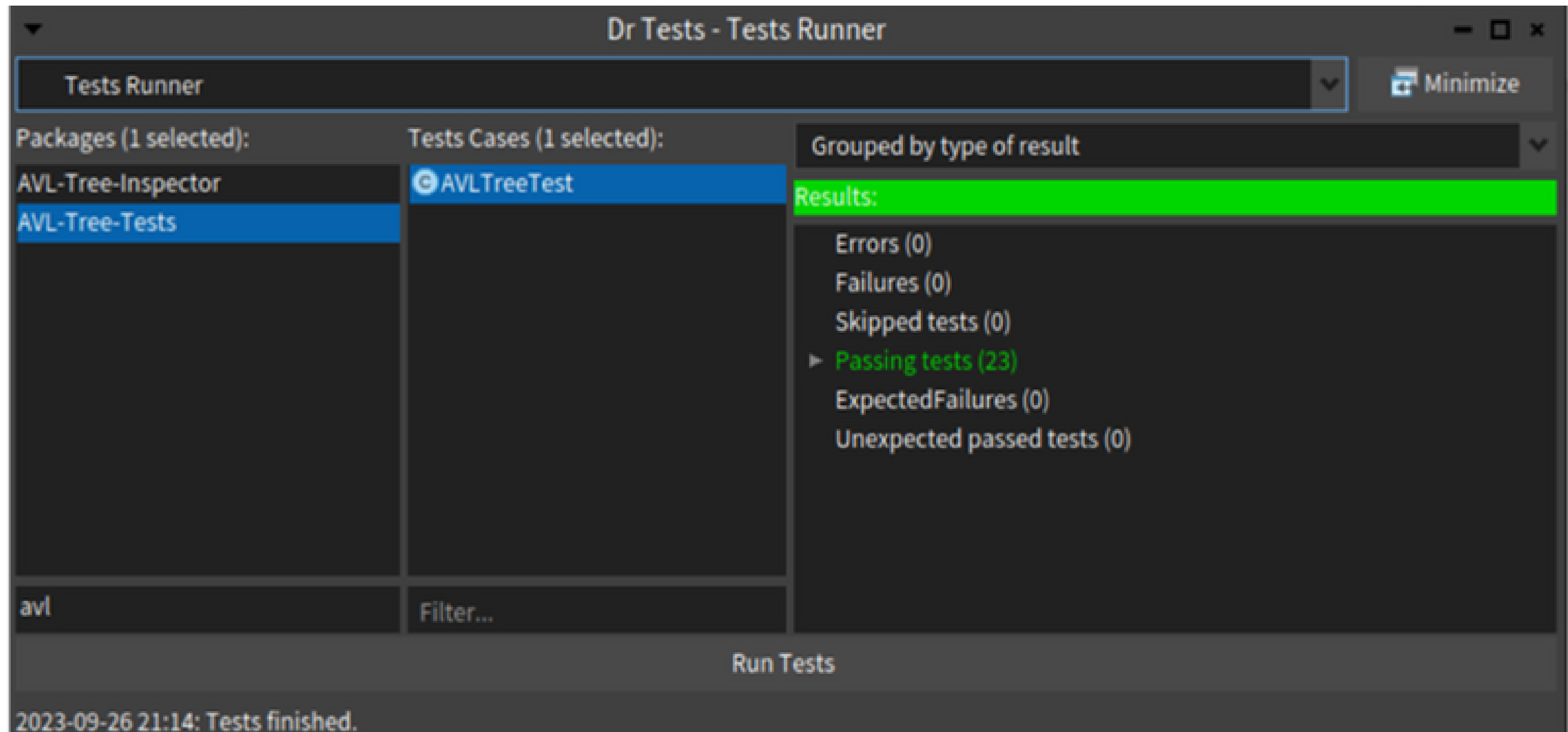
```
graph TD
    30((30)) --> 10((10))
    30 --> 40((40))
    10 --> 5((5))
    10 --> 20((20))
    40 --> 35((35))
    40 --> 50((50))
    50 --> 60((60))
```

1 self

Line: 9:1

- Suppression du noeud 5.
- Un seul 5 a été supprimer

- 23 Tests en **vert**



Arbre AVL

Les Tests

The screenshot displays the AVLTreeInspectorTest IDE interface. The top bar shows the title 'AVLTreeInspectorTest>>testCreateCanvas'. The left sidebar contains a project tree with the following structure:

- AVL-Tree
 - AVL-Tree-Inspector
 - Extensions
 - AVL-Tree-Tests
 - BaselineOfAVLTree

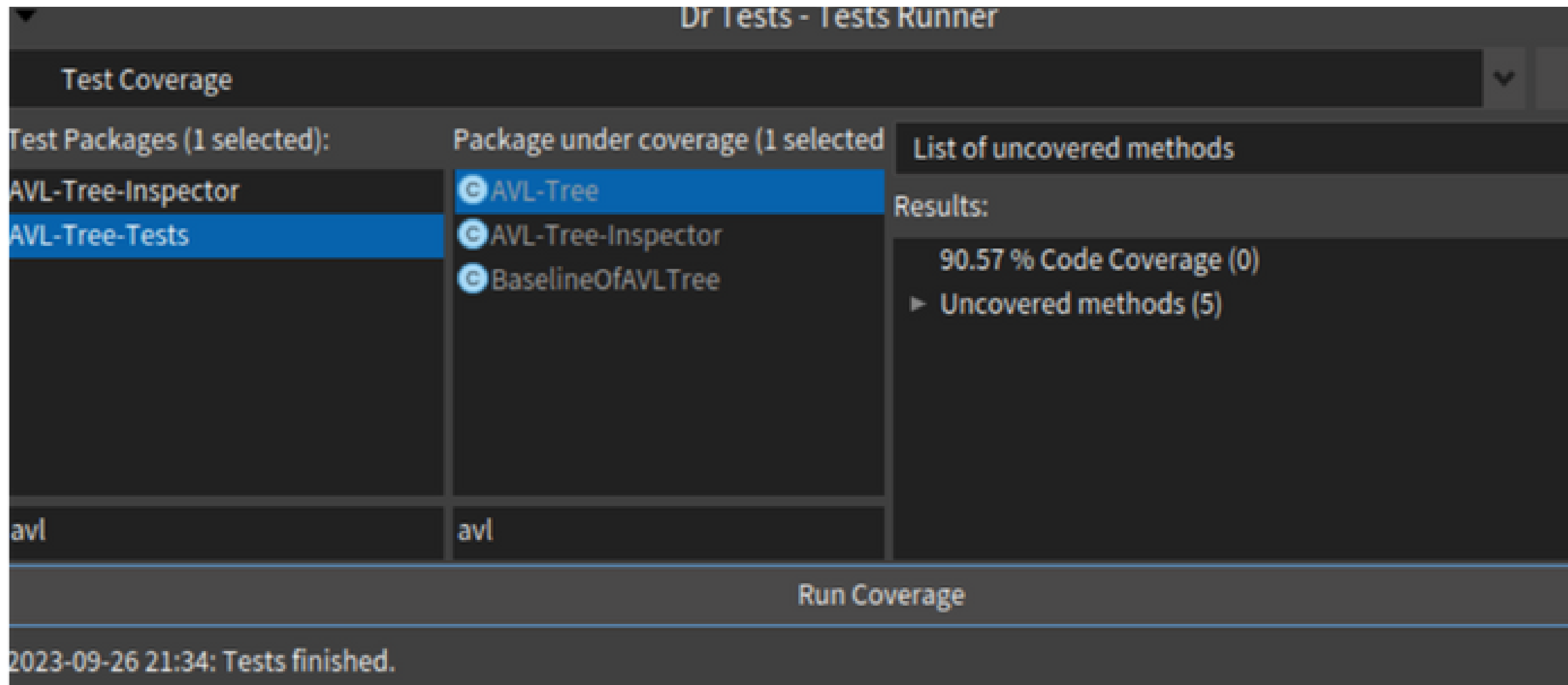
The main editor area is divided into three panes. The left pane shows the class hierarchy for AVLTreeInspectorTest, including AVLTreeVisualizer and AVLTree. The middle pane shows the 'instance side' view with a 'tests' section. The right pane shows the 'testCreateCanvas' method.

The bottom status bar displays the following information:

- Test case subclass: #AVLTreeInspectorTest
- instanceVariableNames: ''
- classVariableNames: ''
- package: 'AVL-Tree-Inspector'

A warning message at the bottom indicates: 'Test class not in a package with name ending with '-Tests' X ?'.

- Les tests couvrent 90.57% du code.





- **Il y a 5 tests manquants:**

`AVLNode>>#balance: path:`

`AVLNode>>#children`

`AVLNode>>#printOn`

`AVLAbstractNode>>#isNilNode`

`AVLAbstractNode>>#children`



- Outil de génération de document PDF.
- Construction de PDF par des éléments.
- Feuilles de style réutilisables.
- Compression.

Artefact

Point de vue de l'utilisateur



- Les classes ont plus ou moins des commentaires mais pas toutes.
- Présence d'un README clair.
- Les starting points sont très utiles.
- Le projet comporte des exemples.

LES CLASSE CLÉS:

```
| pdfdoc aPage |
pdfdoc := PDFDocument new.
aPage := PDFPage new.

aPage
  add:
    ((PDFCircleElement from: 50 mm @ 10 mm to: 150 mm @ 180 mm)
      fillColor: (PDFColor r: 255 g: 0 b: 0);
      drawColor: (PDFColor r: 255 g: 0 b: 0)).

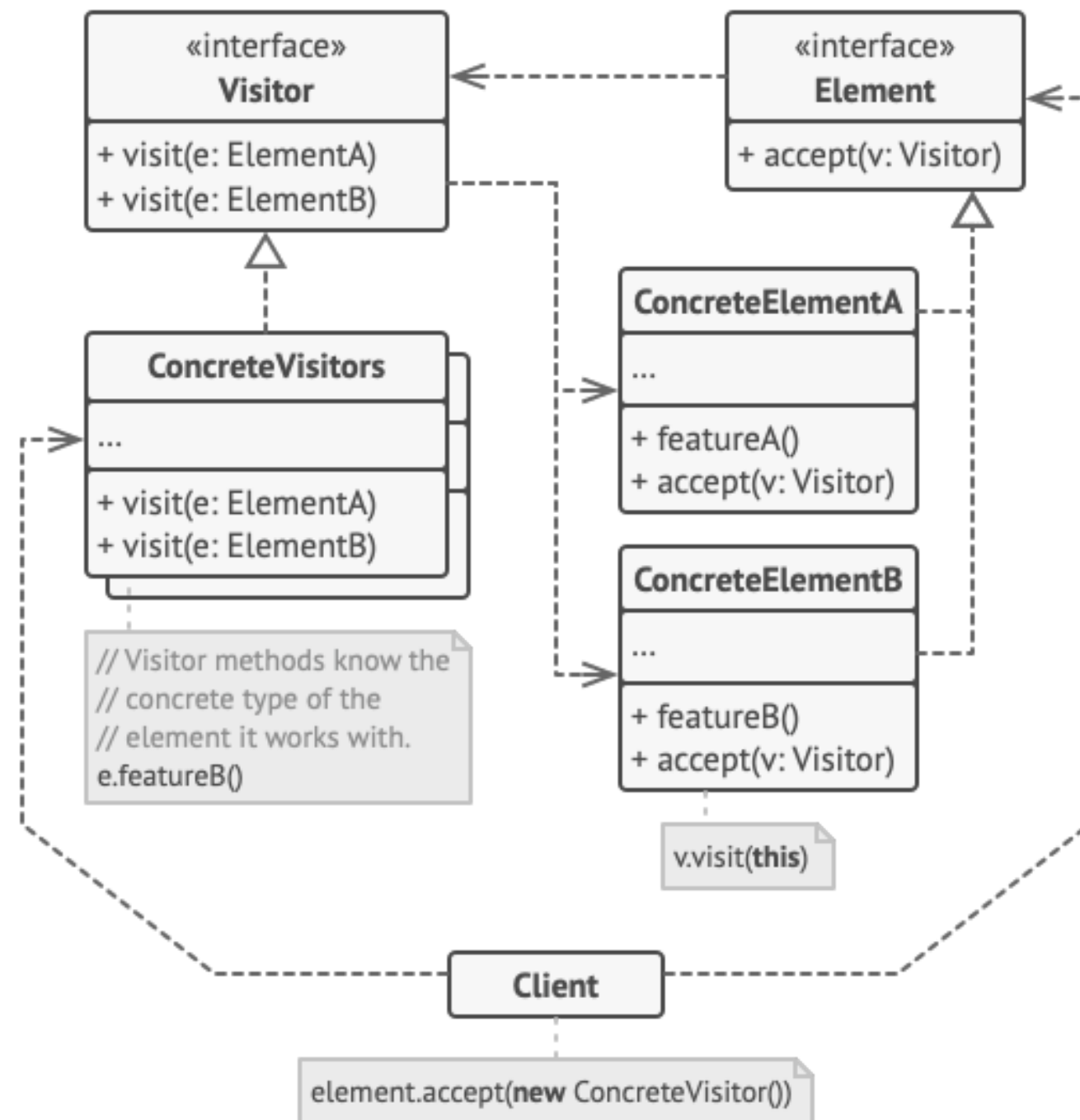
aPage
  add:
    ((PDFArrowElement from: 50 mm @ 50 mm to: 100 mm @ 30 mm)
      drawColor: (PDFColor r: 0 g: 0 b: 255)).

aPage add: (PDFParagraphElement new
  from: 90 mm @ 40 mm;
  dimension: 10 mm @ 20 mm;
  text: '1 Émensis itaque difficultatibus multis et nive obrutis callibus plurimis ubi prope Rauracum ventum est ad supercilia fluminis Rheni, resistente
multitudine Alamanna pontem suspendere navium conpage Romani vi nimia vetabantur ritu grandinis undique convolantibus telis, et cum id impossibile videretur, imperator
cogitationibus magnis attonitus, quid capesseret ambigebat.').

pdfdoc exportTo: 'test8.pdf' asFileReference binaryWriteStream
```



- PDFDocument: Classe qui représente le document a générer
- PDFPage : Classe qui représente une page dans un document
- PDFElement : Classe qui représente un element dans une page
- PDFStreamPrinter: Classe qui gère l'impression du PDF.



Patrons de design

- On a reconnu le patron du visiteur. le projet repose sur le double (voire multiple) dispatch pour implémenter différentes fonctionnalités.

Artefact

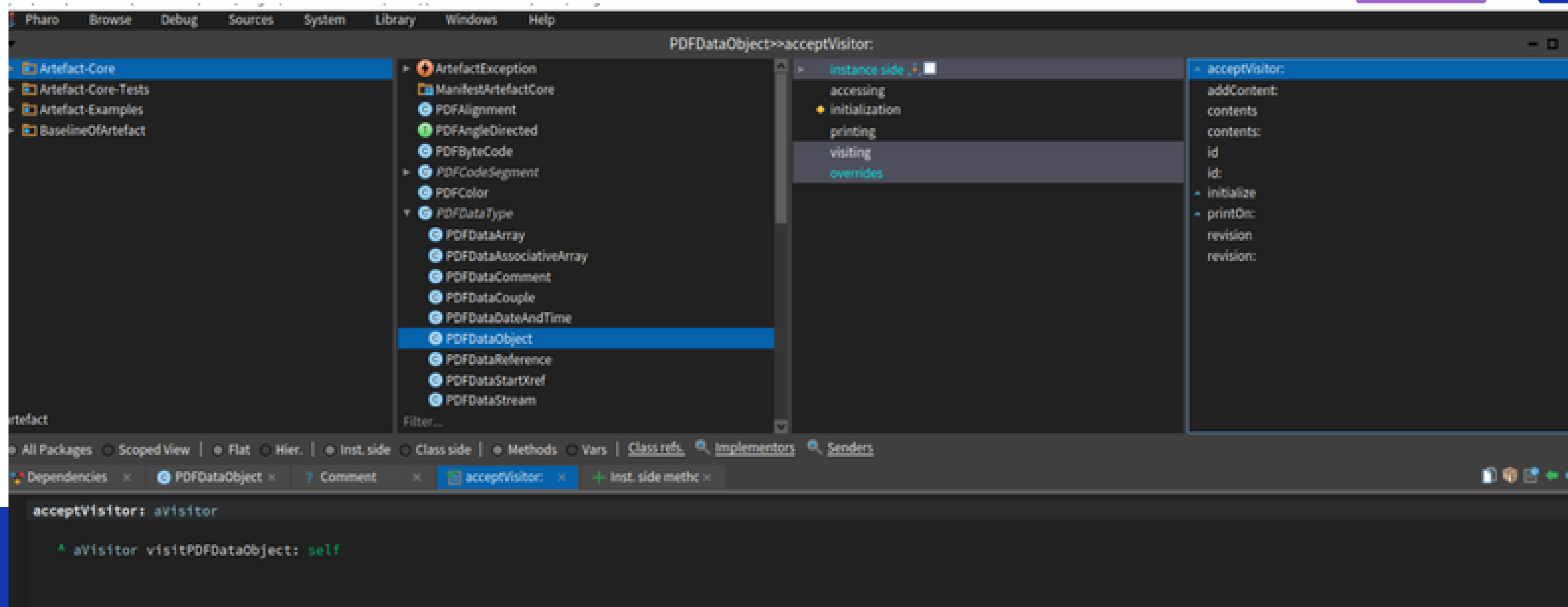
Point de vue de l'implémentation

The screenshot displays the Pharo IDE interface. The top menu bar includes 'Pharo', 'Browse', 'Debug', 'Sources', 'System', 'Library', 'Windows', and 'Help'. The left sidebar shows a project structure with folders: 'Artefact-Core', 'Artefact-Core-Tests', 'Artefact-Examples', and 'BaselineOfArtefact'. The central pane shows a class hierarchy for 'PDFStreamPrinter', with methods like 'PDFDataXObjectStream', 'PDFDataRef', 'PDFDataRefEntry', 'PDFDocument', 'PDFDotted', 'PDFElement', 'PDFFont', 'PDFFormat', 'PDFGenerator', 'PDFMetadata', 'PDFOpacity', 'PDFPage', 'PDFStreamPrinter' (selected), 'JPGReadWriter', 'StyleSheet', and 'UnitValue'. The right pane shows the 'instance side' of the 'visitPDFDataObject:' method, listing various methods such as 'accessing', 'compression', 'converting', 'initialization', 'print', 'utilities', 'visiting', and 'overrides'. The bottom pane shows the implementation of the 'visitPDFDataObject:' method in the 'PDFStreamPrinter' class, with the following code:

```
visitPDFDataObject: aPDFDataObject
    self positions at: aPDFDataObject put: stream position.
    characterStream
        nextPutAll: aPDFDataObject id asString;
        space;
        nextPutAll: aPDFDataObject revision asString;
        space;
        nextPutAll: 'obj';
        lf.
    aPDFDataObject contents
        do: [ :aContent |
            aContent isPrintable
                ifTrue: [ aContent printOn: characterStream ]
```


Artefact

Point de vue de l'implémentation



Artefact

Point de vue de l'implémentation

```
producePageElementCodeWith: aPDFGenerator styleSheet: anObject
"Draw a Bezier curve from xy to destination (3 control points)"

^ String
  streamContents: [ :s |
    s nextPutAll: (self moveTo: self from with: aPDFGenerator).
    self points
    do: [ :p |
      s
        nextPutAll: (self splitCoordinates: (aPDFGenerator determinePositionOnCurrentPage: self from + p));
        space ];
    s
  ]
```

```
producePageElementCodeWith: aPDFGenerator styleSheet: aStyleSheet
"Draw a polygon from xy to destination (a array of positions)"

^ String
  streamContents: [ :s |
    s nextPutAll: (self moveTo: self from with: aPDFGenerator).
    self points , {self points first}
    do: [ :p |
      s
        nextPutAll: (self splitCoordinates: (aPDFGenerator determinePositionOnCurrentPage: self from + p));
        space ];
    s
  ]
```

```
producePageElementCodeWith: aPDFGenerator styleSheet: aStyleSheet
"Draw a rectangle from xy with a specified dimension"

| position |
position := aPDFGenerator determinePositionOnCurrentPage: self from.

^(String streamContents: [ :s |
  s nextPutAll: (self splitCoordinates: position);
  space;
  print: (aPDFGenerator convertToPoints: self dimension x);
  space;
])
```

PDFBezierCurveElement

PDFPolygonCurveElement

PDFRectElement

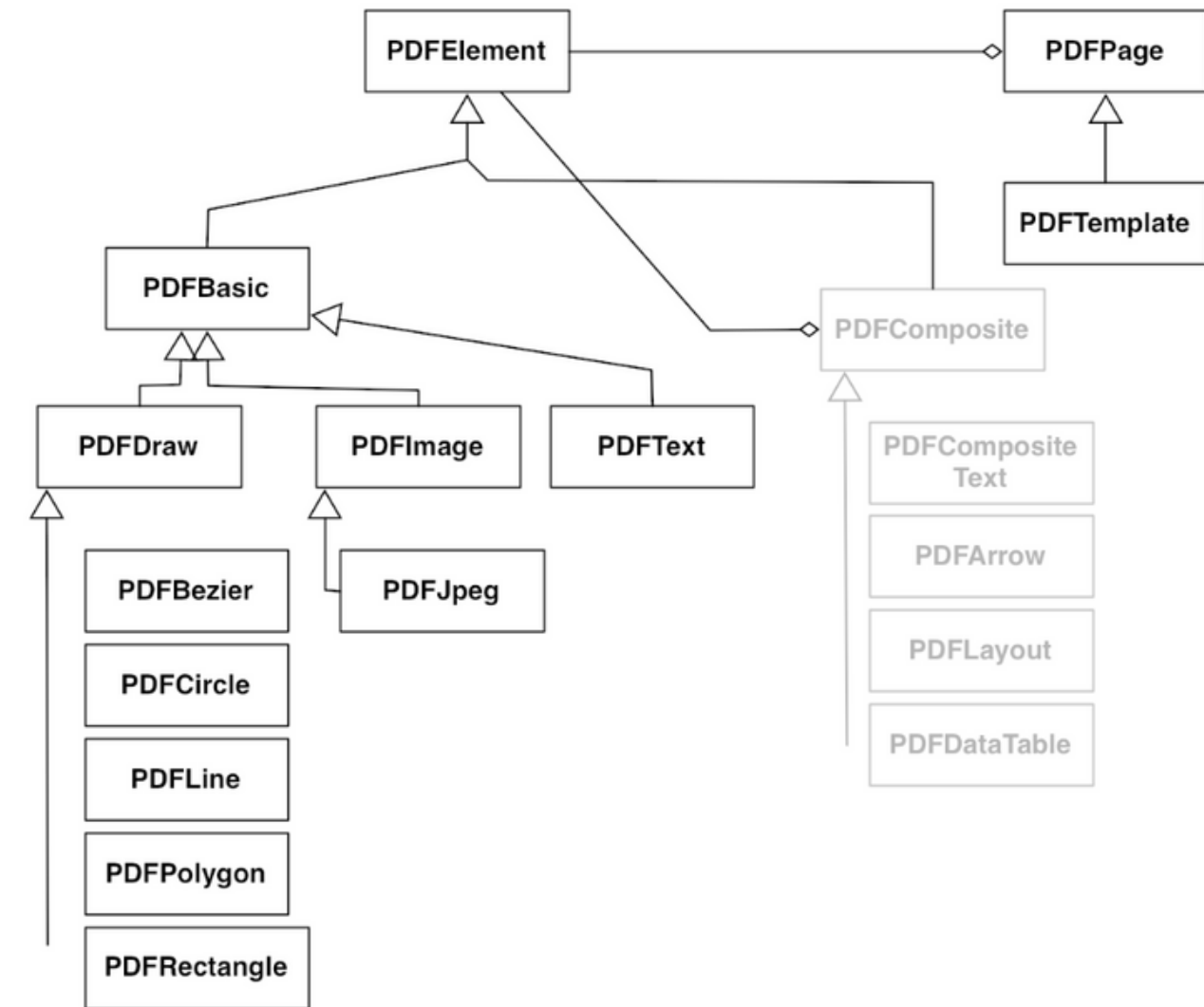
```
generateCodeSegmentWith: aPDFGenerator styleSheet: aStyleSheet format: aFormat

^ self codeSegmentClass
  code: (self producePageElementCodeWith: aPDFGenerator styleSheet: aStyleSheet)
  styleSheet: aStyleSheet
  format: aFormat
  isDrawElement: self isDrawElement
  fontId: ([[aPDFGenerator getFontIdFor: aStyleSheet font]] on: KeyNotFound do: [nil])
  opacityId: ([[aPDFGenerator getOpacityIdFor: aStyleSheet opacity]] on: KeyNotFound do: [nil])
  fromElement: self
```

- Lecture du code plus simple
- Possibilité de rajouter d'autres formes sans créer de bug

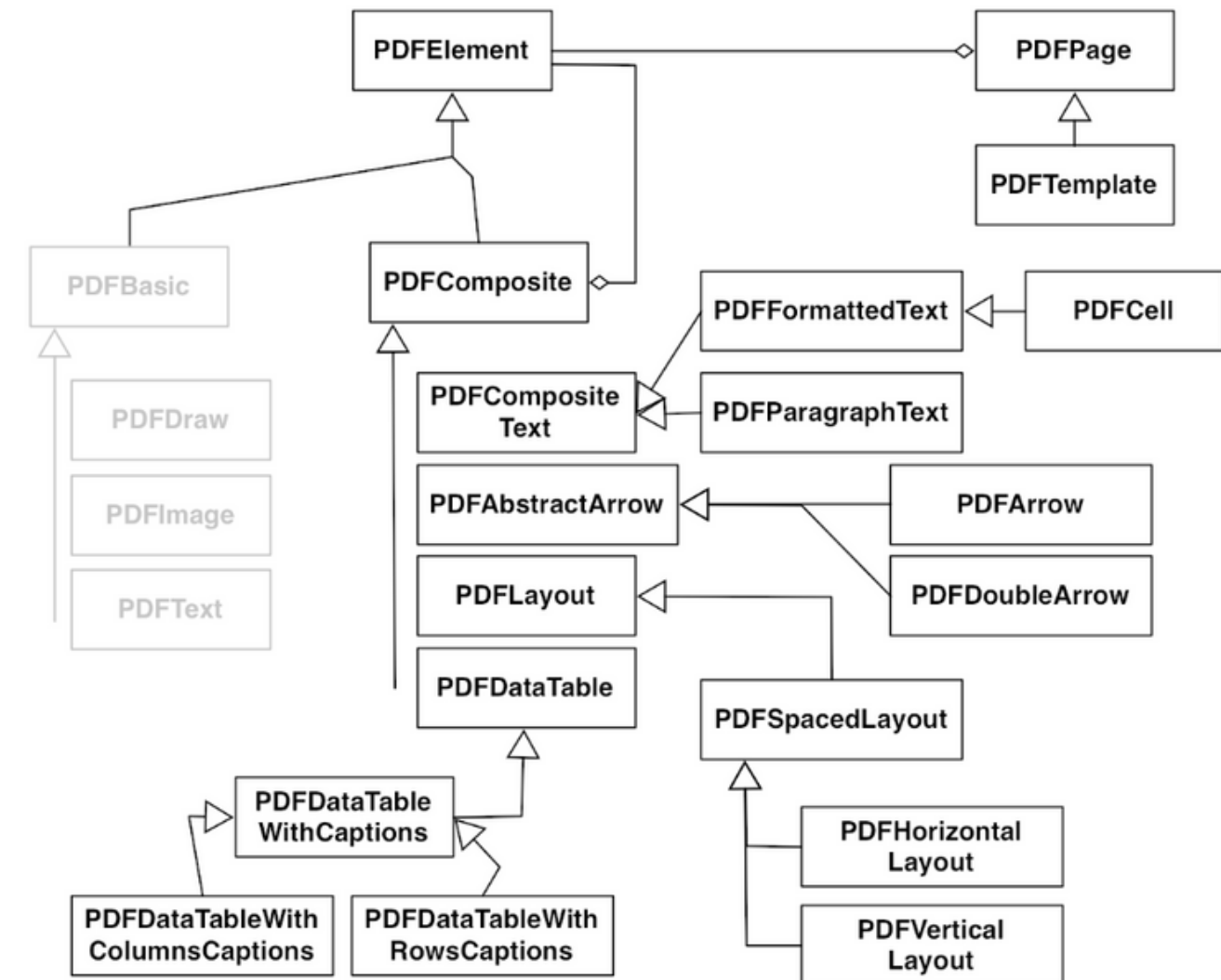
Artefact

Point de vue de l'implémentation



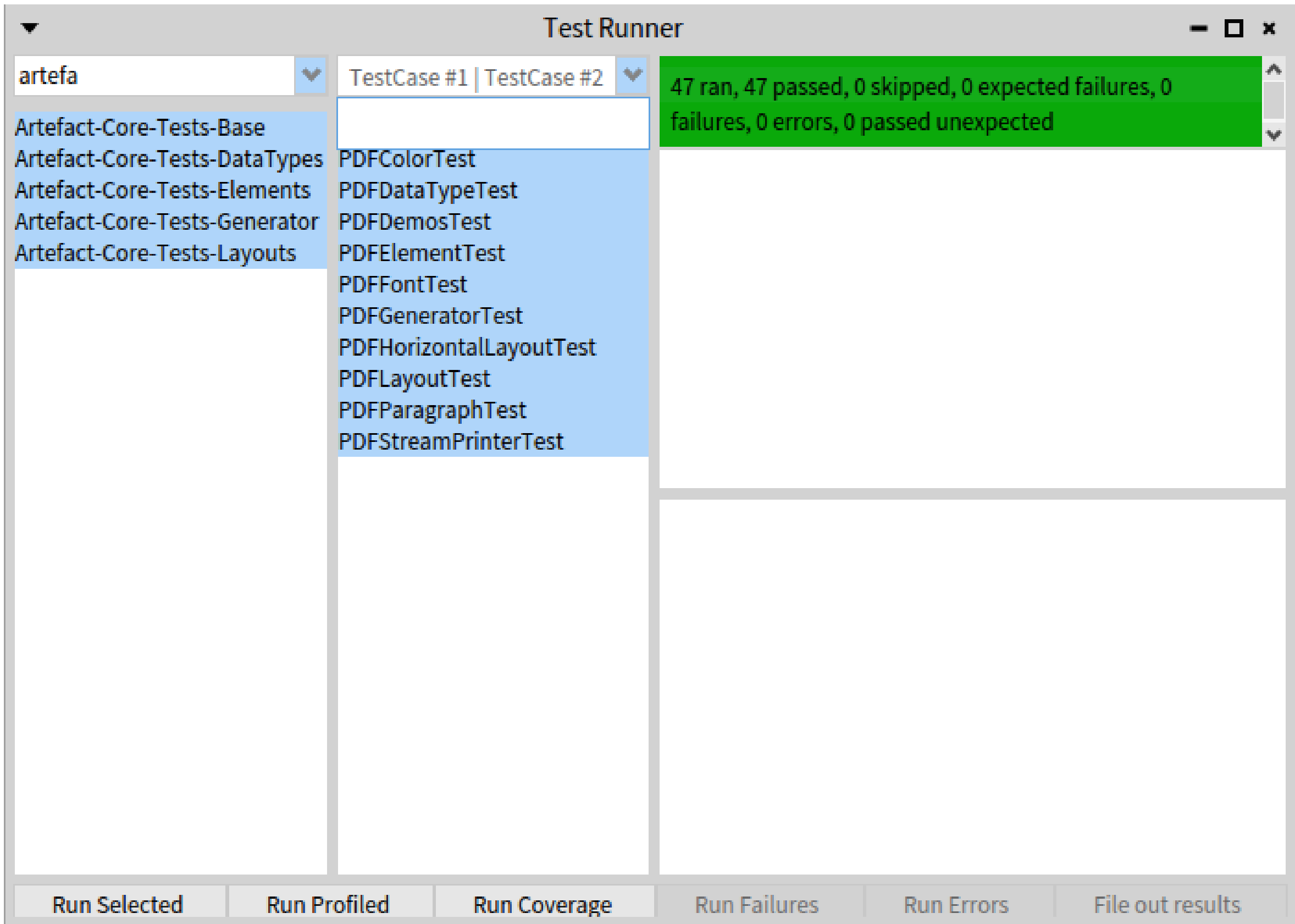
ÉLÉMENTS SIMPLES

ÉLÉMENTS COMPOSITES

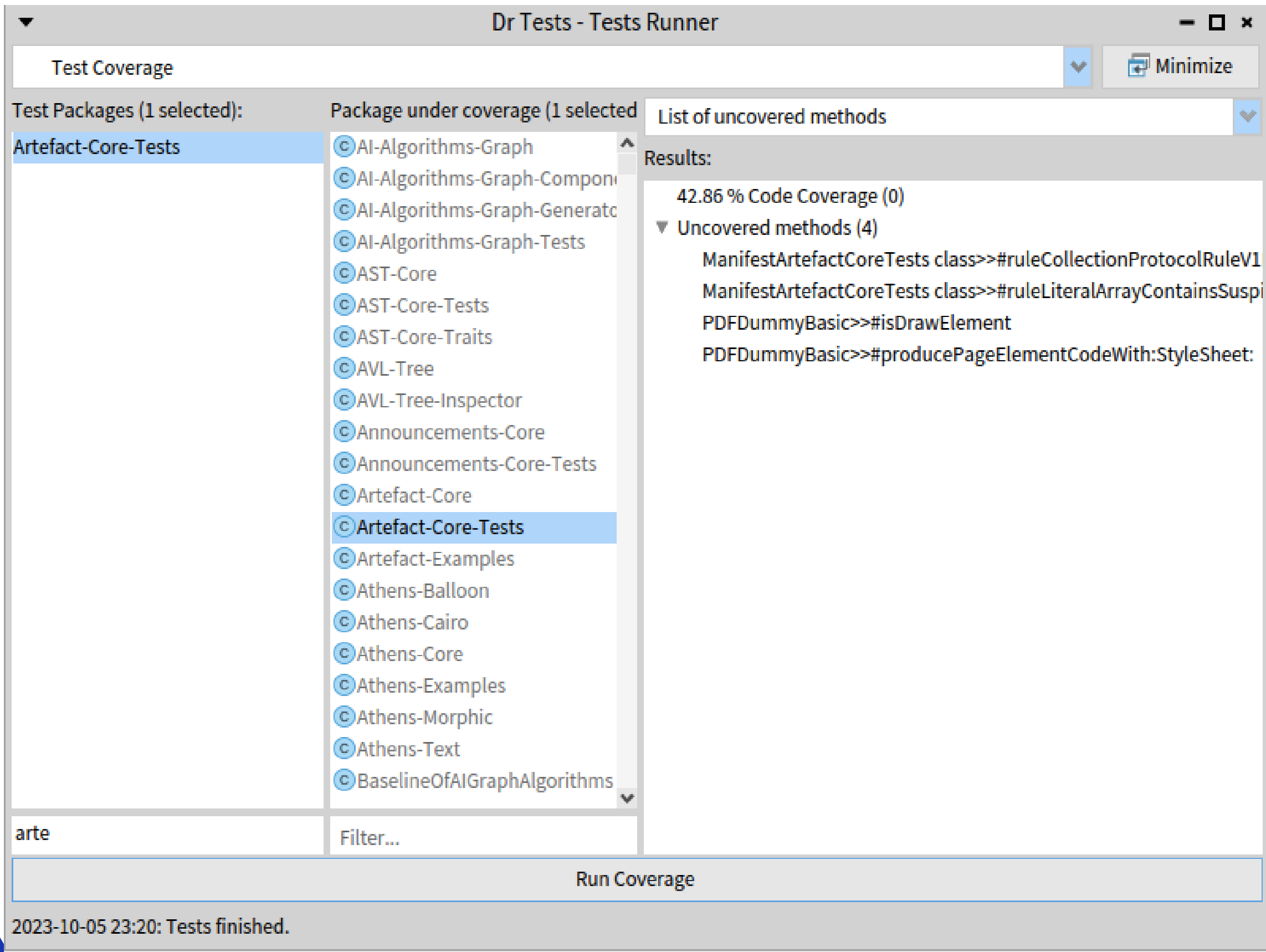


Artefact

Les Tests



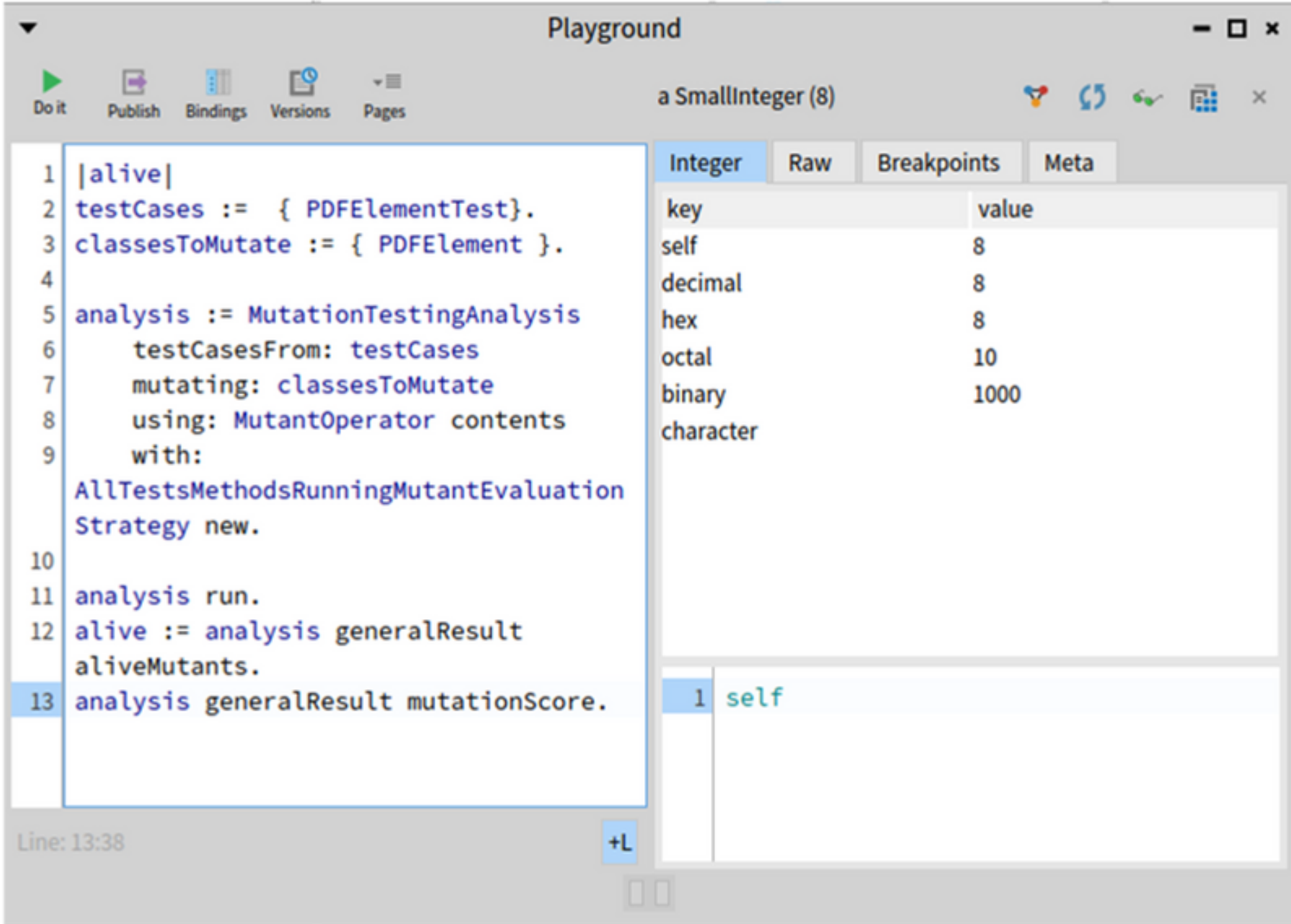
47 tests en vert



Taux de Couverture
faible : 42.86%

Artefact

Les Tests



The screenshot shows the Scala Playground interface. The left pane contains Scala code for a mutation testing analysis. The right pane shows the results of the analysis, including a table of mutation scores for various input types and a list of surviving mutants.

```
1 |alive|
2 |testCases := { PDFElementTest }.
3 |classesToMutate := { PDFElement }.
4
5 |analysis := MutationTestingAnalysis
6   |testCasesFrom: testCases
7   |mutating: classesToMutate
8   |using: MutantOperator contents
9   |with:
10  |AllTestsMethodsRunningMutantEvaluation
11  |Strategy new.
12
13 |analysis run.
14 |alive := analysis generalResult
15 |aliveMutants.
16 |analysis generalResult mutationScore.
```

key	value
self	8
decimal	8
hex	8
octal	10
binary	1000
character	

1 self

- Un test de mutation sur la classe `PDFElement` a montré un score de mutation faible, seulement 8%
- Il reste 124 mutants survivants sur 135.

Il faut améliorer les tests de la classe `PDFElementTest`

© self

135 mutants, 11 killed, 124 alive, 0 terminated. Mutation Score: 8%.

Conclusion

- Utilisation du reverse engineering
- Savoir parcourir le code
- Travail d'équipe
- Des notions sur les AVL