Présentation des méthodes d'apprentissage et analyse de structures de données

Containers-Grid

Louis Deloffre Sabrina Kernouf Romain Morel

SLR

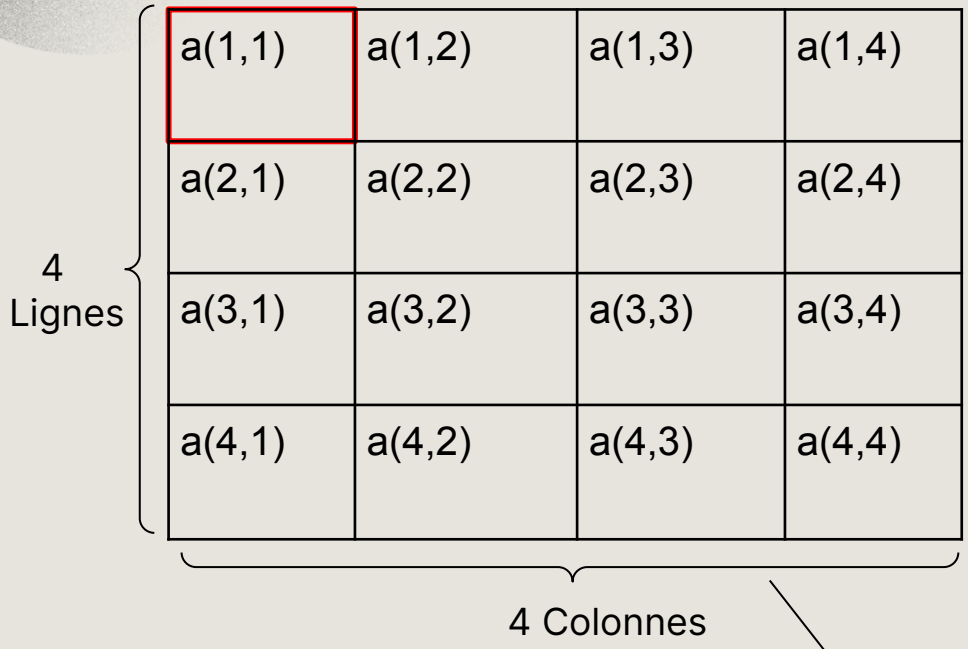
Sommaire

Apprentissage

Autonomie

Formation

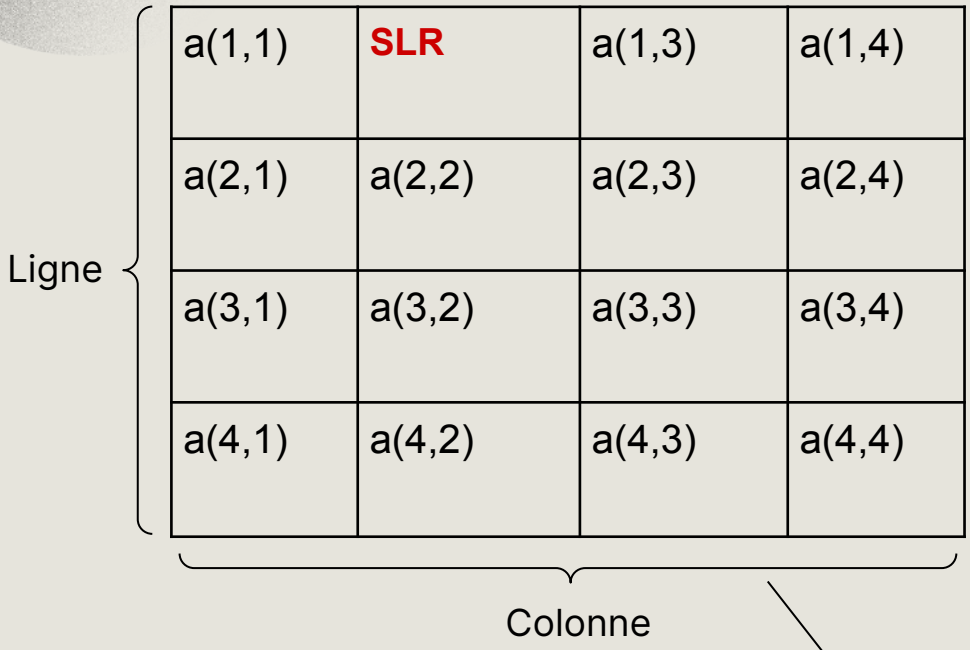
Principes



- paramétrable
  - dimensions

Syntaxe : gridClass rows: 4 columns: 4

Principes



- **Les variables**

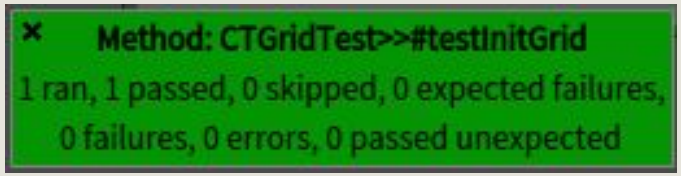
Syntaxe : grid1 atRow: 1 atColumn:2 put: “SLR”.

Principes

Initialiser un tableau

- gridClass new: 2.
- gridClass rows: 3 columns: 2.
- gridClass new: 3 element: -1.
- gridClass rows: 3 columns: 2 tabulate: [:row :column | column * 10 + row]
- gridClass withColumns: #(#('A Time to Kill' 'Blood and Smoke') ('John Grisham' 'Stephen King') ('100' '1000')).
- gridClass withRows: #(#('A Time to Kill' 'John Grisham' '100') #('Blood and Smoke' 'Stephen King' '1000')).

```
testInitGrid
    grid1 := self gridClass new: 2.
    self assert: (grid1 atRow: 1 atColumn: 1) equals: nil.
```



nil	nil
nil	nil

Principes

Initialiser un tableau

- **gridClass rows: 3 columns: 2 tabulate: [:row :column | column * 10 + row]**

```
grid3 := self gridClass rows: 3 columns: 2 tabulate: [ :row :column | column * 10 + row ].
self assert: (grid3 atRow: 1 atColumn: 1) equals: 11.
```

f(x) = Column * 10 + row

	Column = 1	Column = 2
row = 1	11	21
row =2	12	22
row = 3	13	23

Principes

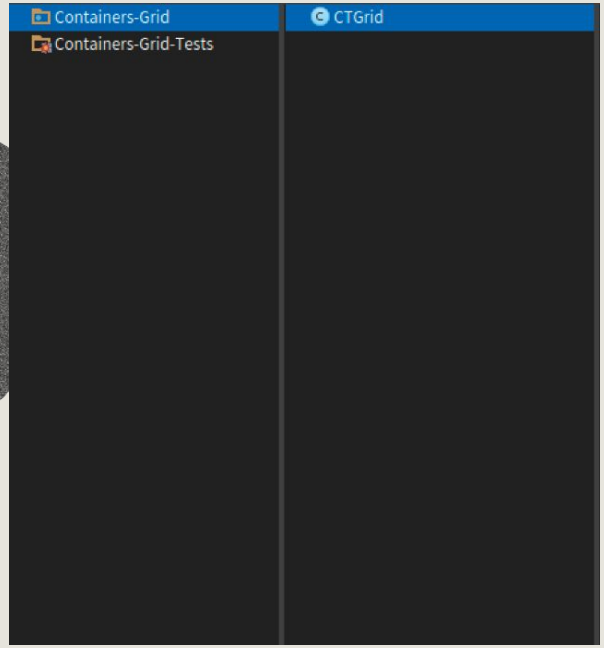
Initialiser un tableau

- **gridClass withColumns: #(#('A Time to Kill' 'Blood and Smoke') ('John Grisham' 'Stephen King') ('100' '1000')).**
- **gridClass withRows: #(#('A Time to Kill' 'John Grisham' '100') #('Blood and Smoke' 'Stephen King' '1000')).**

```
grid3 := self gridClass withColumns: #( #('A Time to Kill' 'Blood and Smoke') #('John Grisham' 'Stephen King') #( '100' '1000') ).
self assert: (grid3 atRow: 1 atColumn: 1) equals: 'A Time to Kill'.
self assert: (grid3 atRow: 1 atColumn: 2) equals: 'Blood and Smoke'.
self assert: (grid3 atRow: 1 atColumn: 3) equals: 'John Grisham'.

grid1 := self gridClass withRows: #( #('A Time to Kill' 'John Grisham' '100') #('Blood and Smoke' 'Stephen King' '1000')).
self assert: (grid1 atRow: 1 atColumn: 1) equals: 'A Time to Kill'.
self assert: (grid1 atRow: 1 atColumn: 2) equals: 'John Grisham'.
```

	Column = 1	Column = 2	Column = 3
row = 1	A Time to Kill	John Grisham	100
row = 2	Blood and Smoke	Stephen King	1000

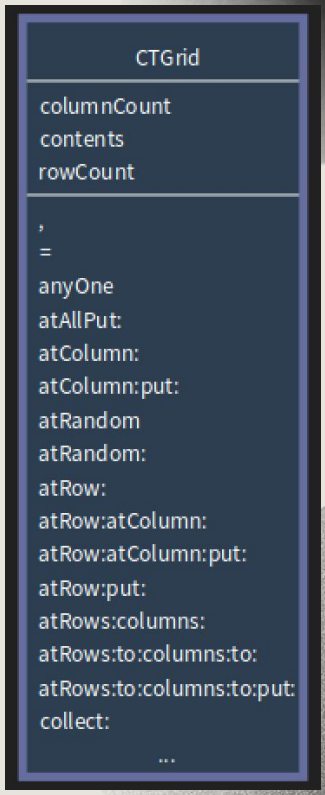
Découvertes



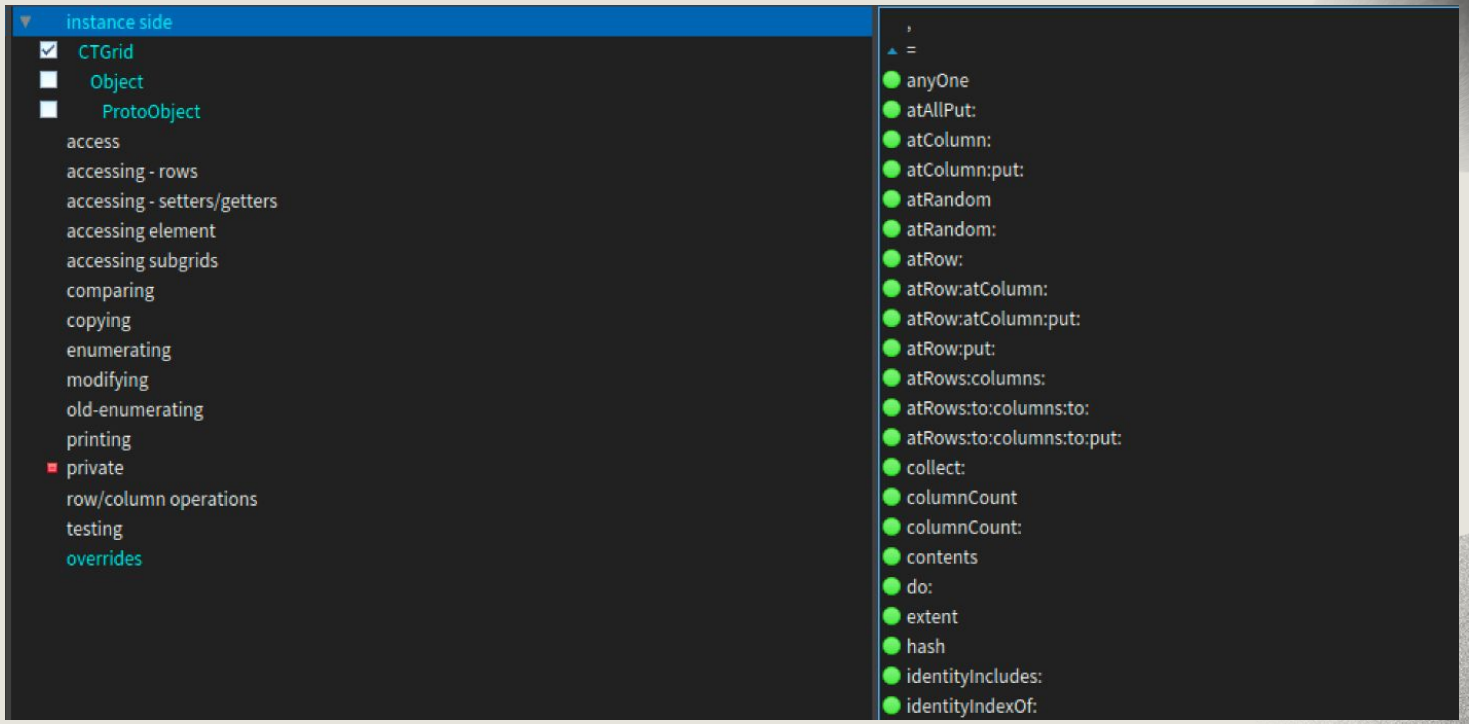
Paquetages

Classes

1 seule classe



Découvertes



Protocoles

Méthodes

Découvertes

atRandom

> Récupère un élément au hasard du tableau

```
atRandom
    ^ contents atRandom
```

```
testAtRandom
    | anInt |
    anInt := grid23 atRandom.

    self assert: (grid23 includes: anInt) equals: true.
```

Découvertes

atAllPut:

> Insère la valeur passée en paramètre dans toutes les cases du tableau

```
atAllPut: value
    contents atAllPut: value
```

```
testAtAllPut
    | m |
    m := self gridClass new: 3.
    m do: [ :each | self assert: each equals: nil ].
    m atAllPut: 1.
    m do: [ :each | self assert: each equals: 1 ]
```

Découvertes

shuffled

> Mélange les éléments du tableau

```
shuffled
    ^ self class rows: rowCount columns: columnCount contents: (contents shuffled)
```

```
testShuffled
        | gri |
    gri := self gridClass new.
    gri := self grid22 shuffled.

    self assert: (gri includes: 1) equals: true.
    self assert: (gri  includes: 2) equals: true.
    self assert: (gri includes: 3) equals: true.
    self assert: (gri includes: 4) equals: true
```

Découvertes

replaceAll:with:

> Remplace toutes les valeurs correspondantes d'un paramètre par une autre valeur (2ème paramètre)

```
replaceAll: oldObject with: newObject
    "Replace all occurrences of oldObject with newObject in the receiver."
    contents replaceAll: oldObject with: newObject
```

L'approche

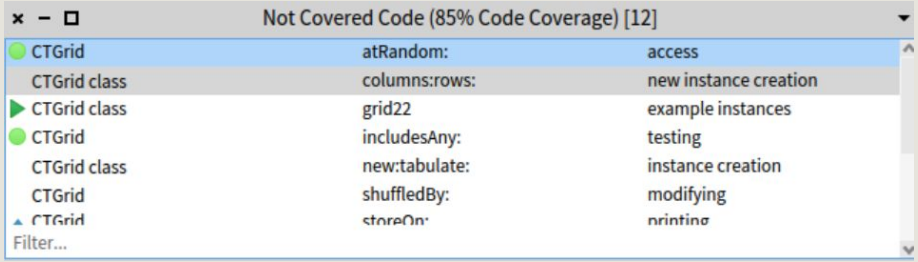
Commentaires :

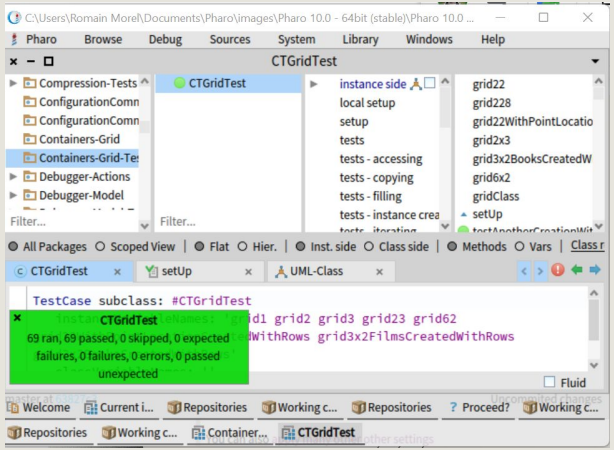
Class: CTGrid

```
I represent  a two-dimensional grid. I provide methods for creating grids, operating on them. A grid
origin is the left topmost corner, hence subsequent lines are located "below".

Examples
Here is a typical grid: 6 columns, 2 rows, growing down.
[[[
 CTGrid grid6x2CreatedWithRowsColumns printString
>>> '(
11 21 31 41 51 61
12 22 32 42 52 62
 )'
]]]
[[[
CTGrid  grid3x2BooksCreatedWithRows  printString
>>>
'(
''A Time to Kill'' ''John Grisham'' 1989
''Blood and Smoke'' ''Stephen King'' 2010
''Spin'' ''Robert Charles Wilson'' 2006
)'
]]]
```

L'approche

Tests

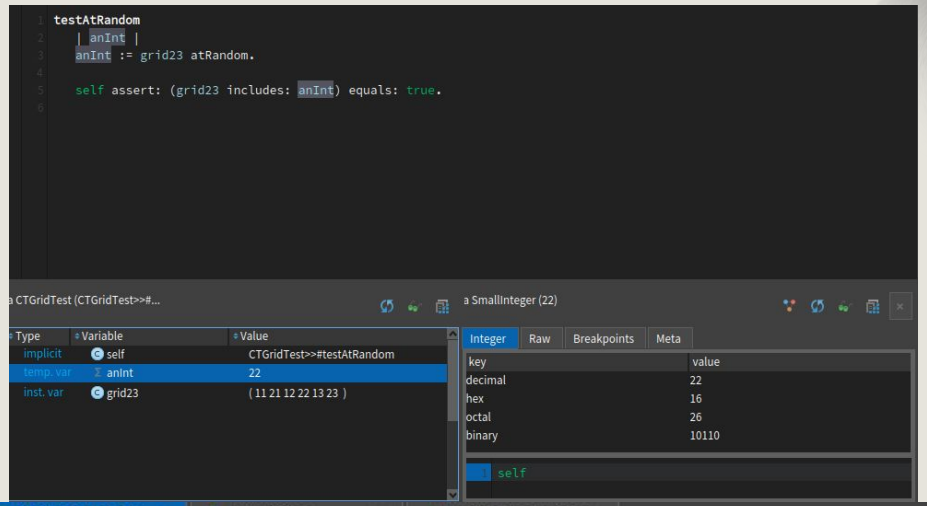


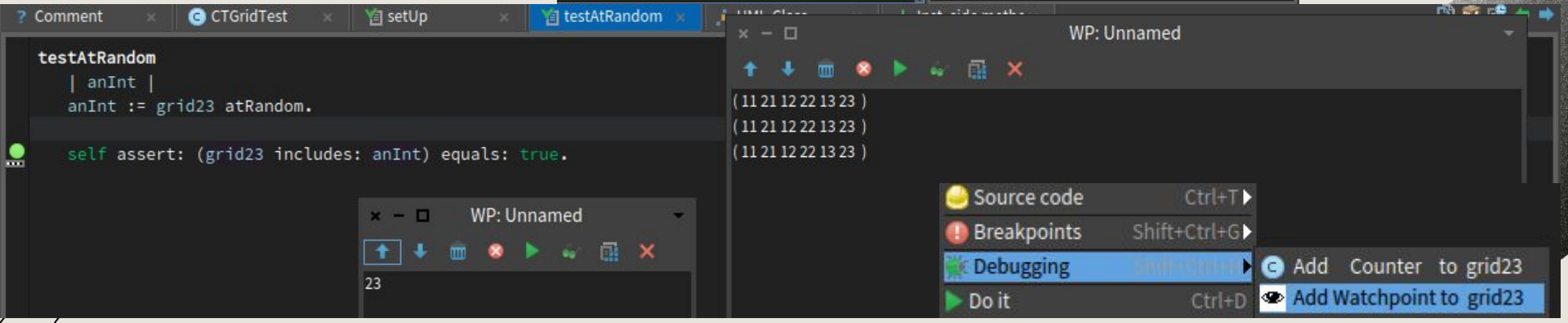



```
setUp
    self grid22.
    self grid228.
    self grid2x3.

    "new setup creation"
    self grid6x2.
    self grid22WithPointLocationCreatedWithRows.
    self grid3x2BooksCreatedWithRows
```

L'approche

Débogage :





L'approche

Setup

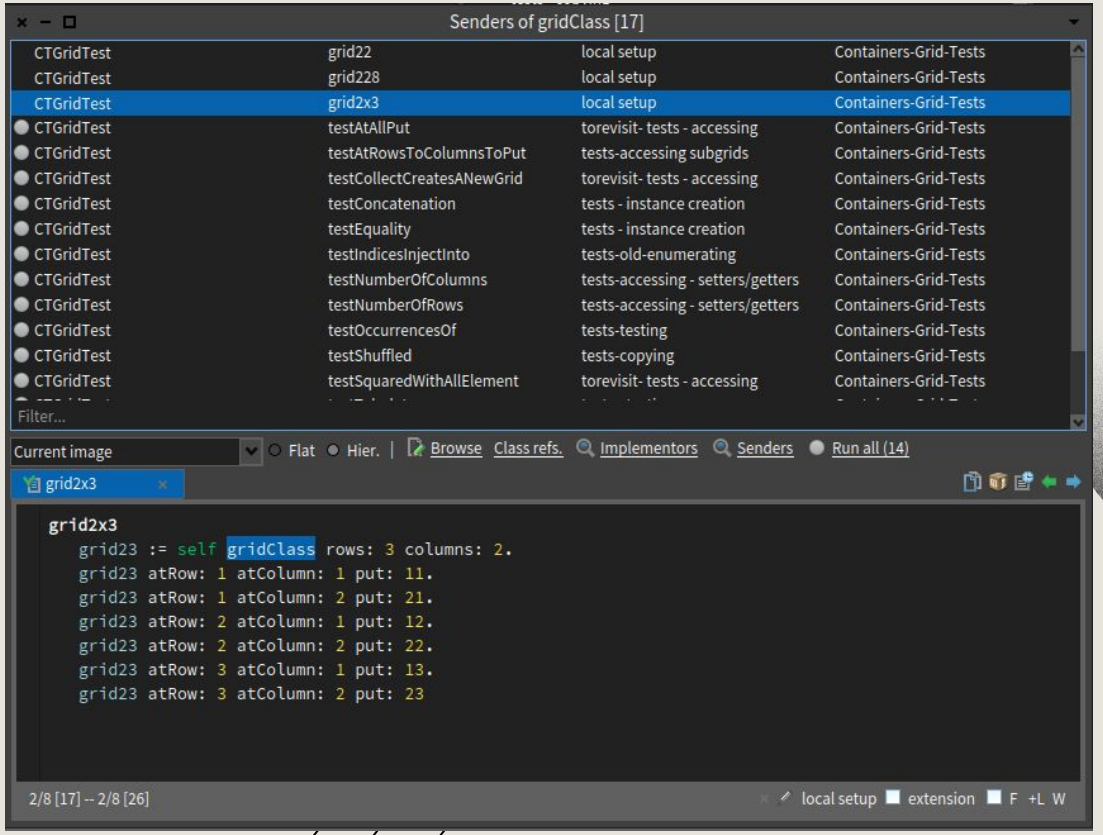
```
setUp
    self grid22.
    self grid228.
    self grid2x3.

    "new setup creation"
    self grid6x2.
    self grid22WithPointLocationCreatedWithRows.
    self grid3x2BooksCreatedWithRows
```


```
grid22
    "self new grid22"
    grid1 := self gridClass new: 2.
    grid1 atRow: 1 atColumn: 1 put: 1.
    grid1 atRow: 1 atColumn: 2 put: 3.
    grid1 atRow: 2 atColumn: 1 put: 2.
    grid1 atRow: 2 atColumn: 2 put: 4.
    ^ grid1
```

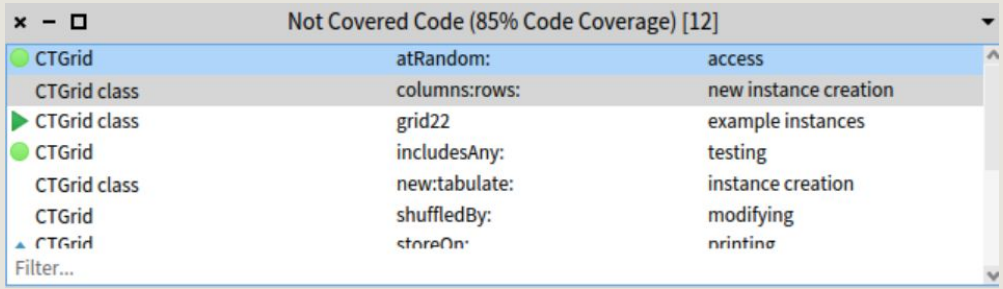
Retourne la variable d'instance

L'approche

Senders



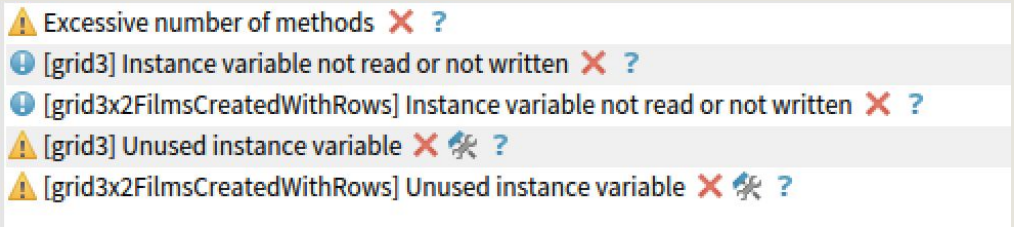
Méthodes non testés



- Constructeur
- L'aléatoire
- Affichage

Conclusion

- Enrichir les commentaires.
- Améliorer le coverage (85% actuellement).
- séparer la God class en hiérarchies.
- Code mort ?



Conclusion

- **Familiarisation à la syntaxe.**
- **Bonne approche qui peut être utilisée sur n'importe quel projet.**
- **Découverte : References to it/Implementors of it/WatchPoint/Debug.**
- **Utilisation de L'UML ?**

Merci