

# PRESENTATION

Conception et **P**aradigmes de **P**rogrammations par la  
**P**ratique

# PLAN



## METHODE D'APPRENTISSAGE

- Présentation de la méthode de chaque constituant du groupe



## CONTAINERS - LINKEDLIST

- Présentation de la structure de données
- Utilisation et Implémentation
- Feedback sur les tests



## CONTAINERS - HASHTABLE

- Présentation de la structure de données
- Utilisation et Implémentation
- Feedback sur les tests

# METHODES D'APPRENTISSAGE



**SITE DU FUN MOOC**  
**<https://www.fun-mooc.fr/>**

- Vidéos
- Quizz



## **ENTRAIDE**

- En classe
- Par travail de groupe



## **EXERCICES ET REVISIONS PERSONNELLES**

- Lecture des cours
- Assimilation avec de petites exercices

The image features a white background with decorative purple wavy lines in the corners. The lines are composed of many thin, overlapping curves that create a sense of motion and depth. They are located in the top-right and bottom-left corners, framing the central text.

# CONTAINERS- HASHTABLE

# CONTAINERS- HASHTABLE

CONCEPT	PERSPECTIVES DE L'UTILISATEUR et IMPLEMENTATION	ANALYSE	LES TESTS
<ul style="list-style-type: none"><li>une structure de données qui permet une association <b>clé-valeur</b></li></ul>	<ul style="list-style-type: none"><li>L'ajout d'un élément.</li><li>La suppression d'un élément.</li><li>La recherche d'un élément.</li></ul>	<ul style="list-style-type: none"><li>Une classe vide.</li><li>Des expressions peuvent être simplifiées.</li><li>Une API riche.</li><li>Les classes héritent de Collection.</li></ul>	Tous les tests passent au vert.



# CONTAINERS- HASHTABLE

## Dictionary

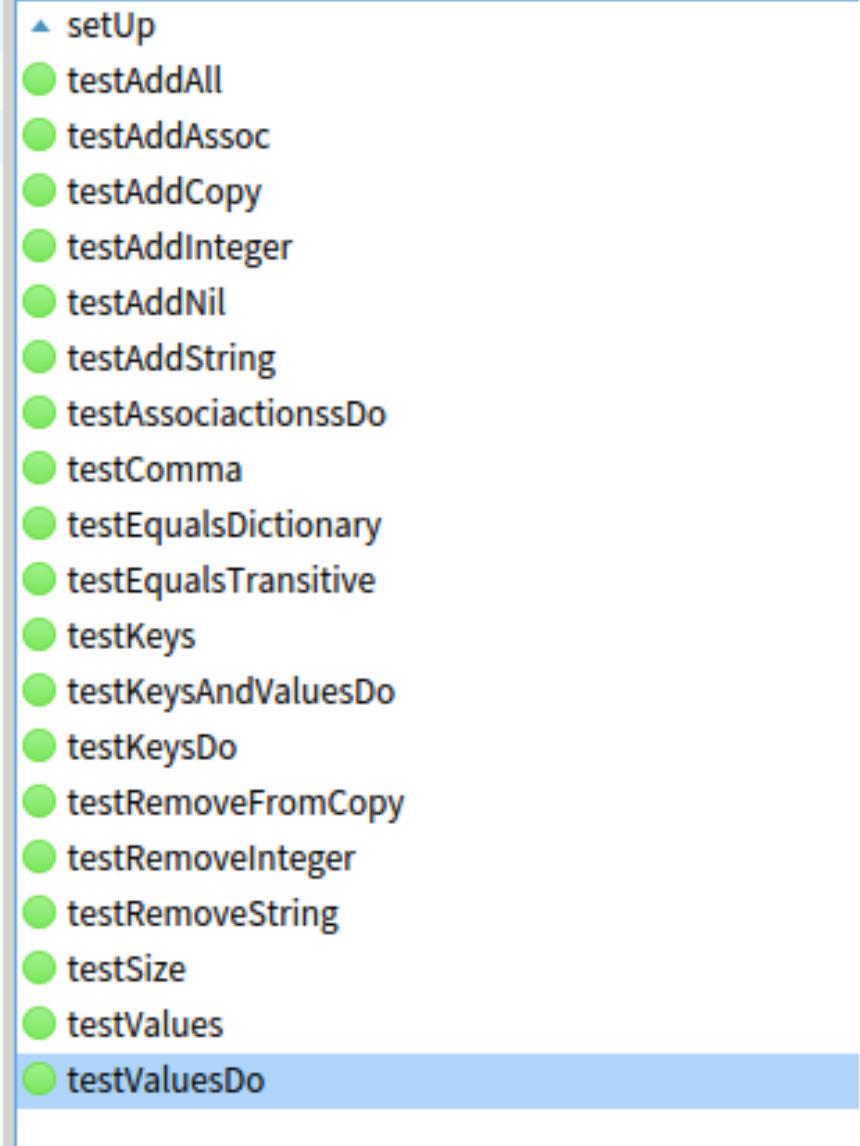
```
collect: aBlock  
"Evaluate aBlock with each of my values as the argument. Collect the  
resulting values into a collection that is like me. Answer with the new  
collection."  
| newCollection |  
newCollection := OrderedCollection new: self size.  
self do: [:each | newCollection add: (aBlock value: each)].  
^ newCollection
```

## CTHashTable

```
collect: aBlock  
"Evaluate aBlock with each of my values as the argument. Collect the  
resulting values into a collection that is like me. Answer with the new  
collection."  
| newCollection |  
newCollection := self species new.  
self associationsDo: [:each |  
    newCollection at: each key put: (aBlock value: each value).  
].  
^newCollection
```

# CONTAINERS- HASHTABLE

Méthode at: put:



- ▲ setUp
- testAddAll
- testAddAssoc
- testAddCopy
- testAddInteger
- testAddNil
- testAddString
- testAssociationssDo
- testComma
- testEqualsDictionary
- testEqualsTransitive
- testKeys
- testKeysAndValuesDo
- testKeysDo
- testRemoveFromCopy
- testRemoveInteger
- testRemoveString
- testSize
- testValues
- testValuesDo

# CONTAINERS- LINKEDLIST

CONCEPT	PERSPECTIVES DE L'UTILISATEUR et IMPLEMENTATION	ANALYSE	LES TESTS
<ul style="list-style-type: none"><li>• Structure de données d'un <b>élément</b> et de <b>l'adresse</b> (référence) <b>vers l'élément suivant / élément précédent</b> (Liste doublement chaînée)</li><li>• Plus <b>flexible</b> qu'un tableau</li><li>• Joue sur la <b>quantité de mémoire</b> disponible</li></ul>	<ul style="list-style-type: none"><li>• Créer une liste vide et tester si une liste est vide</li><li>• Afficher une liste</li><li>• Ajouter un élément en tête de liste.</li><li>• Rechercher un élément</li><li>• Ajouter une liste en fin de liste</li><li>• Supprimer un élément</li></ul>	<ul style="list-style-type: none"><li>• Nécessite <b>beaucoup de mémoires</b> pour la création de plusieurs listes</li><li>• <b>flexible : API RICHE</b></li><li>• <b>Rapide</b> pour le traitement d'un/plusieurs éléments</li></ul>	<ul style="list-style-type: none"><li>• Listes doublement chaînées: <div>✖ CTDoublyLinkedListTests 24 ran, 24 passed, 0 skipped, 0 expected failures, 0 failures, 0 errors, 0 passed unexpected</div></li><li>• Listes chaînées: <i>testTAddWithOccurrences</i> <i>testCopyReplaceAllWithManyOcurrence</i> <div>✖ CTLinkedListTest 249 ran, 247 passed, 0 skipped, 0 expected failures, 0 failures, 2 errors, 0 passed unexpected</div></li></ul>



# CONTAINERS- LINKEDLIST

## LES TESTS

```
testTAddWithOccurrences
| added oldSize collection anElement |
collection := self collectionWithElement.
anElement := self element.
oldSize := collection size.
added := collection add: anElement withOccurrences: 5.

self assert: added == anElement. "test for identity because #add: has not reason to copy its parameter."
self assert: (collection includes: anElement).
self assert: collection size equals: (oldSize + 5)
```

```
testTAddIfNotPresentWithNewElement

| added oldSize collection elem |
collection := self otherCollection.
oldSize := collection size.
elem := self element.
self deny: (collection includes: elem ).

added := collection addIfNotPresent: elem .
self assert: added == elem . "test for identity because #add: has not reason to copy its parameter."
self assert: collection size equals: (oldSize + 1).
```

MERCI DE VOTRE  
ATTENTION