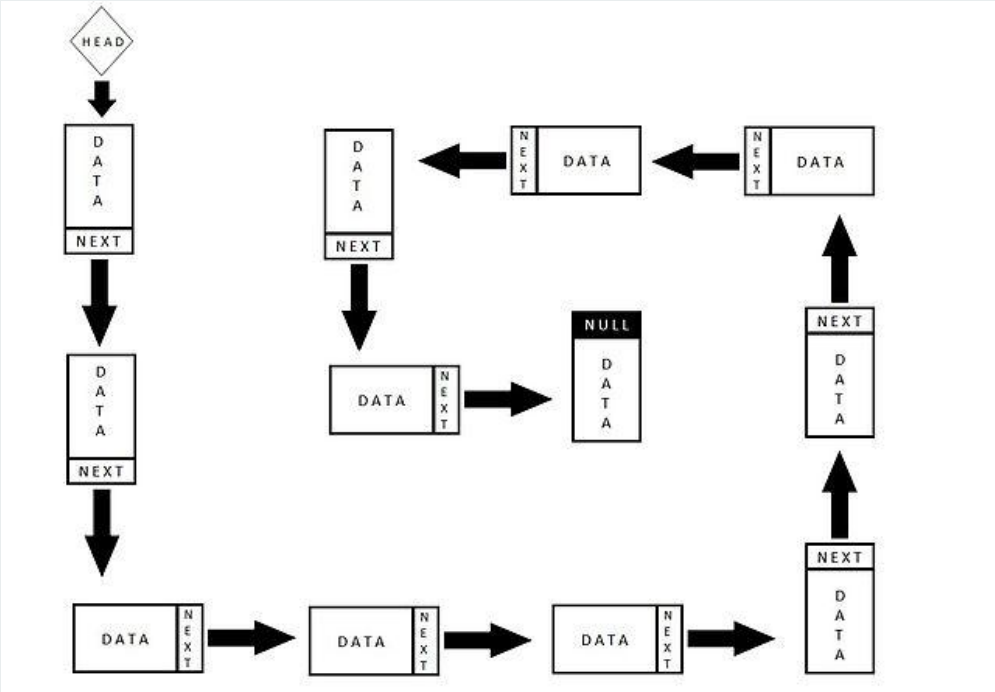
# LinkedList

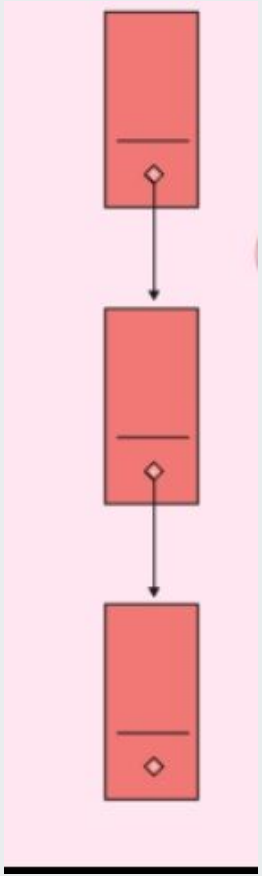
## Participants:

- Chihebeddine Hamrouni
- Alissia Chaouche
- Gaetan Garein
- Isabelle Bivegue



# Plan de présentation:

1. Nos méthodes de travail
2. Utilisation et méthodes les plus utilisées
3. Son implémentation
4. Les tests



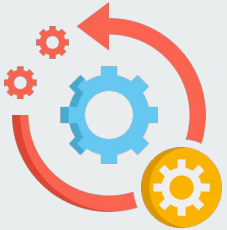
# Nos méthodes de travail


FUN MOOC

# Découverte du sujet

Découverte du sujet

Phase de recherche



Reverse engineering

Regroupement des informations

WIKIPÉDIA
L'encyclopédie libre

GeeksforGeeks

# Documentation

? Comment × | CTDoubleLink × | UML-Class × | Inst. side methc ×

## Class: CTDoubleLink

I am DoubleLink, the elementary part of a DoubleLinkedList.

I hold a value, as well as a link to my successor (nextLink) and to my predecessor (previousLink) - both can be nil.

? Comment × | CTDoubleLinke × | add:beforeLink: × | UML-Class × | Inst. side methc ×

## Class: CTDoubleLinkedList

I am CTDoubleLinkedList, an ordered list data structure consisting of objects, most likely DoubleLinks or something compatible, connected to each other by forward and backwards links.

Note that some of my API deals with the elements that I hold, like any other collection, while some of my API references the links that I use internally (those usually have the word link in the selector name). Some methods accepts both values or links as argument (like #add:). Because I expose some of my internal structure, I can be broken quite easily.

# C'est quoi une linkedlist?

- Liste simplement chaînée
- Liste Doublement chaînée

```
linkedList := CTLinkedList new.
linkedList add: 10.
linkedList add: 20.
linkedList addLast: 30.

^ linkedList
```

a CTLinkedList(10 15 20 30)

```
double := CTDoubleLinkedList new.
double add: 10.
double add: 20.
double addLast: 30.

^ double
```

a CTDoubleLinkedList(10 20 30)

| Variable | Value |
|---|---|
| self | a CTDoubleLinkedList(10 20 30) |
| ▶ head | a CTDoubleLink (10) |
| ▶ tail | a CTDoubleLink (30) |

```
doubleLinkedList := CTDoubleLinkedList doubleLinkedListWith10Integers.
```

a CTDoubleLinkedList(1 2 3 4 5 6 7 8 9 10)

# Méthodes les plus utilisées

```
double := CTDoubleLinkedList new.
double add: 10.
double add: 20.
double addLast: 30.
double addFirst: 5.
double addLast: 35.

^ double
```

a CTDoubleLinkedList(5 10 20 30 35)

```
double asArray .
```

#(5 10 20 30 35)

```
double emptyCheck .
```

a CTDoubleLinkedList(5 10 20 30 35)

```
double firstLink .
```

a CTDoubleLink (5)

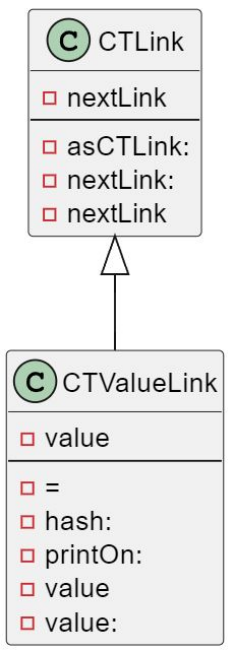
```
double isEmpty .
```

false

```
double size.
```

5

- CTDoubleLink
- CTDoubleLinkedList




```
llink := CTValueLink new value: 20.
^llink
```
a CTValueLink(20)

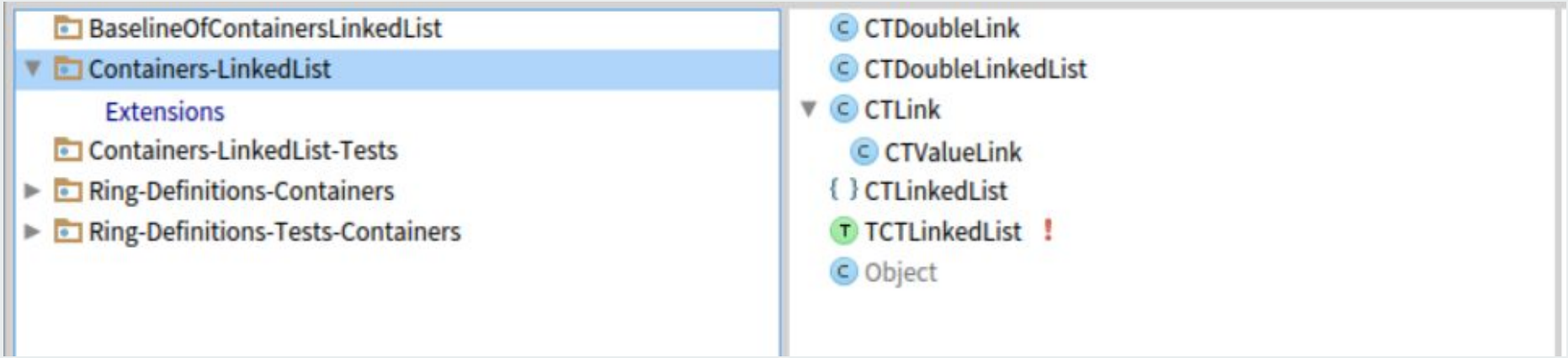
```
llink value: 30.
```
a CTValueLink(30)

```
llink value.
```
20

```
llink nextLink.
```
nil

- Capacité = Taille de la liste
- surcharge mémoire # problèmes de performances majeur

# Implementation



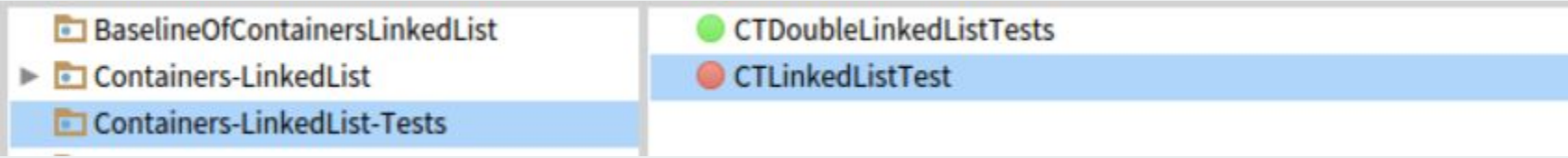
# La méthode *add: afterLink:* pour la liste doublement chaînée

```
add: anObjectOrLink afterLink: otherLink
    "Add anObjectOrLink right after otherLink in me.
    When otherLink is not part of me, the result is undefined.
    Return the internal link object."

    | link otherLinkSuccessor |
    otherLink = tail
        ifTrue: [ ^ self addLast: anObjectOrLink ].
    link := self linkOn: anObjectOrLink.
    otherLinkSuccessor := otherLink nextLink.
    otherLink nextLink: link.
    link previousLink: otherLink.
    link nextLink: otherLinkSuccessor.
    otherLinkSuccessor previousLink: link.
    ^ link
```

# La méthode *add: beforeLink:* pour la liste simplement chaînée

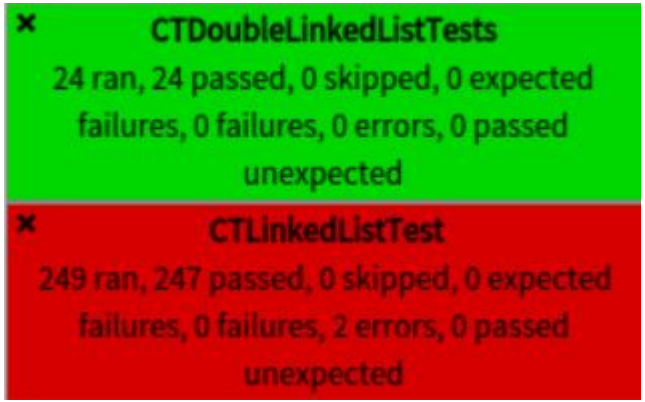
```
add: aLinkOrObject beforeLink: otherLink

   | currentLink|

   firstLink == otherLink ifTrue: [^ self addFirst: aLinkOrObject].

   currentLink := firstLink.
   [currentLink == nil] whileFalse: [
      currentLink nextLink == otherLink ifTrue: [
         | aLink |
         aLink := self linkOn: aLinkOrObject.
         aLink nextLink: currentLink nextLink.
         currentLink nextLink: aLink.
         ^ aLink
      ].
       currentLink := currentLink nextLink.
   ].
   ^ self errorNotFound: otherLink
```

# Les tests


BaselineOfContainersLinkedList
Containers-LinkedList
Containers-LinkedList-Tests
CTDoubleLinkedListTests
CTLinkedListTest

# Couverture des tests

- Couverture du projet complète
- Environ 99,26% de réussite (271 sur 273)



# Exemple de test

```
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 identiy bec
    self assert: (collection includes: anElement).
    self assert: collection size equals: oldSize + 5
```

# Exemple de correction

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

    self assert: added equals: anElement.   "test for identiy because #add: 
    self assert: (collection includes: anElement).
    self assert: collection size equals: (oldSize + 5)
```

# Merci pour votre écoute