Ordered Multi Map

La structure de données & notre approche

1. La Structure de Données

README.md + Commentaire de Classe

Signification

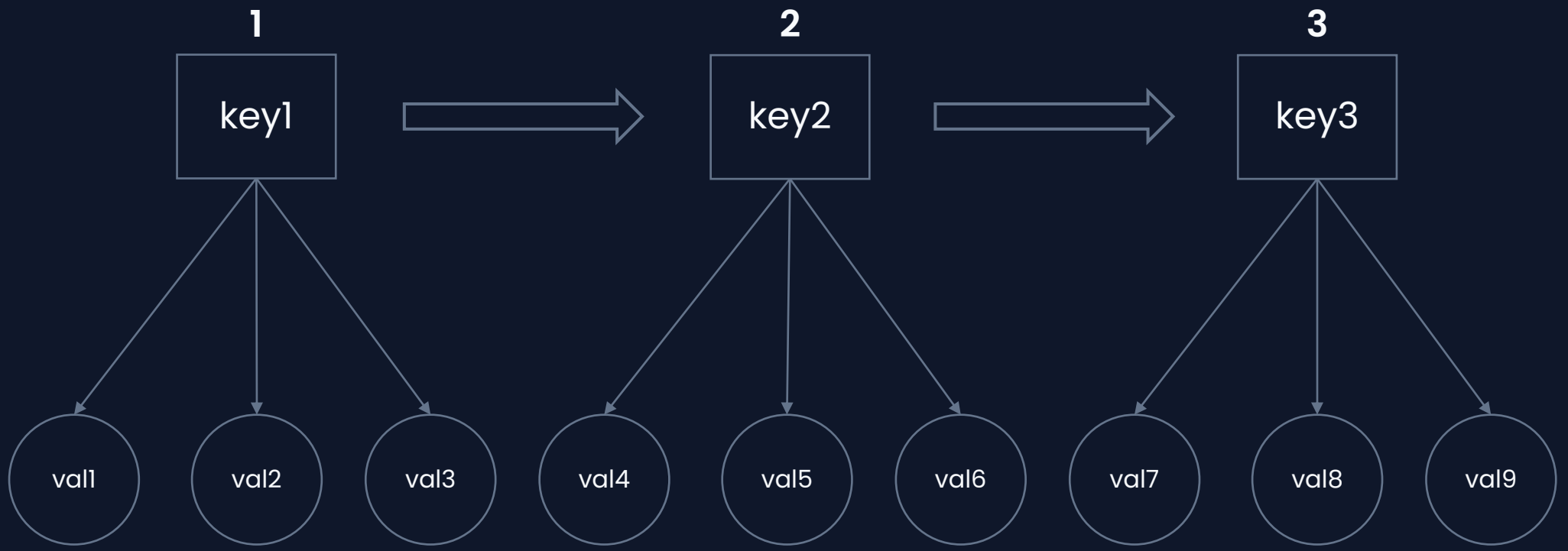
"OrderedMultiMap is a dictionary whose values are a collection"

Map : Associations Clés/Valeurs

Multi : Plusieurs valeurs par clé

Ordered : Maintient l'ordre d'insertion des valeurs

Exemple


1
2
3
key1
key2
key3
val1
val2
val3
val4
val5
val6
val7
val8
val9

2. L’Utilisation

Commentaire de Classe + Tests + Playground

Messages Lecture/Ecriture

at:add: / add:

Les clés et valeurs peuvent être n'importe quoi, y compris nil

allAt:

```
| map |
map := CTOrderedMultiMap   new
  at: 'key1' add: 'value1';
  at: 'key2' add: 'value2';
  add: 'key2' -> 'value3';
  add: 'key1' -> 'value4';
  yourself.

map allAt: 'key1'. "#('value1' 'value4')"
map allAt: 'key2'. "#('value2' 'value3')"
```

Messages Suppression/Vérification

removeKey:
includesKey:

```
| map |
map := CTOrderedMultiMap   new
  at: 'key1' add: 'value1';
  at: 'key2' add: 'value2';
  at: 'key2' add: 'value3';
  at: 'key1' add: 'value4';
  yourself.

map includesKey: 'key2'. "true"
map removeKey: 'key2'. "#('value2' 'value3')"
map includesKey: 'key2'. "false"
```

Messages Utilitaires

isEmpty
size

keys:
values:

```
| map |
map := CTOrderedMultiMap   new
  at: 'key1' add: 'value1';
  at: 'key2' add: 'value2';
  at: 'key2' add: 'value3';
  at: 'key1' add: 'value4';
  yourself.

map isEmpty. "false"
map size. "4"

map keys. "#('key1' 'key2' 'key2' 'key1')"
map values. "#('value1' 'value2' 'value3' 'value4')"
```

Messages Itérations

- do:
- keysDo:
- keysAndValuesDo:

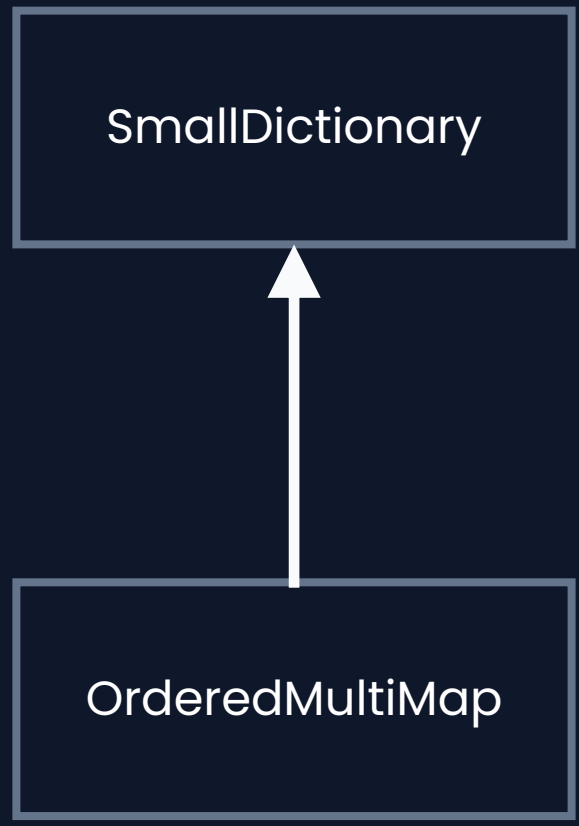
3. La Conception

Tests + UML + Code

SmallDictionary

- SmallDictionary :
  une valeur par clé

- OrderedMultiMap :
  plusieurs valeurs par clé



Comparaison Messages

Action	Message SmallDictionary	Message OrderedMultiMap
Ecrire	at:put: / add:	at:add: / add:
Lire	at:	allAt:
Supprimer	removeKey:	removeKey:

Exemple Messages SmallDictionary

```
| map |
map := CTOrderedMultiMap   new
  at: 'key1' put: 'value1';
  at: 'key2' put: 'value2';
  at: 'key2' put: 'value3';
  at: 'key1' put: 'value4';
  yourself.

map at: 'key1'. "'value4'"
map at: 'key2'. "'value3'"
```

4. L'Implémentation

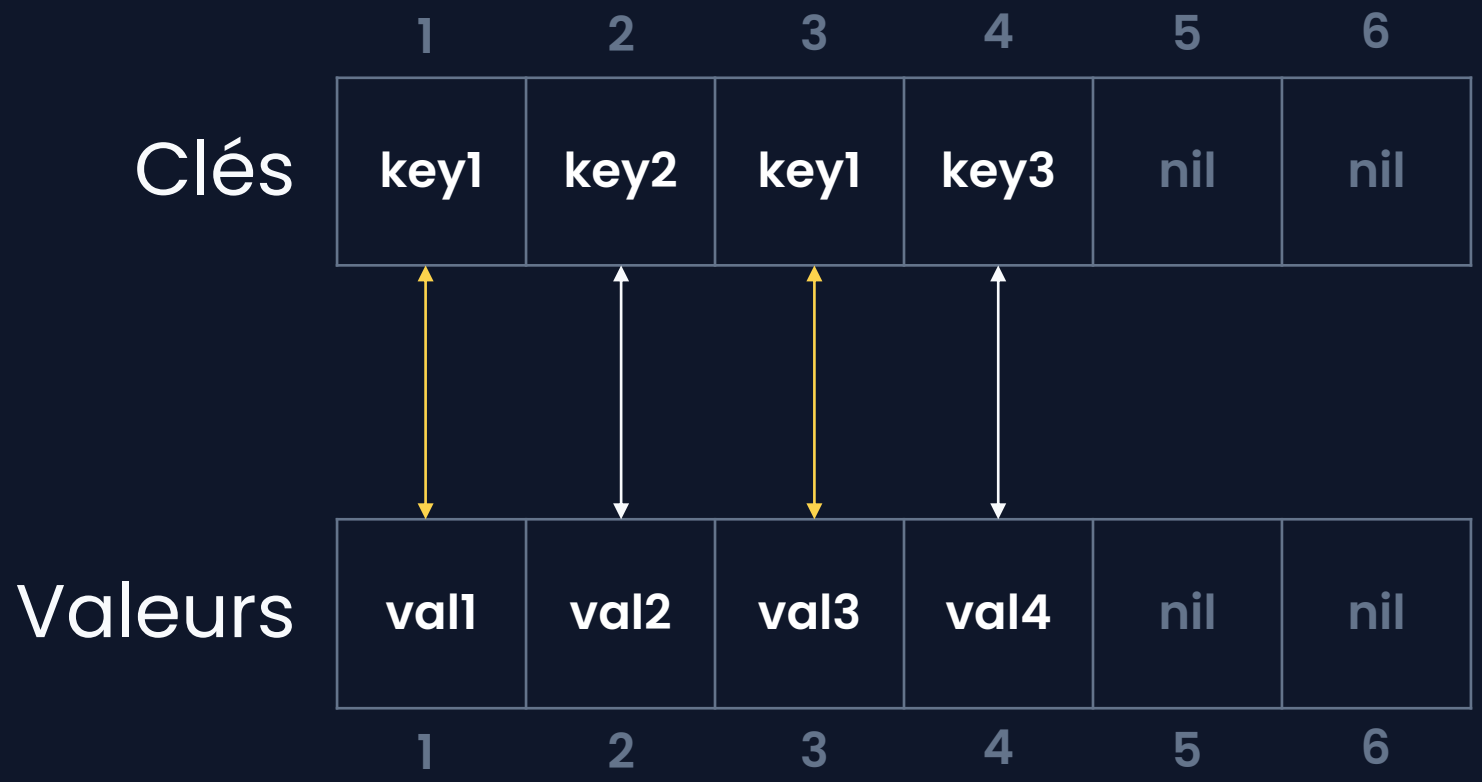
Code + Commentaires

Commentaires

Peu de commentaires dans SmallDictionary

```
"This is inefficient and could be optimized."
```

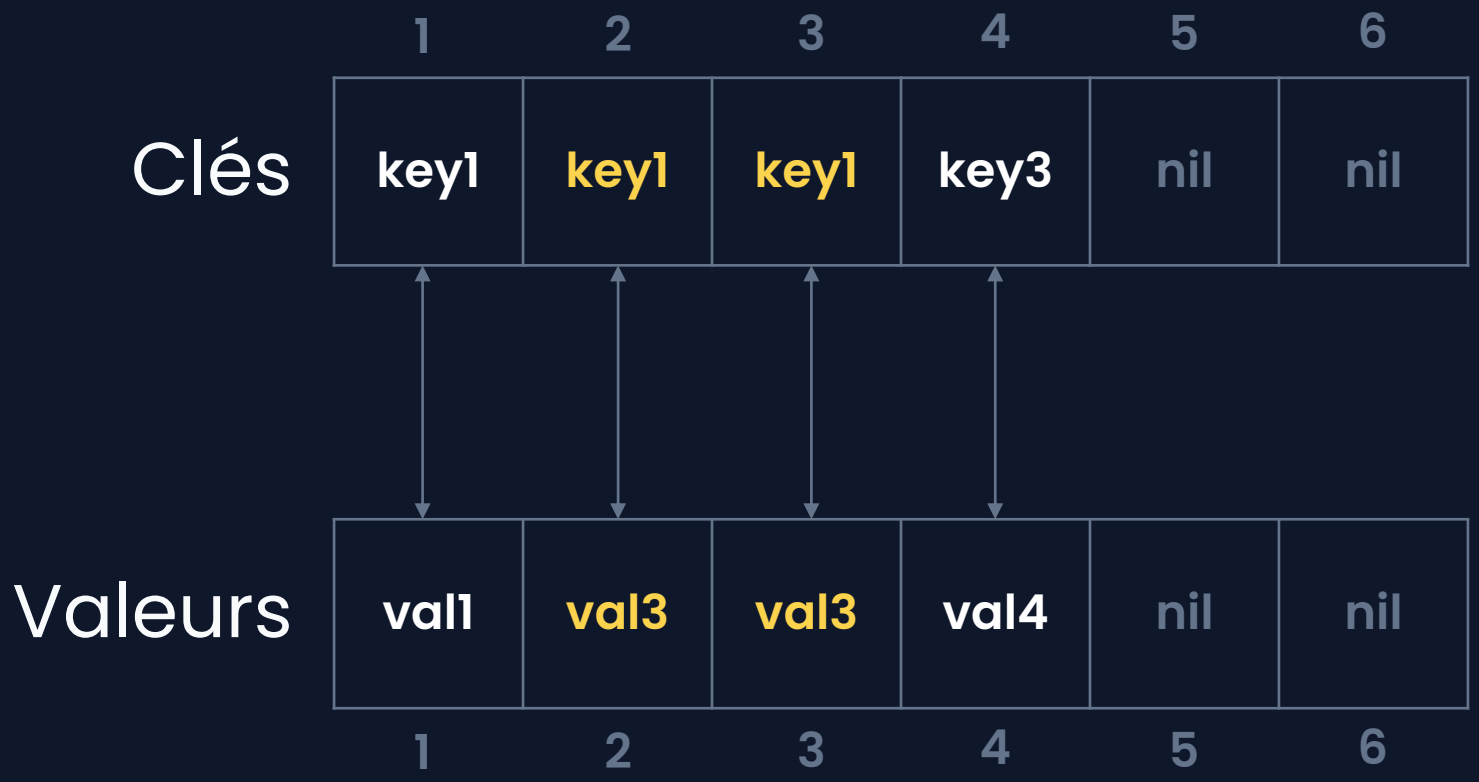
Fonctionnement Clés/Valeurs



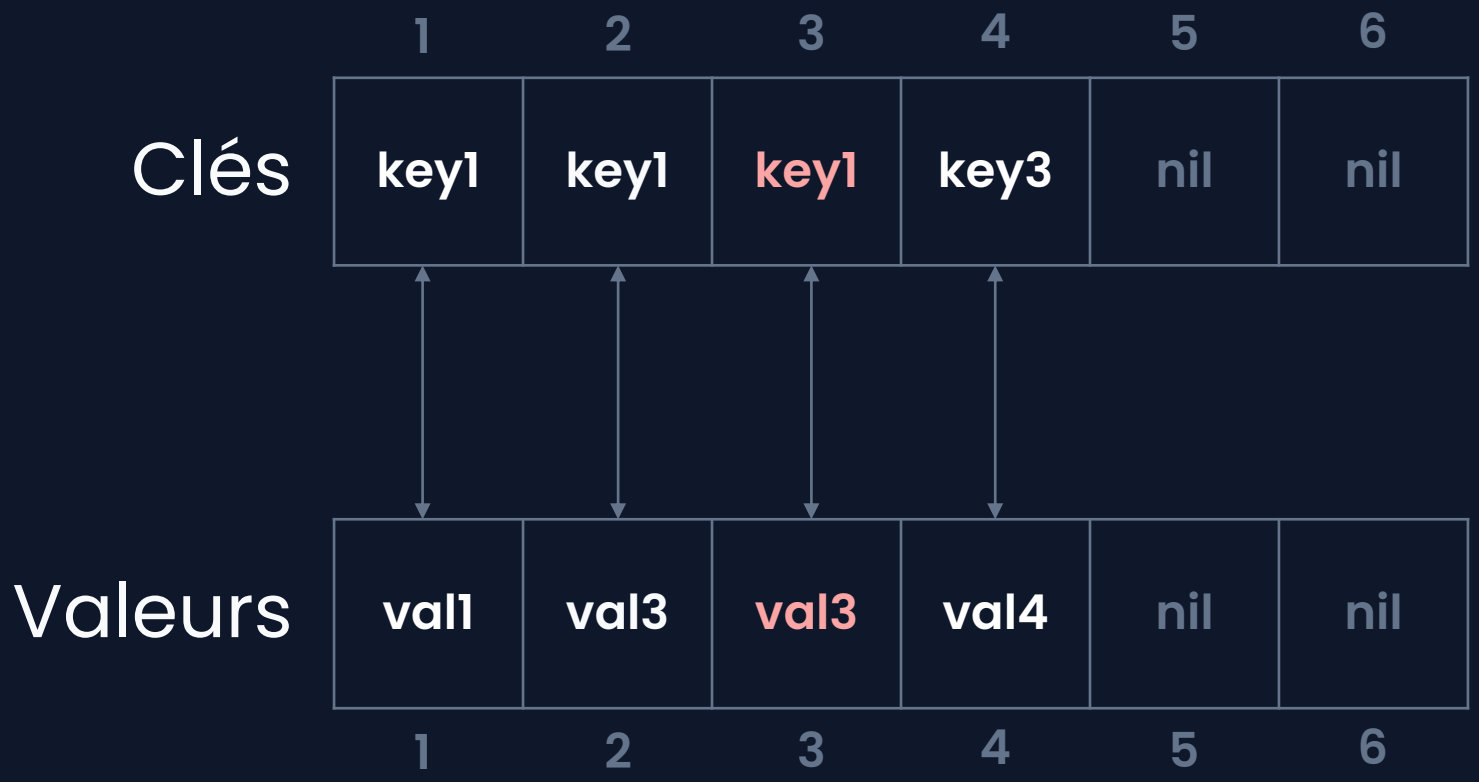
Fonctionnement Suppression

Fonctionnement Suppression

Fonctionnement Suppression



Fonctionnement Suppression



Fonctionnement Suppression

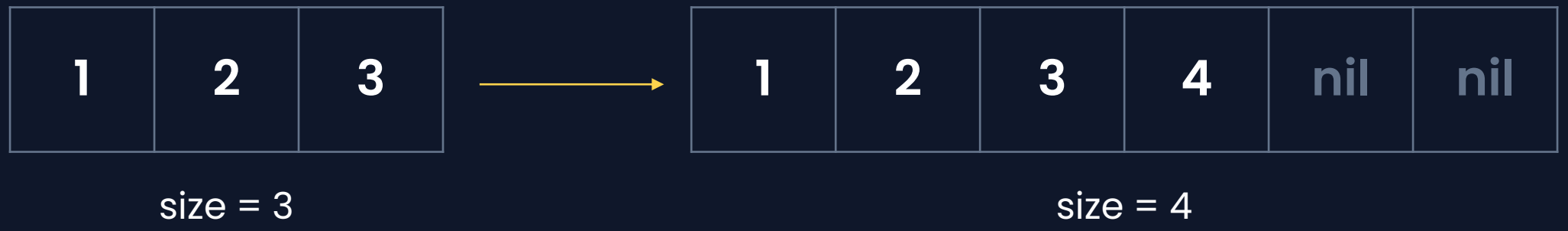
Fonctionnement Suppression

Fonctionnement Suppression

Fonctionnement Suppression

Système « Grow »

- Taille initiale allouée de 3
- Lorsque c’est nécessaire, double la taille



5. Les Tests

Tests

Tests

- Code coverage de 95% (manque printOn:)
- Pas de tests pour les clés/valeurs nil
- at:put: avec OrderedMultiMap testé à moitié…

Cohérence at:put:

```
| map |
map := CTOrderedMultiMap   new
  at: 'key1' add: 'value1';
  at: 'key1' add: 'value2';
  yourself.

map allAt: 'key1'. "#('value1' 'value2')"
map at: 'key1' put: 'value3'.
map allAt: 'key1'. "#('value3' 'value2')"
```

```
testAtPutReplaceExistingValues

  collection at: '1' put: 'foo'.
  self assert: (collection allAt: '1') equals: #('foo').
  collection at: '1' put: 'bar'.
  self assert: (collection allAt: '1') equals: #('bar').
  self assertAssociations: (Array  with: '1' -> 'bar')
```

6. Les Compromis

Code + Recherches

Compromis

“very efficient for small sizes”

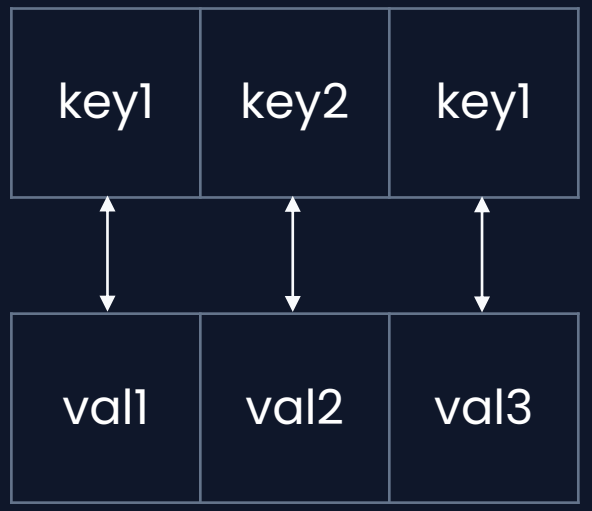
- Recherche en O(n)
- Possible d’avoir une recherche en O(1) ?

Exemple LinkedHashMap Java

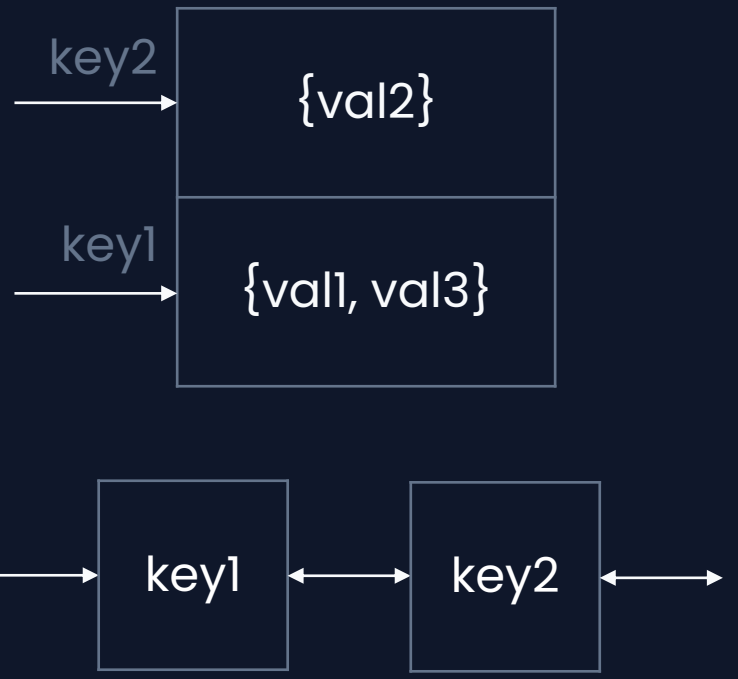
- LinkedHashMap : table de hachage ordonné
- Ordonné avec une liste doublement chaînée

Comparaison

OrderedMultiMap
Recherche en O(n)



LinkedHashMap
Recherche en O(1)



Conclusion

Questions ?