

ÉNONCÉ

Coder et tester la classe *LinkedList*. Cette dernière permet de mettre en place une liste doublement chaînée.

Afin de permettre une standardisation des différentes classes représentant une liste ou un tableau dans l'entreprise, votre chef oblige votre classe à implémenter l'interface *Collection*. Afin de vous simplifier la tâche, votre chef a déjà codé l'interface *Collection* ainsi que la classe *Node*, représentant un nœud dans la liste doublement chaînée.

Il vous a fourni le code ainsi que les diagrammes pour la classe *Node* et l'interface *Collection* :

```
src.domain.Node.Node
__init__(self, prev: 'Node', value, next: 'Node')
get_next(self)
set_next(self, next: 'Node')
get_prev(self)
set_prev(self, prev: 'Node')
get_value(self)
set_value(self, value)
have_next(self)
have_prev(self)
__str__(self)
__eq__(self, other)
__prev
__value
__next
```

```
src.metier.Collection.Collection
add(self, element, index: int = None)
get(self, index: int)
set(self, element, index: int)
remove(self, element = None, index: int = None)
clear(self)
size(self)
is_empty(self)
contains(self, element)
index_of(self, element)
insertion_sort(self)
merge_sort(self)
dichotomic_search(self, element)
```

Il vous a également fourni le fichier *LinkedList.py*, contenant la classe *LinkedList*. **Afin d'éviter tout problème, votre chef vous a interdit de modifier la classe *Node* ainsi que l'interface *Collection*.**

Pour tester votre code, vous avez une méthode main dans le fichier *main.py*. Il peut être intéressant de comparer le temps de calcul entre les méthodes *index_of* / *dichotomic_search* et *insertion_sort* / *merge_sort* !

Votre chef vous demande aussi de créer la méthode *__str__* afin que l'affichage de la liste soit au format suivant :

$$i^1 \leftrightarrow i^2 \leftrightarrow \dots \leftrightarrow i^n$$

Exemple :

814251 <-> 447480 <-> 258903 <-> 643905 <-> 567630 <-> 891293 <-> 774999 <-> 700023 <-> 134171

DATE DE RENDU

Dimanche 15 octobre 2020 à 23h00