

« ListeSD »

SD : sans doublon

Interface ListeSD <> extends Iterable<E>

```
int  taille()
```

```
boolean estVide()
```

```
String toString()
```

```
boolean contient(E element)
```

```
E premier()
```

```
E dernier()
```

Interface ListeSD<E> extends Iterable<E>

`boolean insererEnTete(E element)`

`boolean insererEnQueue(E element)`

`boolean insererAvant(E e, E eAInserer)`

`boolean insererApres(E e, E eAInserer)`

`boolean supprimer(E element)`

`E donnerPrecedent(E element)`

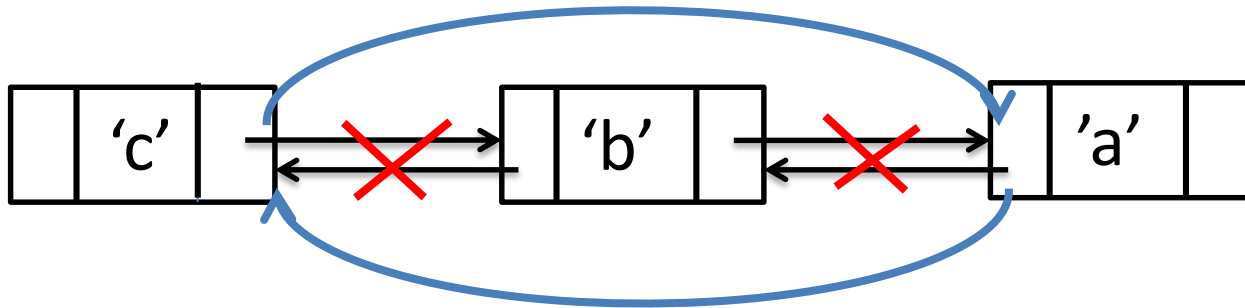
`E donnerSuivant(E element)`

`boolean permuter(E element1, E element2)`

La LinkedList est coûteuse!

Opération	LinkedList
<code>int indexOf(Object o)</code>	$O(N)$
<code>E get (int index)</code>	$O(N)$
<code>boolean contains (Object o)</code>	$O(N)$
<code>void add (int index, E element)</code>	$O(N)$
<code>boolean remove (Object o)</code>	$O(N)$

Suppression d'un nœud :




$O(1)$

Suppression d'un élément dans une liste doublement chaînée:

$$O(N)$$

recherche du nœud via parcours : $O(N)$
+ suppression du nœud : $O(1)$

Suppression d'un élément
dans une liste doublement chaînée
+ $\text{HashMap}\langle E, \text{Nœud} \rangle$: 

$O(1)$

recherche du nœud via map : $O(1)$
+ suppression du nœud : $O(1)$

Choix d'implémentation pour la listeSD :

