CLASS LISTETABLEAU

`public class` **`ListeTableau`**

Version restreinte et francisée de la classe `java.util.ArrayList` (classe très proche de la classe `java.util.Vector`). Comme la classe `java.util.ArrayList` (qui implémente l'interface `java.util.List`), cette classe utilise un tableau pour implémenter une structure de liste permettant de mémoriser des objets de classe arbitraire, dans ce contexte, la valeur spéciale `null` est considérée comme un élément comme les autres pouvant appartenir à une liste au même titre qu'un objet quelconque. En plus des opérations basiques sur les listes, cette classe fournit des méthodes pour manipuler la taille du tableau utilisé pour mémoriser la liste. Comme il est habituel pour les listes, une instance de `ListeTableau` peut contenir plusieurs exemplaires (i.e. plusieurs éléments identiques au sens de "equals") d'un même élément.

Chaque instance de `ListeTableau` possède une capacité. La capacité est la taille du tableau utilisé pour mémoriser les éléments de la liste. Elle est toujours au moins aussi grande que la taille de la liste (i.e. le nombre d'éléments de la liste). Au fur et à mesure que des éléments sont ajoutés à une `ListeTableau`, sa capacité augmente automatiquement.

Lorsqu'une application peut prévoir à l'avance qu'elle va procéder à l'ajout d'un grand nombre d'éléments à une instance de `ListeTableau`, il lui est possible d'augmenter la capacité à l'aide de la méthode `assurerCapaciteMinimum`. Une telle précaution peut permettre de remplacer une succession de réallocations de faible ampleur par un nombre plus réduit de réallocations plus importantes.

Class Specifications

invariant `capacite() >= taille();`
invariant `taille() >= 0;`
invariant `estVide() <==> (taille() == 0) ;`

Constructor Detail

ListeTableau

`public` **`ListeTableau`**`()`

Initialise une liste vide avec une capacité initiale fixée à dix.
Specifications:
ensures capacite() == 10;
ensures estVide();

ListeTableau

`public` **`ListeTableau`**`(int capaciteInitiale)`

Initialise une liste vide avec une capacité initiale fixée à la valeur spécifiée.
Parameters:
`capaciteInitiale` - la capacité initiale de la liste
Specifications:
requires capaciteInitiale >= 0;
ensures capacite() == capaciteInitiale;
ensures estVide();

Method Detail

get

`public Object` **`get`**`(int index)`

Renvoie l'élément situé à l'index spécifié dans cette liste.
Parameters:
`index` - index de l'élément à renvoyer
Returns: l'élément à la position spécifié dans la liste.
Specifications: pure
requires index >= 0 && index < taille();
ensures contient(\result);
ensures indexDe(\result) <= index;

contient

`public boolean` **`contient`**`(Object element)`

Renvoie `true` si cette liste contient l'élément spécifié. Plus précisément, renvoie `true` si et seulement si cette liste contient au moins un élément e tel que:

- `e == null` si `element == null`
- `element.equals(e)` si `element != null`

Parameters:

`element` - élément dont la présence dans cette liste doit etre testée.

Returns: `true` si l'élément spécifié est présent ; `false` sinon.

Specifications: *(à compléter)*

---

indexDe

`public int` **`indexDe`**`(java.lang.Object element)`

Recherche dans cette liste de la première occurrence de l'élément spécifié. C'est la méthode `equals` qui est utilisée pour tester l'égalité de l'élément cherché avec les éléments de la liste.Plus précisément, renvoie le plus petit index `i` tel que:

- `get(i) == null` si `element == null`
- `element.equals(get(i))` si `element != null`
- `i == -1` si aucun index ne satisfait les conditions précédentes

Parameters: `element` - élément recherché

Returns: l'index de la première occurrence de l'élément dans cette liste ; -1 si l'objet n'est pas trouvé.

Specifications: *(à compléter)*

---

ajouter

`public void` **`ajouter`**`(int index, Object element)`

Insertion dans la liste de l'élément spécifié à la position spécifiée. Le cas échéant, décale vers la droite l'élément actuellement à cette position ainsi que tous les éléments suivant.

Parameters:

`index` - index auquel l'élément spécifié va être inséré

`element` - élément à insérer

Specifications:

requires 0 <= index && index < taille();
ensures element == get(index);
ensures (\forall int i; 0 <= i && i < index; get(i) == \old(get(i)));
ensures (\forall int i; index <= i && i < \old(taille()); get(i+1) == \old(get(i)));

---

ajouter

`public void` **`ajouter`**`(Object element)`

Ajoute l'élément spécifié à la fin de la liste.

Parameters:

`element` - element à ajouter à cette liste.

Specifications:

ensures taille() == \old(taille())+1;
ensures (get(\old(taille())) == element);
ensures (\forall int i; 0 <= i && i < \old(taille()); get(i) == \old(get(i)));

---

set

`public Object` **`set`**`(int index, Object element)`

Remplace l'élément de cette liste placé à la position spécifiée par l'élément spécifié.

Parameters:

`index` - index de l'élément à remplacer.

`element` - élément à placer à la position spécifiée.

Returns: l'élément précédemment situé à la position spécifiée.

Specifications:

requires index >= 0 && index < taille();
ensures taille() == \old(taille());
ensures \result == \old(get(index));
ensures get(index) == element;
ensures (\forall int i; 0 <= i && i != index && i < this.taille(); get(i) == \old(get(i)));

retirer

`public Object` **`retirer`**`(int index)`

Enlève de cette liste l'élément situé à la position spécifiée. Décale vers la gauche tous les éléments situés après l'élément supprimé.

Parameters:

`index` - l'index de l'élément à enlever

Returns:

l'élément enlevé de la liste

Specifications:

requires index >= 0 && index < taille();
ensures \result == \old(get(index));
ensures taille() == \old(taille()) - 1;
ensures (\forall int i; 0 <= i && i < index; get(i) == \old(get(i)));
ensures (\forall int i; index < i && i < \old(taille()); get(i-1) == \old(get(i)));

---

vider

`public void` **`vider`**`()`

Enlève tous les éléments de cette liste.

Specifications:

ensures estVide();

---

estVide

`public boolean` **`estVide`**`()`

Teste si cette liste ne contient aucun élément.

Returns:

`true` si cette liste ne contient aucun élément ; `false` sinon.

Specifications: pure

ensures \result <==> (taille() == 0);

---

taille

`public int` **`taille`**`()`

Renvoie le nombre d'éléments dans cette liste.

Returns:

le nombre d'éléments dans cette liste.

Specifications: pure

ensures \result >= 0;

---

capacite

`public int` **`capacite`**`()`

Renvoie la capacité actuelle de cette liste.

Returns:

la capacité actuelle de cette liste

Specifications: pure

ensures \result >= taille();

---

assurerCapaciteMinimum

`public void` **`assurerCapaciteMinimum`**`(int capaciteMini)`

Augmente, si nécessaire, la capacité de cette instance de `ListeTableau` pour assurer qu'elle puisse contenir au moins le nombre d'elements spécifié par le paramètre.

Parameters:

`capaciteMini` - la capacité minimum souhaitée

Specifications:

ensures capacite() >= capaciteMini;
ensures taille() == \old(taille());
ensures (\forall int i; i >= 0 && i < taille(); get(i) == \old(get(i)));

---

ajusterALaTaille

`public void` **`ajusterALaTaille()`**

Ajuste la capacité de cette instance de `ListeTableau` au nombre d'éléments actuellement présents dans la liste. Une application peut utiliser cette opération pour minimiser l'espace mémoire utilisé par cette instance de `ListeTableau`.

Specifications:

ensures capacite() == taille();
ensures taille() == \old(taille());
ensures (\forall int i; i >= 0 && i < taille(); get(i) == \old(get(i)));

---

equals

`public boolean` **`equals`**`(Object obj)`

Teste l'égalité entre l'objet spécifié et cette liste. Renvoie `true` si et seulement si l'objet spécifié est aussi une `ListeTableau,` , que les deux listes ont le même nombre d'éléments et que toutes les paires d'éléments correspondants dans les deux listes sont *equal* . En d'autres termes, deux listes sont dites *equal* si elle contiennent les mêmes elements dans le même ordre.

Overrides:

`equals` in class `Object`

Parameters:

`obj` - l'objet à comparer avec cette liste.

Returns:

`true` si l'objet spécifié est *equal* à cette liste.

Specifications: pure

also

ensures (obj instanceof ListeTableau && taille() == ((ListeTableau) obj).taille())
==> \result <==> (\forall int i; 0 <= i && i < taille();
(get(i) == null && ((ListeTableau)obj).get(i) == null)
|| (get(i).equals(((ListeTableau)obj).get(i))));
ensures (this == obj) ==> \result ;
ensures !(obj instanceof ListeTableau && taille() == ((ListeTableau)obj).taille()) ==> !\result ;
ensures (obj == null) ==> !\result ;

---

clone

`public Object` **`clone()`**

Renvoie une copie de surface de cette instance de `ListeTableau` (Les éléments eux-mêmes ne sont pas copiés).

Overrides:

`clone` in class `Object`

Returns:

un clone de cette instance de `ListeTableau` .

Specifications: pure

also

ensures \result != null;
ensures \result != this;
ensures \result.equals(this);
ensures \typeof(\result) == \typeof(this);
ensures ((ListeTableau)\result).taille() == taille();
ensures (\forall int i; i >= 0 && i < taille(); ((ListeTableau)\result).get(i) == get(i));
