Epreuve de Programmation Orientée Objet
Licence Informatique *

7 septembre 2004

Durée : 2 heures.
Tous les documents sont interdits.

1 Programmation JAVA et définition d'assertions

Ce qui est évalué dans ces deux exercices

Votre capacité à écrire un programme JAVA syntaxiquement correct et qui fonctionne sans erreurs. Plus précisément, il s'agit d'évaluer vos capacités :

- à donner une implémentation partielle d'une classe à partir de son interface (i.e. commentaires et assertions de la classe et des méthodes) ;
- à effectuer des choix corrects dans le choix des attributs à définir et de leur type ;
- à effectuer des choix corrects pour la visibilité des attributs ;
- à définir et utiliser des tableaux ;
- à définir les assertions d'une méthode en conformité avec les commentaires donnés et avec le contrat de la classe.

Exercice 1.1 Implémentation partielle de la classe `ListeTableau`

Donnez le code source de la classe `ListeTableau` dont l'interface (i.e. commentaires et assertions) est donnée en annexe. En ce qui concerne les attributs, vous devez définir **tous** les attributs nécessaires à l'implémentation de **l'ensemble** de la classe. Pour l'implémentation des méthodes, vous vous limiterez aux méthodes suivantes :

- les deux constructeurs ;
- `void ajouter(Object element)`
- `void ajusterALaTaille()`
- `Object clone()`
- `boolean equals(Object obj)`
- `Object set(int index, Object element)`
- `Object retirer(int index)`

Par ailleurs, votre implémentation devra respecter les contraintes suivantes :

- l'implémentation de chaque méthode doit être cohérente avec le commentaire associé présent dans la définition de `ListeTableau` ;
- l'implémentation de chaque méthode doit respecter scrupuleusement les assertions de `ListeTableau` ;
- en cas d'ambiguïté dans les commentaires votre implémentation devra se conformer aux assertions ;

Exercice 1.2 Compléter le contrat de la classe `ListeTableau` en y ajoutant les assertions des méthodes :

- `boolean contient(Object element)`
- `int indexDe(Object element)`

Ces assertions devront être compatibles et cohérentes avec les commentaires de ces deux méthodes ainsi qu'avec le contrat global du reste de la classe.

*M. Champesme – Département d'Informatique – Institut Galilée

2 Héritage et programmation par contrat

Ce qui est évalué dans cet exercice

Votre capacité à définir une classe en utilisant l'héritage et à écrire les assertions spécifiant de manière aussi complète que possible le comportement des constructeurs et des méthodes de cette classe en cohérence avec le contrat de la classe.

Présentation du problème

On souhaite maintenant ajouter des fonctionnalités à la classe `ListeTableau` en lui ajoutant des méthodes prenant en paramètres des listes d'éléments, plutôt que des éléments isolés. Pour cela, plutôt que de modifier la classe `ListeTableau`, nous allons utiliser l'héritage pour définir une nouvelle classe nommée `ListeTableauEtendue` possédant **toutes** les fonctionnalités de la classe `ListeTableau` (y compris pour la création d'instances) plus de nouvelles méthodes.

Exercice 2.1 Donnez le code source complet ainsi que les assertions de la classe `ListeTableauEtendue`. Les nouvelles méthodes de cette classes sont définies de la manière suivante :

---

ajouterTout

`public boolean` **ajouterTout**`(ListeTableau liste)`

Ajoute tous les éléments de la liste spécifiée à la fin de cette liste en respectant l'ordre des éléments dans la liste spécifiée. Le comportement de cette opération est indéfini si la liste spécifiée est modifiée pendant l'exécution de cette méthode (cela implique que le comportement de cette opération est indéfini si la liste spécifiée est cette liste et que cette liste n'est pas vide).

Paramètres :

`liste` - les éléments à ajouter à cette liste.

Renvoie : `true` si l'exécution de cette méthode a modifié cette liste.

---

contientTout

`public boolean` **contientTout**`(ListeTableau liste)`

Renvoie `true` si cette liste contient tous les éléments de la liste spécifiée.

Paramètres :

`liste` - la liste dont on doit tester l'inclusion dans cette liste.

Renvoie : `true` si cette liste contient tous les éléments de la liste spécifiée.

---

retirerTout

`public boolean` **retirerTout**`(ListeTableau liste)`

Enlève de cette liste tous les éléments qui sont contenus dans la liste spécifiée.

Paramètres :

`liste` - la liste qui détermine quels éléments doivent être enlevés de cette liste.

Renvoie : `true` si l'exécution de cette méthode a modifié cette liste.

A Annexes