Semaine table non triée

Matière
- Table de taille variable
- Algorithmes classiques sur les tableaux non triés

Objectifs
- Comprendre la différence entre la taille logique et la taille physique d'une table
- Découvrir les méthodes classiques telles que l'insertion, la suppression et la recherche dans un tableau non trié
- Ecrire des méthodes plus complexes ($O(N^2)$)

Mise en garde
Vous allez implémenter des algorithmes classiques.
La tentation est grande de recopier tel quel un algorithme trouvé dans le cours ou sur le net.
Ce n'est pas une bonne idée !
Vous devriez, plutôt que de recopier ou mémoriser du *code*, **comprendre** et retenir les ***principes moteurs*** de l'algorithme et vous **entraîner** à reconstituer celui-ci.

Remarque sur les classes de tests :
Il faut pouvoir tirer des conclusions du message qui s'affiche lorsqu'une méthode échoue.
N'allez pas chercher des erreurs là où il n'y en a pas !

Par exemple :

```
test 4 : ajout d'un participant deja inscrit (ajout impossible)
le nombre max de participants : 5
les participants deja inscrits : [p1, p2, p3]
le participant a ajouter : p2
le test 4 a echoue (boolean renvoye). Attendu=false recu=true
```

Le test 4 a échoué, mais cela signifie que les 3 premiers tests ont réussi !
Le test 4 propose un ajout impossible. Un participant ne peut se retrouver qu'une seule fois dans la table. Avez-vous pensé à ce cas ?

Exercices obligatoires

A La croisière

Une croisière est organisée par la chaîne de télévision IPL-TVI. Certains heureux téléspectateurs pourront y participer. Une fois le nombre de participants fixé, les inscriptions peuvent débuter !

Version 1

Complétez la classe *ParticipantsV1*.
Cette classe a comme attributs une table des participants et la taille logique de cette table.
Il s'agit d'une table non triée sans trou !
Pour une question d'efficacité, lors de la suppression, le participant est remplacé par celui qui se trouve en dernière position.

Testez la classe avec la classe *TestParticipantsV1*.

Version 2

Complétez la classe *ParticipantsV2*.
Dans cette table l'ordre des ajouts doit être respecté.
Les ajouts se font bien les uns à la suite des autres.
Mais, attention, lors de la suppression, des décalages seront donc nécessaires.
Testez la classe avec la classe *TestParticipantsV2*.

B Un tableau non trié d'entiers

Complétez les méthodes de la classe *TableauNonTrieDEntiers*.
La table est non triée et sans trou.
L'ordre des ajouts ne doit pas être obligatoirement respecté.
Pour toutes les méthodes de suppression, vous essayerez d'optimiser le nombre de permutations.
Testez ces méthodes via la classe *TestTableauNonTrieDEntiers*.

Exercice supplémentaire

C Un tableau non trié d'entiers – ordre des ajouts respecté

C1 Complétez les méthodes `supprimerPremiereOccurrence()` et `supprimerToutesLesOccurrences()` de la classe *TableauNonTrieDEntiersV2*.
La table est non triée et sans trou.
L'ordre des ajouts doit être respecté.
Testez ces méthodes via la classe *TestTableauNonTrieDEntiersV2*.

Exercices défis

C2

La méthode `supprimerToutesLesOccurrences()` de la classe *TableauNonTrieDEntiersV2* peut être écrite en O(N) !

Si votre méthode fait appel à la méthode `supprimerLaPremiereOccurrence()` ou si elle contient une boucle du style :

```
for (int j = i; j < nombreDEntiers-1; j++) {
   tableDEntiers[j]=tableDEntiers[j+1];
}
```

Votre méthode est en $O(N^2)$!

Revoyez cette méthode si c'est le cas.
Demandez un peu d'aide à un professeur…

C3

Ecrivez la méthode `supprimerTousLesExAequos()` de la classe *TableauNonTrieDEntiersV2* .