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 ($O(N^2)$)

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 !

Complétez la classe *Participants*. Cette classe contient une table des participants et la taille logique de cette table. Il s'agit d'une table 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 *TestParticipants*.
Il faut pouvoir tirer des conclusions du message qui s'affiche lorsque la 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 ?

B Les drapeaux

B1 Un drapeau belge d'enfants.

Mise en situation :

L'école a demandé à tous les enfants de venir habillés en noir, jaune ou rouge afin de constituer un immense drapeau belge.

Chaque écolier va être placé un à un dans le drapeau.
Pour éviter le gros chambardement, il faut que ces ajouts engendrent le moins de déplacements possibles.

Imaginez toutes des chaises alignées.
Pour placer un enfant habillé en noir sur la première chaise, il faudrait que tous les enfants déjà installés se décalent (O(N)).
On pourrait simplement déplacer le premier enfant habillé en rouge sur la première chaise inoccupée en fin de drapeau.
Ce déplacement libère une chaise sur laquelle, on pourrait placer le premier enfant habillé en jaune.
Ce déplacement libère une place pour l'enfant habillé en noir. Avec cette solution, seulement 2 enfants devront se déplacer ! (O(1))
Exemple : on veut ajouter nick (noir)

tableTriee avant :

nora	nico	noel	nestor	julie	jo	rene	remi	robin	

Pour ajouter nick, 2 déplacements sont nécessaires :

nora	nico	noel	nestor	julie	jo	rene	remi	robin	

tableTriee après ajout de nick :

nora	nico	noel	nestor	**nick**	jo	**julie**	remi	robin	**rene**

Il faut également prévoir le désistement d'un enfant.
Comme on ne veut pas de chaise inoccupée en plein drapeau, il faudrait que tous les enfants qui le suivent se décalent d'une chaise (O(N)). En réfléchissant un peu, on se rend compte qu'il y a moyen d'occuper la chaise libérée en ne faisant bouger que 3 enfants maximum ! (O(1))

Travail à réaliser :

Pour cet exercice nous vous donnons les classes suivantes :

Ecolier, qui permet de représenter un enfant ayant une couleur. Chaque écolier possède un nom et une couleur parmi noir, jaune ou rouge.

NoirJauneRouge, qui permet de gérer le drapeau. Elle contient un tableau d'écoliers et le nombre d'écoliers de chaque couleur.
Le nombre d'écoliers est limité à `NOMBRE_MAX_ECOLIERS`.
On ne peut retrouver 2 écoliers qui ont le même nom.
La particularité de cette classe est qu'elle maintient le tableau trié suivant les couleurs : d'abord tous les noirs, ensuite tous les jaunes, ensuite tous les rouges.
L'ordre des écoliers d'une même couleur n'a pas d'importance.
Vous allez compléter la **méthode d'ajout** en respectant la *JavaDoc*.
L'algorithme est imposé. Le principe de base est expliqué et illustré dans le document *DrapeauBelge*.

TestEnonceNoirJauneRouge, qui reprend les exemples repris dans le document *DrapeauBelge* qui se trouve sur moodle. **Attention, seuls quelques exemples sont testés**. Ce n'est donc pas parce que quelques exemples passent que forcément vos méthodes sont correctes.

TestPersoNoirJauneRouge qui permet d'introduire vos propres tests.
Pensez à tester des cas particuliers : ajout d'un nom existant, ajout d'un 11ème écolier, …

Pensez à toujours bien tester vos classes.

C Un tableau d'entiers non trié

Complétez les méthodes de la classe *TableauNonTrieDEntiers*.
Testez ces méthodes via la classe *TestTableauNonTrieDEntiers*.

Exercices supplémentaires

B1 Complétez la **méthode de suppression** de la classe *NoirJauneRouge*.
L'algorithme est imposé. Le principe de base est expliqué et illustré dans le document *DrapeauBelge*.

B2 Drapeau bicolore à trier

On remplit aléatoirement toutes les cellules d'un tableau de n cases avec des entiers.

43	18	39	76	27	85	52

On trie ensuite ce tableau afin d'obtenir tous les nombres pairs en premier ensuite les impairs.

52	18	76	27	85	39	43

On vous demande de compléter la classe *DrapeauBicolore* qui contient un tableau d'entiers.
Vous devez pour cette classe compléter la méthode `triBicolore()` qui réorganise le tableau pour que les pairs apparaissent avant les impairs.

L'algorithme de tri est imposé.
Il est suggéré dans le code de la méthode à compléter.
Essayez de le comprendre sans consulter le document *TriBicolore*.
Le document *TriBicolore* reprend toutes les étapes qui ont permis de trier le tableau ci-dessus.

La classe *TestTriBicolore* permet de tester la classe.
Vous devez vérifier les affichages.

Exercices défi

B2 Améliorez la méthode `triBicolore()`.
On se rend compte que beaucoup de permutations ont été faites pour rien en utilisant l'algorithme proposé ! Les impairs sont systématiquement déplacés, alors qu'ils ne le devraient pas nécessairement.

B3 

Le drapeau Hollandais - Dijksta

L'algorithme du néerlandais E.W. Dijkstra répond au problème suivant :
On remplit aléatoirement toutes les cellules d'un tableau de n cases avec des boules bleues, blanches et rouges.

On trie ensuite ce tableau afin d'obtenir le drapeau hollandais.
Pour cela on doit :

- Utiliser ce seul tableau et le parcourir une seule fois.
- Évaluer pour chaque boule, chaque couleur une seule fois au maximum.

Dans un premier temps, essayez d'imaginer l'algorithme. Ensuite, essayez de comprendre l'algorithme dans le document *AlgorithmeDrapeauHollandais*.

La classe *DrapeauTricolore* contient un tableau de char : b (blue), r (red) et w (white).
Le constructeur de cette classe crée un tableau et le remplit complètement de boules dont les couleurs sont choisies aléatoirement parmi les trois couleurs suivantes : b, r et w
Complétez la méthode `triTricolore()` qui réorganise le tableau pour que les boules apparaissent dans l'ordre : b, w, r.
Complétez la classe *TestDrapeauTricolore* à votre guise.