L'ordre de complexité d'un algorithme

Exercices obligatoires

A Calcul du coût d'algorithmes donnés

Sur moodle, répondez au questionnaire à choix multiples appelé *PileImpl.*
Donnez l'ordre de complexité des méthodes proposées en supposant une implémentation **via table**.
Essayez de faire cet exercices sans avoir les méthodes implémentées sous les yeux.
(La classe *PileImpl* est donnée dans le répertoire classes Java Plus de la semaine1.)

B PileImpl // ArrayList

Comme ceci a été fait dans le diaporama *VecteurImpl_ArrayList*, complétez le tableau suivant :

PileImpl<E>	**ArrayList<E>**
`PileImpl()`	`ArrayList()`
`PileImpl(int capacite)`	
`int taille()`	
`boolean estVide()`	
`void push(E element)`	
`E pop()`	
`E sommet()`	

Vérifiez vos réponses :
La solution se trouve sur moodle dans le dossier solution1. Le document s'appelle *BSol.*

C Applications utilisant la classe de l'API Java *ArrayList*

C1 Consigne de gare

Une solution de l'exercice de la semaine dernière se trouve dans le répertoire *classes Java1* de cette semaine.
Comparez votre solution à celle donnée.
Si c'est nécessaire, **modifiez** votre solution.

a) Vous allez compléter la classe *ConsigneLIFO*.
Elle propose les mêmes méthodes que celles de la classe *Consigne*.
Pour la pile, vous utiliserez un objet de la classe *ArrayList*.
La classe *TestConsigneLIFO* permet de tester cette classe.

b) Dans la classe que vous avez implémentée, une pile sert à stocker les casiers inoccupés. Malheureusement, nous pouvons remarquer à l'usage que le système qui en découle n'est pas bien pratique.
Le prochain casier attribué correspond toujours au dernier libéré. Si le voyageur qui récupère ses bagages, prend du temps, le voyageur qui s'est vu attribuer ce même casier doit attendre !
Ecrivez une nouvelle classe afin de remédier à ce problème. Complétez la classe *ConsigneFIFO*.
Vous utiliserez toujours un objet de la classe *ArrayList* pour y placer les casiers inoccupés, mais il faudra l'utiliser autrement.
La classe *TestConsigneFIFO* permet de tester cette classe.

D *ArrayList* : coûts des méthodes

D1

Sur moodle, répondez au questionnaire à choix multiples appelé *ArrayList.*

Pour les plus curieux :
Cliquez sur le lien suivant (Ctrl+clic) et découvrez le code de la classe *ArrayList* :

http://hg.openjdk.java.net/jdk8/jdk8/jdk/file/tip/src/share/classes/java/util/ArrayList.java

ou une version plus récente, mais moins accessible
https://github.com/openjdk/jdk/blob/master/src/java.base/share/classes/java/util/ArrayList.java

Si vous avez des difficultés pour vous en sortir parmi ce gros millier de lignes de code, consultez le document *ArrayListExtraitsSources*.

Pour les moins curieux :
L'*ArrayList* et notre implémentation de la classe *Vecteur* possèdent les mêmes coûts !

D2

La table suivante donne les coûts **espérés** des méthodes de la classe Consigne :

METHODE	COUT
`resteUnCasierLibre()`	O(1)
`attribuerCasierLibre()`	O(1)
`libererCasier()`	O(1)

Vérifiez si les méthodes des différentes implémentations écrites possèdent bien ces coûts !

Consigne (semaine 1) :

METHODE	COUT	Oui ?
`resteUnCasierLibre()`	O(1)	✓
`attribuerCasierLibre()`	O(1)	✓
`libererCasier()`	O(1)	✓

ConsigneLIFO :

METHODE	COUT	Oui ?
`resteUnCasierLibre()`	O(1)	
`attribuerCasierLibre()`	O(1)	
`libererCasier()`	O(1)	

ConsigneFIFO :

METHODE	COUT	Oui ?
`resteUnCasierLibre()`	O(1)	
`attribuerCasierLibre()`	O(1)	
`libererCasier()`	O(1)	

Les solutions se trouvent sur moodle dans le dossier solution1. Le document s'appelle *D2Sol.*