Les files implémentées via des tables

Exercices obligatoires

A Implémentation de l'interface *File*

A1 Complétez le document *A1*.
La solution de cet exercice se trouve sur moodle dans le dossier *solutions*. Elle est là pour vérifier vos réponses après avoir terminé l'exercice ! Le document s'appelle *A1Sol*.

A2 Implémentez l'interface *File*.
Complétez la classe *FileImplViaTable*.
Testez cette classe via la classe *TestFileImplViaTable*. Cette classe propose un menu.
Ce menu permet de tester les **exemples du diaporama**.
Ce menu propose également de tester le scénario repris dans l'exercice A1.
Testez d'abord les méthodes `defile()` et `enfile()` séparément avant de vérifier les implications des unes sur les autres via le scénario.

A3 Sur moodle, répondez au questionnaire à choix multiples *FileImplViaTable*, en supposant que cette implémentation de l'interface *File* utilise une table circulaire.

B. Applications utilisant la classe de l'API Java *ArrayDeque*

B1 Consigne de gare

ConsigneFIFO en utilisant pour la file une *ArrayList* :

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

ConsigneFIFO en utilisant pour la file une *ArrayDeque* :

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

Modifiez votre classe *ConsigneFIFO* en utilisant pour la file une *ArrayDeque* !

B2 Serveur de demandes d'impressions avec priorité.

Un serveur d'impression peut recevoir à tout moment des demandes d'impression de documents. Chaque demande est placée dans une file, en attente d'être traitée.
Dès que l'imprimante est disponible, la demande la plus prioritaire est prise en charge.
Si deux demandes ont la même priorité, c'est la plus ancienne qui sera imprimée (FIFO).

La classe *DemandeImpressio*n permet de mémoriser pour chaque demande d'impression :

- le nom du document à imprimer
- son chemin d'accès au document à imprimer
- le nom de l'utilisateur à l'origine de cette demande
- une priorité

L'attribut `priorite` permet de mémoriser la priorité avec laquelle la demande doit être traitée [0..9].
Une demande de priorité 9 est plus prioritaire qu'une demande de priorité 0. Elle sera imprimée avant.

On vous demande de compléter la classe *ServeurImpressions.*

Cette classe va contenir 10 files d'attentes d'impression (*ArrayDeque*).
Ces 10 files sont stockées dans un tableau.

Pensez à tester cette classe à l'aide de la classe *TestServeurImpressions*.
Celle-ci propose un scénario.

(Remarque : la classe *ServeurImpressions* reçoit des demandes qui ont bien une priorité entre 0 et 9. C'est le constructeur de la classe *DemandeImpression* et la méthode *setPriorite()* qui s'occupent de tester la validé des paramètres!)

C *Deque*

Un *deque* (*double ended queue*) est une structure de données dans laquelle les ajouts et les retraits peuvent se faire aux 2 extrémités.

C1 Complétez le document *Deque* qui se trouve sur moodle.
Pour mieux comprendre le contenu qu'on attend de votre part, nous avons placé sur moodle un document *File* complété avec les mêmes demandes.
💣Mise en garde :
L'implémentation qui va garantir un maximum d'efficacité n'est pas évidente !
Faites-la **valider** par un professeur !

C2 Ecrivez une interface *Deque*.
Ecrivez la classe *DequeImpl* qui implémente cette interface.
Pour tester cette classe, écrivez la classe *JeuDeTestsDeque*.
Cette classe propose le jeu de tests que vous avez écrit.

Pour vous aider à démarrer, on vous a donné la classe *JeuDeTestsFile*.
Cette classe propose le jeu de tests qui se trouve dans le document *File*.
Elle utilise une méthode `assertEquals()`pour vérifier les résultats attendus.
Lisez bien la *JavaDoc* de cette méthode.

Exemples :

N° test	Instruction	Exception ?	return attendu	file.taille() attendu	file.toString() attendu
4	`file.defile()`		a	1	b

```
//test 4
p=file.defile();
assertEquals("test 4 ko", 'a', p);
assertEquals("test 4 ko", 1, file.taille());
assertEquals("test 4 ko", " b", file.toString());
```

N° test	Instruction	Exception ?	return attendu	file.taille() attendu	file.toString() attendu
7	`file.premier()`	FileVideException			

```
//test 7
try{
   file.premier();
   System.out.println("test 7 ko");
   System.exit(0);
}catch (FileVideException ex){
}
```

Exercice supplémentaire

C3
Une implémentation d'un *deque* via table non circulaire avec les éléments centrés !
Faites la question 3a de l'examen de juin 2017 après-midi. N'implémentez pas l'itérateur.
Vous trouverez l'énoncé et les classes à compléter dans le répertoire *Examens* sur moodle.

Exercice défi

B3 Variante :

S'il y a sans cesse des créations de demandes de priorité élevée, une demande de faible priorité déjà présente risque de n'être jamais traitée. Pour remédier à ce problème, nous supposerons qu'une telle demande en tête de file cédera son tour au maximum 3 fois. A la 4ème fois, sa priorité sera augmentée de 1 et elle sera placée en queue de la file de priorité supérieure.
Complétez la classe *ServeurImpressionsVar*.
Tester cette classe à l'aide de la classe *TestServeurImpressionsVar*.