TD Files FIFO

Travail préliminaire

Q 1 . *Types et méthodes génériques* :

- utilisation,
- création (classes et méthodes),
- gestion du typage

Voir le document sur le portail, et toute autre ressource complémentaire que vous trouverez pour vous permettre de mieux comprendre la notion...

Q 2 . *Classes internes*

- C'est quoi ? (faire une recherche préalable au TD)
- Quel(s) avantage(s) ?

Exercice 1 : File (FIFO).

On modélise ici une structure pouvant stocker une suite d'exactement L objets d'un type donné avec $L > 0$. L est constant et est appelé *largeur* de la file.
Cette suite se comporte comme une file FIFO (*first in, first out*) d'objets. Considérons que dans la file les objets sont rangés de la gauche (le premier) vers la droite (le dernier). Cela signifie alors qu'il est possible d'ajouter des objets à la file et que cet ajout se fait par la droite. L'objet ajouté se place en fin de la file à la position la plus à droite. Tous les éléments qui le précèdent sont alors décalés d'une position vers la gauche. L'élément qui se trouvait initialement en première position est alors sorti de la file[1].
Les quatre fonctionnalités de ces files sont :

- `getLargeur` renvoie la largeur de la file,
- `raz` force tous les éléments à la valeur `v` passée en paramètre,
- `ajoute` ajoute un élément, passé en paramètre, à la file. La méthode renvoie l'élément qui a été sorti de la file.
- `toString` renvoie sous forme de `String` une copie du contenu de la suite d'éléments de la file (on concatènera de gauche à droite les `toString` de chacun des éléments).

Q 1 . Définir le type `FileFIFO` : donnez un diagramme UML.

On définira un constructeur avec lequel on fixe la largeur de la file ainsi qu'une valeur par défaut pour tous ses éléments.

Q 2 . Donnez un code pour ce type.

Q 3 . On souhaite que ce type implémente l'interface `Iterable`.

Faites le nécessaire :

- la classe d'itérateur sera placée en classe interne,
- la méthode optionnelle `remove` ne sera pas implémentée,
- vous gérerez l'aspect *fail-fast* de l'itérateur[2].

[1] parmi tous les élements c'est lui qui était entré (*in)* le premier dans la file, il en sort (*out*) donc le premier.
[2] voir la javadoc de `java.util.ConcurrentModificationException`