Laboratoire 05 – Implémentation du TDA File et levée / gestion d'exceptions

OBJECTIFS :

- Pouvoir implémenter un TDA (le TDA File) avec un tableau statique.
- Être capable de lever et gérer des exceptions.
- Vous exercer à bien écrire les commentaires de documentation avec les antécédents, les conséquents, et @throws s'il y a lieu.
- Continuer à vous exercer à coder en Java, en respectant les bonnes pratiques, et les normes de programmation (style Java).

EXERCICE 1

1. Écrivez la définition de la classe `File.java`, qui implémente les services d'une file avec un tableau « circulaire », comme vu au cours. Prenez soin de bien écrire les commentaires de documentation (description, antécédents, conséquents, @param, @return, @throws, etc.). Référez-vous aux notes de cours, et inspirez-vous de la classe `Pile.java` dont on a vu le code.

   ***Notes** :*
   - *Respectez le nom et le type des paramètres des méthodes ci-dessous pour pouvoir utiliser la classe de tests `TestFile.java` qui vous est fournie.*
   - *Vous pouvez (et devriez) écrire des méthodes privées (private) utilitaires pour bien séparer votre code en sous-programmes (bonne séparation fonctionnelle).*
   - *Vous devez importer la classe `java.util.NoSuchElementException` pour pouvoir utiliser cette Exception dans vos méthodes.*

   Votre classe doit contenir les services suivants :

   Constructeurs :
   `public File(int capaciteInit)` : construit une file vide avec la capacité donnée.
   `public File()` : construit une file vide avec une capacité = 10

   Méthodes :
   - `getNbrElements()` : retourne le nombre d'éléments dans la file
   - `estVide()` : retourne si la file est vide
   - `consulterDebut()` :
     - retourne l'élément au début de la file, sans la modifier.
     - lève `NoSuchElementException` si la file est vide avant l'appel
   - `enfiler(Object elem)`
     - enfile `elem` en fin de file
     - si la file est pleine avant l'ajout, il faut agrandir la file de 10 places supplémentaires avant d'enfiler `elem`
   - `defiler()` :
     - retire et retourne l'élément au début de la file
     - lève `NoSuchElementException` si la file est vide avant l'appel
   - `vider()` : retire tous les éléments de la file

2. Dans votre projet, importez la classe `TestFile.java`, fournie avec cet énoncé, pour tester votre implémentation de file.

EXERCICE 2

Créez un autre projet et importez-y votre classe `File.java` complétée ainsi que la classe fournie `TestFile.java`. Modifiez les méthodes `défiler()` et `consulterSommet()` pour qu'elles lèvent une `Exception` plutôt qu'une `NoSuchElementException`.

Vous remarquerez que la classe `TestFile` ne compile plus, car `NoSuchElementException` était une exception implicite, et n'avait pas besoin d'être gérée. Ce n'est pas de même pour l'exception `Exception`, qui est explicite. Modifiez la classe `TestFile` pour qu'elle compile et fonctionne avec ce changement.

Vous devrez aussi modifier, dans la classe `TestFile`, tous les blocs `try… catch` qui y sont déjà pour que les tests soient en accord cette nouvelle spécification.

Lorsque tout compile, retestez votre nouvelle classe `File` avec votre classe `TestFile` modifiée.

BON TRAVAIL !