Laboratoire 06 – Implémentation du TDA Liste avec position courante

OBJECTIFS :

- Coder une implémentation statique du TDA liste avec position courante, à l'aide d'un tableau.
- Vous exercer à lever et gérer des exceptions.
- Continuer à vous exercer à coder en Java, en respectant les bonnes pratiques, et les normes de programmation (style Java).

Avant de commencer, créez un nouveau projet et importez-y les classes fournies : `ListePleineException`, `ListeVideException`, et `ListePosCouranteTableau`.

EXERCICE 1

Votre travail consiste à implémenter les méthodes de la classe `ListePosCouranteTableau` qui, pour le moment, sont vides. Vous devrez aussi déclarer les attributs d'instance en respectant le principe d'encapsulation.

Lisez bien les commentaires Javadoc pour comprendre l'implémentation demandée et comprendre ce que les méthodes doivent faire exactement. Comme toujours, pensez à découper vos méthodes en sous-programmes (privées), lorsque cela est pertinent.

Notez que vous avez à faire une implémentation statique (contrairement à l'implémentation de Pile et File vue la semaine dernière). Cela signifie que votre liste a une capacité fixe qui, une fois atteinte (nbrElements = capacité), ne permet plus l'ajout d'autres éléments.

Comme toujours, codez et testez une méthode à la fois. N'attendez pas d'avoir codé toutes les méthodes avant de faire vos tests.

À la fin, lorsque vous pensez que tout fonctionne, importez la classe fournie `TestsListePosCourante` dans votre projet, et exécutez les tests. Déboguez et corrigez au besoin.

EXERCICE 2

Cet exercice a pour but de vous faire manipuler des variables de type `ListePosCouranteTableau`.

1. Créez une nouvelle classe nommée `UtilisationListePosCourante`.

2. Dans la méthode `main` de cette nouvelle classe, instanciez une variable nommée `liste`, de type `ListePosCouranteTableau`, et ajoutez-y des éléments pour obtenir la liste suivante :
   `[ 5, 3, 10, 7, null, 23, 9, 4, 12 ]`

3. Dans cette même classe, codez la méthode (*static*) `contientElement` qui prend en paramètre une liste de type `ListePosCouranteTableau`, et un élément de type `Object`. La méthode doit tester si l'élément donné appartient à la liste donnée ou non (retourne un booléen).

Notez que les éléments dans votre liste sont des `Integer`, et que cette classe redéfinit la méthode `equals` (de la classe `Object`). Pour tester l'égalité entre deux éléments, utilisez cette méthode `equals` (et non l'opérateur == qui compare les références seulement) : `e1.equals(e2)`.

Pour votre implémentation, supposez que la liste donnée n'est pas `null` (mais elle peut contenir des éléments `null`). Aussi N'UTILISEZ PAS UNE BOUCLE FOR pour parcourir la liste, car on ne sait pas combien d'éléments on aura à consulter avant de trouver (ou non) l'élément cherché.

4. De retour dans la méthode `main`, écrivez les appels suivants pour tester que votre méthode `contientElement` fonctionne correctement :

```
//Tests de la methode contientElement en testant la recherche
//d'un objet existant au debut, a la fin, au milieu,
//d'un objet null, et d'un objet inexistant.
System.out.println(contientElement(liste, 5));     //true
System.out.println(contientElement(liste, 12));    //true
System.out.println(contientElement(liste, 7));     //true
System.out.println(contientElement(liste, null));  //true
System.out.println(contientElement(liste, 11));    //false

//vider la liste
liste.vider();

//test dans une liste vide
System.out.println(contientElement(liste, 1));     //false
```

BON TRAVAIL !