TP 1 : TABLEAU PARTIEL

Introduction

Pour le premier tp, vous allez construire une classe qui permet la *coupe* de tableau. Cette classe, du nom de `TableauPartiel`, va permettre d'extraire un sous-tableau avec un descripteur de `Coupe`. Le sous-tableau extrait sera une référence au tableau original. Une hiérarchie de classe vous est fourni pour décrire les différentes coupes possibles. Une liste des méthodes à implémenter est donnée. Vous pouvez ajouter des méthodes aux classes, mais ne pouvez pas modifier la signature de celles demandées.

Description

Coupe

Une coupe est un descripteur. Elle décrit le sous-tableau que nous voulons extraire du tableau de base. Il y a trois descripteurs qui hérite de la classe de base `Coupe`.

```
public abstract class Coupe {
}
```

Le premier descripteur nous donne une coupe entre deux indices. Le tableau utilisé dans ce cas va inclure les éléments du premier indice inclus jusqu'au deuxième. L'élément du deuxième indice est exclu du nouveau tableau.

```
public class CoupeDeA extends Coupe {
	private int _debut;
	private int _fin;

	public CoupeDeA( int a_debut, int a_fin ){
		assert 0 <= a_debut;
		assert a_debut <= a_fin;

		_debut = a_debut;
		_fin = a_fin;
	}

	public int debut(){
		return _debut;
	}

	public int fin(){
		return _fin;
	}
}
```

Le deuxième descripteur nous donne un tableau à partir du premier élément jusqu'à l'élément indiqué, qui sera exclu.

```
public class CoupeA extends Coupe {
	private int _fin;

	public CoupeA( int a_fin ){
		assert 0 <= a_fin;

		_fin = a_fin;
	}

	public int fin(){
		return _fin;
	}
}
```

Le dernier descripteur nous donne un indice de départ et nous prenons tout jusqu'à la fin du tableau.

```
public class CoupeDe extends Coupe {
	private int _debut;

	public CoupeDe( int a_debut ){
		assert 0 <= a_debut;

		_debut = a_debut;
	}

	public int debut(){
		return _debut;
	}
}
```

Nous allons maintenant regarder la classe pour les tableaux.

TableauPartiel

Cette classe contient les méthodes que vous devez construire. Il est important de prendre note que certaines méthodes/constructeurs vont copier le tableau alors que d'autre vont seulement copier une référence.

Constructeurs :

Le premier constructeur prend un tableau **Java** et le transforme en `TableauPartiel`. Cette transformation va demander de construire une copie du tableau reçu en argument, ne prenez pas seulement la référence. Nous ne voulons pas qu'il y ait de lien entre notre tableau et celui qui est utilisé pour le construire. Le deuxième constructeur fait une copie d'un `TableauPartiel`. Cette copie n'est pas une référence vers le tableau original. C'est une version sans lien. Nous allons maintenant décrire les méthodes que vous devez coder dans cette classe.

```
public class TableauPartiel<E> {
      public TableauPartiel( E[] a_tableau )
      public TableauPartiel( TableauPartiel<E> a_tableauPartiel )

      public int position( E a_element )
                  throws ElementNonPresent
      public boolean contient( E a_element )
      public TableauPartiel<E> coupe( Coupe a_coupe )
                               throws IndexHorsPorte
      public E[] elements()
      public boolean estVide()
      public E get( int a_position )
                throws IndexHorsPorte
      public void remplacer( E a_ancien, E a_nouveau )
                  throws ElementNonPresent
      public void set( int a_position, E a_element )
                  throws IndexHorsPorte
      public int taille()
}
```

Méthodes :

Vous devez construire neuf méthodes.

1. **`public boolean estVide()`.**
   Cette méthode retourne **`true`** si le tableau courant est vide. Un tableau vide est un tableau de taille zéro.
2. **`public int taille()`.**
   Cette méthode retourne le nombre de case que contient le tableau.
3. **`public E get( int a_position ) throws IndexHorsPorte.`**
   Cette méthode retourne une référence sur l'élément situé à la position donnée. Si la position donnée n'est pas valide, vous devez lever une exception `IndexHorsPorte`. Vous devez aussi construire cette exception.
4. **`public void set( int a_position, E a_element ) throws IndexHorsPorte.`**
   Cette méthode assigne une valeur à la position indiquée. Vérifiez la position pour qu'elle soit dans l'intervalle permise pour le tableau (`IndexHorsPorte`).
5. **`public boolean contient( E a_element ).`**
   Cette méthode cherche un élément dans le tableau et retourne **`true`** si l'élément est présent, false sinon.
6. **`public int position( E a_element ) throws ElementNonPresent.`**
   Cette méthode retourne la position (l'index) d'un élément. Si l'élément est présent plus d'une fois, alors l'élément ayant la position la plus petite est retourné. Si l'élément n'est pas dans le tableau, une exception est lancée : `ElementNonPresent`.
7. **`public void remplacer( E a_ancien, E a_nouveau ).`**
   Cette méthode trouve les occurrences d'un élément et les remplaces par le nouvel élément.

8. `public E[] elements()`.
   Cette méthode retourne un nouveau tableau **Java** contenant les éléments du `TableauPartiel`. C'est une copie du tableau qui doit être retournée.
9. `public TableauPartiel<E> coupe( Coupe a_coupe ) throws IndexHorsPorte`.
   Finalement, la méthode `coupe` donne un nouveau `TableauPartiel` à partir de la coupe donnée. Ce nouveau tableau contient une référence sur le tableau d'origine. Donc, les cases modifiées dans ce nouveau tableau seront aussi modifiées dans l'ancien tableau. Le premier élément d'un `TableauPartiel` est adressé par l'indice zéro, même si son indice diffère dans le tableau d'origine. Si la `Coupe` contient un ou des indices non valides, l'exception `IndexHorsPorte` est lancée.

La méthode `equals` de la classe `Object` est utilisée pour les comparaisons dans les méthodes `contient`, `position` et `remplacer`.

Directives

1. Le tp est à faire seul.
2. Vous pouvez ajouter des méthodes aux classes. Vous ne pouvez pas modifier la signature des méthodes déjà données.
3. Commentaire :
   a. Commentez l'entête de chaque classe et méthode. Ces commentaires doivent contenir la description de la méthode et le rôle de ces paramètres.
   b. Une ligne contient soit un commentaire, soit du code, pas les deux.
   c. Utilisez des noms d'identificateur significatif.
   d. Utilisez le français.
4. Code :
   a. Pas de `goto`, `continue`.
   b. Les `break` ne peuvent apparaitre que dans les `switch`.
   c. Un seul `return` par méthode.
5. Indentez votre code. Assurez-vous que l'indentation est faite avec des espaces.

Remise

Remettre le tp par l'entremise de Moodle. Placez vos fichiers `'*.java'` dans un dossier compressé de Window, vous devez remettre l'archive. Le tp est à remettre avant le 3 juin 23 :55.

Évaluation

- Fonctionnalité (7 pts) : des tests partiels vous serons remis. Un test plus complet sera appliqué à votre tp.
- Structure (1 pt) : veillez à utiliser correctement le mécanisme d'héritage et de méthode. Vous ne pouvez pas utiliser `instanceof`.
- Lisibilité (3 pts) : commentaire, indentation et noms d'identificateur significatif.