TP 8bis la classe `DynArray`
Algorithmique – Programmation FIP (ING39)

V. Aponte, P. Courtieu

Les tableaux dynamiques – ou tableaux infinis

Le but de ce TP est d'apprendre à implanter une classe à partir de sa documentation.

On cherche à concevoir une classe permettant de définir et utiliser des objets ayant les mêmes fonctionnalités que les tableaux, mais dans lequels on n'a pas besoin de se soucier du dépassement de capacité (tableau trop petit), ni d'indices en dehors du tableau car *trop grands*. On les appellera *tableaux dynamiques*. Leur principe est le suivant :

- *tout entier i positif ou nul* est considéré comme un *indice valide* de case et donc, on peut lire le contenu de cette case ou le modifier.
- Si on accède à une case d'indice valide (nul ou positif quelconque), la valeur retournée est la dernière à avoir été mise dans cette case, ou elle est 0, si cette case n'a jamais été modifiée.
- Toute case remplie avec la valeur zéro est considérée comme étant *vide*. On peut donc *vider* une case en la mettant à zéro.

Il s'agit donc, de tableaux *virtuellement infinis*, où tout accès à un indice positif ou nul donne lieu à une valeur (0 si jamais modifié, sinon dernière valeur modifiée). Les opérations d'un tableau dynamique permettent d'accéder à une case (`get`), et de modifier une case avec une valeur (`set`). Lorsqu'on accède (`get`) à une case vide (non remplie ou remplie avec zéro), on obtient la valeur par défaut : 0, sinon on obtient la dernière valeur stockée dans cette case. L'opération `values`, retourne un **tableau classique**, contenant toutes les valeurs des cases (vides et non vides) comprises entre l'indice 0 et la dernière case non vide. Par exemple, si le tableau dynamique t contient :

$$t = \{0, 0, 1, 0, 7, 0, 0, 0, 0, 0, 0, 0 \ldots\}$$

où toutes les valeurs après l'indice 4 sont vides. L'opération `t.values()` doit envoyer un tableau classique égal à : `t.values()`$\Rightarrow$ `{0,0,1,0,7}`. Le tableau retourné par `values` est appelé *préfixe non vide* du tableau dynamique. Enfin la méthode `getExtent()` permet d'obtenir la taille du préfixe non vide.
Dans notre exemple, `t.getExtent()`$\Rightarrow$ `5`. Notez que la taille du préfixe de t peut augmenter selon que l'on ajoute une case non vide plus loin que la dernière non vide, ou diminuer si l'on met à zéro la dernière case non vide.

Ce qui vous est demandé

1. Dans un premier temps on se donne une documentation précise des fonctionnalités, sous la forme d'une interface `DynArrayInterface` comportant les méthodes publiques documentées au format (`javadoc`). Dans le projet qui a été créé à votre nom (Consultez `gitlab` et voyez le projet `TpDynArray_votrelogin`).
2. Dans un deuxième temps vous devez concevoir le jeu de tests associé à une classe qui implante l'interface `DynArrayInterface`. Ceci devrait vous prendre environ 30mn. Pour permettre à `netbeans` de typer vos jeux de tests, le mieux est de faire une implantation vide de l'interface (toutes les méthodes lèveront une exception).
3. Dans un troisième temps on fournit une vraie *implantation* de cette classe, en ajoutant des champs (variables d'instance) et en implantant les méthodes de manière à respecter la documentation. ceci devrait vous prendre environ 1h. **Votre implantation doit respecter les contraintes décrites plus bas**.

Contraintes d'implantation

Étant donné qu'on ne peux pas stocker une infinité de cases en mémoire, on implantera cette classe en représentant de manière *finie* ce tableau virtuellement infini. Il y a plusieurs solutions à ce problème. Dans ce TP vous devez impérativement utiliser la solution décrite ici.

L'implantation doit être faite de la façon suivante. La classe contient une variable d'instance `tab` de type `ArrayList <Integer>` représentant le préfixe non vide du tableau infini. `tab` devra nécessairement contenir *au moins* toutes les cases (vides ou non vides) du tableau jusqu'à la dernière case non vide, autrement dit l'intégralité du préfixe non vide du tableau. Il est également très utile de connaître à tout moment la taille exacte de ce préfixe.

Il est évidemment hors de question de stocker **toutes** les valeurs égales à zéro au delà du préfixe non vide du tableau (ce serait impossible !). La principale difficulté sera de faire en sorte que la méthode `set` agrandisse si nécessaire la liste interne à l'objet en modifiant la variable d'instance `tab` et d'implanter correctement la méthode qui renvoie la taille et le tableau préfixe. Voici un exemple du contenu réel de `tab` et le tableau infini qu'il représente :

`tab` :	2	0	0	6	0	0					
Tableau infini :	2	0	0	6	0	0	0	…	0	…	∞

Si on effectue `set(7,22)` on obtiendra (si on ajoute exactement les cases nécessaires) :

`tab` :	2	0	0	6	0	0	0	22				
Tableau infini :	2	0	0	6	0	0	0	22	0	…	0	… ∞