# TD 1

# Le type abstrait File

Dans ce TD, nous allons nous attacher à la programmation d'une file. Une file est une suite d'éléments dans laquelle nous pouvons extraire l'élément placé en tête de la file et ajouter un élément en fin de file.

Dans les deux premières parties de ce TD, nous allons implémenter une file dont la spécification est la suivante :

```
1  public interface File
   {
       /** estVide : déterminer si la file est vide.
        * @return vrai si la file est vide */
       public boolean estVide();
6
       /** estPleine : déterminer si la file est pleine.
        * @return vrai si la file est pleine */
       public boolean estPleine();

11     /** getTaille : taille de la file.
        * @return nombre d'éléments présents dans la file */
       public int getTaille();

       /** ajouter : ajouter x en fin de file.
16      * @pre la file ne doit pas être pleine
        * @param x : valeur à ajouter en fin de file */
       public void ajouter(float x);

       /** supprimer : supprimer l'élément de tête de file.
21      * @pre la file ne doit pas être vide
        */
       public void supprimer();

       /** getTete : renvoie l'élément en tête de file.
26      * @pre la file ne doit pas être vide
        * @return valeur de l'élément en tête de file */
       public float getTete();
   }// File
```

## 1 Implémentation d'une file de réels à l'aide d'un tableau avec décalage

Nous allons implémenter une file de réels en utilisant un tableau pour stocker les éléments. Le mode de fonctionnement sera le suivant : lorsqu'un élément est ajouté à la file, ce dernier est stocké dans la première case libre (case non utilisée pour stocker un élément de la file) du tableau. Lorsqu'un élément est extrait de la file, tous les éléments du tableau sont décalés de manière à ce que la case d'indice 0 du tableau contienne la nouvelle tête de file.

**Exemple** (le symbole / désigne une case non utilisée du tableau – sa valeur est donc indéfinie) :

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| État du tableau après quelques ajouts et suppressions | 5.5 | 1.3 | 15.2 | 1.5 | / | / | / | / | / | / |
| État du tableau après l'ajout de la valeur 20.2 dans la file | 5.5 | 1.3 | 15.2 | 1.5 | 20.2 | / | / | / | / | / |
| État du tableau après le retrait de la valeur située en tête de file | 1.3 | 15.2 | 1.5 | 20.2 | / | / | / | / | / | / |

**Question 1 :** Quels seront les attributs de la classe FileTableau ? Écrivez la déclaration de la classe FileTableau ainsi que de ses attributs.

**Question 2 :** Écrivez le constructeur de la classe FileTableau.

**Question 3 :** Écrivez la méthode supprimer.

**Question 4 :** Écrivez un programme de test. Ce programme devra demander à l'utilisateur de saisir une suite de nombres à insérer dans la file. La saisie s'arrêtera lorsque la file sera pleine ou lorsque l'utilisateur aura saisi un nombre négatif. Une fois la saisie effectuée, le programme affichera le contenu de la file à l'écran.

# 2 Implémentation d'une file de réels à l'aide d'un anneau

Nous allons implémenter une file de réels en utilisant un anneau. Un anneau est défini de manière à ce que chaque élément de l'anneau possède obligatoirement un prédécesseur ainsi qu'un successeur. Il existe plusieurs manières de représenter un anneau ; dans le cas présent, nous allons utiliser un tableau. De manière à définir un anneau de K cases, nous créerons donc un tableau de K éléments.

La gestion de l'anneau s'effectue par un jeu sur les indices lors du parcours du tableau. Le successeur d'un élément d'indice i tel que $0 \leq i < K-1$ est l'élément d'indice $i+1$ ; de la même manière, le prédécesseur d'un élément d'indice $0 < i \leq K-1$ est l'élément d'indice $i-1$.

De plus, le successeur de l'élément d'indice $K-1$ (dernière case du tableau) est l'élément d'indice 0 et le prédécesseur de l'élément d'indice 0 est l'élément d'indice $K-1$.

Sur cette base, les indices du successeur et du prédécesseur d'un élément d'indice i, dans un tableau de taille K, peuvent être calculés via les formules suivantes :

$Successeur(i) = (i+1)\, modulo\, K$

$Predecesseur(i) = (i-1)\, modulo\, K$

Notre implémentation va donc utiliser un anneau pour stocker les éléments de la file. Afin de pouvoir se repérer dans cet anneau et de pouvoir ajouter / supprimer des éléments dans la file, nous aurons constamment besoin de connaitre l'indice de l'élément de tête de file ainsi que l'indice de l'élément de fin de file. La suppression se fera alors par changement de l'indice de l'élément de tête de file et l'insertion s'effectuera sur le successeur du dernier élément de la file.

Voici un exemple de l'état interne de la file utilisant un anneau comme représentation, à l'issue de diverses opérations de suppression et d'insertion (le symbole / désigne une case non utilisée du tableau – sa valeur est donc *indéfinie*).

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| État de la file après quelques ajouts et suppressions | 5.5 | 1.3 | 15.2 | 1.5 | 6.2 | 14.3 | 18.1 | 12 | 5.2 | 8.1 |
| | tête | | | | | | | | | fin |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Suppression de l'élément de tête | / | 1.3 | 15.2 | 1.5 | 6.2 | 14.3 | 18.1 | 12 | 5.2 | 8.1 |
| | | tête | | | | | | | | fin |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Suppression de l'élément de tête | / | / | 15.2 | 1.5 | 6.2 | 14.3 | 18.1 | 12 | 5.2 | 8.1 |
| | | | tête | | | | | | | fin |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Ajout de 17.2 dans la file | 17.2 | / | 15.2 | 1.5 | 6.2 | 14.3 | 18.1 | 12 | 5.2 | 8.1 |
| | fin | | tête | | | | | | | |

**Question 1 :** Quels seront les attributs de la classe FileAnneau ? Écrivez la déclaration de la classe FileAnneau ainsi que de ses attributs.

**Question 2 :** Écrivez le constructeur de la classe FileAnneau. Attention à l'initialisation des attributs !

**Question 3 :** Écrivez les méthodes estPleine et ajouter.

**Question 4 :** Quelle(s) modification(s) faut-il apporter au programme de test de la question précédente pour pouvoir tester cette nouvelle implémentation de File ?

# 3 Une file générique.

**Question 1 :** Changez la spécification du type File pour la rendre générique du type de l'élément stocké dans la file.

**Question 2 :** Changez l'implémentation de la classe FileAnneau de manière à la rendre générique du type de l'élément stocké dans la file.

**Question 3 :** Changez le programme de test de manière à refléter ces modifications.

# Annexe : spécification du TA générique Array

```
1  /**
   * Array : tableau de capacité fixe dont les éléments sont d'un type T
   */
  public class Array<T> {
    /**
6    * Initialiser une nouvelle instance de Array.
     * @param capacite : capacité (nombre de "cases") du tableau
     */
    public Array(int capacite);

11  /**
     * Donne la capacité du tableau
     * @return nombre de "cases" du tableau
     */
    public int length();
16
    /**
     * Donne l'élément d'indice i
     * @param i : indice de l'élément à consulter
     * @pre 0 ≤ i < this.length()
21   * @return valeur de l'élément d'indice i
     */
23  public T get(int i);

    /**
26   * Modifie l'élément d'indice i
     * @param i : indice de l'élément à modifier
     * @pre 0 ≤ i < this.length()
29   * @param v : nouvelle valeur de l'élément d'indice i
     */
31  public void set(int i, T v);
  } // Array
```