# Programmation 1 bis

# Sujet 2 : Création d'une classe Ensemble

## 1 Introduction

On se propose de créer une nouvelle classe `EE` (pour Ensemble d'Entiers) qui permet de manipuler des ensembles de nombres entiers. Il s'agit d'un ensemble au sens mathématique du terme, c'est-à-dire où chaque élément est représenté de manière unique.

Le travail consiste à écrire des méthodes offrant les opérations élémentaires sur les ensembles : appartenance d'un élément à un ensemble, ajout/retrait d'un élément à un ensemble, union, intersection entre ensembles, etc.

Les variables d'instance sont privées et, sauf mention contraire, les méthodes sont publiques. Autre point important : essayer d'utiliser au maximum les méthodes déjà écrites pour faciliter l'écriture d'une nouvelle méthode.

## 2 Création du type `EE`

La classe `EE` devra contenir deux variables d'instance principales. Un tableau `ensTab` contenant les éléments de l'ensemble, sans répétition. (La taille maximum d'un ensemble sera connue dans la classe par `ensTab.length` et sera précisée en paramètre de tous les constructeurs.) L'autre attribut `cardinal` donne le cardinal (nombre d'éléments) de l'ensemble. Ainsi, les éléments de l'ensemble `this` seront les `this.cardinal` premiers éléments du tableau `this.ensTab`, placés dans un ordre quelconque.

L'écriture d'une classe `TestEE` permettra de tester les différentes méthodes au fur et à mesure.

On rappelle qu'un constructeur est une méthode (procédure) qui initialise les variables d'instance de `this` dès que la place mémoire est créée par un `new`. Créer les constructeurs suivants :

1. un constructeur qui crée un ensemble vide ; ce constructeur prend comme paramètre un entier positif `max` qui correspond à la taille maximum de l'ensemble créé, et donc à la taille du tableau créé ; (Les autres constructeurs commenceront par appeler ce constructeur pour initialiser l'ensemble.)
2. un constructeur admettant comme paramètre (en plus de `max`) un tableau d'entiers dont les valeurs sont les éléments de l'ensemble à représenter ;
3. un constructeur par recopie admettant comme paramètre un autre ensemble (la taille maximum de l'ensemble construit est la même que celle de l'ensemble recopié) ;
4. (optionnel) un constructeur admettant comme paramètre (en plus de `max`) une chaîne contenant des entiers séparés par des blancs[1].

[1] On pourra utiliser le constructeur de la classe `Scanner` vu en cours qui permet de lire sur des chaînes (ou bien la méthode `split` d'une instance de `String` et la méthode `Integer.parseInt`).

## 3 Méthodes utiles

1. Ecrire une méthode `toString` renvoyant une chaîne de caractères représentant un ensemble du type $\{n_1, n_2, ..., n_k\}$.[2]
2. Ecrire une méthode `getCardinal` qui retourne le cardinal de l'ensemble.
3. Ecrire une méthode privée `contientPratique` qui retourne -1 si l'entier passé en paramètre n'appartient pas à l'ensemble et retourne sinon l'indice du tableau `this.ensTab` où se trouve l'élément.
4. Ecrire une méthode `contient` qui retourne un booléen indiquant si l'entier en paramètre appartient à l'ensemble.
5. Ecrire une méthode privée `void ajoutPratique (int e)` qui ajoute un nouvel élément $e$ dans l'ensemble courant. Cette méthode a comme pré-requis que l'élément ne se trouve pas déjà dans l'ensemble et que le tableau `this.ensTab` n'est pas plein. Cette méthode privée est un outil de base utilisable dans les méthodes qui suivent.
6. Ecrire une méthode privée `int retraitPratique (int i)`, avec comme pré-requis $0 \leq i < this.cardinal$, qui supprime `this.ensTab[i]` de l'ensemble courant et le renvoie.
7. Ecrire une méthode `estVide` qui teste si l'ensemble courant est vide.
8. Ecrire une méthode `deborde` qui teste si le tableau `ensTab` de l'ensemble courant a atteint la taille maximale autorisée (et ne peut donc pas contenir un élément de plus).

## 4 Méthodes de calcul sur les ensembles

1. Ecrire une méthode `retraitElt` qui retire de l'ensemble courant l'élément entier en paramètre. La fonction renvoie `true` si l'élément appartient à l'ensemble et est retiré physiquement ; `false` sinon.
2. Ecrire une méthode `ajoutElt` qui ajoute un élément en paramètre à l'ensemble courant. La fonction renvoie $-1$ si l'élément ne peut pas être ajouté car le tableau `ensTab` déborde ; elle renvoie 0 si l'élément est déjà présent et n'a pas besoin d'être ajouté physiquement ; elle renvoie 1 si l'élément est ajouté physiquement.
3. Ecrire une méthode `estInclus` qui teste si l'ensemble courant est inclus dans (ou égal à) un autre ensemble passé en paramètre.
4. Ecrire une méthode `estEgal` qui teste si l'ensemble courant est égal à un autre ensemble passé en paramètre.
5. Ecrire une méthode `estDisjoint` qui teste si l'ensemble courant et un autre ensemble passé en paramètre sont disjoints (aucun élément en commun).

   Pour les questions suivantes, vous compléterez les spécifications des méthodes en précisant la taille maximum de l'ensemble renvoyé en fonction de celles de l'ensemble courant et de l'ensemble passé en paramètre.
6. Ecrire une méthode `intersection` qui retourne l'ensemble intersection entre l'ensemble courant et un autre ensemble passé en paramètre.
7. Ecrire une méthode `reunion` qui retourne l'ensemble union entre l'ensemble courant et un autre ensemble passé en paramètre.
8. Ecrire une méthode `difference` qui retourne la différence ensembliste (souvent notée '\') de l'ensemble courant moins un autre ensemble passé en paramètre.

[2] La méthode `toString` est appelée automatiquement lors de l'affichage d'une variable de type EE (par un `System.out.println(ensemble1)` par exemple).