# Ensemble et ensemble chaîné

On souhaite modéliser les notions d'ensemble et d'ensemble ordonnée. Le point de départ est la spécification de l'ensemble donnée au listing 1.

Listing 1 – L'interface Ensemble

```
/** Définition d'un ensemble d'entier. */
public interface Ensemble {
   //@ public invariant estVide() <==> cardinal() == 0;
   //@ public invariant 0 <= cardinal();

   /** Obtenir le nombre d'éléments dans l'ensemble.
    * @return nombre d'éléments dans l'ensemble. */
   /*@ pure helper @*/ int cardinal();

   /** Savoir si l'ensemble est vide.
    * @return Est-ce que l'ensemble est vide ? */
   /*@ pure helper @*/ boolean estVide();

   /** Savoir si un élément est présent dans l'ensemble.
    * @param x l'élément cherché
    * @return x est dans l'ensemble */
   /*@ pure helper @*/ boolean contient(int x);

   /** Ajouter un élément dans l'ensemble.
    * @param x l'élément à ajouter */
   //@ ensures contient(x); // élément ajouté
   void ajouter(int x);

   /** Enlever un élément de l'ensemble.
     * @param x l'élément à supprimer */
   //@ ensures ! contient(x); // élément supprimé
   void supprimer(int x);

}
```

**Exercice 1 : Ensemble chaîné d'entier**

On décide de réaliser l'ensemble en utilisant une structure chaînée pour stocker ses éléments. On s'appuie donc sur une classe Cellule.

**1.1.** Dessiner le diagramme de classe qui faire apparaître Ensemble, Cellule et EnsembleChaine.

**1.2.** Écrire les classes Cellule et EnsembleChaine.

**Exercice 2 : Ensemble ordonné**

Un ensemble ordonné est un ensemble dont les éléments sont équipés d'une relation d'ordre. Ainsi, on peut avoir des opérations qui exploitent cette relation d'ordre. Ici nous nous limiterons à une seule opération, celle qui permet d'obtenir le plus petit élément de l'ensemble.

**2.1.** Compléter le diagramme de classe UML pour faire apparaître EnsembleOrdonne.

**2.2.** Écrire en Java l'interface EnsembleOrdonne.
**2.3.** Écrire en Java la classe EnsembleOrdonneChaine qui réalise EnsembleOrdonne.
**2.4.** Est-ce qu'un ensemble chainé ordonné est en moyenne plus efficace qu'un ensemble chaîné ? Est-ce que votre implantation l'est ?

**Exercice 3 : Généricité**
Pour l'instant nous n'avons considéré que des ensembles d'entiers. Utilisons la généricité pour paramétrer nos classes et interfaces par le type des éléments des ensembles. Procédons par étapes...
**3.1.** Modifier Ensemble, Cellule et EnsembleChaine pour les rendre génériques sur le type des éléments. Adapter le programme de test.
**3.2.** Modifier EnsembleOrdonne et EnsembleChaineOrdonne.

**Exercice 4 : Un peu de recul**
Prenons un peu de recul.
**4.1.** Est-ce que l'implantation chaînée d'un ensemble ou d'un ensemble ordonné est efficace ?
**4.2.** Est-ce que ce serait plus efficace en prenant un tableau ?
**4.3.** Quelles seraient des implantations efficaces ?
**4.4.** Consulter la documentation des API Java pour retrouver la notion d'ensemble et les réalisations qui en sont proposées.

**Exercice 5 : Pour aller plus loin...**
**5.1.** Définir une méthode appelée `justePlusGrandQue` qui, appliquée sur un ensemble ordonné, retourne le plus petit élément strictement plus grand que celui passé en paramètre de cette méthode. Elle retourne **`null`** si cet élément n'existe pas.
**5.2.** Cette méthode est déjà présente dans les API Java. Quel est son nom et où est-elle définie ?