TD 6
Définition du Type Ensemble

TAD
Semestre 2

Question 1 à 4 : spécification fonctionnelle du TAD
Question 5 à 7 : spécification algorithmique du TAD (type abstrait → type concret)

1 Identifier les opérations

- nombre d'éléments (cardinal)
- insérer un élément à un ensemble
- supprimer un élément d'un ensemble
- Tester l'appartenance d'un élément à un ensemble
- créer un ensemble vide

2 Définir la syntaxe des opérations

$$
\begin{aligned}
ensembleVide &: \rightarrow Ensemble[T] \\
estVide &: Ensemble[T] \rightarrow Boolen \\
appartient &: Ensemble[T] \times T \rightarrow Boolen \\
ajouter &: Ensemble[T] \times T \rightarrow Ensemble[T] \\
supprimer &: Ensemble[T] \times T \rightarrow Ensemble[T] \\
cardinal &: Ensemble[T] \rightarrow Entier
\end{aligned}
$$

3 Préciser le domaine de définition des opérations

- L'opération d'ajout n'est possible que si l'élément n'appartient pas déjà à l'ensemble.
- L'opération de suppression nécessite que l'élément soit présent dans l'ensemble.

Pour ens de type Ensemble[T] et e de type T.

ajouter(ens, e) est défini si et seulement si non appartient(ens, e) ;
supprimer(ens, e) est défini si et seulement si appartient(ens, e) ;

4 Définir la sémantique des opérations

ensembleVide	Générateur de base
estVide	Observateur
apartient	Observateur
ajouter	Générateur de base
suprimer	Générateur secondaire
cardinal	Observateur

Pour ens de type Ensemble[T] et e de type T.

$$
\begin{aligned}
estVide(ensembleVide) &= VRAI && (1)\\
appartient(ensembleVide, e) &= FAUX && (2)\\
cardinal(ensembleVide) &= 0 && (3)\\
estVide(ajouter(ens,\ e)) &= FAUX && (4)\\
appartient(ajouter(ens,\ e),\ e') &= si\ e = e'\ alors\ VRAI\ sinon\ appartient(ens,\ e') && (5)\\
cardinal(ajouter(ens,\ e)) &= cardinal(ens) + 1 && (6)\\
suprimer(ensembleVide,\ e) &= NON\ VALIDE.\ A\ SUPPRIMER && (7)\\
suprimer(ajouter(ens,\ e),\ e') &= si\ e = e'\ alors\ ens\ sinon\ ajouter(suprimer(ens, e'), e) && (8)
\end{aligned}
$$

5 Incarner le type abstrait en un type concret impératif

```
1  procedure créerensembleVide(sortie ens <Ensemble[T]>);
2
3  fonction estVide(entree ens <Ensemble[T]>)
4    retourne <Booleen>;
5
6  fonction appartient(entree ens <Ensemble[T]>, entree e <T>)
7    retourne <Booleen>;
8
9  procedure ajouter(maj ens <Ensemble[T]>, entree e <T>)
10   declenche elementExistant, ensemblePlein;
11
12 procedure supprimer(maj ens <Ensemble[T]>, entree e <T>);
13   declenche elementInexistant, ensembleVide;
14
15 fonction cardinal(entree ens <Ensemble[T]>)
16   retourne <Entier>;
```

Listing 1 – Entête des opérations

```
1  -- + propriétés en commentaire
2
3  procedure creerensembleVide(sortie ens <Ensemble[T]>);
4
5  fonction estVide(entree ens <Ensemble[T]>)
6    retourne <Booleen>;
7
8  fonction appartient(entree ens <Ensemble[T]>, entree e <T>)
9    retourne <Booleen>;
10
11 procedure ajouter(maj ens <Ensemble[T]>, entree e <T>)
12   declenche elementExistant, ensemblePlein;
13
14 procedure supprimer(maj ens <Ensemble[T]>, entree e <T>);
15   declenche elementInexistant, ensembleVide;
16
17 fonction cardinal(entree ens <Ensemble[T]>)
18   retourne <Entier>;
```

Listing 2 – Spécification impérative

6 Incarner le type abstrait en un type concret fonctionnel

```
1  -- Type concret fonctionnel
2  -- ne possède que des fonctions
3  fonction ensembleVide()
4    retourne <Ensemble[T]>;
5
6  fonction estVide(entree ens <Ensemble[T]>)
7    retourne <Booleen>;
8
9  fonction appartient(entree ens <Ensemble[T]>, entree e <T>)
10   retourne <Booleen>;
11
12 fonction ajout(entree ens <Ensemble[T]>, entree e <T>)
13   retourne <Ensemble[T]>
14   declenche elementExistant, ensemblePlein;
15
16 fonction retrait(entree ens <Ensemble[T]>, entree e <T>);
17   retourne <Ensemble[T]>
18   declenche elementInexistant, ensembleVide;
19
20 fonction cardinal(entree ens <Ensemble[T]>)
21   retourne <Entier>;
```

Listing 3 – spécification fonctionnelle

7 Type concret impératif ou fonctionnel ?

On suppose défini chez le client le S/P.

```
1  procedure afficher (entree ens <Ensemble[Entier]>);
2  glossaire
3    ens <Ensemble[Entier]>;
4  debut
5    creerensembleVide(ens);
6    ajouter(ens, 3);
7    ajouter(ens,6);
8    ajouter(ens, 9);
9    afficher(ens);
10 fin
```