Types abstraits de données

Un type abstrait de données est un ensemble de données - dont on ne précise par la représentation concrète - doté d'un ensemble d'opérations, dont on précise seulement les spécifications. On fait donc à la fois abstraction de la manière dont les données sont représentées et de la manière dont les opérations sont programmées.

Le concept de type abstrait n'est pas lié à un langage de programmation particulier. Il peut être mis en oeuvre via des mécanismes de programmation, dont en particulier la programmation objet, la programmation modulaire ou la programmation avec des fonctions.

Parmi les opérations sur un type de données abstrait, on distingue usuellement :

- les constructeurs, qui permettent de créer des données,
- les sélecteurs, qui permettent d'accéder à tout ou partie de l'information contenue dans une donnée,
- les opérateurs, qui permettent d'opérer entre données du type (opération internes) ou avec d'autres types de données (opérations externes),
- les prédicats, qui permettent de tester une propriété.

L'ensemble des fonctionnalités disponibles pour un type de données en constitue l'**interface** - la partie visible pour qui veut utiliser ce type de données.

L'**implémentation** consiste à *concrétiser* - réaliser effectivement - un type de données en définissant la représentation des données avec des types de données existant, et en écrivant les programmes des opérations.

Exemple : On peut définir un type abstrait `Rationnel`, en définissant l'ensemble de ses valeurs possibles par référence à l'ensemble des rationnels en mathématiques Q et en proposant l'interface suivante :

- Un constructeur : `faitrationnel : Entier x Entier -> Rationnel`
- Des sélecteurs :
  - `numérateur : Rationnel -> Entier`
  - `dénominateur : Rationnel -> Entier`
- Des opérateurs : `+ : Rationnel x Rationnel -> Rationnel`, ...
- Des prédicats : `egal : Rationnel x Rationnel -> Booléen`, ...

Le programmeur utilisant ce type de données doit pouvoir écrire :
`egal(faitrationnel(1,2)+ faitrationnel(1,6), faitrationnel(2,3))` et obtenir un résultat correct, sans se soucier de la manière dont les différents traitements sont programmés. Il peut *faire abstraction* de la représentation et du détail des calculs, et ne se soucie donc pas du moment où peut être effectuée ou non la simplification d'un rationnel.

Exercice :

1. Ecrire la spécification d'un type abstrait `Temps`, permettant de construire des temps en heures, minutes, secondes, de faire des opérations (addition, soustraction), de comparer deux temps et d'afficher un temps sous un format usuel sous forme de chaîne de caractères.

   -
   -
   -
   -

2. Proposer 2 implémentations distinctes de ce type abstrait. (l'une avec des tableaux , l'autre avec des dictionnaires) (à réaliser dans un notebook avec des tests)

3. Bonus : si vous avez fini, une 3ème en POO