TD 4
Objets de type classe, encapsulation, abstraction

1 Rappel de Cours

Ce TD a pour but d'illustrer deux notions fondamentales sur les objets de type classe :

- **l'encapsulation** : les données et les fonctions permettant de manipuler les objets sont regroupées dans une classe. Les programmes utilisant ce type d'objet et les fonctions spécifiques à ces programmes sont dans une autre classe.
- **l'abstraction** : la représentation interne n'est pas visible pour un utilisateur de la classe. Les seules informations nécessaires et suffisantes pour l'utilisateur sont les spécifications. Ainsi, la représentation interne peut être modifiée sans que cela perturbe l'utilisateur.

Où :

- l'utilisateur est un programme utilisant des variables qui sont des objets de ce type.
- la représentation interne est l'ensemble des champs (ou "variables" ou "attributs" ou "données") choisis pour représenter les valeurs et l'état de l'objet.

2 Exercices

Exercice 1. Représentation des complexes par une classe

Un nombre complexe peut être représenté par un objet de type classe à deux attributs, `x` (partie réelle) et `y` (partie imaginaire).

L'utilisateur de la classe n'aura pas directement accès aux informations qui représentent la valeur d'un nombre complexe donné : il n'a pas à savoir si un complexe est représenté par (partie réelle, partie imaginaire), ou (module, argument), ...
Les attributs (ici `x` et `y`) seront donc `private`.

La manipulation des nombres complexes se fera alors via des fonctions définies à l'intérieur de cette classe, et qui seront rendues visibles aux utilisateurs.
Les fonctions destinées à l'utilisateur seront donc `public`.

Pour satisfaire aux goûts divers et variés des utilisateurs potentiels, on pourra réaliser **deux versions** de chaque fonction de calcul : une version **fonction de classe statique** et une version **méthode d'objet**.

Exemple :

```
/* fonction de classe statique*/

public static double module(Complexe z)
    // résultat : le module de z

public static Complexe add(Complexe z1, Complexe z2)
    // résultat : la somme de z1 et de z2


/* méthode d'objet */

public double module()
    // résultat : le module de this

public Complexe add(Complexe z)
    // résultat : la somme de this et de z
```

Question 1. Définir cette représentation au moyen d'une classe `Complexe`.

Question 2. Rédiger les fonctions :
- `module` qui rend en résultat le module d'un complexe.
- `add` qui rend en résultat la somme de deux complexes.
- `mul` qui rend en résultat le produit de deux complexes.
- `enClair(Complexe z)` qui rend en résultat la chaîne de caractères de la forme "x+y.i" (la forme usuelle d'un nombre complexe), **et** sa version méthode d'objet `toString()`.

Question 3. Rédiger une procédure principale qui :
- lit deux réels x et y,
- crée le nombre complexe z=x+y.i et,
- calcule et affiche le nombre complexe z.(z+1)+1.

Remarque : il va falloir "fabriquer" le "un" complexe (alias 1+0.i).

Exercice 2. Abstraction : représentation des nombres rationnels

On se propose de rédiger une classe `Rationnel` qui représente les nombres rationnels (c'est-à-dire les fractions $\frac{p}{q}$) dotée des opérations classiques : somme, produit, quotient, test d'égalité. Voici la spécification de la classe `Rationnel` :

```
public class Rationnel {
    public Rationnel(int a, int b)
        // pré-requis : b!=0 (a et b de signe quelconque)
        // constructeur : crée un rationnel égal à a/b
    public Rationnel(int a)
        // constructeur : crée un rationnel égal à a
    public String toString()
        // résultat : chaîne de caractère figurant this,
        // par exemple "-235/12" ou "235"
    public static boolean egal(Rationnel x, Rationnel y)
        // résultat : indique si x et y sont égaux
    public static Rationnel somme(Rationnel x, Rationnel y)
        // résultat : somme de x et y
    public static Rationnel produit(Rationnel x, Rationnel y)
        // résultat : produit de x par y
    public static Rationnel quotient(Rationnel x,Rationnel y)
        // pré-requis : y différent de zéro
        // résultat : quotient de x par y
}
```

Question 1. Telles que spécifiées, quelles sont les fonctions de classe et les méthodes d'objet ?

Mise en oeuvre : On décide de représenter un rationnel au moyen de deux entiers, le numérateur `p` et le dénominateur `q`. Afin de faciliter le test d'égalité et pour profiter au maximum de l'intervalle de représentation des entiers, on impose les contraintes suivantes sur `p` et `q` :

- Le dénominateur `q` est > 0. Le signe du nombre rationnel est donc celui de l'entier `p` (qui est quelconque).
- Le rationnel 0 est représenté par $p = 0$ et $q = 1$.
- Pour un rationnel différent de 0, `p` et `q` sont tels que la fraction $\frac{p}{q}$ est irréductible (`p` et `q` sont premiers entre eux). Par exemple, $\frac{27}{18}$ devra être simplifié en $\frac{3}{2}$.

Question 2. En suivant ces spécifications, rédigez la classe complète `Rationnel` ainsi qu'une fonction permettant de tester votre code.

Mise en oeuvre alternative :
Un choix de mise en oeuvre de cette classe consiste à n'imposer aucune contrainte sur `p` et `q` autre que `q` est non nul.

Question 3. Quelles méthodes doivent être modifiées pour adopter cette mise en oeuvre ?
NB : La fonction toString doit renvoyer la chaîne correspondant à la fraction réduite.