Contrôle de programmation par Objets en Java

1A GrB

Avril 2016

Durée: 40 mn

Question I)

1) Définir un type abstrait de données NombreComplexe avec des opérations pour

- initialiser la partie imaginaire (reps. réelle)
- consulter la partie imaginaire (reps. réelle)
- additionner deux nombres complexes
- tester si deux complexe sont égaux

Réponse : notons $\mathbb{C}$ ce type NombreComplexe, R le type des réels.

Type $\mathbb{C}$

setRe : $\mathbb{C}$ X $\mathbb{R}$ $\longmapsto$ $\mathbb{C}$
Initialise la partie réelle d'un complexe

setIm : $\mathbb{C}$ X $\mathbb{R}$ $\longmapsto$ $\mathbb{C}$
Initialise la partie imaginaire d'un complexe

getRe : $\mathbb{C}$ $\longmapsto$ $\mathbb{R}$
Consulte la partie réelle d'un complexe

getIm : $\mathbb{C}$ $\longmapsto$ $\mathbb{R}$
Consulte la partie imaginaire d'un complexe

add : $\mathbb{C}$ X $\mathbb{C}$ $\longmapsto$ $\mathbb{C}$
Additionne deux complexes

egale : $\mathbb{C}$ X $\mathbb{C}$ $\longmapsto$ $\mathbb{B}$
Teste si deux complexes sont égaux

On pourrait rajouter aussi une opération d'initialisation à 0
(non demandé ici)

init : $\longmapsto$ $\mathbb{C}$

2) Donner aussi quelques axiomes algébriques pour spécifier le comportement de ces opérations

Réponse:

Quelques exemples d'axiomes simples:

$getRe\ (setRe\ (c, r)) = r$
$getRe\ (setIm\ (c, i)) = getRe\ (c)$
$getIm\ (setRe\ (c, r)) = getIm\ (c)$
$getIm\ (setIm\ (c, i)) = i$

On pourra aussi rajouter (non demandé ici):

$add\ (c,\ setIm\ (init(), r)\ ) = setIm\ (\ c\ , getIm(c) + r)$
$add\ (c,\ setIm\ (init(), i)\ ) = setIm\ (\ c\ , getIm(c) + i)$
$add\ (c,\ d) = add\ (d,\ c)$

etc...

Question II)

1. Créer une classe qui implémente en Java ce TAD.
2. Définir des constructeurs.
3. Rajouter à cette classe une méthode pour afficher un complexe.
4. Donner au moins deux profiles de méthodes *additionner1* et *additionner2* pour additionner deux complexes.
5. Comment utiliser chacune de ces deux méthodes?
6. Ecrire un programme qui
    1. crée un nombre complexe (utiliser la classe Scanner pour lire les parties réelles et imaginaires), ajoute 1 à sa partie imaginaire et l'imprime.
    2. crée de la même façon deux complexes, les additionne et imprime le complexe résultat.
    3. teste si deux complexes créés sont égaux.

Réponse très détaillée : (cf. aussi classe `Point` du TP)

```
      /**
       * Classe Complexe du plan avec ses coordonnées x et y
       */

       class Complexe extends Object{
          private double re, im;

(1)       public Complexe () { re=im=0;}

(2)       public Complexe (double p, double q) {
              re = p; im = q;
          }

          /**
           * Methodes qui affectent la partie réelle
           * et la partie imaginaire
          */
          public void setRe(double p) {
              re = p;
          }
```

```
        public void setIm(double p) {
            im = p;
        }

        /**
         * Methodes qui consultent la partie réelle
         * et la partie imaginaire
        */
        public double getRe() {
            return re;
        }

        public double getIm() {
            return im;
        }

(3)     public void afficher() {
            System.out.println("(Re,Im) = ("+ re +","+im+")");
        }

(4)      public Complexe additionner1(Complexe x){
              return new Complexe(this.re + x.re, this.im + x.im);
         }
(5)      public static Complexe additionner2 (Complexe x, Complexe y){
             return new Complexe(y.re + x.re, y.im + x.im);
         }

(5')     public boolean equal (Complexe o){
            return (re == o.re) || (im == o.im);
         }

    };
    //
    // Programme de test.
(6) //
    class Main {
        public static void main (String[] args){
            Complexe a,b,c;
            a=new Complexe(1,2);
            b=new Complexe(3,4);
(7)         c = a.additionner1(b);
            c.afficher();
(8)         c = Complexe.additionner2(a,b);
            c.afficher();

            c = new Complexe(3,4);
(9)         System.out.println (c.equal(a));
            System.out.println (c.equal(b));
        }
    }

    /*
(10) (Re,Im) = (4.0,6.0)
    (Re,Im) = (4.0,6.0)
    false
    true
    */
```

NB.

lignes `(1)` et `(2)` réponses à la question 2)

ligne `(3)` réponse à la question 3)

lignes `(4)` et `(5)` réponses à la question 4)

Dans le cas `static` il y a deux paramètres.

lignes `(7)` et `(8)` réponses à la question 5)

ligne `(5')` et `(9)` réponses à la question 6.3)

lignes `(10)` résultat de ce programme

Remarque:

La méthode *additionner1* pourrait être écrite aussi:

```
public Complexe additioner1(Complexe x){
   Complexe z = new Complexe();
   z.setRe (re + x.getRe());
   z.setIm (im + x.getIm());
   return z;
}
```

That's all folks.