Programmation Orientée Objet

TP interfaces - Expressiez-vous! Part II

Ensimag 2^{ème} année 2015-16

Ce TP continue celui sur les expressions arithmétique, avec un exemple d'utilisation d'une interface Java

6 De nouveaux opérateurs

Question 1 Maintenant que vos classes sont en place, ajoutez :

- un opérateur binaire "puissance", symbole "^" (pour l'évaluation, pensez à utiliser Math. pow())
- un opérateur unaire "exp(x)"

Observez comment, grâce aux principe de l'héritage, il est aisé d'étendre la couverture fonctionnelle de nos expressions.

7 A la découverte des interfaces Java

Mr. X dispose d'un ensemble d'objets Java qui tous ont la particularité de pouvoir "s'évaluer" par une valeur double. Autrement dit, la propriété commune à tous ces objets est qu'ils disposent d'une méthode de signature public double evaluer(). C'est le seul point commun entre ces objets, qui à part cela peuvent ne rien avoir en commun. On peut imaginer par exemple qu'il est possible d'évaluer par un double aussi bien des expressions arithmétiques que des étudiants (note de POO?), des cartes graphiques (nombre de coeurs CUDA), des animaux (poids de nourriture par jour) ou des restaurants (avis des clients).

Il souhaite chercher, parmi tous ces objets, celui d'évaluation minimale. Il veut donc pouvoir ranger tous ces objets dans un tableau (ou une ArrayList<>), pour pouvoir lancer un simple algorithme de recherche de minimum sur le tableau.

Pour simplifier, on considèrera seulement deux types d'objets (pour lesquels la comparaison des évaluations est ici sémantiquement pertinente) :

- des *expressions évaluables*, qui encapsulent une expression arithmétique ExpAbstraite et un environnement Env stockant les valeurs des variables associées.
- des nombres rationnels évaluables, basés sur la classe Rational du premier TP de l'année.

Question 2 Interface Evaluable

Pour dénoter en Java la propriété "la classe dispose d'une méthode d'évaluation sous forme d'une valeur flottante" commune à tous les objets, on aura typiquement recours à une *interface*. Implantez l'interface Evaluable qui déclare une unique méthode <u>public double</u> evaluer().

Question 3 Expression arithmétique évaluable

Ecrire une classe ExpressionEvaluable qui encapsule (contient comme attributs) un objet ExpAbstraite et un contexte Env contenant les valeurs des variables de cette expression.

Cette classe doit bien sûr réaliser l'interface Evaluable, et donc (re)définir une méthode evaluer(). Vous utiliserez pour cela les méthodes déjà existantes dans la hiérarchie de classe des expressions.

Question 4 Fraction évaluable

Commencez par récupérer la classe Rational du 1er TP (celle que vous avez écrite ou une correction disponible sur le kiosk).

Pour pouvoir utiliser une fraction en tant qu'objet évaluable, deux solutions sont possibles :

- 1. vous pouvez modifier la classe Rational pour qu'elle réalise également l'interface Evaluable :
- 2. vous pouvez créer une nouvelle classe RationalEvaluable qui hérite de Rational et réalise Evaluable. L'avantage est ici de réutiliser une classe existante sans la modifier (ce qui est souvent intéressant, par exemple lorsqu'on utilise des classes d'un paquetage dont les sources ne sont pas disponibles).

Question 5 Test d'utilisation des objets Evaluable

Testez vos classes au moyen du programme TestInterfaceEvaluable.java disponible sur le Kiosk :

```
import java.util.*;
   public class TestInterfaceEvaluable {
3
4
       public static void main(String[] args) {
5
           ArrayList < Evaluable > list = new ArrayList < Evaluable > ();
6
           // création de l'environnement stockant les valeurs des
7
              variables
           Env env = new Env();
8
           env.associer("y", 2);
           env.associer("x", 1);
           env.associer("a", 9);
11
           env.associer("b", 3);
13
           // on ajoute quelques expressions...
14
           ExpAbstraite exp;
15
16
           exp = new ExpBinaireMult(new Variable("y"), new Constante(3));
17
           list.add(new ExpressionEvaluable(exp, env));
18
19
           exp = new ExpBinaireMult(new ExpBinairePlus(new Variable("x"),
                    new Variable("x")), new Constante(5));
21
           list.add(new ExpressionEvaluable(exp, env));
23
           exp = new ExpBinaireMult(new Constante(-3.5), new ExpUnaireSin(
24
                    new ExpBinairePlus(new Variable("a"), new Variable("b"))
25
                       ));
           list.add(new ExpressionEvaluable(exp, env));
26
27
           // on ajoute quelques rationnels...
28
           list.add(new RationalEvaluable(17, 2));
29
           list.add(new RationalEvaluable(9));
30
31
32
           // affichage
33
           System.out.println("Ensemble des objets evaluables:");
34
35
           int i = 1;
           for (Evaluable e : list) {
36
                System.out.println(i++ + ". " + e
37
```

```
+ "\t --> evaluation: " + e.evaluer());
38
39
            }
            System.out.println("");
40
41
            // recherche du minimum
42
            Evaluable min = rechercherMin(list);
43
            if (min != null) {
44
                System.out.println("L'objet d'evaluation minimale est: " +
45
                    min):
                System.out.println("Sa valeur double est: " + min.evaluer())
46
47
            }
       }
48
49
       private static Evaluable rechercherMin(ArrayList < Evaluable > list) {
50
51
            if (list.isEmpty()) {
52
                return null;
            }
55
            Evaluable min = list.get(0);
            for (Evaluable e : list) {
57
                if (e.evaluer() < min.evaluer()) {</pre>
                    min = e;
                }
            }
61
            return min;
       }
62
63
   }
64
```

8 Extension. Des variables de type quelconque dans l'environnement...

On dispose maintenant d'une belle hiérarchie de classes permettant de représenter des expressions à valeur double. Dans l'environnement Env, pour le moment, toutes les variables sont de type double. On veut désormais que nos expressions acceptent plusieurs types de variables (et pas seulement des variables dont les valeurs sont de type double) - par exemple, des Rational.

La seule chose nécessaire pour calculer nos expressions est qu'à chaque variable puisse être associée une valeur double. En d'autres termes, il faut et il suffit que les objets associés par l'environnement aux variables réalisent tous l'interface Evaluable.

Question 6 Modifiez la classe Env de telle sorte qu'elle stocke non plus des valeurs double mais des Evaluable. Pour cela :

- modifier l'attribut de Env, par exemple pour un attribut de type HashMap<String, Evaluable>
- modifier la signature de la méthode associer. Le prototype de cette méthode devient void associer(String nom, Evaluable valeur)
- modifier le corps de la méthode double obtenirValeur(String nom); pour que la méthode appelle double evaluer() sur l'objet associé à la variable nom.

Question 7 Modifiez la classe Constante de telle sorte qu'elle réalise l'interface Evaluable Cela permet de stocker des simples constantes double dans l'environnement.

Question 8 Testez votre code.

Ouverture... Plus généralement, il devient désormais possible que d'autres classes implémentent Evaluable - par exemple des Animaux, des Zoos ou n'importe quel autres objets pour lesquels il serait pertinent d'avoir une méthode double evaluer() qui retourne une représentation double de l'objet. Dès lors, il devient possible de calculer des expressions sur tous ces objets...