Programmation Java Avancée - Master TTT
22 mars 2010

Les documents hors livres sont autorisés.
On prendra soin de bien préciser les visibilités des membres des classes.
On peut faire les exercices dans n'importe quel ordre.

On désire écrire un programme qui manipule des fonctions à une variable et à valeur réelle.

Exercice n° 1 △

Écrire une interface `Fonction` qui déclare les méthodes suivantes :

- une méthode `evalue` qui étant donnée une valeur x devra retourner la valeur de la fonction en x ;
- une méthode `derivee` qui retourne une fonction qui est la dérivée de l'objet fonction.
- une méthode `setNomVariable` qui prend en argument une chaîne de caractère qui sera utilisée comme nom pour la variable lorsqu'on affiche la fonction (par défaut cette chaîne vaut "x").

Exercice n° 2 △△

Écrire une classe `Polynome` qui implémente `Fonction` et qui représente des polynômes à coefficients réels, de sorte que :

- l'unique constructeur prend en argument un tableau qui indique la valeur des coefficients du polynôme ; ainsi, le tableau `{0.,3.,1.,0.}` correspond au polynôme $3X + X^2$;
- une méthode `degre` retourne le degré du polynome ;
- des méthodes `getCoeff` et `setCoeff` permettent d'obtenir ou de fixer un coefficient du polynôme en donnant le degré du monôme correspondant. Attention, si on fixe le coefficient d'un monôme supérieur au degré, ceci augmente le degré du polynôme, mais ne provoque pas d'erreur ;
- si on écrit les lignes suivantes dans la fonction main,

```
double[] t={3.,0.,1.,0.};
Polynome p=new Polynome(t);
p.setNomVariable("Y");
System.out.println(t);
```

  le programme affiche `3.0Y+Y^2`.

Exercice n° 3 △ △ △

Écrire une classe `Graphe` qui contient un constructeur prenant quatre arguments `xmin`, `xmax`, `ymin`, et `ymax`. Lorsqu'on crée un tel objet, une fenêtre vide s'affiche de taille 200×100, qu'on peut redimensionner. Un objet de type `Graphe` possède une méthode `afficheFonction` qui prend en argument une `Fonction` et affiche celle-ci dans la fenêtre correspondant à l'objet de sorte que le bord gauche correspond à l'abscisse `xmin`, le bord

droit à `xmax`, le haut à l'ordonnée `ymax` et le bas à `ymin`. La fonction reste affichée si on redimensionne la fenêtre ; le repère est calculé de sorte que les bords correspondent toujours aux limites fixées. On peut afficher plusieurs fonctions dans la même fenêtre en appelant plusieurs fois la méthode `afficheFonction`.

Exercice n° 4 △

On désire construire des fonctions en assemblant des fonctions qui existent déjà par addition, multiplication, composition, *etc.*. Toutes ces opérations ont en commun de manipuler deux sous-fonctions ; écrire une classe abstraite `OperateurBinaire` qui implémente `Fonction` de sorte que :

- l'unique constructeur prend en argument deux fonctions ;
- des méthodes `getFirst` et `getSecond` permettent d'accéder à ces fonctions.

Exercice n° 5 △

Écrire la classe `Somme` qui étend `OperateurBinaire`. Si `f` et `g` sont des fonctions, `Fonction s=new Somme(f,g);` représente $f + g$. On supposera dans la suite qu'il existe une classe similaire, `Produit` qui représente le produit et qu'on ne demande pas d'écrire.

Exercice n° 6 △△

Écrire la classe `Composition` qui étend `OperateurBinaire`. Si `f` et `g` sont des fonctions, `Fonction h=new Composition(f,g);` représente $f \circ g$. De plus, on veut que les lignes :

```
double[] t={3.,0.,1.};
Fonction p=new Polynome(t); // p = 3+x^2
double[] u={1.,2.};
Fonction q=new Polynome(u); // q = 1+x
Fonction h=new Composition(p,q);
System.out.println(t);
```

affiche `3.0(1.0+2.0x)+(1.0+2.0x)^2`.

Exercice n° 7 △△

a) Montrer que la dérivée de la fonction Sinus est la fonction Sinus$\circ P$, avec $P(x) = \pi/2+x$.
b) Écrire la classe `Sinus` qui implémente `Fonction`.
c) Comme il n'y a qu'une fonction Sinus (alors qu'il y a beaucoup de polynômes différents, par exemple), on veut éviter de construire plusieurs objets de la classe `Sinus`. Faire en sorte que l'utilisateur ne puisse pas appeler le constructeur de la classe et qu'il existe une méthode statique `getSinus` qui retourne toujours le même objet de type `Sinus`.

Exercice n° 8 △△

Au lieu d'écrire de nombreuses classes différentes, on veut remplacer les classes `Somme`, `Produit`, `Quotient`, *etc.* par une unique classe `Operation`. Pour cela, déclarer l'énumération `Operateur` qui contient les valeur `Plus`, `Mult` et `Div`, et écrire la classe `Operation` qui implémente `Fonction` et dont l'unique constructeur prend en argument un opérateur et deux fonctions.