Feuille de TP N°02

Les classes et les objets

Exercice 1 *Polynôme*

Dans ce TP, il s'agit d'implémenter la classe `Polynome` qui représente un polynôme de degré au plus égal à 2 ($ax^2 + bx + c$ avec a, b et c potentiellement nuls).

1 Structure et Interface de la classe Polynome

La classe `Polynome` considérée dans ce TP correspond aux spécifications suivantes.

1.1 Variables d'instance

On représentera un polynôme par les trois variables d'instance a, b et c qui sont trois réels définissant respectivement les cœfficients de degré 2, 1 et 0 du polynôme.

1.2 Constructeurs

- `Polynome(double initA, double initB, double initC)` : constructeur avec valeurs initiales explicites pour les trois cœfficients du polynôme.
- `Polynome`() : constructeur sans arguments, donnant la même valeur initiale par défaut aux trois cœfficients du polynôme ;

1.3 Accesseurs et modifieurs (getters & setters)

- Des accesseurs en lecture (ou *getters*)
  `getA()`, `getB()` et `getC()` sont définis pour chaque variable d'instance.
- Un modifieur (ou *setter*)
  `setAsetBsetC(double newA, double newB, double newC)` est défini pour les trois variables d'instance simultanément.

1.4 Méthodes d'instance

- `double evaluer(int val)` : retourne la valeur du polynôme pour l'entier `val` ;
- `void multCoeff(int val)` : multiplie tous les cœfficients du polynôme par la valeur `val` ;
- `void ajouter(Polynome` autrePoly)' : modifie les cœfficients du receveur en y ajoutant ceux du polynôme `autrePoly` ;
- `Polynome somme(Polynome autrePoly)` retourne le polynôme resultant de l'addition du receveur et de `autrePoly` (il n'y a aucune modification de `autrePoly` ou du receveur) ;

- `Polynome derive()` renvoie le polynôme dérivé ;
- `double evaluerDerive(int val)` renvoie la valeur de la dérivée du premier ordre du polynôme receveur pour la valeur `val` ;
- `double discriminant()` renvoie le discriminant du polynôme ;
- `void afficherRacines()` affiche la ou les racines (réelles ou imaginaires) du polynôme ; Pour cette méthode, vous pouvez utiliser la méthode de classe statique `double Math.sqrt(double val)` qui calcule la racine carrée d'un nombre réel positif. Elle est présente dans la classe `java.lang.Math` (voir annexe)
- `Polynome produit(Polynome autrePoly)` : retourne le résultat de la multiplication du polynôme receveur par `autrePoly` si ce résultat est un polynôme de degré au plus deux. Un message d'erreur est affiché et la méthode retourne `null` Sinon ;
- `String toString()` : retourne une chaîne de caractères `[ a.x^2 + b.x + c ]` représentant le polynôme textuellement ;
- `void afficher()` : affiche une représentation textuelle du polynôme ;

Rappel de formules :

On appelle **discriminant** du polyôme $ax^2 + bx + c$ ($a \neq 0$), le réel $\Delta = b^2 - 4ac$.
- si $\Delta > 0$: deux racines réelles : $x_1 = \frac{-b-\sqrt{\Delta}}{2a}$ et $x_2 = \frac{-b+\sqrt{\Delta}}{2a}$
- si $\Delta = 0$: une racine réelle dite "double" : $x_0 = \frac{b}{2a}$
- si $\Delta < 0$: pas de racines réelles. Cependant le polynôme possède 2 racines complexes conjuguées z_1 et $z_2 = \bar{z_1}$. Si $\Delta = (i\delta)^2$, on peut écrire : $z_1 = \frac{-b-i\delta}{2a}$ et $z_2 = \frac{-b+i\delta}{2a}$.

2 Travail à réaliser

2.1 Implémentation de la classe Polynome

Créez un nouveau projet, nommé TpPolynome dans lequel vous importerez les fichiers `TestMain.java` et `Polynome.java` du répertoire `tp2` dans votre workspace.
On vous demande de compléter le fichier `Polynome.java` contenant les sources de votre propre implémentation de la classe `Polynome`, sources conformes aux spécifications préalablement fournies.

2.2 Utilisation de la classe Polynome : La méthode main

Testez la classe `Polynome` à l'intérieur de la méthode main du fichier `TestMain.java`, afin de réaliser les traitements suivants :

1. déclarer et construire un `Polynome p1` avec les valeurs par défaut ;
2. donner la valeur `1` à tous les cœfficients de `p1` ;
3. déclarer et construire un `Polynome p2` avec pour cœfficents `1`, `-2` et `-3` ;
4. afficher `p1` puis `p2` ;
5. afficher la valeur de `p2` pour l'entier 2 ;
6. afficher la valeur de la dérivée `p2` pour l'entier `2` ;
7. déclarer un polynôme `p3` et l'initialiser à la dérivée de `p2` ;
8. afficher `p3` ;
9. afficher le discriminant de `p2` ;

10. afficher les racines de `p2` ;
11. afficher le discriminant de `p1` ;
12. afficher les racines de `p1` ;
13. ajouter `p2` à `p1` puis afficher `p1` ;
14. mettre à zéro le cœfficents de degré 2 de `p1` ;
15. afficher le résultat de la multiplication de `p1` par `p3`.
16. mettre à zéro les cœfficents de degré 2 et 1 de `p3` ;
17. afficher le résultat de la multiplication de `p2` par `p3`.

3 Résultat visuel

à l'issue de cet exercice, l'exécution du programme de test demandé doit ressembler (fortement) à ceci :

- `Le polynome p1 est : [1.0x^2 + 1.0x + 1.0]`
- `Le polynome p2 est : [1.0x^2 + -2.0x + -3.0]`
- `La valeur de p2 en 2 est : -3.0`
- `La valeur en 2 du polynome derive de p2 est : 2.0`
- `Le polynome p3 (derive de p2) est : [0.0x^2 + 2.0x + -2.0]`
- `Le discriminant de p2 est : 16.0`
- `Racines du polynome p2 :`
  `Ce polynome a deux racines reelles qui sont 3.0 et -1.0`
- `Le discriminant de p1 est : -3.0`
- `Racines du polynome p1 : Ce polynome a deux racines imaginaires qui sont`
  `-0.5 - 0.8660254037844386.i et -0.5 + 0.8660254037844386.i`
- `Le polynome p1 est, apres ajout de p2 : [2.0x^2 + -1.0x + -2.0]`
- `Le resultat de la multiplication de p1 par p3 est : [-2.0x^2 + -2.0x + 4.0]`
- `Le resultat de la multiplication de p2 par p3 est : [-2.0x^2 + 4.0x + 6.0]`

Bon travail !