Programmation Orientée Objet avec JAVA
TP N°3
Manipuler les fractions et les complexes

Objectif du TP :

- **Utiliser des classes pour manipuler les fractions**
- **Utiliser les classes pour faire des calculs sur les nombres complexes**

Exercice 1 : contrôle des variables private par les modificateurs

I- Conception d'une classe

1- Nous nous proposons de définir une classe Fraction pour représenter les fractions mathématiques. Une fraction est un couple d'entiers (numérateur, dénominateur) habituellement notée : numérateur / dénominateur. Le dénominateur d'une fraction est toujours différent de 0.
Une fraction est dite réduite ou irréductible lorsque son numérateur et son dénominateur sont premiers entre eux. Pour réduire une fraction, on calcule le plus grand commun diviseur de son numérateur et de son dénominateur, puis on divise le numérateur et le dénominateur par le plus grand commun diviseur calculé précédemment.
Les opérateurs sont définis de la façon suivante :

$$\frac{n1}{d1} + \frac{n2}{d2} = \frac{n1 * d2 + n2 * d1}{d1 * d2}$$

$$\frac{n1}{d1} - \frac{n2}{d2} = \frac{n1 * d2 - n2 * d1}{d1 * d2}$$

$$\frac{n1}{d1} * \frac{n2}{d2} = \frac{n1 * n2}{d1 * d2}$$

$$\frac{n1}{d1} / \frac{n2}{d2} = \frac{n1 * d2}{d1 * n2}$$

Après toute opération, on calculera la fraction irréductible équivalente.

On définira les constructeurs d'une instance de la classe fraction ci dessous:

- **Fraction()** initialise par défaut la fraction avec la valeur (0,1),
- **Fraction(int n, int d)** initialise la fraction avec la valeur (n,d),
- **Fraction(int n)** initialise la fraction avec la valeur (n,1),
- **Fraction(String s)** initialise la fraction avec la fraction contenue sous forme de chaîne dans s.

On définira les méthodes suivantes :

- **public static int ppcm(int a, int b)** qui calcule e plus petit commun multiple de deux nombres a et b par la formule : ppcm(a,b)=a/pgcd(a,b)*b;.

- **public static int pgcd(int a,intb)** qui calcule le plus grand commun diviseur(a, b) en utilisant l'algorithme suivant :
  - On calcule le reste de la division de a par b.
  - Si ce reste est 0, le pgcd est b.
  - Si ce reste est différent de 0, le pgcd(a, b) est le même que le pgcd(b, reste);

- **public Fraction reduire()** qui retourne la fraction irréductible équivalente à la fraction qui reçoit le message.

- **String toString()** qui permet de convertir une fraction en chaîne de caractères : le numérateur,'|' puis le dénominateur,

- les méthodes d'instance **plus, moins, multiplie**, et **divise** qui permettent de réaliser les opérateurs d'addition, soustraction, multiplication et division de deux fractions.

- deux méthodes **equals** (static et non static) qui permettent de déterminer si deux fractions sont égales ou non.

- deux méthodes **compareTo** (static et non static) pour comparer deux fractions. compareTo retourne -1 si la première fraction est inférieure à la deuxième, 0 si elles sont égales ou 1 si la première est supérieure à la deuxième.

- **Fraction valueOf(String s)** qui permet d'obtenir une fraction à partir d'une chaîne de caractères contenant un entier, le caractère '|', puis un autre entier.

- **int[] nChiffres(int n)** qui renvoie, sous la forme d'un tableau, les n premiers chiffres après la virgule d'une fraction.

- **Boolean estDecimale()** qui renvoie true si la fraction est décimale et false dans le cas contraire.
  On dit qu'une fraction est décimale si elle admet un nombre fini de chiffres après la virgule. A titre d'exemple, 1/4=0.25 est décimal, par contre, 1/3=0.3333... (s'écrivant avec une infinité de 3) ne l'est pas.

2- Concevoir une classe TestFraction de test permettant ce qui suit :

- Construire deux fractions f1 et f2 fournies dans la ligne de commandes : java TestFraction 2/3 + 3/5
- Construire une fraction f3 de numérateur 6 et de dénominateur 14 et une fraction f4 de numérateur 1 et de dénominateur 5.
- Calculer la somme, la différence, le produit, et le rapport de f3 et f4.
- Afficher les 5 premiers chiffres après la virgule de la somme de f3 et f4.
- Calculer des expressions arithmétiques de fractions toutes strictement positives : par exemple :
  12|34+123|321*5|6=6695|10914

La première partie, jusqu'à = est tapée par l'utilisateur du programme, puis après la frappe de = le programme affiche le résultat.
Pour faire ce calcul, on fera les hypothèses suivantes :

- ✓ Il n'y a jamais d'erreur dans les données tapées par l'utilisateur du programme.
- ✓ Les opérateurs ont la même priorité, et l'opérateur le plus à gauche est évalué en premier.

L'expression précédente correspond donc au parenthésage : (12|34+123|321) * 5|6

Exercice 3 : Nombres complexes

I. Créez une classe Complexe pour représenter des nombres complexes (un nombre complexe est représenté par deux réels le premier représente la partie réelle, et le deuxième représente la partie imaginaire).
La classe doit contenir :
Un constructeur pour initialiser le nombre
2 méthodes accesseurs et 2 méthodes modificateurs
Une méthode module qui calcule le module du nombre)
Une méthode ajouteToi, (public Complexe ajouteToi(Complexe c)) qui reçoit comme paramètre un complexe qu'elle ajoute au complexe courant (this), et retourne comme valeur le complexe courant modifié.

Une méthode multplieToi (public Complexe multiplieToiPar(Complexe c)) qui multiplie le complexe courant avec le complexe passé en paramètre. Le complexe courant est modifié et le résultat de la multiplication est renvoyé comme valeur de retour.

Une méthode opposeToi (public Complexe opposeToi()) qui modifie le complexe courant en son opposé et le renvoie comme valeur de retour.

Ajoutez une méthode toString() (comme dans le TP 1) qui renvoie la représentation habituelle des nombres complexes : 5, 2 + 5i, 3 - 8i, 6i (les nombres pourront comporter des décimales, mais ne devront pas être de la forme 1 + -2 i, mais de la forme 1 - 2 i).

II Créez une classe TextComplexe pour :

1- Vérifiez que (1 + i)(1 + i) est bien égal à 2i .Testez en mettant 1 + i dans une variable z et en calculant z.multiplieToiPar(z).

2- Faire le calcul suivant en utilisant votre classe (vous devriez trouver -7 -2i) :
z1 = 1 + 2i
z2 = 3 - 5i
z3 = 1 + i
z4 = (z1 - z2) (z3 -1)

Votre programme affichera les valeurs finales de z1, z2, z3 et z4. Essayez de comprendre les valeurs affichées.

3. Comment faire si on veut aussi afficher à la fin les valeurs initiales de z1, z2 et de z3 ?

4. Faites maintenant le calcul suivant en partant des mêmes valeurs de départ de z1 et z2 :
(z1 - z2) (z1 + z2 -1)
Vous devriez trouver 15 + 27 i. Ce n'est pas ce que vous trouvez ? Essayez de comprendre pourquoi.

III-Modifiez la classe complexe pour ajouter :

Une méthode ajouter (public Complexe ajouter(Complexe c)) qui ajoute un complexe au complexe courant et retourne comme valeur le complexe résultat et ceci sans modifier le complexe courant.

Une méthode multplie(public Complexe multiplier(Complexe c)) qui multiplie le complexe courant avec le complexe passé en paramètre. Le complexe courant n'est pas modifié et le résultat de la multiplication est renvoyé comme valeur de retour.

Une méthode oppose (public Complexe oppose()) qui calcule le complexe opposé du complexe courant sans le modifier. La valeur retournée est le complexe opposé.

Modifiez la classe TestComplexe pour refaire le calcul (z1 - z2) (z1 + z2 -1) en utilisant les méthodes ajoutées.

Quelles méthodes préférez-vous, ajouteToi et multiplieToiPar, ou ajouter et multiplier ? Pourquoi ?

API :

Classe String	
int	indexOf(int ch) Returns the index within this string of the first occurrence of the specified character.
int	indexOf(int ch, int fromIndex) Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index.
int	indexOf(String str) Returns the index within this string of the first occurrence of the specified substring.
int	indexOf(String str, int fromIndex) Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.
String	substring(int beginIndex) Returns a new string that is a substring of this string.
String	substring(int beginIndex, int endIndex) Returns a new string that is a substring of this string.

Classe Integer	
static int	parseInt(String s) Parses the string argument as a signed decimal integer.