TP Java n°3

- Pour chaque question, créez également une classe permettant de tester la classe qu'il vous est demandé d'implémenter.
- Vous ne pouvez utiliser que les classes Java vues en cours.

Exercice 1

Écrire une classe *Fraction* permettant de représenter des fractions décimales (numérateur et dénominateur entiers).

Cette classe devra fournir au minimum les opérations suivantes :

- Un constructeur par défaut et un constructeur permettant d'initialiser le numérateur et le dénominateur (et d'autres si vous le jugez utile).
- Les accesseurs *getNum()* et *getDen()* permettant de renvoyer, respectivement, le numérateur et le dénominateur.
- Les méthodes *add()* et *mult()* permettant, respectivement, d'ajouter et de multiplier deux *Fraction* (vous pouvez également implémenter *sub()* et *div()*). Ces méthodes doivent être surchargées pour permettre d'ajouter/multiplier une *Fraction* ou un entier à la *Fraction* sur laquelle elles sont appelées. On doit donc pouvoir écrire *fraction.add(fraction2)* et *fraction.add(3)*…
- Elle devra également implémenter *compareTo()* et redéfinir les méthodes *clone()*, *equals()* et *toString()* (voir remarques plus bas).
- Dans le programme de test, créez un tableau de 5 fractions et triez-le (à l'aide d'une méthode de `Arrays`) : qu'en déduisez-vous ?

- Pour redéfinir une méthode (comme *toString()*, *clone*() et *equals()*, qui sont déjà définies dans *Object*), il est fortement conseillé de placer l'annotation *@Override* avant la définition de la méthode. Les définitions des méthodes redéfinies doivent être rigoureusement identiques aux définitions initiales (à la visibilité près).
- Pour implémenter *compareTo()*, ajoutez `implements Comparable<Fraction>` dans l'entête de la classe : `class Fraction implements Comparable<Fraction> {…}` puis écrivez une méthode `public int compareTo(Fraction f)`
- On rappelle que la fraction *12/4* est égale à la fraction *6/2*, qui est égale à la fraction *3/1*… Pour faciliter les comparaisons et produire des résultats lisibles, il est donc conseillé de stocker la fraction *sous sa forme simplifiée* (la création de la fraction 12/4 doit donc faire en sorte que le numérateur soit 3 et le dénominateur soit 1). Il en va de même pour le résultat renvoyé par *add()*, *mult()*, etc. : toutes ces méthodes doivent renvoyer une fraction simplifiée…
- Pour simplifier une fraction, le plus simple consiste à trouver le *pgcd* de son numérateur et de son dénominateur et de les diviser par le résultat obtenu : *pgcd(12, 4) = 4*, donc le numérateur sera stocké comme *12/4 = 3* et le dénominateur comme *4/4 = 1*.
- Pensez aussi à la gestion du signe…
- Réfléchissez bien à l'endroit le plus adéquat pour simplifier la fraction…

Exercice 2

Écrire une classe *Pile* permettant de représenter une pile d'entiers. Cette pile devra être implémentée de *façon dynamique* (c'est-à-dire, sans utiliser de tableau…).

Un moyen d'y parvenir consiste à considérer qu'une pile est une liste d'*Entree*, chaque *Entree* contenant un entier (la "valeur") et un champ référençant l'*Entree* suivante (comme une liste chaînée, mais où les pointeurs ont été remplacés par des références…). La pile en elle–même n'a donc besoin de connaître que la référence de la première *Entree* (son "sommet").

Vous devez donc implémenter une classe *Entree* permettant de représenter chaque élément de la pile et une classe *Pile* qui utilisera un champ de type *Entree* pour représenter son sommet.

La classe *Pile* devra fournir les opérations suivantes :

- Un constructeur par défaut, qui créera une pile vide.
- La méthode *sommet()* qui renvoie l'entier au sommet de la pile (*null* si la pile est vide).
- La méthode *pileVide()* qui indique si la pile est vide
- Les méthodes *empiler()* et *depiler()* qui ajoute/supprime et renvoie l'élément placé au sommet de la pile. La méthode *depiler()* devra lancer l'exception *InvalidOperationException* si la pile est vide. On rappelle que la syntaxe pour lever cette exception est de la forme *throw new InvalidOperationException()*.
- La classe *Pile* doit également implémenter *clone()* (ou un constructeur de copie) afin qu'elle effectue une copie de la pile (le programme de test devra le montrer…).
- Vous pouvez également ajouter un constructeur pour construire une *Pile* à partir d'un tableau d'entiers et une méthode *getArray()* permettant de renvoyer un tableau d'entiers à partir d'une *Pile*.