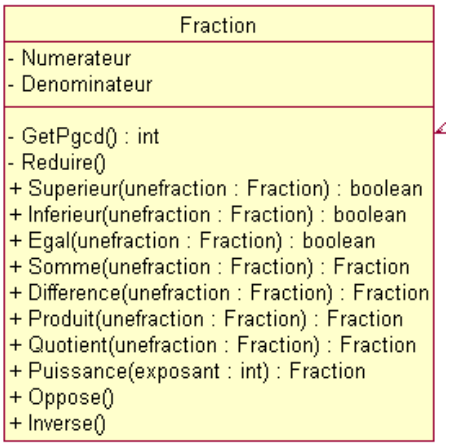

TP à faire Consolidation du concept de de Classe et d'instance d'Objet avec une classe "métier" ou "model » plus complexe.
TP FRACTION :

Objectif : construction d'une classe fraction.

Une fraction (ou nombre rationnel) est un ensemble ordonné de deux entiers - numérateur et dénominateur -. Nous nous proposons ici de définir une classe fraction munie de ses principales opérations.

Question 1.

Travail à faire
En utilisant le formalisme UML, redessiner le diagramme complet de la classe Fraction ci-dessous avec Enterprise Architecte en indiquant les attributs ainsi que leur niveau de visibilité.

Question 2.

On désire pouvoir instancier une fraction de trois manières différentes :
Fraction f1 = new Fraction (12, 7) ; // dans ce cas le numérateur vaudra 12 et le dénominateur 7
Fraction f2 = new Fraction () ; // fraction nulle de dénominateur 1
Fraction f3 = new Fraction (9) ; // fraction dont le numérateur vaut 9 et le dénominateur 1
Travail à faire.
Ecrire la classe Fraction en JAVA.
Pour tester la fraction, vous déclarerez une classe publique AppFraction avec une seule méthode Main et vous effectuerez les tests nécessaires à la classe.
Ajouter deux accesseurs publics (méthodes Get).

Question 3. Gestion de l'affichage.

Afin de ne pas mélanger ce qui concerne les fractions et leurs opérations d'une part et l'affichage d'autre part on vous propose de créer une classe Ecran qui assurera cette

responsabilité d'affichage. Cette classe ne contiendra qu'une seule méthode statique Affiche (Fraction f).

Travail à faire.

Ajouter au diagramme la nouvelle classe, indiquer la relation entre les deux classes.
Solution

Ecrire la classe Ecran et sa méthode publique affiche() tel que:

```
class AppFraction{
        static void Main(){
                Fraction f = new Fraction(12,7);
                Fraction f1 = new Fraction(9);
                Fraction f2 = new Fraction();
                Ecran.Affiche(f);
                Ecran.Affiche(f1);
                Ecran.Affiche(f2);
         }
}
```

Question 4 Opérations unaires

Ecrire une méthode publique Oppose() qui permettra d'écrire:

```
Fraction f = new Fraction(4,7);

f.Oppose();                    // inverse le signe de la fraction, qui devient -4/7
```

Ecrire une méthode publique Inverse() qui permettra d'écrire:

```
Fraction f = new Fraction(4,7);

f.Inverse();                    // inverse numérateur et dénominateur, qui devient
7/4
```

Question 5 opérateurs relationnels

Opérateur Supérieur

```
Fraction f = new Fraction(11,7);
 Fraction f1 = new Fraction(5,4);
 f.Superieur( f1 ) retourne vrai
```

Opérateur Inférieur

```
f.Inferieur( f1 ) retourne faux.
```

Opérateur d'égalité.

```
Fraction f = new Fraction(11,7);
Fraction f1 = new Fraction(22,14);
```

f.Egal (f1) retourne vrai.

Question 6 pgcd

On vous demande d'écrire une méthode privée GetPgcd() qui retourne le PGCD des numérateur et dénominateur.
Algorithme du pgcd de deux nombres (Euclide).
soit a et b deux nombres
Tant que a est différent de b, si a est supérieur à b, a prend comme valeur a-b, sinon b prend la valeur b-a.
Il faudra par ailleurs s'assurer que numérateur et dénominateur ne sont pas nuls et ne traiter que deux nombres positifs.
Tester avec une fraction construite avec comme numérateur -75 et comme dénominateur 90 (doit retourner 15)

Question 7 Réduction et signe de la fraction

La méthode privée reduire() réduit la fraction courante en divisant numérateur et dénominateur par leur pgcd et traite le problème du signe de la fraction, le signe de la fraction est le signe de son numérateur, si le numérateur et le dénominateur sont négatifs, la fraction n'a pas de signe (implicitement +).
Ecrire cette méthode et la tester avec :
Fraction f3 = new Fraction(120,-150);

Ecran.affiche(f3); // affiche -4/5

*****

Méthode pour obtenir le PGCD à utiliser pour la méthode réduire()

```
private int GetPgcd()
{
int a = this.Numerateur;
int b = this.Denominateur;
int pgcd=-1;
if ( a!=0 && b!=0)
{
if ( a<0 ) a =-a;
if ( b<0 ) b =-b;
while ( a!=b )
if ( a<b )
b -= a;
else
a -=b;
pgcd = a;
}
return pgcd;
}
```