DM1 Java

1 Le Chronomètre

1.1

Voici les signatures des chronos que nous avons défini

```
public class chrono
{
    private int secondes;
    private int minutes;
    private int heures;
    public void chrono ()
    public void afficher()
    public void rebours ()
    public int getSecondes ()
    public int getMinutes ()
    public int getHeures ()
    public void setSecondes (int sec)
    public void setMinutes (int min)
    public void setHeures (int heu)
}


public class chrono2
{
    private int nbSecondes;
    public void chrono ()
    public void afficher()
    public void rebours ()
    public int getSecondes ()
    public int getMinutes ()
    public int getHeures ()
    public void setSecondes (int sec)
    public void setMinutes (int min)
    public void setHeures (int heu)
}
```

1.2

Voici nos constructeurs

Chrono 1

```
public void chrono ()
```

```
{
    secondes = 0;
    minutes = 0;
    heures = 0;
}
```

Chrono 2

```
public void chrono ()
{
    nbSecondes = 0;
}
```

1.3

Voici nos méthodes retours()

Afin d'éviter un nombre de secondes négatif, il suffit de rajouter une condition si le compteur est à 0

Chrono 1

```
public void rebours ()
    {
        if (secondes > 0) secondes = secondes - 1;
        else if (minutes > 0)
        {
            minutes = minutes - 1;
            secondes = 59;
        }
        else if (heures > 0)
        {
            heures = heures - 1;
            minutes = 59;
            secondes = 59;
        }
        else System.out.println("Impossible d'enlever une seconde");
    }
```

Chrono 2

```
public void rebours ()
{
    if (nbSecondes > 0) nbSecondes = nbSecondes - 1;
    else System.out.println("Impossible d'enlever une seconde");
}
```

1.4

Il nous semble que la meilleure implémentation dépend de l'utilisation

Le chrono 1 est moins efficace pour enlever des secondes (plus d'instructions) mais est plus efficace pour afficher et modifier les valeurs du chrono.

Le chrono 2 est donc plus efficace pour compter à rebours, et moins efficace pour afficher et modifier les valeurs.

1.5

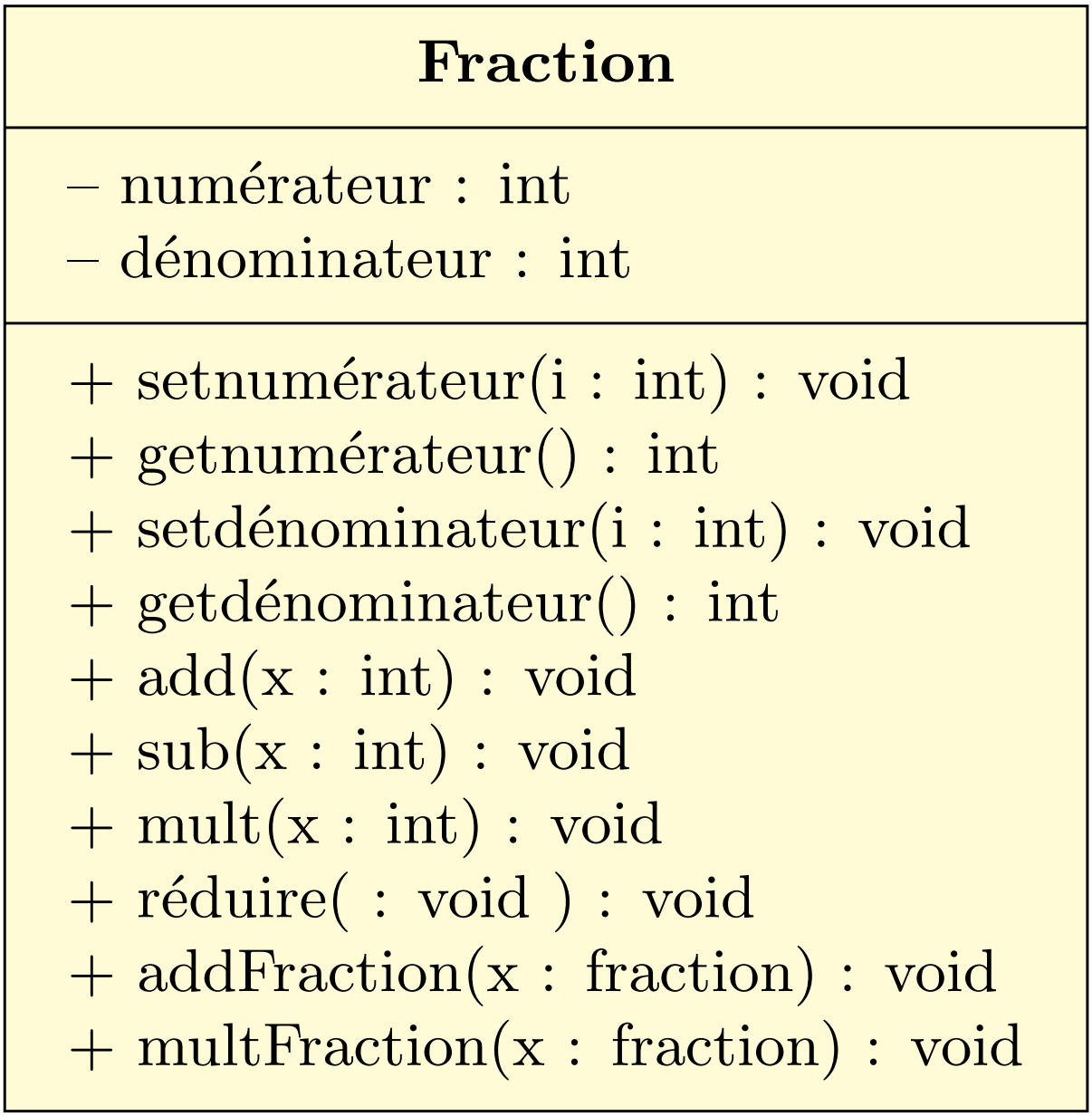
Exécutez la commande "make main" et laissez la magie opérer.

2 Arithmétique rationnelle exacte en Java

2.1

Oui et Non. Oui car il n'y a pas d'approximation sur les nombres entier, et Non pour les nombres réels.

2.2



2.3

```
public class fraction
{

//      ATTRIBUTS
    private int numerateur;

    private int denominateur;

//      CONSTRUCTEUR
    public void fraction (int a, int b)
    {
        this.setNumerateur(a);
        this.setDenominateur(b);
    }

//      METHODES
    public void afficher()
    {
        System.out.println(numerateur + " / " + denominateur);
    }

    public void add (int x)
    {
        numerateur = numerateur + x * denominateur;
    }

    public void sub (int x)
    {
        numerateur = numerateur - x * denominateur;
    }

    public void mult (int x)
    {
        numerateur = numerateur * x;
    }

    public void reduire ()
    {
        int min;
        if (numerateur < denominateur)  min = numerateur;
        else                            min = denominateur;
        for (int i = 1; i < min; i++)
        {
            if (numerateur % i == 0 && denominateur % i == 0)
            {
                numerateur = numerateur / i;
                denominateur = denominateur / i;
            }
        }
```

```
    }

    public void addFraction (fraction f)
    {
        numerateur = numerateur * f.getDenominateur()+ f.getNumerateur() *
            ↪  denominateur;
        denominateur = denominateur * f.getDenominateur();
    }

    public void multFraction (fraction f)
    {
        numerateur = numerateur * f.getNumerateur();
        denominateur = denominateur * f.getDenominateur();
    }

    public int getNumerateur ()
    {
        return numerateur;
    }

    public int getDenominateur ()
    {
        return denominateur;
    }

    public void setNumerateur (int num)
    {
        numerateur = num;
    }

    public void setDenominateur (int den)
    {
        denominateur = den;
    }
}
```

2.4

on utilisera :

```
public class division
{
    public static void main(String[] args) {
        System.out.println("TESTS DE L'EXACTITUDE DES OPERATIONS");
        System.out.println("int   : 6 / 5 = "+ (int)(6/5));
        System.out.println("float : 6 / 5 = "+ (float)(6/5));

        System.out.println("TESTS DE LA CLASSE FRACTION");
        System.out.println("resolution de (123/456+789/10 +11)*12 ");

        fraction f1 = new fraction();
```

```
        fraction f2 = new fraction();
        f1.setNumerateur(123);
        f1.setDenominateur(456);
        f2.setNumerateur(789);
        f2.setDenominateur(10);

        f1.addFraction(f2);
        f1.add(11);
        f1.mult(12);
        f1.reduire();
        f1.afficher();

    }
}
```

```
C:\Users\Utilisateur\Desktop\java>java division
TESTS DE L'EXACTITUDE DES OPERATIONS
int   : 6 / 5 = 1
float : 6 / 5 = 1.0
TESTS DE LA CLASSE FRACTION
resolution de (123/456+789/10 +11)*12
205587 / 190
```

2.5 pgcd

```
public void pgcd ()
    {
        a = getdenominateur();
        b = getnumerateur();
        while !((a*b)==0){
            if(a>b){
                a = a-b;
                }
            else{
                b = b-a;
                }
        }
        if (a =0){
            return b;
        }
        else{
            return a;
        }
    }
```