Université A. Belkaïd
Tlemcen
Faculté des Sciences
Département d'Informatique
L2 informatique 2016-2017

Examen de Rattrapage de Programmation Objet

Nom : .....

Prénom : .....

Mercredi 5 avril 2017
Durée : 1h30

Éteignez vos téléphones SVP - Documents non autorisés

La classe Fraction: *(répondez sur le sujet)*

La classe Fraction permet de modéliser en Java des nombres rationnels qui sont des fractions de nombres entiers relatifs. Exemple : dans la fraction $\frac{-5}{8}$ le numérateur est -5 et le dénominateur est 8. Nous la représenterons ainsi : [(-5)/(8)]. Quand le dénominateur est 1, on représentera uniquement le numérateur. Exemple : (42) au lieu de [(42)/(1)].

1) On a aussi décidé que les attributs des objets de type Fraction, une fois initialisés, ne pourront pas être modifiés. **C'est-à-dire que ces objets seront .....** **(1 pt)**

2) La classe suivante comporte deux attributs f1 et f2 de type Fraction et une méthode pour tester la classe Fraction. **Lisez-la *attentivement* puis complétez les déclarations de f1 et f2 en remplissant directement les cadres vides :**

```
public class TestFraction {
    [                              ] f1;
    [                              ] f2;                    (1 pt)
    public static void test(){
        f1 = new Fraction(-2,3);
        f2 = new Fraction(3,4);
        System.out.println(f1+" + "+f2+" = "+f1.plus(f2));
        System.out.println(f1+" / "+f2+" = "+f1.divisee(f2));
        f2 = f1.fois(f2);
        System.out.println(f2+" = "+f2.reduire()+" = "+f2.toDouble());
    }
}
```

Voici le résultat de l'exécution de la méthode `test()` : *(Vous devrez le prendre en compte dans le reste de l'exercice)*

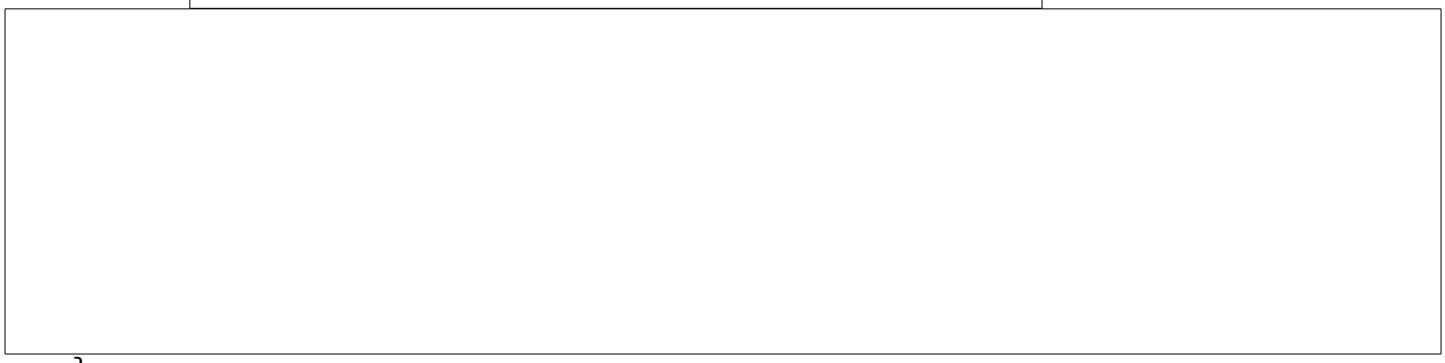
```
[(-2)/(3)] + [(3)/(4)] = [(1)/(12)]
[(-2)/(3)] / [(3)/(4)] = [(-8)/(9)]
[(-6)/(12)] = [(-1)/(2)] = -0.5
```

3) Complétez le code Java de la classe Fraction en remplissant directement les cadres vides : **(12 pts)**

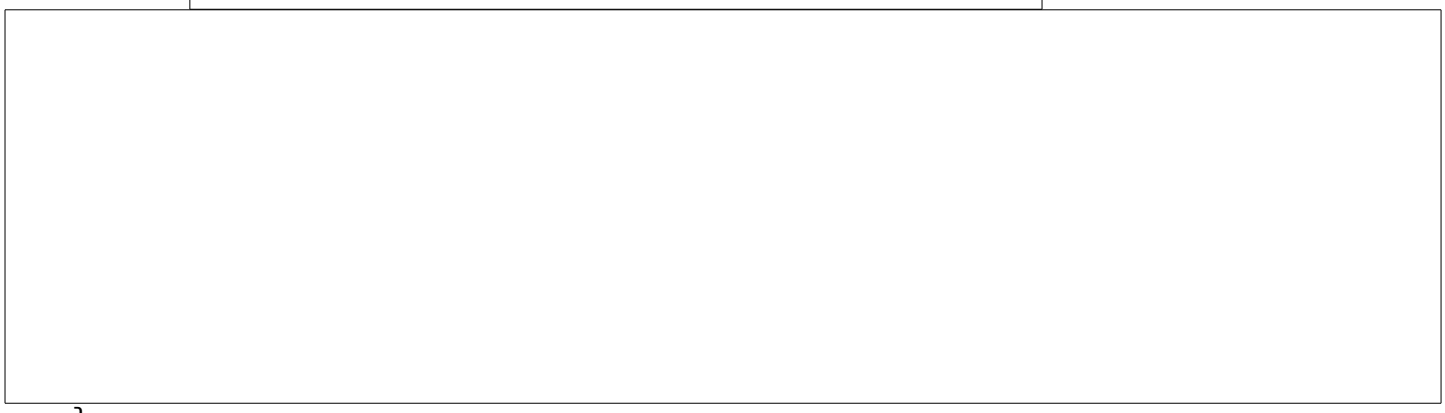
```
public class Fraction {  // lisez bien les commentaires
    private [                    ] int num; // numérateur   (ne doit pas changer)
    private [                    ] int den; // denominateur (ne doit pas changer)
    [                            ] Fraction(int num, int den){
        [                        ] = num;
        if (den != 0)
            [                    ] = den;
        else // lance une exception
            throw new IllegalArgumentException("Fraction avec dénominateur zéro !");
    }
    /**  méthode toDouble : donne la valeur réelle (convertir avant de diviser!)*/
    public [                                                ] {
        [                                                    ]
    }
```

```
/** méthode fois : multiplie avec une autre fraction (avec a/b x c/d = ac/bd)
    EN ÉVITANT LES NullPointerExceptions !                                  */
public [                    ] {
[                                                                          ]
}
/** méthode plus : additionne (avec a/b + c/d = (ad+cb)/bd)
    EN ÉVITANT LES NullPointerExceptions !                                */
public [                    ] {
[                                                                          ]
}
public [                    ] {
    String s;
    if (den==1)
        s = "("+num+")";
    else
        s = "[("+num+")/("+den+")]";
    [                    ]
}
/** méthode inverse : a/b -> b/a  si a!=0  ou  null si a==0    */
public [                    ] {
[                                                                          ]
}
public boolean estNegative() {
    return num!=0 && (num<0) != (den<0);
}
private int pgcd(int a, int b){
    return a%b==0?b:pgcd(b,a%b);
}
```

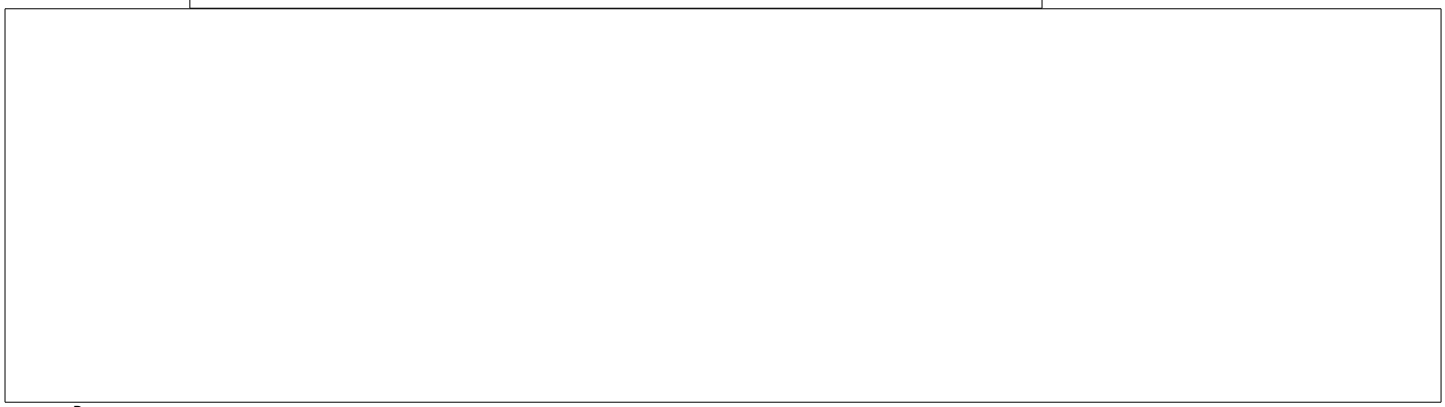
```
/** méthode reduire : prend la valeur absolue du numérateur et du dénominateur et
les divise par leur pgcd, puis si la fraction était < 0, rend le numérateur < 0 */
public                                                    {
```



```
}
/** méthode divisee, utilise fois et inverse : (a/b) / (c/d) = (a/b) x (d/c)
    EN ÉVITANT LES NullPointerExceptions  !                                  */
public                                                    {
```



```
}
/** méthode egale, renvoie true si (a/b) = (c/d)  Ne compare pas le résultat de
la division réelle (imprécis) mais réduit d'abord les fractions puis compare le
numérateur et le dénominateur.   ... EN ÉVITANT LES NullPointerExceptions  !   */
public                                                    {
```



```
}
```

```
    /** méthode equals, teste d'abord l'égalité superficielle puis si l'objet
    comparé est une instance de (instanceof) Fraction avec le même état, sans
    réduire: 2/4 et 1/2 ne sont pas "equals", (a/b) equals (c/d) ssi a=c et b=d   */
    @Override public boolean equals(Object obj) {
```

```
    }
}
```

4) On ajoute les lignes suivantes à la fin de la méthode `test` de la classe `TestFraction` : **(6 pts)**

```
11. ____f2.inverse();
12. ____System.out.println("f2 = "+f2);
13. ____Object obj1=new Object();
14. ____Object obj2=new Fraction(13,3);
15. ____System.out.println(obj1);
16. ____System.out.println(obj2);
17. ____System.out.println((Object)obj2);
18. ____System.out.println((Object)f1);
19. ____Fraction f3=obj1;
20. ____Fraction f4=obj2;
21. ____Fraction f5=(Fraction)obj1;
22. ____Fraction f6=(Fraction)obj2;
23. ____System.out.println(obj2.inverse());
24. ____System.out.println(f4.inverse());
25. ____System.out.println(f5.inverse());
26. ____System.out.println(f6.inverse());
27. }
```

a) Certaines lignes contiennent des erreurs. Indiquez **lesquelles** (leur numéro), **avec le type d'erreur** (compilation, exécution, logique) et **une brève explication** (répondez sur votre double-feuille).

b) Supprimez les lignes erronées et donnez le résultat de l'exécution de celles qui restent en indiquant à côté de chaque ligne affichée le numéro de ligne du code source ci-dessus correspondant (Attention : toute réponse sans ces indications ne sera pas corrigée).

Bon courage.

Amine Brikci-Nigassa

nh2@libretlemcen.org
nh2blog.wordpress.com
twitter.com/nh2