

★ **Exercice 1.** L'objectif de ce premier exercice est de concevoir et réaliser les classes de bases d'un programme manipulant des points, des segments, des triangles et d'autres formes géométriques.

▷ **Question 1. L'objet Point**

- Décrivez (en langage naturel) le comportement que l'on peut attendre d'un objet **Point**. Réalisez la carte CRC correspondante.
- Spécifiez l'interface publique d'un objet **Point**.
- Spécifiez les profils des constructeurs de la classe **Point**.
- Implémentez la classe **Point** en écrivant le corps des méthodes de l'interface publique.

▷ **Question 2. La classe de test TestPoint de la classe Point**

Écrivez une classe de test qui teste le comportement réel de la classe **Point** et le compare au comportement attendu. Ci-dessous, vous trouverez un exemple de sortie d'une classe de test.

```

1 [ok] (new Point()).getX()
2 [ok] (new Point()).getY()
3 [ok] (new Point(3., 2.)).getX()
4 [ok] (new Point(3., 2.)).getY()
5 [ok] (new Point(new Point(3., 2.))).getX()
6 [ok] (new Point(new Point(3., 2.))).getY()
7 [ok] p.setX(4.).getX()
8 [ok] p.setY(7.).getY()
9 [ok] (new Point(2.,3.)).translater(4.,7.)
10 [ok] (new Point(2.,3.)).translater(4.,7.)

```

▷ **Question 3. L'objet Segment**

- Décrivez (en langage naturel) le comportement que l'on peut attendre d'un objet **Segment**. Réaliser la carte CRC correspondante.
- Spécifiez l'interface publique d'un objet **Segment**.
- Spécifiez les profils des constructeurs de la classe **Segment**.
- Implémentez la classe **Segment** en écrivant le corps des méthodes de l'interface publique.

▷ **Question 4. La classe de test TestSegment de la classe Segment**

Écrivez une classe de test qui teste le comportement réel de la classe **Segment** et le compare automatiquement au comportement attendu.

▷ **Question 5. L'objet Triangle**

Suivez la même procédure que précédemment pour réaliser la classe **Triangle**.

★ **Exercice 2.**

▷ **Question 6.** Concevez une classe **Fraction** permettant de représenter des nombres rationnels. Cette classe doit permettre :

- de consulter et de modifier le numérateur et le dénominateur du nombre,
- de multiplier un nombre rationnel par un coefficient entier,
- d'ajouter, de soustraire, de multiplier, et de diviser deux nombres rationnels,
- d'inverser un nombre rationnel,
- de comparer deux nombres rationnels,
- de réduire un nombre rationnel.

Vous trouverez ci-dessous l'interface publique de la classe **Fraction** à implémenter.

```

1 /**
2  * Calcule le pgcd de deux entiers
3  * @param a un des deux entiers
4  * @param b un des deux entiers
5  * @return le pgcd des deux entiers
6  */
7 public static int pgcd (int a, int b) ;
8
9 /**

```

```

10  * Constructeur d'une Fraction
11  * @param num Numerateur
12  * @param den Denominateur
13  */
14  public Fraction(int num, int den) ;
15
16  /**
17  * Retourne le numerateur de la Fraction
18  * @return le numerateur
19  */
20  public int getNumerateur() ;
21
22  /**
23  * Retourne le denominateur de la Fraction
24  * @return le denominateur
25  */
26  public int getDenominateur() ;
27
28  /**
29  * Change la valeur du denominteur
30  * @param nd le nouveau denominateur
31  */
32  public void setDenominateur(int nd) ;
33
34  /**
35  * Change la valeur du numerateur
36  * @param nd le nouveau numerateur
37  */
38  public void setNumerateur(int nn) ;
39
40  /**
41  * Test si la Fraction est egale a une autre Fraction
42  * @param f Fraction a comparer
43  * @return vrai si les deux Fractions sont egales
44  */
45  public boolean egaleA(Fraction f) ;
46
47  /**
48  * Ajoute une Fraction a la Fraction
49  * @param Fraction a ajouter
50  */
51  public void ajoute(Fraction f) ;
52
53  /**
54  * Reduit/Simplifie la Fraction
55  */
56  public void reduire() ;
57
58  /**
59  * Inverse la Fraction<br>
60  * Le signe est au numerateur.
61  */
62  public void inverse() ;
63
64  /**
65  * Multiplie la Fraction par un Coefficient
66  * @param i Coefficient multiplicateur
67  */
68  public void multiplierParCoeff(int i) ;

```

Pour réaliser la méthode `public void reduire()`, vous utiliserez la méthode donnée ci-dessous (recopiez la définition de cette méthode dans votre classe `Fraction`) :

```

1  public static int pgcd(int a, int b) {
2      int c;
3      while (b != 0) {
4          c = a % b;
5          a = b;
6          b = c;
7      }
8      return a;
9  }

```

**Conseil :** Procédez par étapes :

- (a) écrivez une méthode de la classe `Fraction`,
- (b) décommentez uniquement les tests correspondants à cette méthode dans la classe `TestFraction` qui vous est fournie (cf. `/home/depot/1A/P00/TP1/TestFraction.java`),
- (c) puis continuez avec la prochaine méthode que vous devez implémenter.

▷ **Question 7.** Complétez la classe de test `TestFraction` afin de tester le comportement des méthodes `multiplie(Fraction f)`, `soustrait(Fraction f)` et `divise(Fraction f)`.