

Examen de Programmation Objet

Durée : 1h

Aucun document autorisé

Mobiles interdits

Nom :

Prénom :

On souhaite représenter un point $p = (x, y)$ du plan cartésien 2 dimensions.

► 1. Écrivez en JAVA la classe Point avec :

- deux variables privées `x` et `y` de type **double**;
- un ou plusieurs constructeurs pour initialiser l'objet courant;
- les accesseurs `getX` et `getY`;
- les mutateurs `setX` et `setY`;
- la fonction `distance` qui renvoie la distance entre le point courant et un point passé en paramètre. Rappel : la distance entre 2 points $p_1 = (x_1, y_1)$ et $p_2 = (x_2, y_2)$ est $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$;
- la fonction booléenne `égal` qui renvoie vrai si point courant et un point passé en paramètre sont égaux et faux sinon
- la fonction `toString` qui renvoie la représentation du point courant sous forme d'une String au format (x, y) .

```
/**
 * La classe Point représente un point de l'espace cartésien à 2 dimensions
 */
public class Point {
    //abscisse
    private double x;
    //ordonnée
    private double y;

    /**
     * Rôle : crée le point (x, y)
     *
     * @param x abscisse
     * @param y ordonnée
     */
    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }

    /**
     * Rôle : crée le point (0, 0)
     */
    public Point() {
```

```
        this.x = this.y = 0.0;
    }

    /**
     * Rôle : renvoie l'abscisse du point courant
     *
     * @return x
     */
    public double getX() {
        return this.x;
    }

    /**
     * Rôle : renvoie l'ordonnée du point courant
     *
     * @return y
     */
    public double getY() {
        return this.y;
    }

    /**
     * Rôle : change l'abscisse du point courant
     */
    public void setX(double x) {
        this.x = x;
    }

    /**
     * Rôle : change l'ordonnée du point courant
     *
     * @param y
     */
    public void setY(double y) {
        this.y = y;
    }

    /**
     * Rôle : @return représentation sous forme de chaîne de caractères
     *
     * de l'objet courant
     */
    public String toString() {
        return "(" + this.x + "," + this.y + ")";
    }

    /**
     * Rôle : teste l'égalité this == p
     *
     * @param p
     */
    public boolean égal(Point p) {
        return this.x == p.x && this.y == p.y;
    }

    /**
     * Rôle : renvoie la distance entre le point courant et le point p
     *
     * @param p
```

```

    */
    public double distance(Point p) {
        double dx = p.x - this.x;
        double dy = p.y - this.y;
        return Math.sqrt(dx*dx+dy*dy);
    }
}

```

.....

On souhaite représenter les quadrilatères du plan (polygones à 4 sommets).

- 2. Écrivez la classe `Quadrilatère` avec :
- une variable privée `lesPoints` de type `tableau` de `Point` ;
 - un constructeur qui initialise le tableau `lesPoints` à partir de 4 points passés en paramètre du constructeur. Vous vérifierez que les 4 points sont différents.
 - la fonction `périmètre` qui renvoie le périmètre du `Quadrilatère` courant. Vous devez utiliser un énoncé itératif.

```

public class Quadrilatère {
    private Point [] lesPoints;

    /**
     * Rôle : crée un Quadrilatère initialisé avec les 4 points p0, p1, p2 et p3
     *         s'ils sont différents
     */
    public Quadrilatère(Point p0, Point p1, Point p2, Point p3) {
        //vérifier si les points sont différents
        assert(!p1.égal(p0));
        assert(!p2.égal(p0) && !p2.égal(p1));
        assert(!p3.égal(p0) && !p3.égal(p1) && !p3.égal(p2));
        //ok les points sont différents => créer le tableau de 4 Point(s)
        lesPoints = new Point[4];
        //memoriser les points dans le tableau
        this.lesPoints[0] = p0;
        this.lesPoints[1] = p1;
        this.lesPoints[2] = p2;
        this.lesPoints[3] = p3;
    }

    /**
     * Rôle : renvoie le périmètre du quadrilatère courant
     */
    public double périmètre() {
        double d=0.0;
        //faire la somme des longueurs des lesPoints.length-1 premiers côtés
        for (int i=0; i<lesPoints.length-1; i++)
            d+=lesPoints[i].distance(lesPoints[i+1]);
        //ajouter la longueur du dernier côté
        return d+lesPoints[lesPoints.length-1].distance(lesPoints[0]);
    }
}

```

```

}
.....

```