Licence Informatique — **UE J1IN4W01**
Université de Bordeaux — **Programmation 2**
2015-2016 — **Travaux Dirigés - feuille 3**

Interface. Relation entre implémentation et interface. Type réel/déclaré

Exercice 1. On souhaite créer une classe `Point2DPolaire` qui permet la représentation d'un point dans un plan grâce respectivement à des coordonnées polaires (`module` et `argument`).

1. Écrire les méthodes qui définissent la forme canonique de la classe `Point2DPolaire`.
2. Rajouter les méthodes :

   ```
   public double getAbscisse();
   public double getOrdonnee();
   public void setAbscisse(double x);
   public voir setOrdonnee(double y)
   ```

   Pour un point P de coordonnées polaires (ρ, θ), ses coordonnées cartésiennes correspondantes sont : $x = \rho \cos\theta$ et $y = \rho \sin\theta$.
   Pour les méthodes `setAbscisse` et `setOrdonnee` il faudrait recalculer le module et l'argument en fonction des nouvelles valeurs pour l'abscisse et l'ordonnée (pour la transformation, voir la feuille 2).

   ```
   public void translation(double dx, double dy);
   ```

   Pour effectuer une translation d'un point P représenté par des coordonnées polaires, il faudra passer par ses coordonnées cartésiennes (voir la feuille 2).

   ```
   public void homothetie(double rapport);
   ```

   Soit $P(\rho, \theta)$ un point dans un repère polaire. Soit $P'(\rho', \theta')$ l'image de P par l'`homothetie` de rapport k, alors $\rho' = \rho * k$ et $\theta' = \theta$.

   ```
   public void rotation(double angle);
   ```

   Soit $P(\rho, \theta)$ un point dans un repère polaire. Soit $P'(\rho', \theta')$ l'image de P par la `rotation` d'angle α, alors $\rho' = \rho$ et $\theta' = \theta + \alpha$.
3. Écrire une classe `TestPoint2DPolaire` qui teste les méthodes proposées.

Exercice 2. Proposer une **interface** `Point2D` déclarant des méthodes permettant de manipuler des points $2D$ définis respectivement soit en coordonnées cartésiennes soit en coordonnées polaires. On s'inspirera des classes `Point2DCartesien` et `Point2DPolaire` .

```
public class Point2DPolaire {
    public Point2DPolaire(double module, double argument){}
    public Point2DPolaire(Point2DPolaire p){}
    public double getModule(){}
    public double getArgument(){}
    public void setModule(double x){}
    public void setArgument(double y){}
    public double getAbscisse(){}
    public double getOrdonnee(){}
    public void setAbscisse(double x){}
    public void setOrdonnee(double y){}
    public Point2DPolaire clone(){}
    public boolean equals(Point2DPolaire p){}
    public String toString(){}
    public void translation(double dx, double dy) {}
    public void homothetie(double rapport){}
    public void rotation(double angle){}
```

```
    private double module;
    private double argument;
}

public class Point2DCartesien {
    public Point2DCartesien(double x, double y){}
    public Point2DCartesien(Point2DCartesien p){}
    public double getModule(){}
    public double getArgument(){}
    public void setModule(double x){}
    public void setArgument(double y){}
    public double getAbscisse(){}
    public double getOrdonnee(){}
    public void setAbscisse(double x){}
    public void setOrdonnee(double y){}
    public Point2DCartesien clone(){}
    public boolean equals(Point2DCartesien p){}
    public String toString(){}
    public void translation(double dx, double dy) {}
    public void homothetie(double rapport){}
    public void rotation(double angle){}

    private double x;
    private double y;
}
```

Exercice 3. On veut maintenant que les classes `Point2DCartesien` et `Point2DPolaire` implementent l'interface `Point2D`. Quelles modifications doivent subir les méthodes de ces classes ? Est-ce que l'on peut comparer un objet dont le type réel est `Point2DCartesien` avec un objet dont le type réel est `Point2DPolaire` ?

Exercice 4. Écrire une classe `GeoLib` qui contient des exemples d'utilisation des méthodes de l'interface `Point2D`. On pourra, par exemple, définir une méthode de calcul de la distance entre deux points `Point2D` ainsi qu'une méthode qui calcule le centre de gravité d'un tableau de points.