P.O.O. : Java — Sujet n°2
Le Java Orienté Objet

Pensez à créer un **nouveau projet** (`Sujet2`) qui contiendra **un package pour chaque exercice**.

`Exo01` - *Des polygones*

On considère dans cet exercice, uniquement des polygones dont les côtés sont tous de même longueur. On se propose d'écrire un programme qui va permettre de :

- créer des polygones,
- compter le nombre de polygones créés,
- afficher les caractéristiques d'un polygone,
- calculer le périmètre d'un polygone,
- crée un polygone en ajoutant/enlevant des côtés à un polygone existant,
- comparer deux polygones et dire celui qui a le plus petit périmètre.

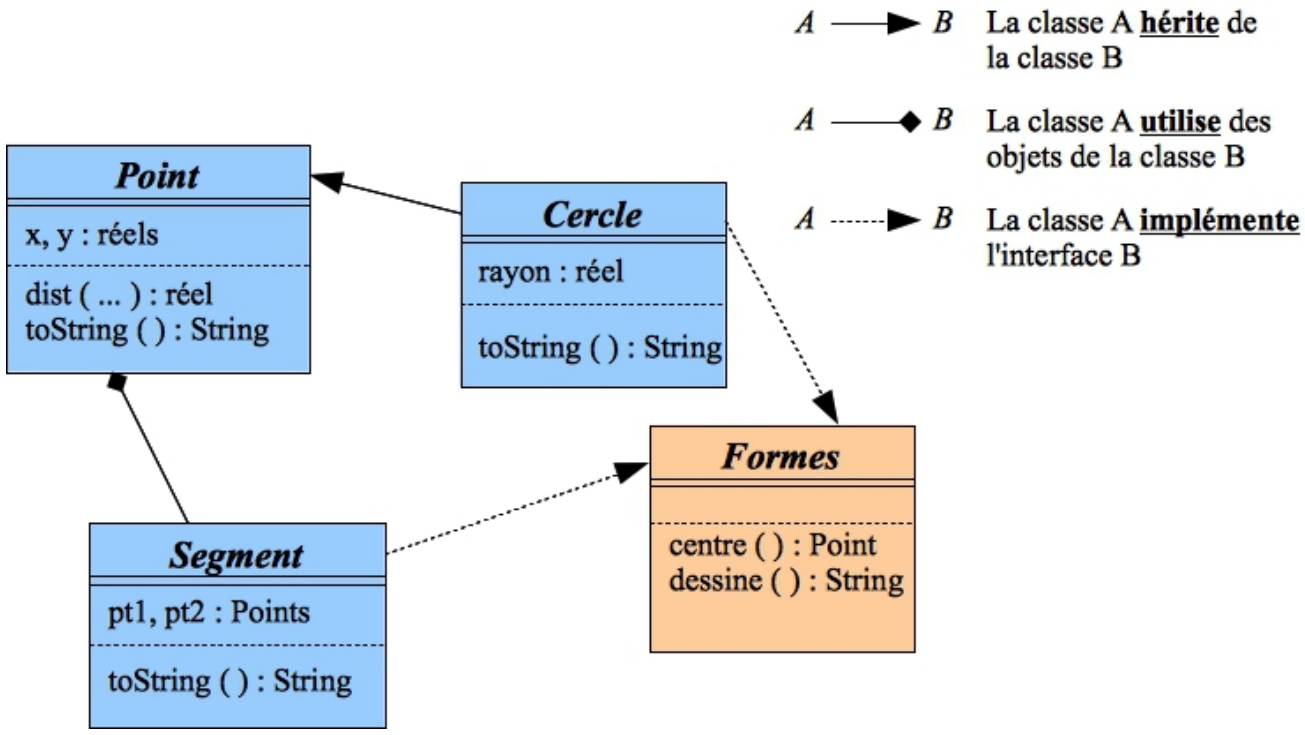
Les étapes à suivre sont les suivantes :

1. a) Écrire la déclaration de la classe `Polygone` : une variable de classe, les variables d'instances, un constructeur par défaut, un constructeur avec paramètres, les accesseurs et mutateurs.
   b) Écrire la classe `Principal` qui contiendra la méthode `main`. Cette dernière
   - Créera un polygone, `p1`, en utilisant le constructeur avec paramètres, puis en utilisant les accesseurs affichera le nombre de côtés et leur longueur.
   - Créera un polygone, `p2`, en utilisant le constructeur par défaut, puis en utilisant les mutateurs donnera une valeur aux nombres de côtés et à la longueur. Enfin, ces caractéristiques seront affichées.
   - Affichera le nombre de polygones créés.
2. a) Redéfinir la méthode `toString` de la classe `Object` (polymorphisme) pour qu'elle permette d'afficher les caractéristiques d'un polygone.
   b) Modifier la méthode `main` pour qu'elle affiche toutes les caractéristiques des deux polygones créés à la question précédente en utilisant la méthode `toString`.
3. a) Écrire la méthode `calcPerim` qui initialise une variable d'instances correspondant au périmètre d'un polygone. Cette méthode sera privée car elle ne sera jamais appelée depuis une autre classe.
   b) Faire les modifications nécessaires pour que la variable d'instances correspondant au périmètre, soit initialisée dès que les caractéristiques d'un polygone sont modifiées (création et mutateurs)
   c) Modifier la méthode `toString` pour que cette nouvelle caractéristique puisse également être affichée.
4. a) Écrire la méthode `presqueClone` qui crée un nouveau polygone en ajoutant/enlevant des côtés à un polygone déjà existant (ce nombre de côtés sera passé en paramètre).
   b) Modifier la méthode `main` pour créer un nouveau polygone en modifiant `p1` : le nombre de côtés à ajouter/enlever sera donné par l'utilisateur. Permettre ensuite l'affichage des caractéristiques de ce nouveau polygone.

5. a) Écrire la méthode `plusPetit` qui compare deux polygones et retourne celui ayant le plus petit périmètre.
   b) Modifier la méthode `main` pour afficher les caractéristiques du polygone ayant le plus petit périmètre entre `p1` et `p2`.

`Exo02` - *Dessiner des formes*

On souhaite écrire un programme Java qui « dessine » des formes géométriques très simples : des cercles et des droites. Pour cela les classes (en bleu) et l'interface (en orange) désignées par le graphique suivant seront utilisées :



La classe `Point` :

Un point est caractérisé dans un repère orthonormé, par un couple de coordonnées réelles.

1. Écrire la déclaration de la classe `Point` : *variables d'instances*, *constructeur* permettant de donner des valeurs aux coordonnées d'un point et les *accesseurs* aux variables d'instances.
2. Redéfinir la méthode `toString` (polymorphisme) afin qu'elle décrive un point en affichant ces coordonnées sous la forme : (x, y).
3. Écrire la méthode `dist` qui permet de *calculer* la distance entre un point donné et le « point courant ». Cette méthode n'affiche rien et ne demande rien à l'utilisateur.
   Indication : la méthode `sqrt` permet le calcul de la racine carrée. Elle se trouve dans la classe `Math`. Son prototype est : `double Math.sqrt(double) ;`
4. Écrire une classe contenant la méthode `main` afin de tester la classe `Point` : créer 2 points qui vous serviront pour la suite.

La classe `Cercle` :

Un cercle est vu comme un point (le centre du cercle) auquel est associé un rayon.

1. Écrire la déclaration de la classe `Cercle` : *variable d'instance*, *constructeur* permettant de donner des valeurs aux différentes caractéristiques d'un cercle.
2. Redéfinir la méthode `toString` (polymorphisme) afin qu'elle décrive un cercle : « Cercle de centre (x, y) et de rayon r ».
3. Modifier la méthode `main` pour tester la classe `Cercle` : utiliser un des points précédemment créés pour construire un cercle dont le rayon sera donné par l'utilisateur.

La classe `Segment` :

Un segment est caractérisé par les deux points situés à ses extrémités.

1. Écrire la déclaration de la classe `Segment` : *variables d'instances*, *constructeur* permettant de donner des valeurs aux différentes caractéristiques d'un segment à partir de deux points donnés et les *accesseurs* aux variables d'instances.
2. Redéfinir la méthode `toString` (polymorphisme) afin qu'elle décrive un segment : « Segment délimité par les points (x1, y1) et (x2, y2) ».
3. Modifier la méthode `main` pour tester la classe `Segment` : utiliser les deux points déclarés précédemment pour construire un segment.

L'interface `Forme`

L'interface `Forme` déclare les méthodes abstraites permettant de calculer le point représentant le « centre » d'une forme et de « dessiner » une forme.

1. Écrire l'interface `Forme` qui déclare les deux méthodes abstraites :
   a) `centre` retournera un point,
   b) `dessine` fera un affichage.
   Indication : Dans Eclipse, la création d'une interface se fait par le biais du menu `File/New/Interface`, ou simplement en choisissant `Interface` dans le menu déroulant accessible près de l'icône permettant la création d'une nouvelle classe (flèche vers le bas à droite de l'icône C+ vert).
2. Dans la classe `Cercle` :
   a) écrire la méthode `centre` retournera le centre du cercle,
   b) écrire la méthode longueur calculera la circonférence du cercle,
   c) écrire la méthode `dessine` affichera le message « Au compas : » suivi des caractéristiques du cercle (méthode toString).
3. Dans la classe `Segment` :
   a) écrire la méthode `centre` retournera le milieu du segment,
   b) écrire la méthode longueur calculera la longueur du segment,
   c) écrire la méthode `dessine` affichera le message « A la règle : » suivi des caractéristiques du segment (méthode toString).
4. Dans la classe contenant la méthode `main` :
   a) déclarer un tableau de trois formes, puis mettre le cercle dans la première case, le segment dans la seconde case et créer un nouveau segment ou cercle que vous placerez en dernière position.
   b) « dessiner » toutes les formes de votre œuvre d'art.

Aller plus loin avec la classe `Cercle` ...

On va compléter la classe cercle en écrivant un nouveau constructeur : son but est de créer un cercle dont un segment donné en paramètre en est le diamètre.

1. Écrire dans la classe `Cercle` un constructeur dont la signature est :
   ```
   public Cercle (Segment unSeg) ;
   ```
   Ce nouveau constructeur fera un appel au constructeur de la classe `Cercle` écrit précédement.
2. Modifier la classe `main` pour :
   a) mettre une 4ème case au tableau des formes,
   b) initialiser cette 4ème case avec le cercle créé à partir du segment déjà créé,
   c) « dessiner » toutes les formes du dessin.