Programmation Orientée Objets

TD-TP n°5 : Classes abstraites et interfaces

Exercice 1 : Diagramme de classes

Compléter le diagramme de classes modélisant l'intégralité des données du sujet de TP, incluant le TP n°5.

Exercice 2 : Classe « Cercle »

Écrire la classe `Cercle` (`centre`, `rayon`, `perimetre()`, `surface()`, `dedans(Point p)`, `plusGrand(Cercle c)`, `translater(double dx, double dy)`).

Exercice 3 : Classe abstraite « Forme »

On souhaite pouvoir manipuler facilement, ensemble, des « `Cercle` » et des « `Polygone` ».
Pour cela, on doit créer une super-classe abstraite « `Forme` » commune à toutes les formes géométriques (et introduisant toutes les méthodes de `Cercle`).
Effectuer les modifications permettant à toutes les formes géométriques d'hériter de la classe `Forme` : on doit pouvoir appeler `translater()` et les 4 autres méthodes de `Cercle` pour toutes les formes.
Copier-coller et adapter `EnsemblePolygone` et `EnsemblePolygoneAlea` afin de les généraliser en deux classes `EnsembleForme` et `EnsembleFormeAlea` (qui vont les remplacer).

Exercice 4 : Transtypage descendant et identification du type à l'exécution (RTTI).

Écrire le code nécessaire pour tenir à jour un compteur de chaque type de formes géométriques.
Effectuer le minimum de modifications afin d'afficher le numéro de chaque forme et le numéro spécifique au type de forme dans la méthode `toString()` :

```
Forme (n°12), Polygone (n°3) : (0,0), (2,0), (2,2), (1,9)
```

Empêcher la méthode `sommeSurfaces()` d'appeler `estConvexe()` dans le cas des cercles.

Exercice 5 : Quelques méthodes.

Écrire la méthode `zoom(int zoomX)` dans la classe `Cercle`, qui multiplie le rayon par `zoomX`.
Écrire la méthode `zoomZoomable(int zoomX)` dans la classe `EnsembleForme`, qui appelle `zoom()` sur tous les cercles de l'ensemble.
Faire de même pour les rectangles et les polygones réguliers.

Exercice 6 : Interface « Forme » et classe abstraite « FormeSurface »

On souhaite maintenant regrouper dans une interface tout ce qui est abstrait dans la classe `Forme`. Pour cela, nous allons décomposer la classe abstraite `Forme` en :

- Une interface `Forme` (100% abstraite) ;
- Une classe abstraite `FormeSurface`.

Créer la classe abstraite `FormeSurface` qui regroupe tout ce qui est concret dans la classe `Forme`.
Transformer la classe abstraite `Forme` en interface.
Est-ce que `FormeSurface` peut devenir une classe concrète ? Faut-il modifier `EnsembleForme` ?

Exercice 7 : Interfaces « Zoomable » et « Dessinable »

Écrire une 2nde interface `Zoomable` implémentée par `Cercle`, `Rectangle` et `PolygoneRegulier`.
Écrire une 3ème interface `Dessinable` comportant une constante `TAILLE_MAX` (de la fenêtre graphique), une variable `java.awt.Color couleur` (couleur d'initialisation des formes) et une méthode `seDessiner()`.
Modifier `Cercle`, `Rectangle` et `PolygoneRegulier` afin qu'elles implémentent `Zoomable` et `Dessinable`.

Exercice 8 : Interface « Comparable »

Modifier `FormeSurface` afin qu'elle implémente l'interface `Comparable`. Il est nécessaire de redéfinir la méthode `int compareTo(Object o)` qui retourne -1, 0 ou 1 selon que l'objet est considéré comme plus petit, égal ou plus grand que l'objet `o` (en fonction de leur surface).