PCOO – TD 9

Sauvegarde des formes

Nous allons donner à l'utilisateur la possiblité de sauvegarder les formes dans notre éditeur de formes du TD 6. Nous allons ajouter deux classes aux projets :

— `ShapeWriter` pour écrire des formes dans un flux de caractères ;
— `ShapeReader` pour lire des formes dans un flux de caractères.

Dans le fichier de sortie, chaque ligne du fichier décrit une forme. Par exemple, la ligne "`Circle 12.0 34.0 40.0`" décrit un cercle de centre $(12, 34)$ et de rayon 40. La ligne "`Rectangle 23.0 40.0 30.0 40.0`" décrit un rectangle défini par le coin supérieur gauche $(23, 40)$, la largeur 30 et la hauteur 40.

1. Utilisez le patron de conception `Visitor` afin de faciliter l'écriture de la classe `ShapeWriter` et le maintien de celle-ci lors de l'ajout de nouvelles formes. Vous pouvez modifier l'interface `Shape` et ajouter de nouvelles classes et interfaces. `ShapeWriter` possède une méthode statique `void write(File file, Iterable<Shape> shapes)`.
2. Afin de regrouper, pour chaque forme, son écriture (sérialisation) et sa lecture (déserialisation) dans un même fichier, définissez l'interface `ShapeSerializer<S extends Shape>`. Cette interface contient les trois méthodes suivantes :
   — `String code()` : `"Circle"` ou `"Rectangle"` ;
   — `String serialize(S shape)` : retourne une chaîne qui décrit la forme ;
   — `S unserialize(String s)` : reconstruit la forme à partir de `s`.
3. Écrivez les classes `CircleSerializer` et `RectangleSerializer` qui implémentent l'interface `ShapeSerializer` et modifiez le code écrit de la classe `ShapeWriter` afin d'utiliser ces nouvelles classes.
4. Écrivez la classe `ShapeReader` qui possède la méthode statique `List<Shape> read(File file)`. Elle lit les formes dans le fichier `File` et retourne la liste contenant les formes lues. `ShapeWriter` possède une méthode statique `List<Shape> read(File file)`. Quelle structure de données peut être utilisée pour trouver le `ShapeSerializer` associé au code de la forme ?