D14 - Java et la programmation par composants

Durée 3h.

Important : **Lisez tout avant de commencer**, il est fortement recommandé de faire un diagramme de classes simple avant de programmer. La seule documentation accessible comprend : **la javadoc, vos TP, et les supports**. Tout le code sera commenté brièvement, compilable et exécutable via un script maven, **V**ous enverrez une archive des sources à bruno@univ-tln.fr.

Mise en en place

Pour gérer le projet vous mettrez en place un projet appelé <*votrelogin*>ExamenD14.

Un système de gestion de figures

On souhaite représenter des *figures* qui sont soit des *disques* (repérés dans le plan par les coordonnées de leur centre appelé origine)*, des polyèdres quelconques* (représentés comme une liste de sommet consécutifs)*,* des *rectangles*, et des *carrés (*créés à partir des coordonnées de leur coin supérieur gauche appelé origine*)*.

Les figures sont identifiées par un code unique composé d'une paire : lettre ('C' pour les cercles, 'R' pour les rectangles et 'S' pour les carrés) et d'un entier incrémenté automatiquement pour chaque type. Ce code ne doit jamais être modifié. Par exemple, les cercles auront comme identifiant C1, C2, ... et les rectangles R1, R2, ... Si vous ne savez pas comment faire utiliser un entier comment identifiant.

On souhaite calculer le *centre de gravité*/*barycentre* (`getCenter()`) de chaque *figure*. Par défaut deux instances de Figure sont égales si elles ont le même identifiant, la comparaison de base se fait aussi par rapport au code.

1°/Écrire les classes nécessaires et une classe **Test1** qui crée un tableau de figures et affiche la liste des barycentres.

2°/On souhaite maintenant représenter des *diagrammes* qui sont des collections de figures indexées par leur identifiant. La classe diagramme doit avoir une méthode getCenter() qui retourne son centre de gravité et une méthode Rectangle getBoundingBox() qui retourne un rectangle calculé automatiquement qui est la boite englobante de toutes les figures. ATTENTION, le boite englobante ne doit pas être recréée à chaque appel mais uniquement quand elle a changé. Écrire le code de la classe Diagramme. Les diagrammes sont identifiés par un entier unique. Écrire la classe Diagramme et une classe **Test2** qui crée un diagramme et affiche la boite englobante et montre comme on accède à une figure par son identifiant.

3°/Mettre en place une solution « simple » pour trier (i) directement une liste de figures par identifiant mais aussi (ii) ponctuellement par aire croissante. Donner un exemple des deux tris sur la même liste dans une classe **Test4**.

4°/Créer une base de données relationnelles (Mysql, H2 ou Postgresql) simple qui permet de rendre persistantes les figure. La modélisation retenue pour représenter les figures sera d'utiliser une seule relation pour toute la hiérarchie d'héritage avec des attributs facultatifs et un attribut type qui distingue les figures.

5°/Créer une IHM simple (Swing ou Javafx) qui affiche la liste des codes de toutes les figures et quand on clique dessus le détails des informations (sur la console dans un premier temps).

6°/Un défi : proposer une API fluent permettant de créer des diagrammes en utilisant le builder pattern :

```
float aire = Diagramme.getBuilder()
        .add(Polyedre.getBuilder().addSommet(10,10).addSommet(10,20).addSommet(20,15).build())
         .add(Cercle.getBuilder().setOrigine(10,10).setRayon(10).build())
         .add(Rectangle.getBuilder().setOrigine(20,20).setLargeur(10).setLongeur(20).build())
         .add(Carre.getBuilder().setOrigine(30,30).setCote(30).build())
.build().getBoudingBox().getCentre() ;
```