## Mini projet java (suite facultative)

Objectif :

Effectuer quelques transformations avant de générer le fichier dessin2.svg .

### Restructurer le code de tp.MyApp

**public** **class** MyApp {

**public** **static** **void** main(String[] args) {

System.***out***.println("cette application va générer un fichier dessin.svg");

//premiersTests();

*enchainerTransformationsEtGenerationFichierSvg*();

}

**public** **static** List<Figure2D> buildListeFigures() {

List<Figure2D> listeFigures = **new** ArrayList<>();

listeFigures.add(**new** Rectangle(100,180,200,50,"black",5,"blue"));

listeFigures.add(**new** Ligne(150,100,250,100,"green",4,**null**));

listeFigures.add(**new** Cercle(100,100,30,"black",3,"red"));

**return** listeFigures;

}

**public** **static** **void** enchainerTransformationsEtGenerationFichierSvg() {

List<Figure2D> listeInitialeFigures = *buildListeFigures*();

List<Figure2D> listeTransformeeFigures =

listeInitialeFigures.stream()

//effecter une premiere transformation de type zoomer(0.5)

//effecter une seconde transformation de type translation(0,120)

//filtrer selon le type de figure "Cercle" via un test de type instanceof Cercle

.collect(Collectors.*toList*());

MySvgUtil.*generateSvgFile*(listeTransformeeFigures, "dessin2.svg");

}

**public** **static** **void** premiersTests() {

//tests de la partie 1

}

}

Lancer une première fois le code restructuré pour vérifier la génération de dessin2.svg

### Coder des transformations et filtrages

Coder et tester (via un affichage graphique de dessin2.svg)

les transformations et filtrages suggérés par les commentaires de enchainerTransformationsEtGenerationFichierSvg()

### Demander interactivement (via des saisies) les transformations à effectuer

Au début de la méthode enchainerTransformationsEtGenerationFichierSvg() , ajouter du code permettant de demander à l'utilisateur les transformations et filtrage qu'il a envie de déclencher.

On pourra par exemple effectuer des saisies en *mode texte* ou en mode "*showInputDialog"*

On pourra par exemple utiliser des variables locales telles que dx,dy,coeffZoom,typeFig .

### éventuelle restructuration du code via le design pattern "visitor"

Lire les pages 76,77et78 de doc/design-patterns.pdf

et restructurer le code en déplaçant les méthodes toSvgSubString() et toSvgStringWithColor()

dans un visiteur externe au classes Figure2D/Ligne/Rectangle/Cercle de façon à ce que ce visiteur "SvgGeneratorVisitor" soit vu comme une extension qui pourrait cohabiter avec un autre visiteur.

**Attention**: Tp complexe sans solution en fin de journée.

--> effectuer une copie de sauvegarde du code fonctionnant bien avant de tenter une modification !

Solution approchée (plus complète en java) :

partie "**visiteur**" de la partie "***design\_patterns-ext***" du référentiel git

"***https://github.com/didier-mycontrib/design-patterns***"

### Révisions libres sur ce qui vous intéresse en java

### ....