Rapport Projet Orienté Objet 2

Introduction :

Le projet de Programmation Orienté Objet 2017 a été l'un des projets les plus intéressants que l'on a eu à faire, mais également l'un des plus difficiles.
La consigne était de modifier un fichier SVG contenant différents motifs plus ou moins gros. Malheureusement, l'utilisation d'un temporisateur n'est pas utilisé dans notre projet, l'optimisation ne prend pas assez de temps de recherche pour pouvoir se permettre l'utilisation d'un temporisateur.
Nous avons passé énormément de temps à réfléchir à comment faire au plus simple et au plus efficace possible pour parvenir à réussir le projet. La réflexion a sans doute pris plus de temps que la partie d'implémentation.

Principe du projet et implémentation :

L'implémentation du projet a été effectué en JAVA. Le choix du JAVA nous a paru logique pour l'UE de Programmation Orienté Objet.
Nous n'avons pas utilisé d'héritage, car nous n'en n'avons pas vu le besoin, ni eu le besoin.

Notre modélisation repose sur un système de matrices de rectangles qui s'imbriquent. Les rectangles sont des « rectangles de points » englobant chaque motif ou chaque groupe de motif.

► La lecture des fichiers SVG a été peut-être le plus difficile.
L'implémentation du Parseur.java a été l'implémentation ayant pris le plus de temps.
Il a fallu parser les motifs et récupérer leurs contenus, mais également ceux des groupes de motifs, puis, passer le contenu enregistré en point. Chaque lettre (m, M, l, L …) a été du être géré et permettre de construire les différents points. La création d'une forme (Forme.java) était effectué également ici en fonction des différents flags et des points crées.
► Nous avions en tête d'avoir des « rectangles » de points couvrant chaque motif. Chaque motif était entouré d'un rectangle de point dont on connaît les quatres coordonnées. A chaque nouveau motif, on regarde le placement des points précédents (du rectangle précédent) pour s'y placer à la suite. Nous avons également pris en compte la longueur et la largeur totale d'un fichier SVG. La mise en place d'un « rectangle » autour de la figure s'effectue grâce au Rouleau.java.

VETRIVEL Govindaraj
DIVRIOTIS Constantin
CMI ISR L3

De plus, si une « colonne » de motif est complète, on passe à la prochaine « colonne ». On vérifie également si une ligne est complète ou non.
Le fonctionnement ressemble à celui d'une matrice, c'est pourquoi pour le placement un autre programme MatrixRec.java est présent. Notre programme repose sur une matrice de rectangle, il était donc nécessaire pour le placement de ces rectangles d'y ajouter ce programme afin qu'elles s'imbriquent parfaitement.
► Ces coordonnées ont été interprétés et permettent avec le fichier Panneau.java (qui permet l'affichage via Fenêtre.java, fichier utilisant swing) d'afficher les lignes et les courbes grâce aux points et à l'interprétation du fichier SVG original. Pour les courbes, nous avons suivi la formule proposé dans le sujet.
► Enfin, évidemment la base de notre programme les deux programmes Point.java et Rectangle.java permettant de créer et manipuler nos deux objets principaux.

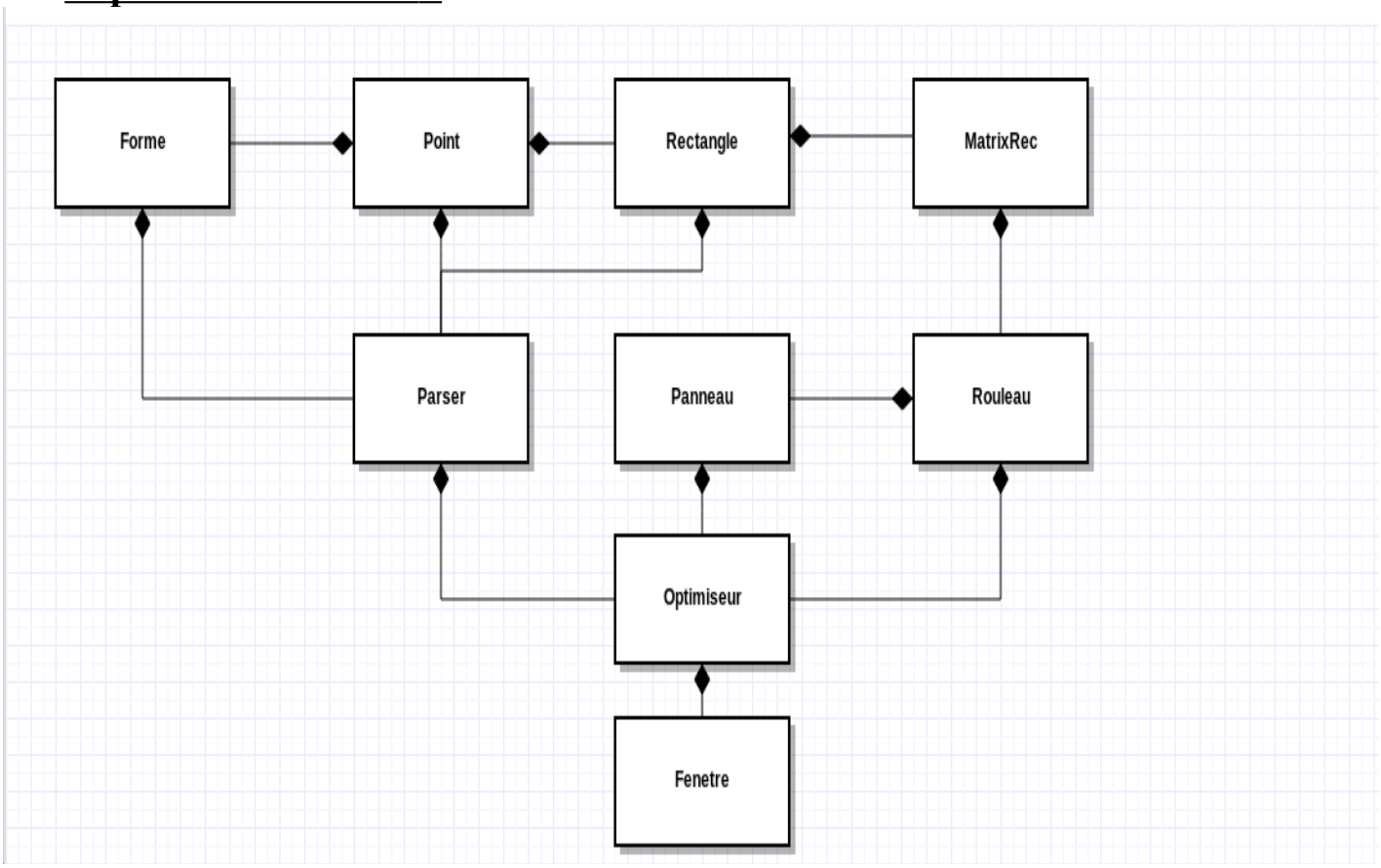
Répartition du travail :
VETRIVEL Govindaraj : MatrixRec.java, Rouleau.java, Optimiseur.java, Forme.java, Parseur.java, Rectangle.java
DIVRIOTIS Constantin : Point.java, Rectangle.java, Panneau.java, Fenetre.java, Parseur.java, Optimiseur.java

Limites :
L'optimisation n'est pas parfaite : en effet, nous aurions pu largement l'améliorer en découpant la matrice (tant que on ne touche pas la figure, on continue à découper le rectangle) afin d'obtenir les points encore plus précis ou la figure est délimité. Ainsi, un gain de place est encore possible.
De plus, placer d'autres motifs qui aurait pu s'insérer à l'intérieur d'un motif n'a pas été étudié plus que ça, le manque de temps ne nous a pas permis de pourvoir avoir la possibilité d'effectuer cette optimisation. Il aurait fallu un nouveau programme ou revoir MatrixRec.java et Rouleau.java afin de pouvoir vérifier la place libre à l'intérieur de chaque motif.

Conclusion
Nous sommes satisfaits de notre rendu, il permet d'exécuter ce que le sujet nous demandait, c'est-à-dire que les objets du fichier SVG orignal devait prendre le moins de place possible après avoir été traité par nos programmes.
Ce rendu aurait pu être meilleur, en effet, nous sommes conscients que l'optimisation aurait pu être bien meilleur, mais le temps nous a manquait pour ce projet.

Annexe :

Représentation UML :



VETRIVEL Govindaraj
DIVRIOTIS Constantin
CMI ISR L3