Rapport de projet PPIL :
Application de dessin Client/Serveur

Manukyan Samvel
Lagler Robin

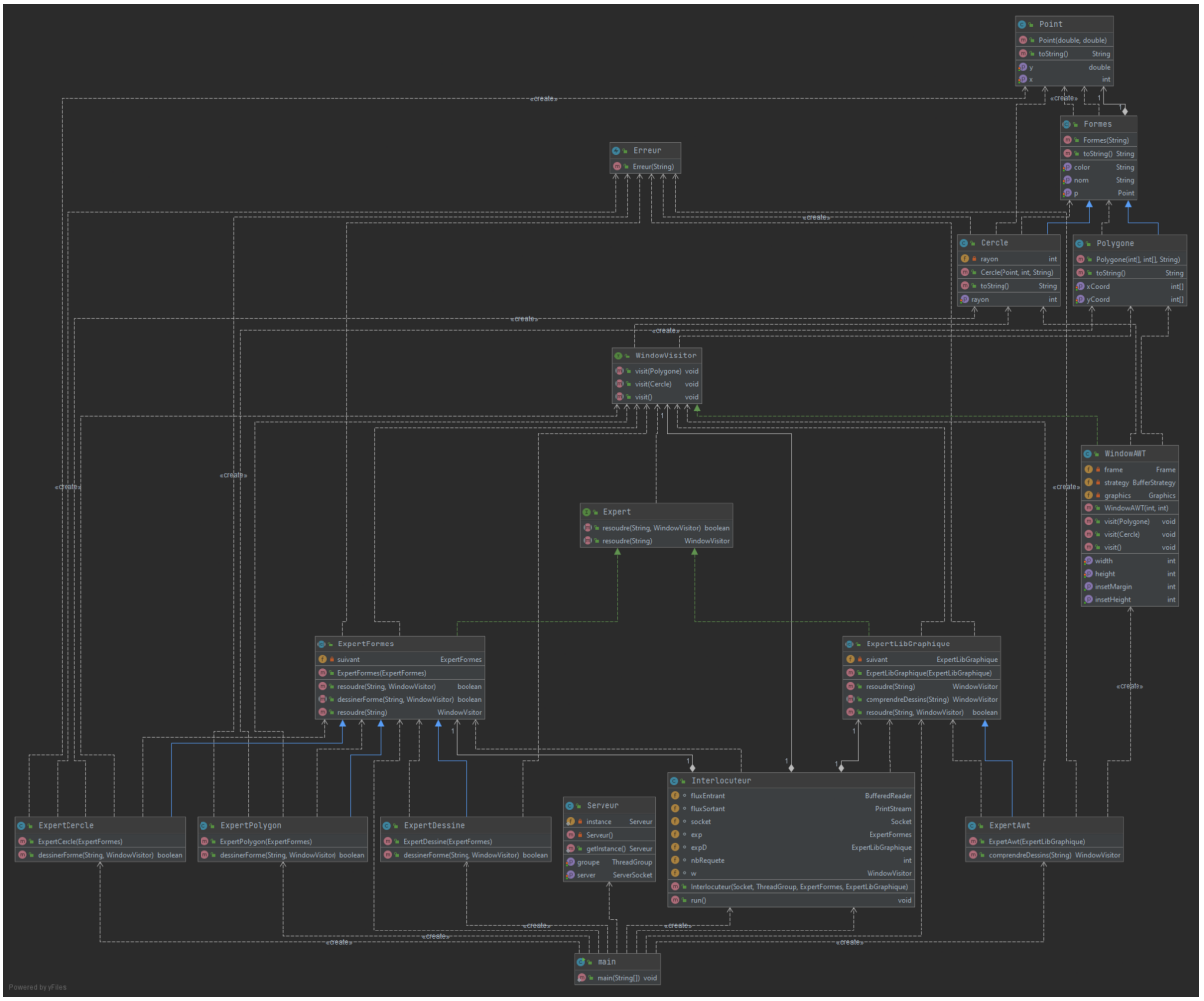
Résumé : Rapport de projet contenant tous les choix et intérprétations du sujet ainsi que la réponse. Il s'agit essentiellement de la structure du projet.

Sommaire :

1. **Structure du projet**
2. **Client**
    a. Client
    b. Coordonnées monde / écran
    c. Fenetre
    d. Vecteur2D
    e. Formes
    f. Méthodes et fonctions générales à une forme.
    g. Chargement / Sauvegarde de fichier (Avec COR Design Pattern)
    h. Dessin d'une forme (Avec Visitor Design Pattern)
3. **Serveur**
    a. Serveur
    b. Choix d'une librairie Graphique (Avec COR Pattern)
    c. Dessin d'une forme (Avec COR et Visitor Design Pattern)
4. **Conclusion**

1. Structure du projet.

Structure du client côté serveur :



2. Client

Comme l'a précisé le sujet, le client doit être réalisé en c++, il s'agit du client qui s'occupe de réaliser tout les calculs. Il envoie ensuite chaque forme au serveur.

a. Client

La classe Client nous sert à nous connecter au Serveur Java. Les informations du serveur sont déjà rentré (Nous pouvons aussi désactiver cette option et rentré manuellement l'adresse ip et le port du serveur).

Il s'occupe également d'enregistrer les formes avant de les envoyé dans le cas ou le client veut enregistrer toutes les formes qu'il a envoyés dans un fichier.

Fichier qu'il pourra également charger plus tard pour pouvoir retrouver ses formes d'origine.

Le client s'occupe également de gérer les coordonnées négative. Il utilise pour cela une autre classe.

b. Coordonnées monde / écran

Pour le passage des coordonnées monde vers les coordonnées écran, la classe TransformationAffine s'occupe de tout cela.

En effet, il s'agit d'une classe à laquelle on passe la taille de ce que l'on voit dans le monde et la taille de l'écran (il s'agit de deux points par vue qui définissent un rectangle, représentant les vues) et grâce à ceux ci, nous arrivons à déterminer certaines valeur qui nous seront utiles pour définir les postions de ces points dans l'écran.

Nous arrivons donc à déterminer : Lambda1, Lambda2, A et B.

Et pour chaque Vecteur2D passé dans la fonction TransAffine, il en résulte un nouveau Vecteur possédant des coordonnées adaptés à l'écran.

c. Fenetre

Nous avons décidé de créer une classe Fenetre, celle-ci sert à demander au serveur JAVA, d'utiliser une certaines librairie graphique parmis celle disponible ainsi que la taille de la fenêtre souhaité.

Ainsi le serveur, lorsqu'il reçoit la chaine de caractère correspondate, instancie une fenêtre de la librairie graphique demandé.

d. Vecteur2D

La classe Vecteur2d représente en réalité, un point. Il s'agit d'un Vecteur car il part des coordonées 0,0 pour aller au coordonnées X,Y du vecteur.

Plusieurs opérations sont possibles sur un vecteur, comme le calcul du déterminant ou encore l'addition de vecteur.

e. Formes

Nous avons décidé de représenté une forme par une couleur, et un Vecteur2D de base.
Il s'agit d'une classe Virtual pure, qui fait hérité certaines méthode (elle est donc non instanciable).

Chacune de nos formes héritent de cette classe. Le groupe de forme, représentant plusieurs forme est aussi une forme, il était important de le définir comme une forme afin d'avoir des groupes de groupe de forme.

Dans ces groupes de formes, il n'y pas grand chose de spécial, seulement que certaines méthodes ne servent à rien, puisqu'il est interdit ou impossible d'arriver dans certains scénario (Exemple : On enverra jamais l'informations d'un groupe au serveur java.)

Les formes disponibles sont : Cercle, Polygone, Triangle, Segment et GroupeForme.

f. Méthode et fonctions générales à une forme.

Les fonctions de transformations :

- Translation (déplacement d'une forme)
- Rotation
- Homotétie (Zoom d'une forme)
- Aire

Il s'agit de fonction que chacune des autres formes doivent pouvoir faire.
Chacune de ces méthodes est réecrite par forme, en effet, la rotation d'un cercle ne change pas le cercle alors que celle d'un triangle est biien réelle).

La méthode toString, qui sert à obtenir la forme sous forme de chaine de caractère, afin d'être envoyé au serveur.

Il existe aussi la méthode draw, qui servent à envoyer a la classe client la forme à dessiner.

La fonction transform, qui effectue la transformation affine sur la forme afin d'être adapté à l'écran

Et les fonction qui permettent d'obtenir le plus grand x et y ainsi que le plus petit x et y. Elles servent à définir la taille de l'environnement monde.

Il se peut que dans la classe de groupeForme, certaines méthode ou fonction aient un contenu vide, en effet, certains cas ne sont pas possible.

g. Chargement / Sauvegarde de fichier (Avec COR Design Pattern)

-Tout d'abord nous allons commencer par la Sauvegarde d'un Forme. Dans notre cas à nous chaque Forme peut être sauvegardés par un visiteur qu'il s'appelle VisiteurDessine (qui hérite de la classe VisiteurAbstrait). VisiteurDessin possède un "vector de Forme*", où il est justement possible de stocker des Formes, ce que fait ici la methode "sauvegarder" dans VisiteurDessine.

Or ici nous n'enregistrons rien pour l'instant (par exemple dans un fichier), pour cela nous avons une autre methode qui s'appelle "enregistrer" toujours dans VisiteurDessine, qui va enregistrer les Formes en en chaîne de caractères d'une Forme dans un fichier que nous nommerons lors de l'appel de la methode. Après cela, dans ce fichier nous aurons les Formes qui seront bien sauvegardées en format "chaine de caractère".

- Puis pour la partie Chargement, pour récupérer les données qui sont inscrites dans ce fichier. Nous avons créé un Design Pattern Chaine Of Responsability, qui va permettre de vérifier la bonne récupération d' une Forme en lui donnant juste une chaîne de caractère.

Pour chaque Forme, nous avons donc créé, un Expert qui va “résoudre” une chaîne de caractère passé en argument, pour cela chaque ExpertForme à différents moyen de "retrouver" sa Forme, et si un ExpertForme, réussit à retrouver la Forme alors une Forme est créé et est retourné.

Ainsi pour le Chargement, nous allons récupérer les informations ligne par ligne dans le fichier et les passer à la classe Expert qui va nous retrouver les Formes.
Et donc nous aurons toutes les Formes que nous avons sauvegardées.

h. Dessin d’une forme (Avec Visitor Design Pattern)

Il s’agit d’envoyé la chaine de caractère (toString) de la forme au serveur.

Un problème s’est posé dès le début sur cette partie.

En effet, nous ne pouvons stocker dans un groupe une liste de chaque type de forme. C’est du code qui est dupliqué et qui donc est inpropre. Nous avons donc réalisé une liste de pointeur de forme (Forme était une classe mère des autres classes de forme).

La liste nous permet ainsi de récupérer directement les bonnes formes mais survient alors un problème, comment définir qu’il s’agit de la bonne classe.

Afin de comprendre la suite, une classe abstraire VisiteurAbstrait à été créer, elle comprend les méthodes pour chaque classe de forme qui existe.

Puis une classe VisiteurDessine héritant de VisiteurAbstrait à été créer. Celle ci contient les méthodes qui permettent de dire au client de dessiner cette forme.

Le problème est donc le suivant : comment définir s’il s’agit d’un groupe dans un groupe, ou une forme simple.

Pour rappel, il est impossible de faire :

- dessine(Forme *f)

Alors qu'il est possible de faire :

- f->dessine

Le programme possède alors de la façon suivante :

- Contexte : j'ai créer un groupe de Forme et un triangle, le triangle est ensuite placé dans le groupe.
- Je décide de dessiner le groupe j'éxecute donc la méthode draw du groupe en lui passant un Objet : Visiteur Dessine.
- La méthode draw, utilise l'objet VisiteurDessine et lui dit de visite(this) (this étant une instance de la classe dans la quelle nous somme, il n'y a alors pas de problème).
- VisiteurDessine, elle fait appel à la bonne méthode de déssin et envoie la forme au client.

Pour résumer, la méthode draw permet de nous retrouver dans le bon type de classe, et la classe visiteur nous permet de faire en sorte d'envoyer les bonnes informations.

Le Pattern Visiteur est donc important lorsque l'on peut savoir ce qu'on a entrée mais pas ce que nous aurons en sortie.

3. Serveur

Le serveur est

a. Serveur

Il s'agit d'un serveur java, en local, que l'on ouvre sur un port. Il respecte le pattern Singleton, afin d'éviter les erreurs, un seul serveur est lancé et uniquement un seul.

b. Choix d'une librairie Graphique (Avec COR Pattern)

Une chaine de responsabilité a été établie afin de lancer la création d'une fenetre graphique. Pour l'instant, une seule librairie graphique a été ajouter : AWT. L'ajout d'une autre fenêtre graphique est tout à fait possible.

Lorsque le serveur reçoit la première chaine de caractères, il l'envoie à travers cette chaine de résponsabilité afin de trouver quelle classe instancié. Il s'agit d'une simple comparaison de chaine de caractères.

Une fenêtre windowAwt est alors créer.

c. Dessin d'une forme (Avec COR et Visitor Design Pattern)

Une fois la librairie graphique choisit parmis celle existente, les prochaines chaine de caractères sont envoyés au serveur.

Celle-ci sont envoyé dans un COR, celui ci s'occupe de déterminer de quelle forme il s'agit.

Nous avons décidé dans un soucis de rassembler les classes segment, Triangles et Polygon dans le même expert, en effet, ils vont sortir la même chose et seront déssiné de la même manière.

Un ExpertDessin à aussi été rajouté, celui-ci sert à dessiner une fois que toutes les formes ont été envoyés.

En effet l'utilisation d'active rendering nous permet de mettre en tampon les formes envoyés et lorsque l'on décide, de les dessiner.

Lorsque l'on a trouvé de quelle forme il s'agit, le visiteur intervient. celui-ci sert à utilisé la bonne méthode en fonction de la forme qui est passer en paramètre de la fonction visit.

Cela permet encore une fois, de généraliser du code dans quelques lignes plutôt que de faire des copier coller.

L'interet du DP COR est d'obtenir une instance d'objet sans savoir ce que l'on a en entrée.

4. Conclusion

Il était très intéressant professionnellement parlant de réaliser un projet de cette taille avec pleins de spécificité.

En effet, le fait de réaliser ce projet en duo nous a permis de nous rendre compte des différente difficulté du travail en entreprise (à moindre mesure, mais par exemple, un équipié n'est pas forcément sur le même travail que nous et ne peut donc nous aider sur un instant T).

Il nous a aussi permis de nous rendre compte de la réalité en entreprise avec l'application sur un serveur avec un client, c'est ce que nous pourrons retrouvé dans la réalité et le fait de se confronter avec ce genre de problème durant notre période scolaire est une réalisation importante afin de mieux comprendre les rouages de cet univers.

L'une des parties les plus importantes du projet est bien sûre d'éviter la redondance de code. Partie important et non des moindres, il a fallu relever les défis de la conception et comprendre les enjeux sur chaque Design Pattern réalisé. (COR = entrée inconnu pour une instance de classe et Visitor : Entrée connu mais impossible de déterminer la bonne fonction a appelé lors d'une généralisation d'appel)

Robin : La collaboration entre collègues fut intéressente, j'ai du me forcer à certaines bonnes pratiques et supprimer toutes mes mauvaises pratiques, parfois avec du mal…

J'ai également pu échanger avec Samvel sur mon point de vue de programmation et en apprendre plus.

Malgrès mes difficultés en C++, j'ai fait de mon mieux et je suis content du résultat produit avec les petites options supplémentaires.

Samvel :

Ce projet était très intéressant pour moi, il m'a permis d'apprendre de nouvelle chose, notamment l'utilisation de Git que Robin m'a expliqué plus en profondeur car je ne l'avait jamais utilisé lors d'un vrai projet, le langage C++ que je ne connaissais pas assez et le travaille en équipe avec Robin.

L'utilisation de deux langages (Java et C++) pour un seul projet, était assez nouveau pour moi. J'ai aussi pu étoffer le langage C++ où j'avais quelques difficultés dessus. Le Java n'a pas été un gros souci de ce côté là car j'avais déjà fait quelque gros projet dessus.

Même avec toutes ces nouveautés pour moi, nous avons réussi à fournir un projet fonctionnel.