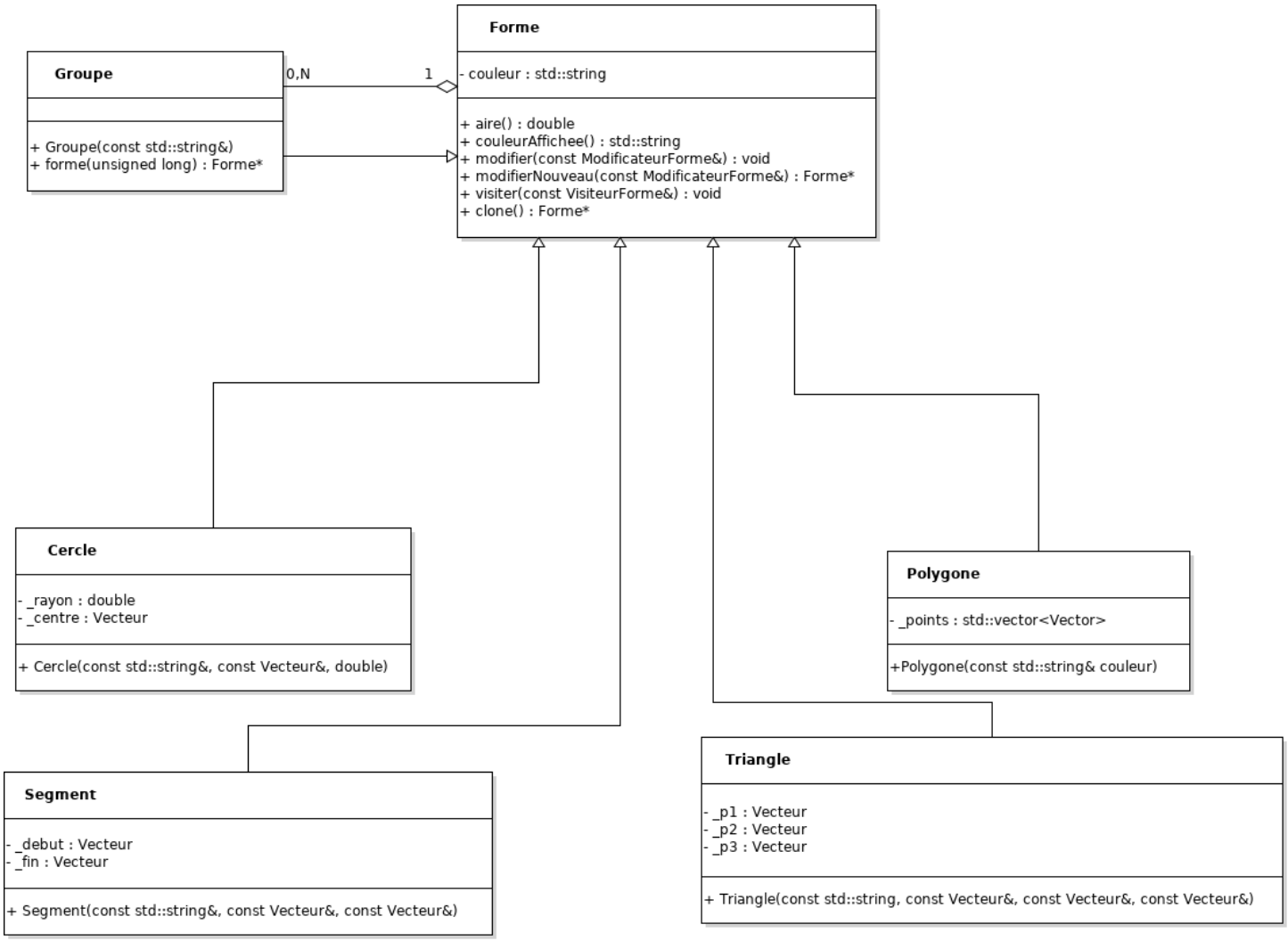
Rapport de Projet de Synthèse

par Roméo Florian et Webert Chris

Bonjour.
Contrairement au projet de Web, je n'ai pas écrit ce rapport sous l'emprise de l'alcool.

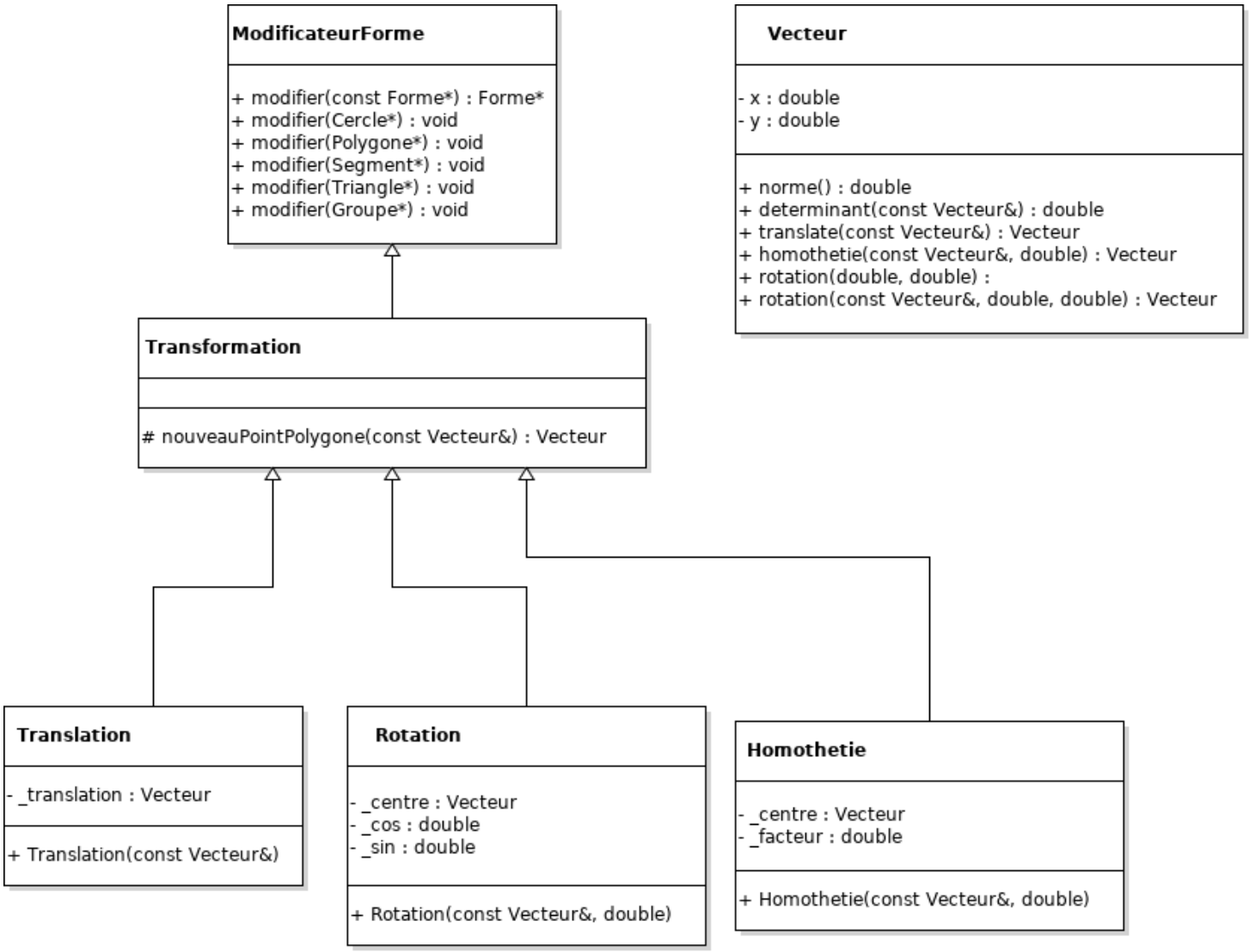
***EDIT** (28/11/2017): Maintenant si.*

Représentation des formes



Le triangle n'hérite pas de Polygone pour ne pas hériter des méthodes permettant d'ajouter ou de supprimer des points, se retrouvant avec un objet invalide.

Transformations



Chaque transformation est représentée par une classe dérivée de Transformation.

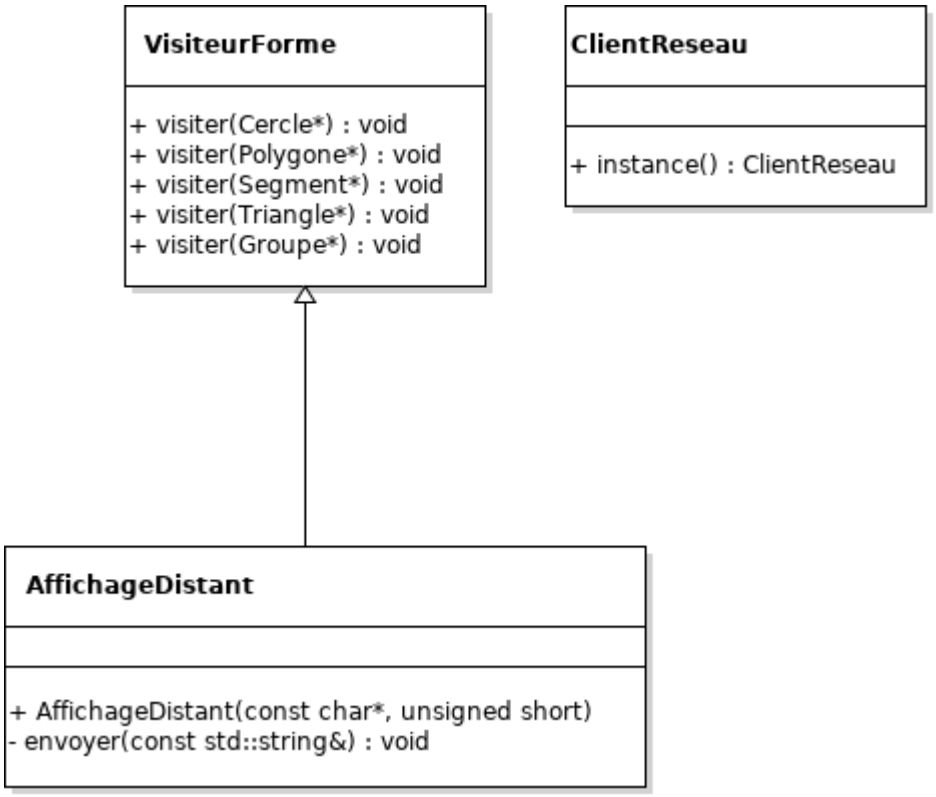
Les transformations peuvent être appliquées de deux façons : directement sur la forme ou alors sur une nouvelle forme.

Le design pattern Visitor a été utilisé ici.

Les formules des transformations sont contenues dans Vecteur (qui représente les points), la transformation est appliquée à chaque point de chaque forme.

Communication entre les deux parties

Transfert depuis le client C++



La classe ClientReseau est un singleton utilisé pour initialiser la bibliothèque Winsock sur Windows.

La classe AffichageDistant est un visiteur de formes, qui va envoyer au serveur donné les formes sous forme de chaînes de caractère.

Protocole

La communication entre le client C++ et le serveur Java se fait avec un protocole basé sur les chaînes de caractères.

Le protocole envoie les formes sous forme de blocs, en commençant par le nom de la forme, et en finissant par « FIN ».

```
SEGMENT
couleur=RED
debut=0;0
fin=100;100
FIN
```

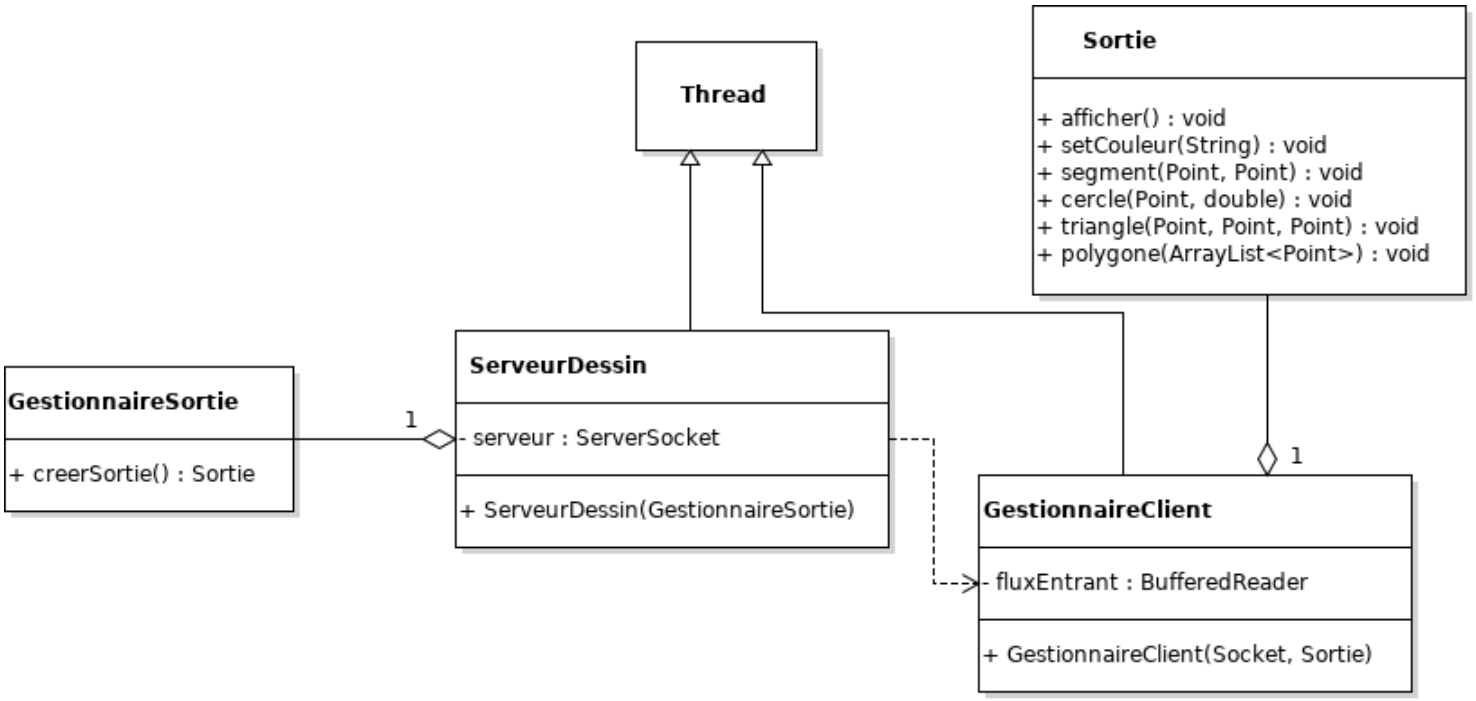
Cette forme à été choisie car elle présente plusieurs avantages :

- On reçoit les variables une par une, sans devoir utiliser split.
- Le protocole est extensible, on peut écrire une nouvelle version de l'application avec de nouvelles variables (par exemple la couleur de la bordure de la forme), et garder une compatibilité avec la version précédente du serveur Java

Les blocs se suivent, le bloc suivant se trouve sur une nouvelle ligne.

Forme	Variables
Cercle	• Couleur • Centre • Rayon
Polygone	• Couleur • Point (présent autant de fois qu'il y a de points)
Segment	• Couleur • Début • Fin
Triangle	• Couleur • Point 1 • Point 2 • Point 3
Groupe	Le groupe n'est pas envoyé, chaque forme est envoyée individuellement avec la couleur du groupe.

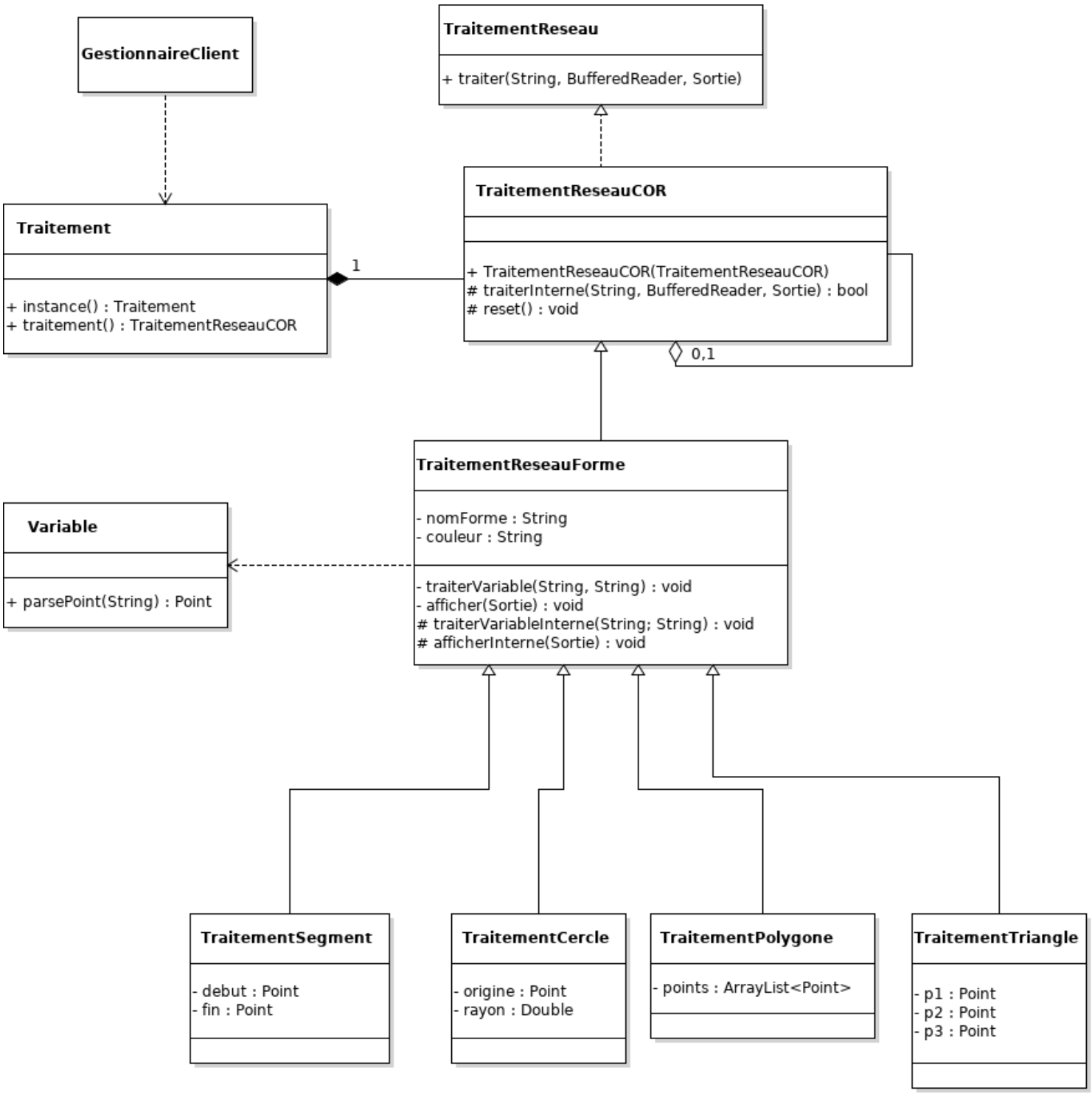
Réception depuis le serveur Java



La classe ServeurDessin s'occupe d'écouter sur un port spécifique, d'accepter les nouveaux clients, qui seront gérés par GestionnaireClient, et de leur attribuer une sortie.

La classe GestionnaireClient est lancée dans un nouveau thread pour ne pas bloquer les autres clients. La classe ServeurDessin l'est aussi pour ne pas bloquer le thread JavaFX.

Traitement des données reçues



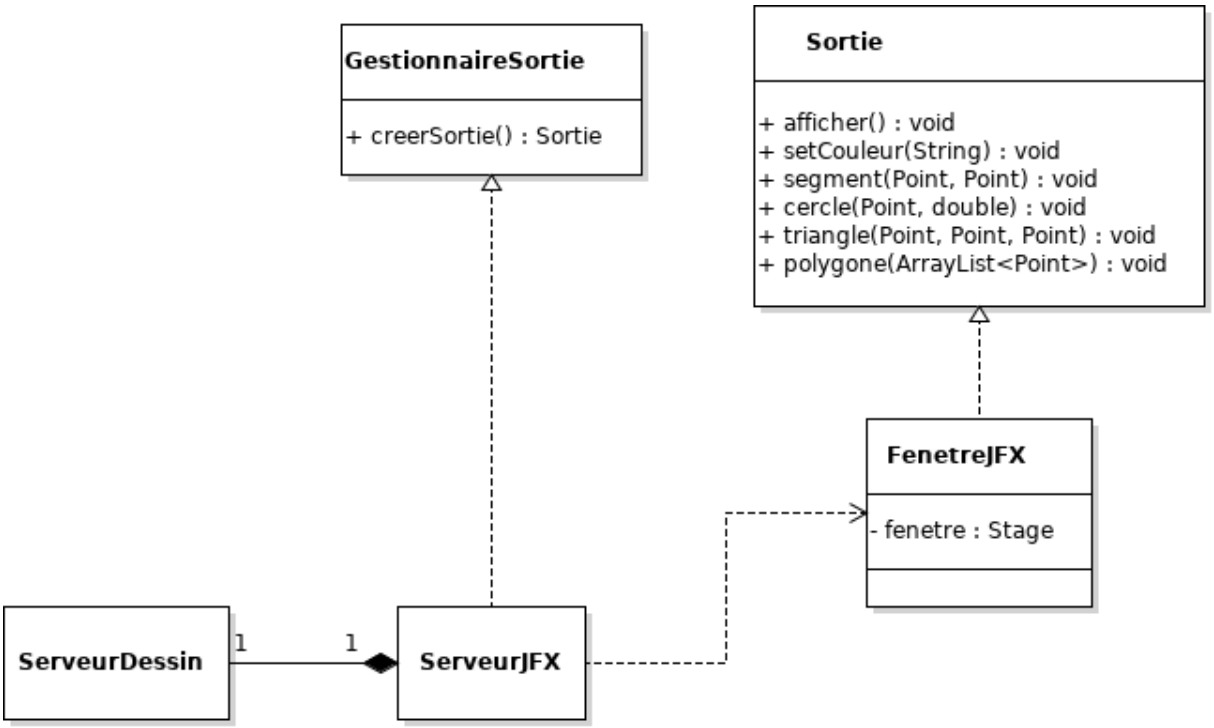
Le serveur écrit en Java traite les données reçues à l'aide du design pattern Chain of Responsibility.

La classe Traitement est un singleton créant et permettant l'accès à la chaîne de traitement complète.

La chaîne est gérée par TraitementReseauCOR, qui décide si la classe peut traiter les données ou s'il faut les envoyer au prochain expert.

TraitementReseauForme s'occupe de traiter les variables reçues sous la forme nom=valeur. La classe dérivée s'occupera de traiter la valeur reçue comme chaîne de caractère, grâce aux méthodes statiques de la classe Variable.

Affichage



L'affichage se fait sur le serveur Java, qui utilise JavaFX.

Le programme lance directement le thread JavaFX, et le serveur écoute dans un autre port.

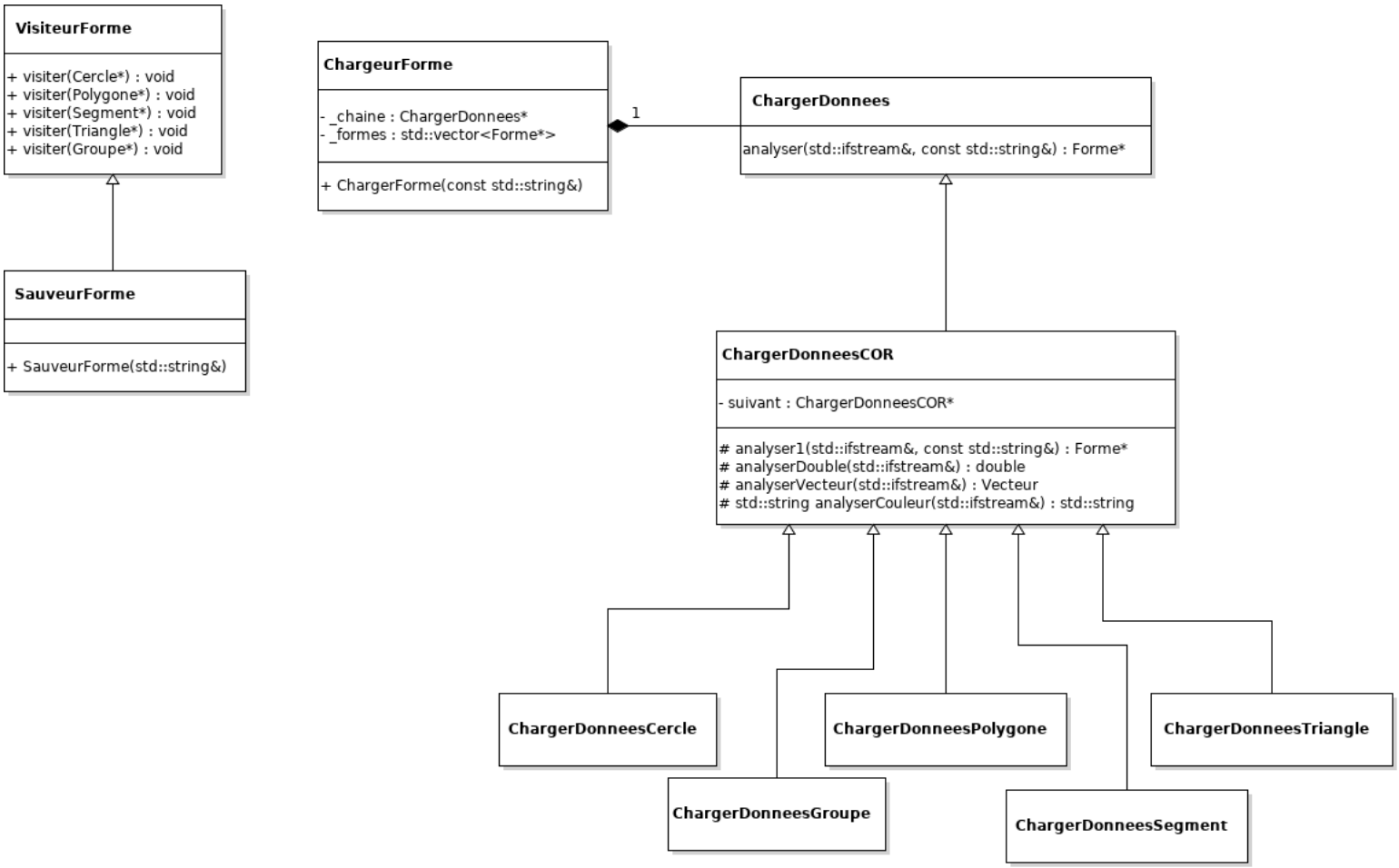
On utilise Platform.runLater() afin de pouvoir créer et afficher de nouvelles fenêtres depuis le thread qui gère le client.

La suppression implicite du thread JavaFX à du être désactivée afin de pouvoir garder le serveur fonctionnel indéfiniment.

Tests

Nous avons écrit des tests unitaires pour éviter les régressions. On utilise la bibliothèque catch, et on teste toutes les opérations de création et modification.

Sauvegarde/chargement d'un fichier



La sauvegarde se fait de la même façon que la communication entre le client et le serveur.

Le code du transfert n'a pas été repris, car la sauvegarde doit supporter les groupes et les couleurs attribuées à la construction de la forme (qui prend la couleur du groupe à l'affichage) :

```
Cercle
{
couleur:red
origine:(357.000000,230.000000)
rayon:20.000000
}
```

Le design pattern Visitor a été utilisé pour la sauvegarde et Chain of Responsibility pour le chargement.

La classe chargeurforme crée la chaine de responsabilité à la construction et charge le fichier passé en paramètre.

Le flux du fichier est passé à la chaîne d'experts qui analysera le nom de la forme et la traitera si l'expert la reconnaît.

Des fonctions ont été crée afin de repérer et lire des double, des Vecteur ou des Couleur. Le chargeur traite les formes les unes après les autres jusqu'à la fin du fichier.

Les formes chargées sont passées dans un membre privé vector<Forme*> de la classe chargeurforme. Elles peuvent maintenant être consultées, modifiées ou envoyées au serveur Java.