

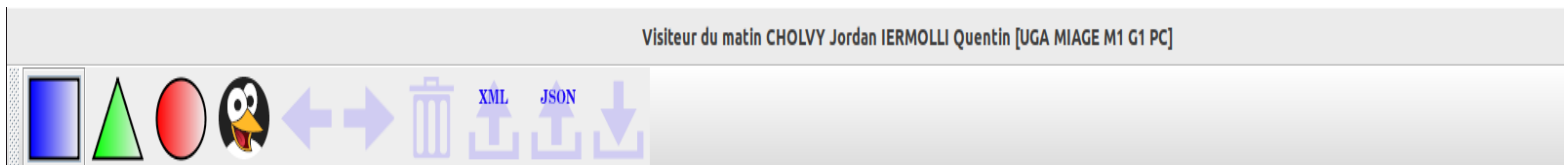
## SOMMAIRE :

- 1) Les figures (patron visiteur & factory)
- 2) Commande Avant Arrière (Patron Commande)
- 3) Déplacement
- 4) Export Import (patron visiteur)
- 5) Le groupage (patron composite)
- 6) Les Testes (Junit)
- 7) La Qualité de code (Sonar)

*Ancienne Interface*



*Nouvelle Interface*



## 1) Les figures (patron visiteur & factory)

### A) Description de l'évolution :

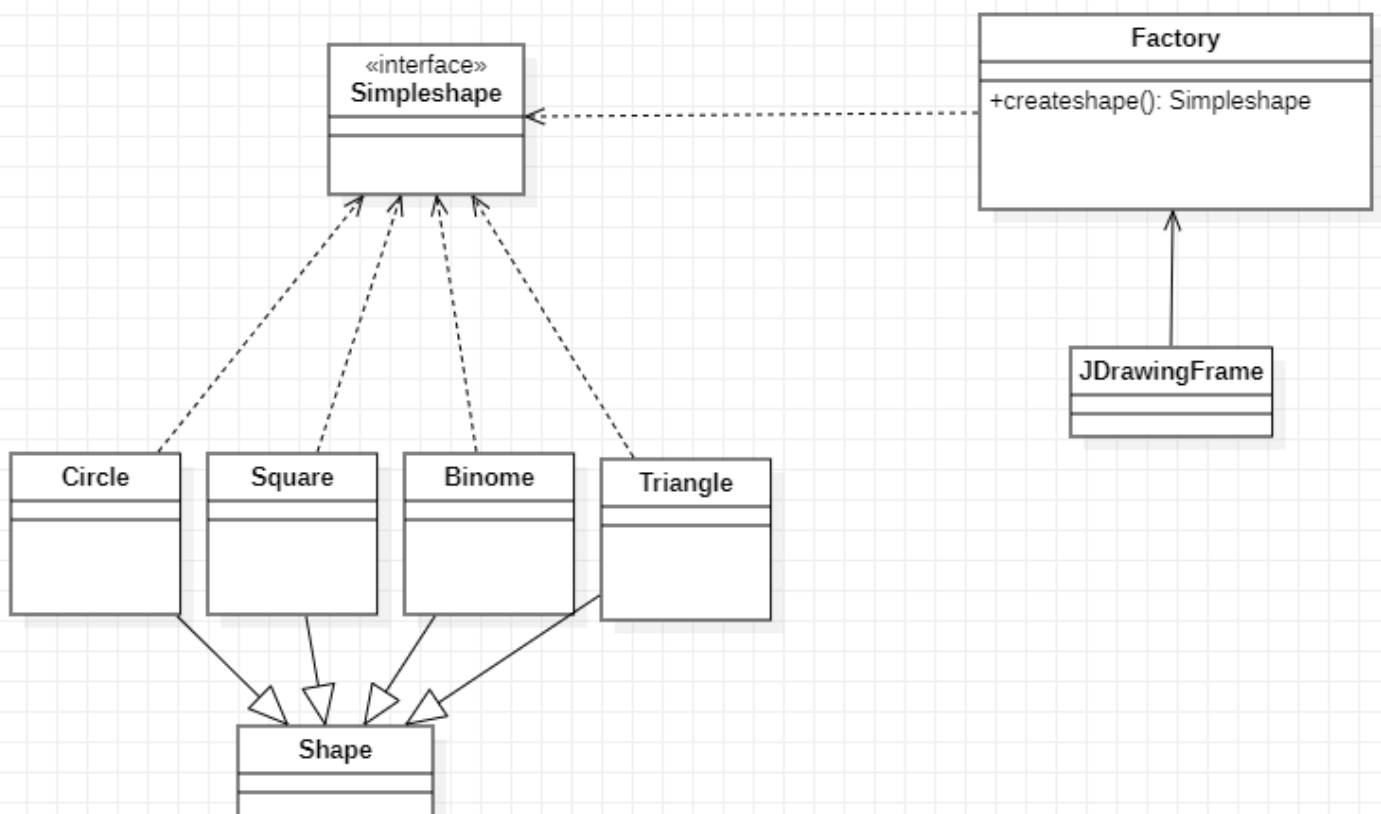
Depuis la dernière itération, nous avons ajouté le triangle et une image personnelle représentant Tux.



### B) Diagramme de classe :

Afin de simplifier le code et éviter les doublons, nous avons rajouté une classe Shape contenant toutes les fonctions communes aux figures.

Diagramme de classe patron simple factory :



C)Diagramme de séquence :

Exemple de dessins d'un carré

Diagramme de séquence :

Ici, on veut créer un cercle c de centre (1;2).

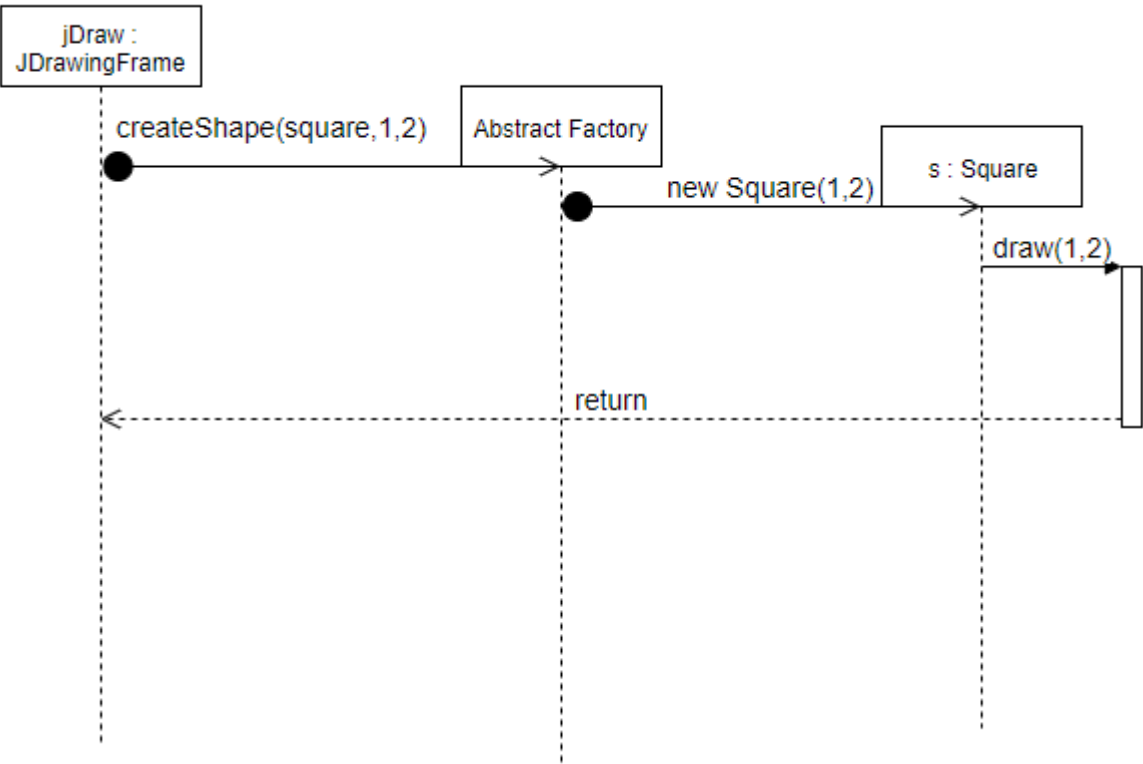



Image résultante :



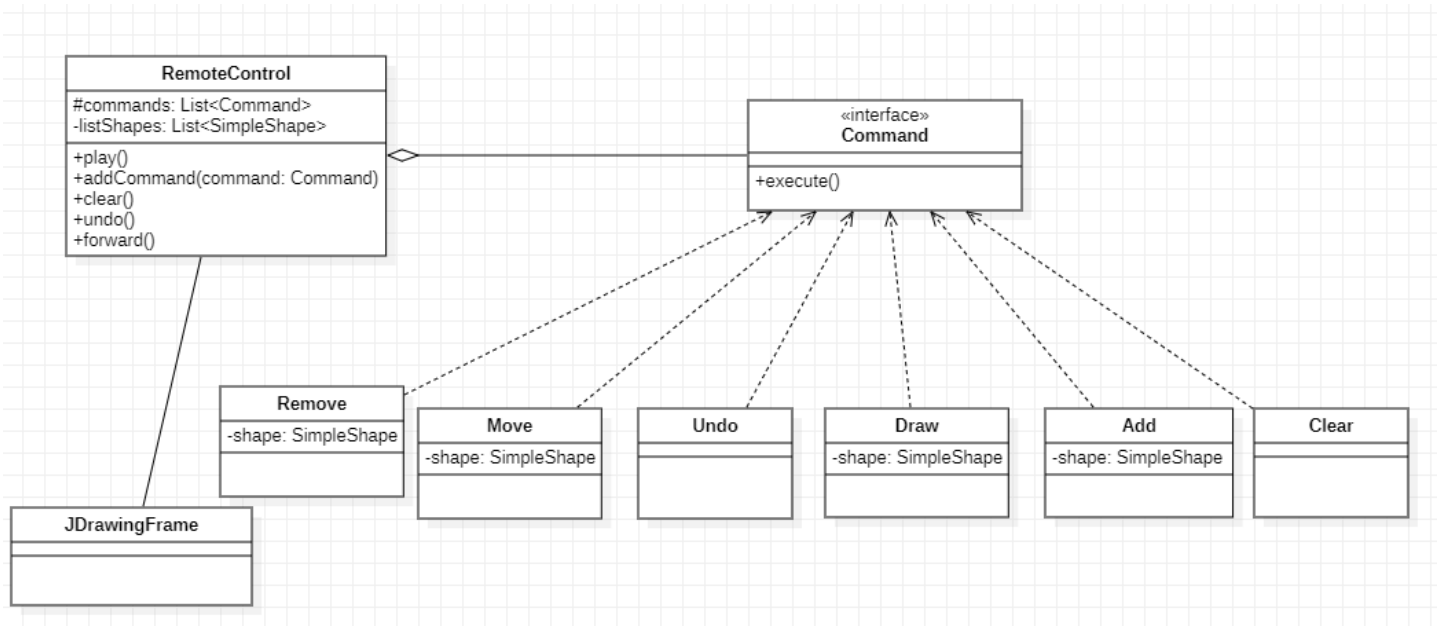
## 2) Commande Avant Arrière (Patron Commande)

### A) Description de l'évolution :

 Afin de pouvoir revenir en arrière, nous avons implémenté le patron commande. Chaque action (draw, add [ajout d'une figure à un groupe] remove, move, clear) s'exécute l'une après l'autre pour former le dessin. Via les flèches de l'interface, on peut annuler (undo) un changement.

### B) Diagramme de classe :

Diagramme de classe patron commande :



### C) Diagramme de séquence :

Diagramme de séquence :

Dans le diagramme de séquence ci-dessous, nous avons ajouté puis exécuté la commande adaptée pour dessiner un carré de coordonnées centrales (1;2).

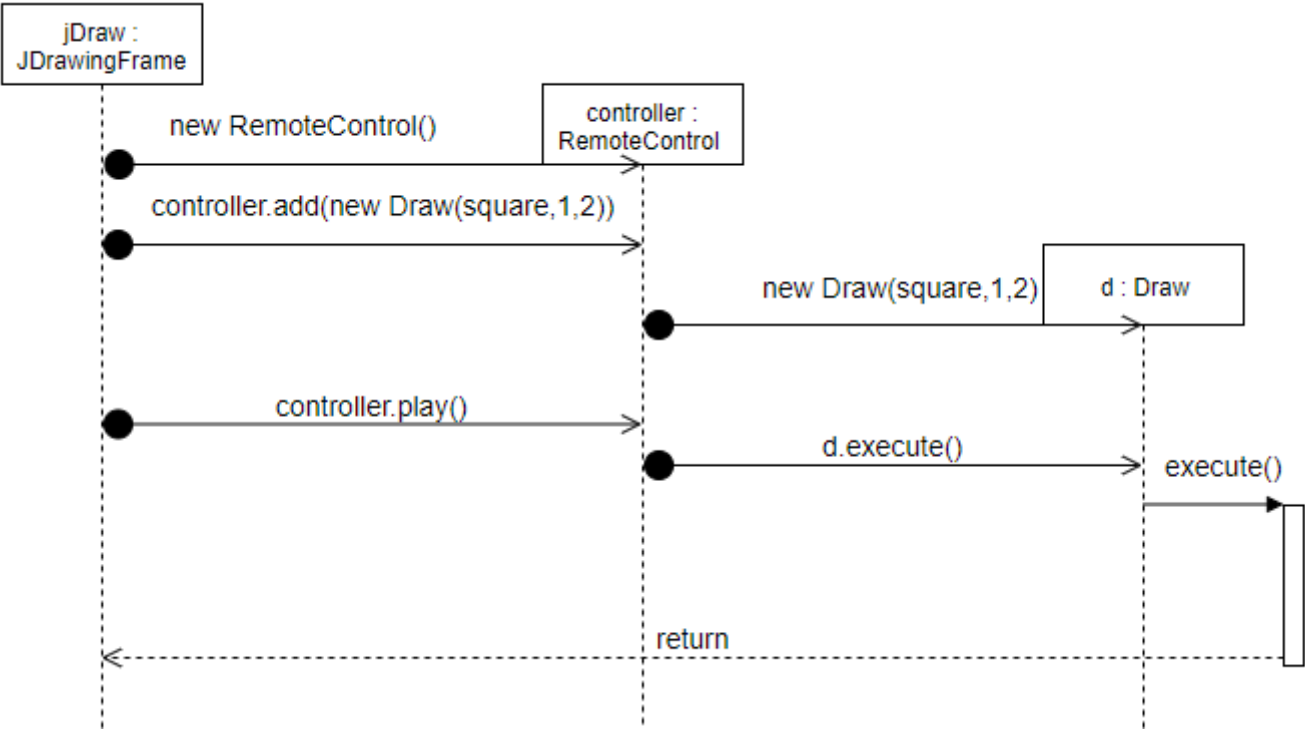
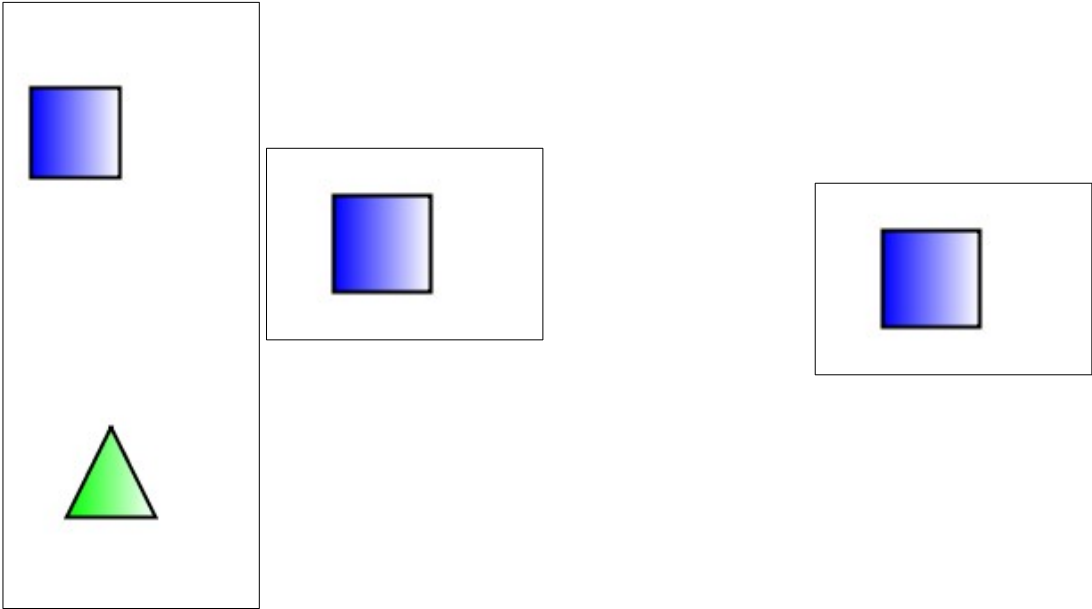


Image résultante :



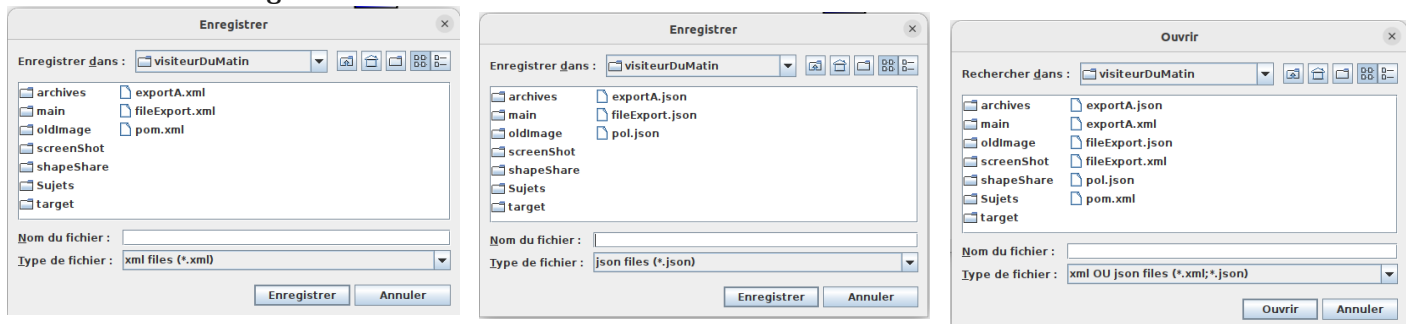
## 4) Export Import (patron visiteur)

### A) Description de l'évolution :



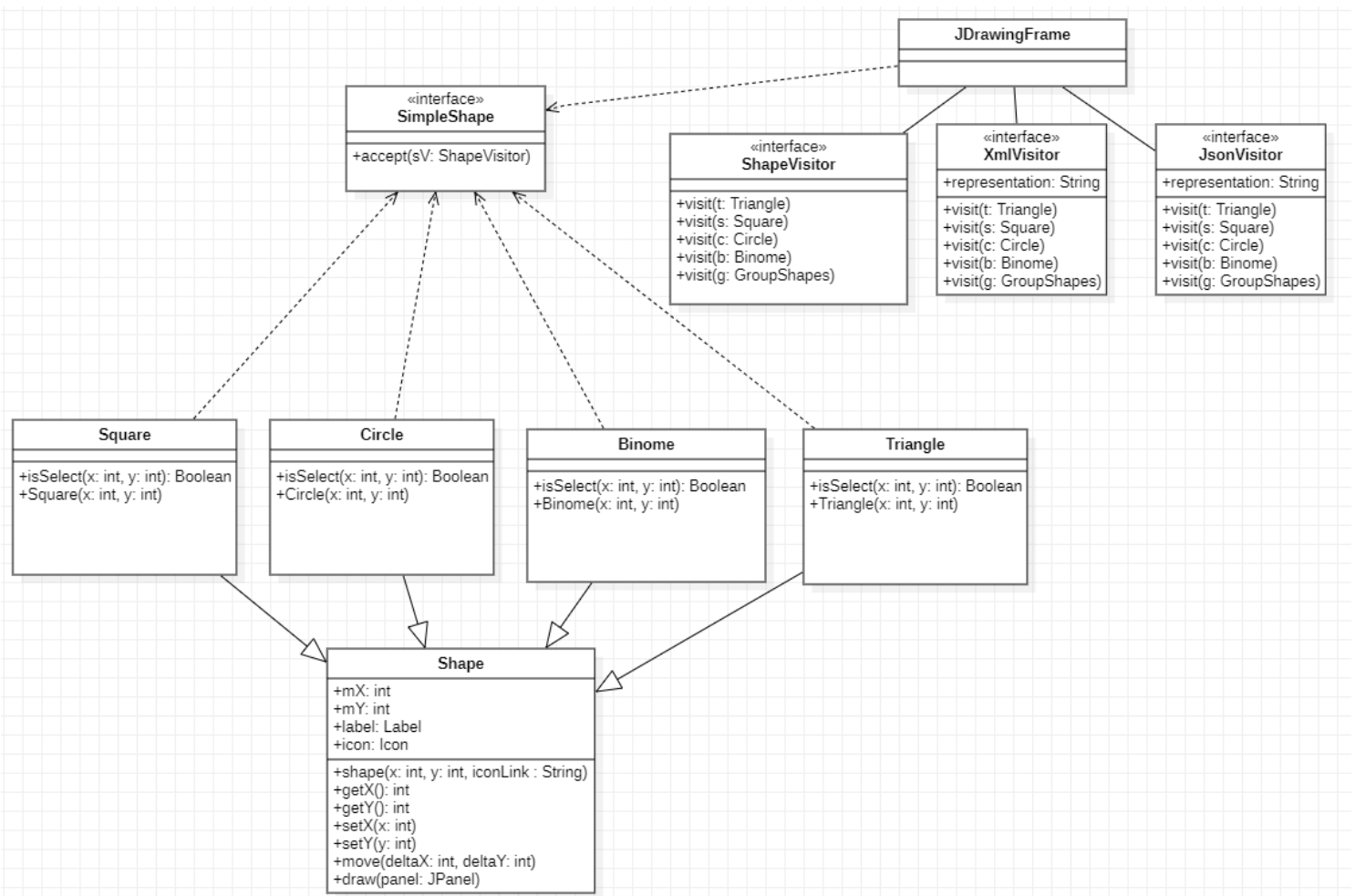
Afin d'importer et exporter les dessins plus facilement, nous avons ajouté un nœud groupe dans les fichiers xml et json. Notre application utilise maintenant les parsers XML JSON pour transformer un dessin en fichier xml, json et inversement.

Dans l'optique d'améliorer l'utilisation du logiciel, nous avons ajouté un système de pop-up pour faciliter la sélection et l'enregistrement des fichiers.



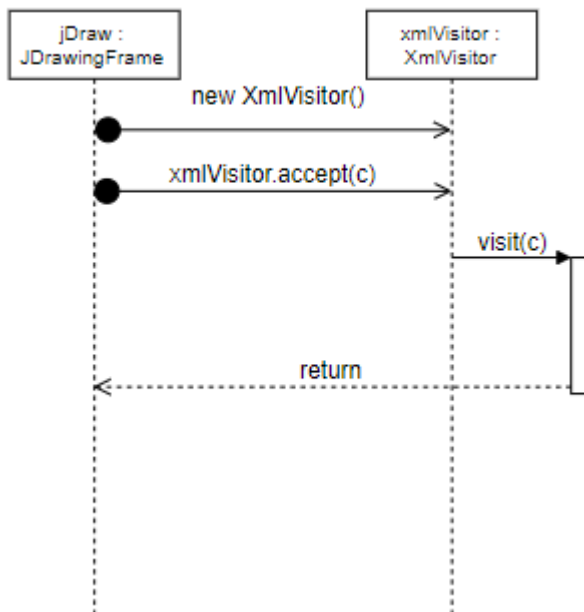
### B) Diagramme de classe :

Diagramme de classe patron visiteur :



### C)Diagramme de séquence :

*Diagramme de séquence :*



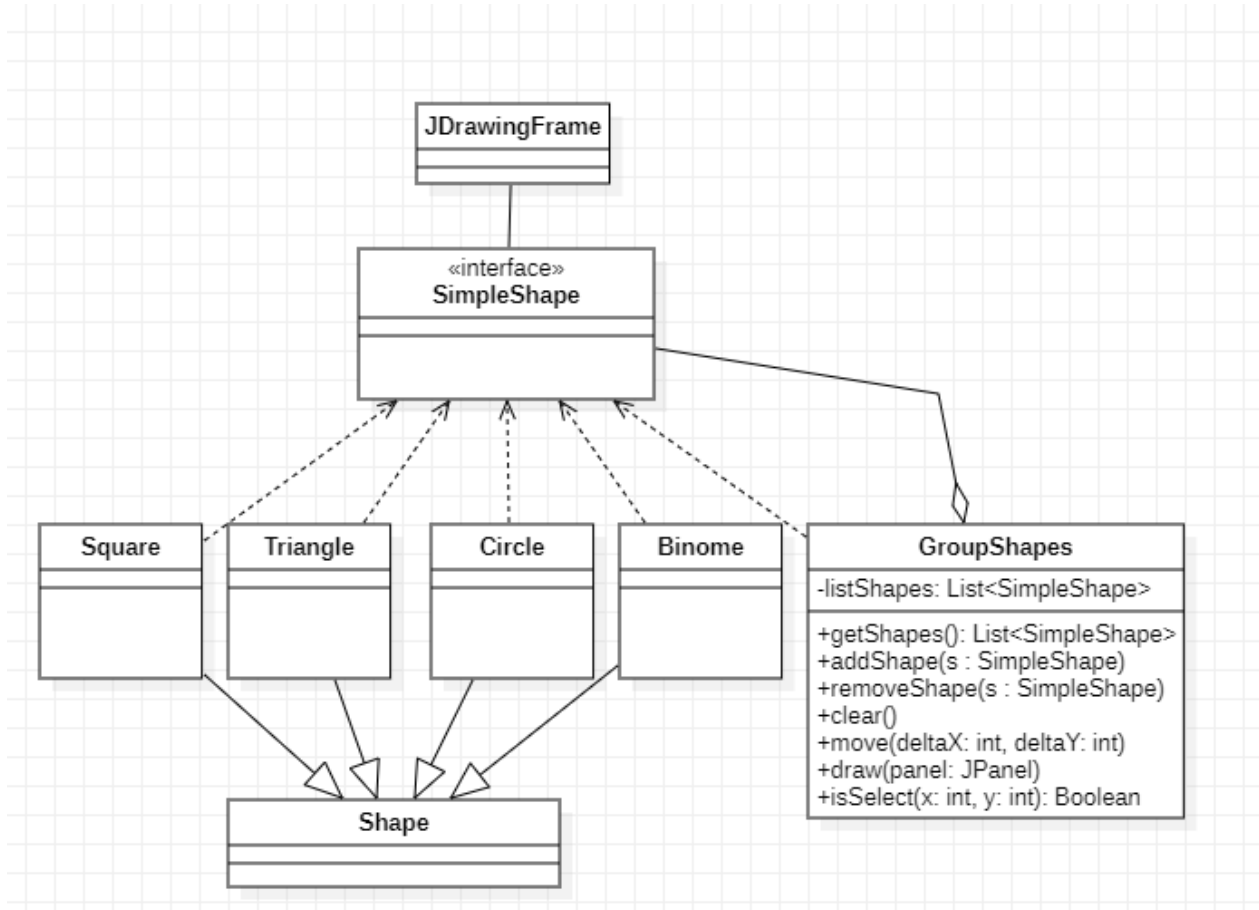
*Image résultante :*

## 5) Le groupage (patron composite)

A) Description de l'évolution :

B) Diagramme de classe :

Diagramme de classe patron composite :



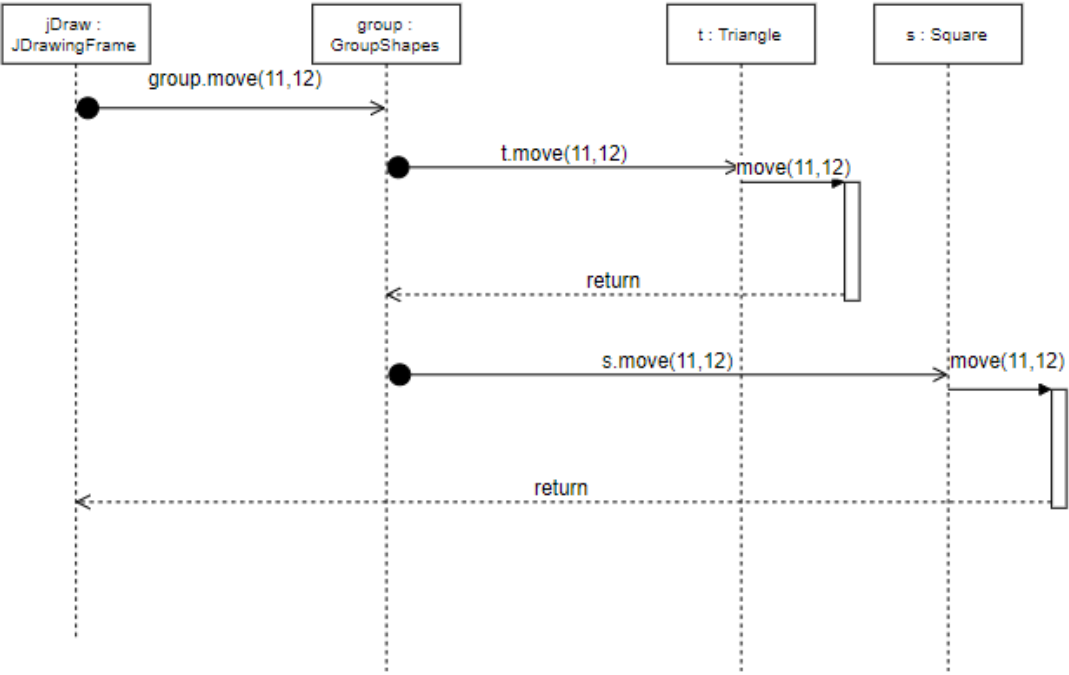
C) Diagramme de séquence :

Diagramme de séquence :

On suppose qu'on a un groupe avec des figures déjà créées : 1 carré appelé *s* et 1 triangle appelé *t*. Ici, on demande à déplacer d'une distance (11;12) le groupe créé.



Image résultante :

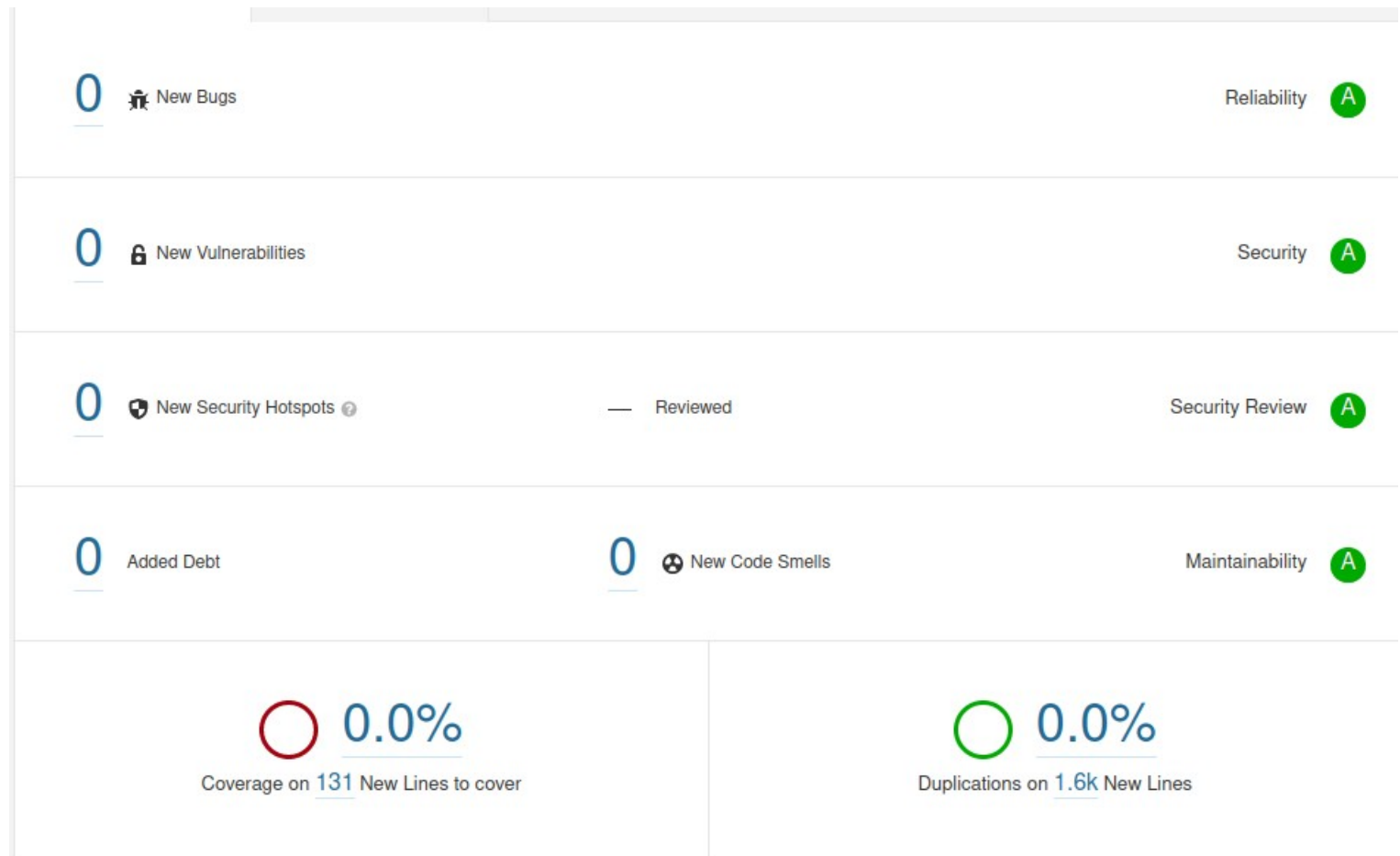


## 6) Les Tests (JUnit)

Element ^	Class, %	Method, %	Line, %
▼ edu	51% (18/35)	55% (89/159)	43% (269/614)
▼ uga	51% (18/35)	55% (89/159)	43% (269/614)
▼ miage	51% (18/35)	55% (89/159)	43% (269/614)
▼ m1	51% (18/35)	55% (89/159)	43% (269/614)
▼ polygones	51% (18/35)	55% (89/159)	43% (269/614)
▼ qui	51% (18/35)	55% (89/159)	43% (269/614)
▼ commands	66% (4/6)	60% (12/20)	66% (51/77)
Add	100% (1/1)	100% (2/2)	100% (12/12)
Clear	0% (0/1)	0% (0/2)	0% (0/3)
Command	100% (0/0)	100% (0/0)	100% (0/0)
Draw	100% (1/1)	100% (2/2)	100% (11/11)
Move	0% (0/1)	0% (0/2)	0% (0/5)
RemoteControl	100% (1/1)	60% (6/10)	48% (17/35)
Remove	100% (1/1)	100% (2/2)	100% (11/11)
▼ factory	50% (1/2)	33% (1/3)	33% (9/27)
Factory	50% (1/2)	33% (1/3)	33% (9/27)
▼ file	80% (4/5)	62% (15/24)	55% (89/160)
Export	100% (1/1)	50% (2/4)	50% (10/20)
FileUtils	100% (1/1)	45% (5/11)	17% (9/52)
Import	0% (0/1)	0% (0/1)	0% (0/14)
JsonFile	100% (1/1)	100% (3/3)	100% (25/25)
XmlFile	100% (1/1)	100% (5/5)	91% (45/49)
▼ graphique	7% (1/13)	2% (1/38)	2% (5/206)
GUIHelper	0% (0/2)	0% (0/2)	0% (0/6)
JDrawingFrame	0% (0/10)	0% (0/35)	0% (0/195)
shapeInteraction	100% (1/1)	100% (1/1)	100% (5/5)
▼ shapes	100% (6/6)	76% (42/55)	73% (79/107)
Binome	100% (1/1)	66% (2/3)	66% (2/3)
Circle	100% (1/1)	66% (2/3)	66% (2/3)
GroupeShape	100% (1/1)	68% (13/19)	62% (23/37)
Shape	100% (1/1)	76% (10/13)	67% (21/31)
SimpleShape	100% (0/0)	100% (0/0)	100% (0/0)
Square	100% (1/1)	75% (3/4)	87% (7/8)
Triangle	100% (1/1)	92% (12/13)	96% (24/25)
> visitor	100% (2/2)	100% (18/18)	100% (36/36)
App	0% (0/1)	0% (0/1)	0% (0/1)

Nous avons implémenté 40 tests qui couvre 51 % de notre code. Nous nous sommes concentrés sur le test des fonctionnalités critiques de notre programme (commande, groupage, export, import). Par manque de temps, nous n'avons pas pu implémenter les tests pour la partie graphique.

## 7) La Qualité de code (Sonar mg2\_6)



Nous avons mis l'accent sur la qualité de notre code. Pour cela nous l'avons complètement reformaté pour que chaque classe aie une fonction unique, clairement explicité par son nom. Nous avons suivi et corrigé toutes les alertes de sonar pour réduire notre dette à 0.

*Ps : Notre taux de couverture est de 0 % car suite à la subdivision de notre code en module pour réaliser la dernière implémentation, sonar ne retrouve plus le dossier de test.*