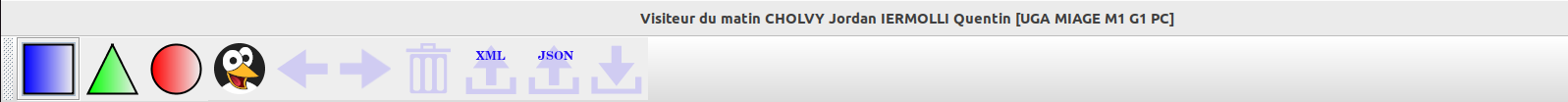
**SOMMAIRE :**

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

*Ancienne Interface*

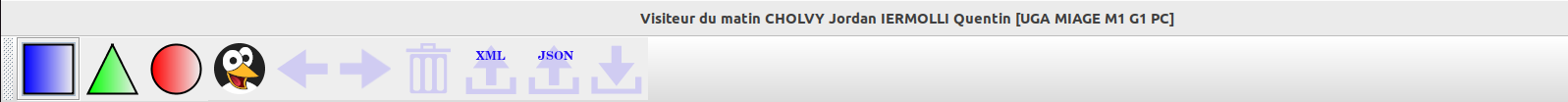


*Nouvelle Interface*

***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 visiteur :*

**C)Diagramme de séquence :**

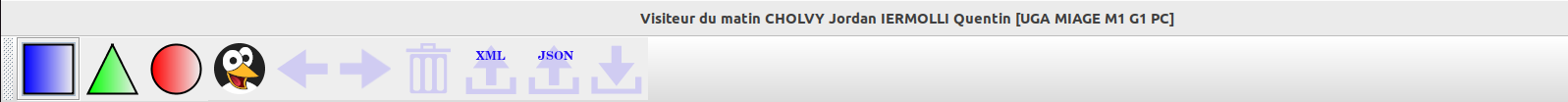
Exemple de dessins d’un carré

*Diagramme de séquence :*

*Image résu**ltante :*

***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 dessins. Via les flèches de l’interface, on peut annuler (undo) un changement.

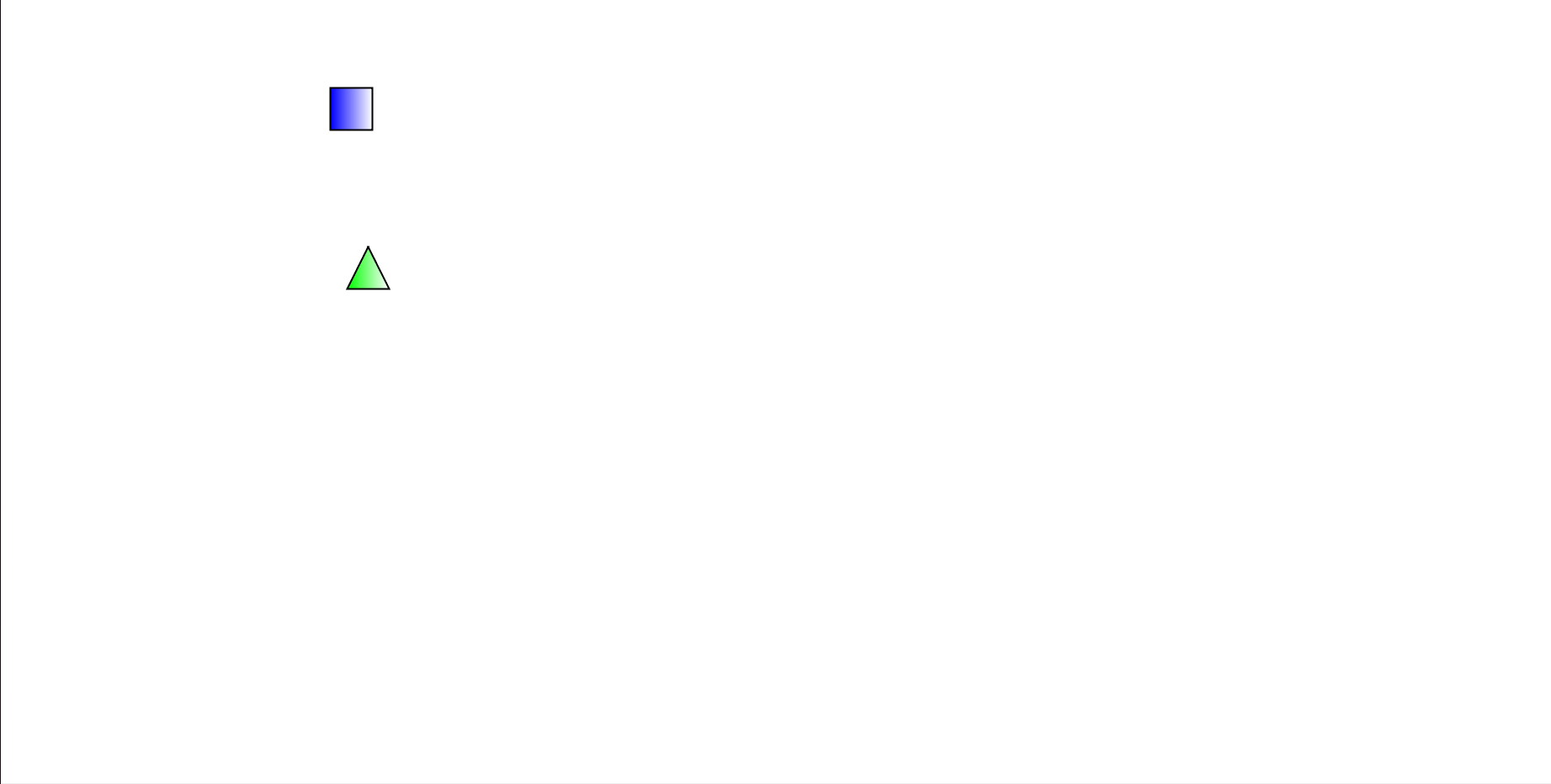
**B)Diagramme de classe :**

*Diagramme de classe patron visiteur :*

**C)Diagramme de séquence :**

*Diagramme de séquence :*

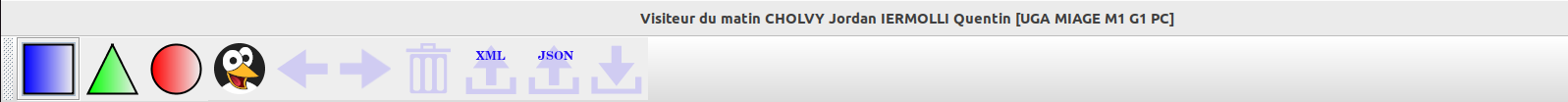
*Image résultante :*

**

**

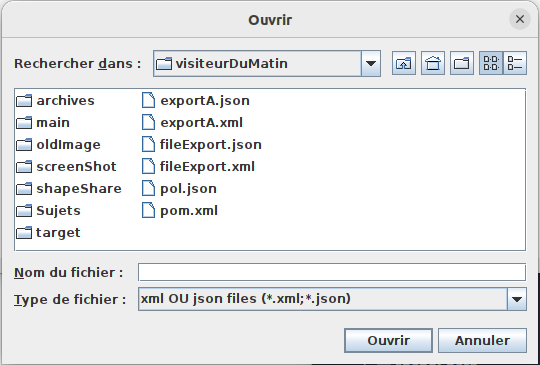
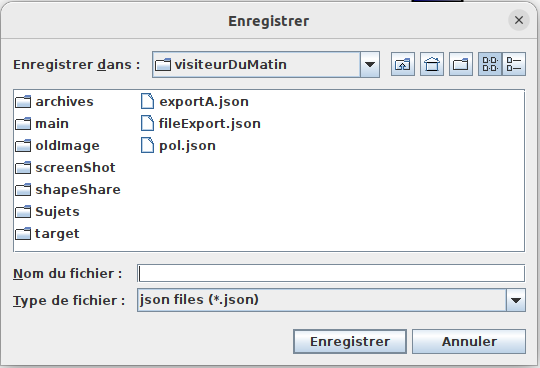
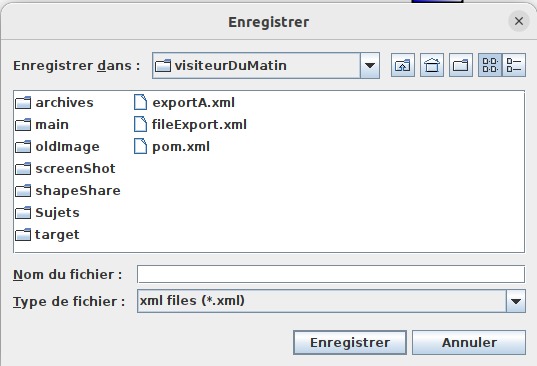
***4)Export Import (patron visiteur)***

**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 parseur 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 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 :**

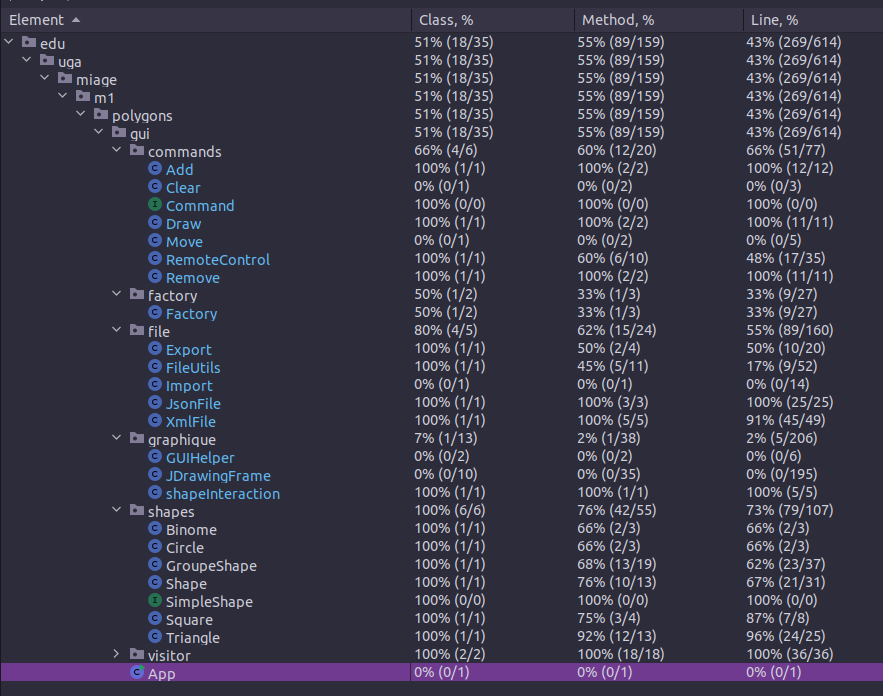
*Diagramme de classe patron visiteur :*

**C)Diagramme de séquence :**

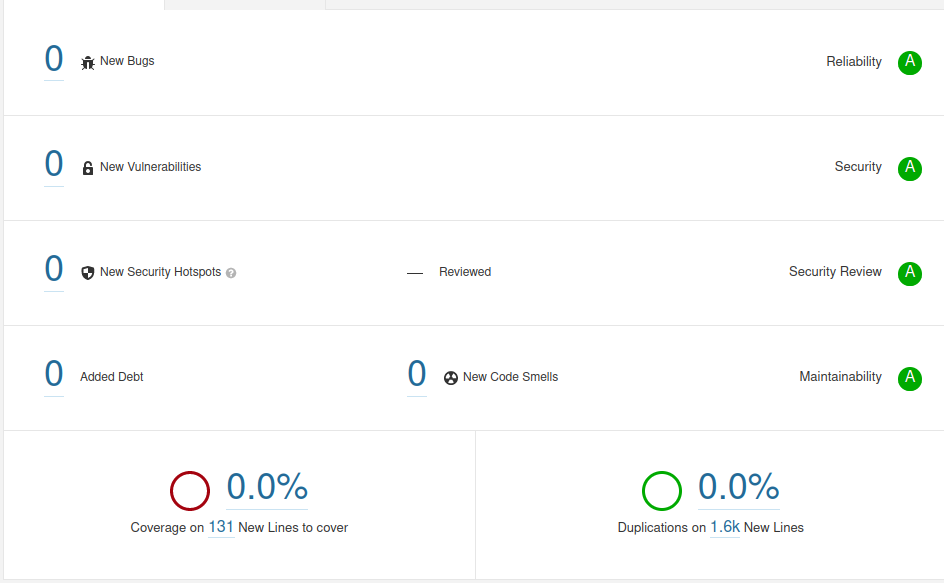
*Diagramme de séquence :*

*Image résultante :*

***6)Les Testes (Junit)***

Nous avons implémenté 40 testes qui couvre 51 % de notre code. Nous nous sommes concentré sur le teste des fonctionnalité critique de notre programme (commande, groupage, export, import). Par manque de temps, nous avons pas pues implémenter les teste 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 reformater pour que chaque classe est une fonction unique, clairement explicité par son nom. Nous avons suivis et corrigé toutes les alertes de sonar pour réduire notre dette à 0

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