Dessine moi un mouton

Création d'une librairie java

Compte rendu

Eline Abat - Marius Ballot

Introduction

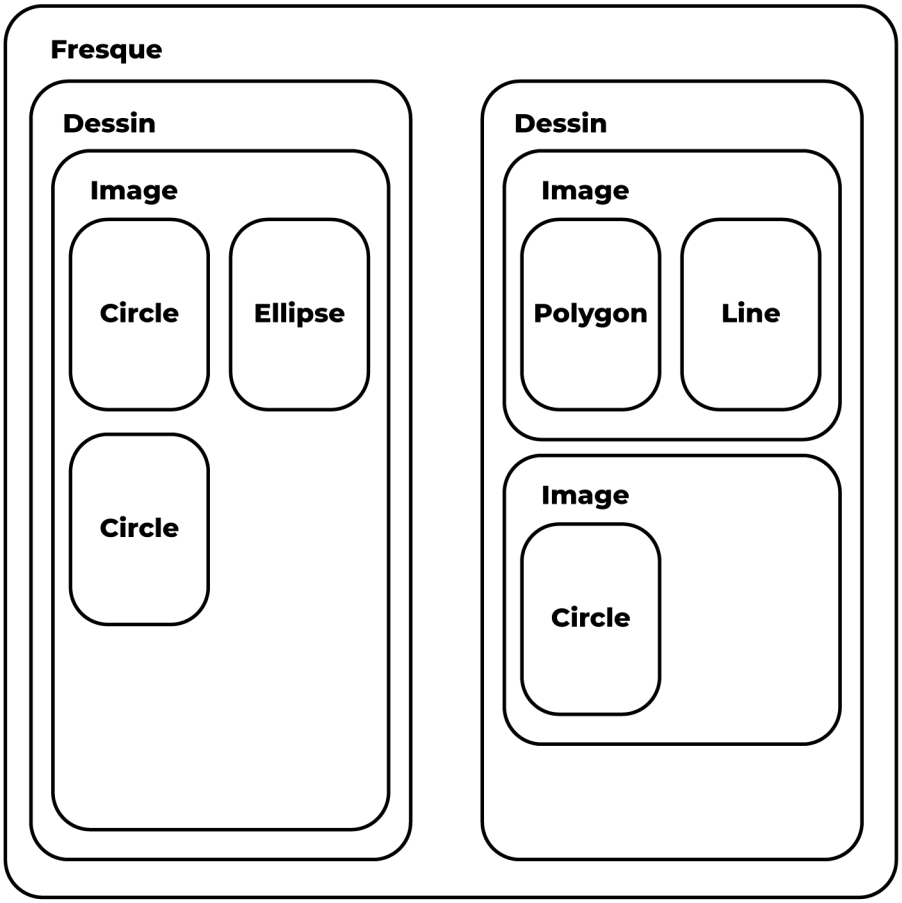
En tant que conclusion d'un module d'apprentissage du langage java, nous avons dû réaliser la création d'une micro librairie permettant de créer une fresque graphique abstraite, l'output de étant seulement autorisé en console.

Une fresque est composée de dessins, eux même composés d'images, eux même composé de formes géométriques variées : Cercles, ellipses, lignes et polygones.

Ces différentes entités, sauf la fresque, son amovibles. Nous pouvons leur appliquer des transformations de translation, modification d'échelle ou homotétie et rotation. Ces transformations sont également appelées PSR (=Position, Scale, Rotation) en CG (Computer Graphics). À cela s'ajoute la possibilité d'appliquer une symétrie par point et par axe.

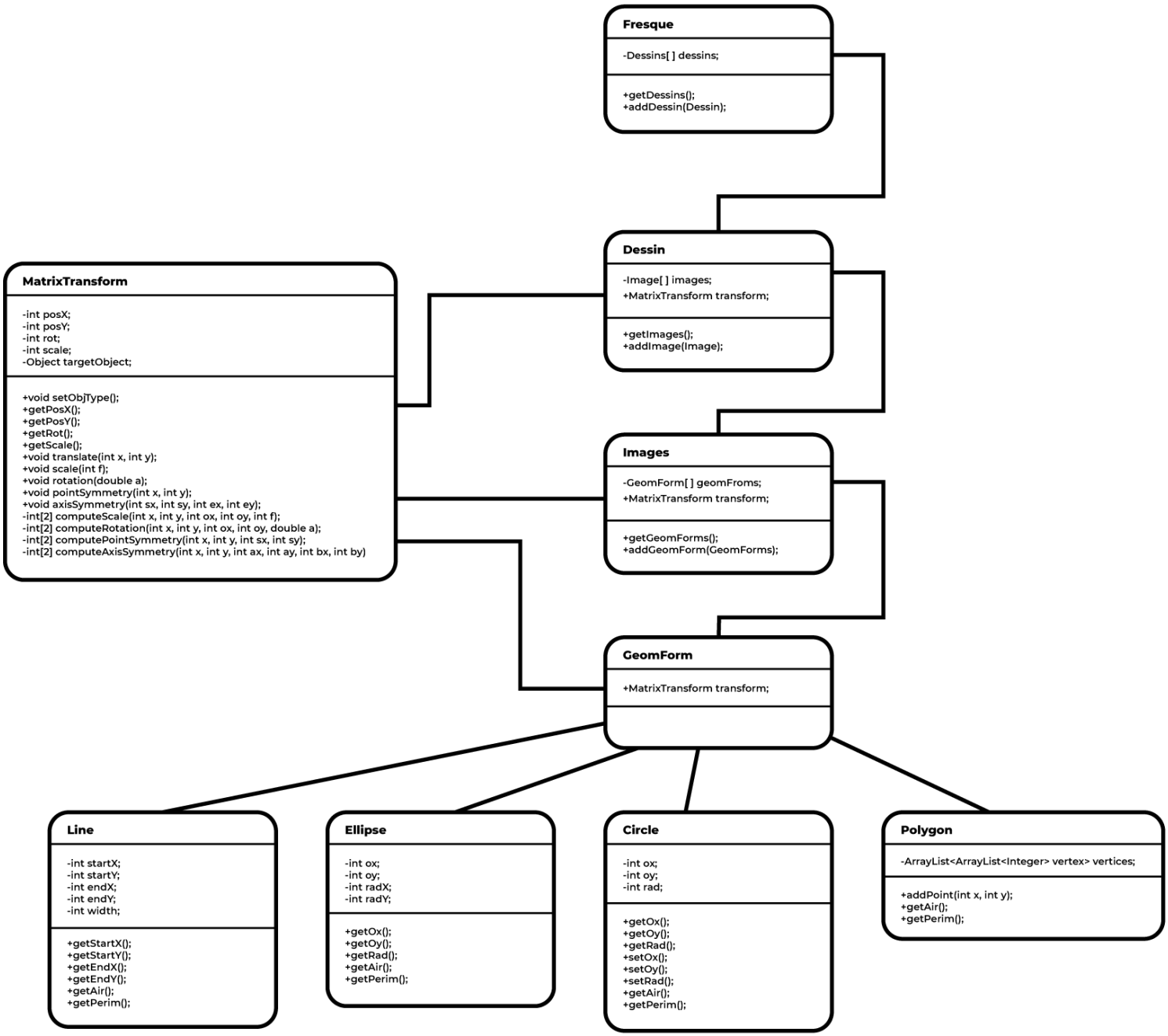
Plusieurs dessins, images et formes géométriques peuvent être ajouté a son contenant parent mais aucune ne peut avoir de doublons dans son contenant.

Schéma d'architecture fonctionnelle



Représentation d'un exemple d'utilisation de la librairie

Diagramme de classe



Code Break Down

Gestion de parenté

Pour gérer nos relations entre parents et enfants, fresque, images etc, nous avons simplement créer pour chaque contenant une ArrayList vide à laquelle nous avons spécifié les entrées autorisé, dessin pour fresque, images pour dessin et forme géométrique pour dessin.
C'est à l'insertion des geomForm dans le parent que nous avons notre routine de comparaison des airs.

Gestion de doublons

Pour cela nous avons du nous intéressé a la notion identité/égalité. Nous avons donc généré nos hascode et equals directement depuis notre ide et ainsi nous permettre de gérer ce qui est un doublon ou non par un algo.

```
void addImage(Image obj) {
        Boolean exists = false;

        for (int i = 0; i < images.size(); i++) {
            exists = obj.equals(images.get(i));
        }

        if (exists == true) {
            System.out.println(obj + "This image already exists");
            return;
        }

        images.add(obj);
    }
```

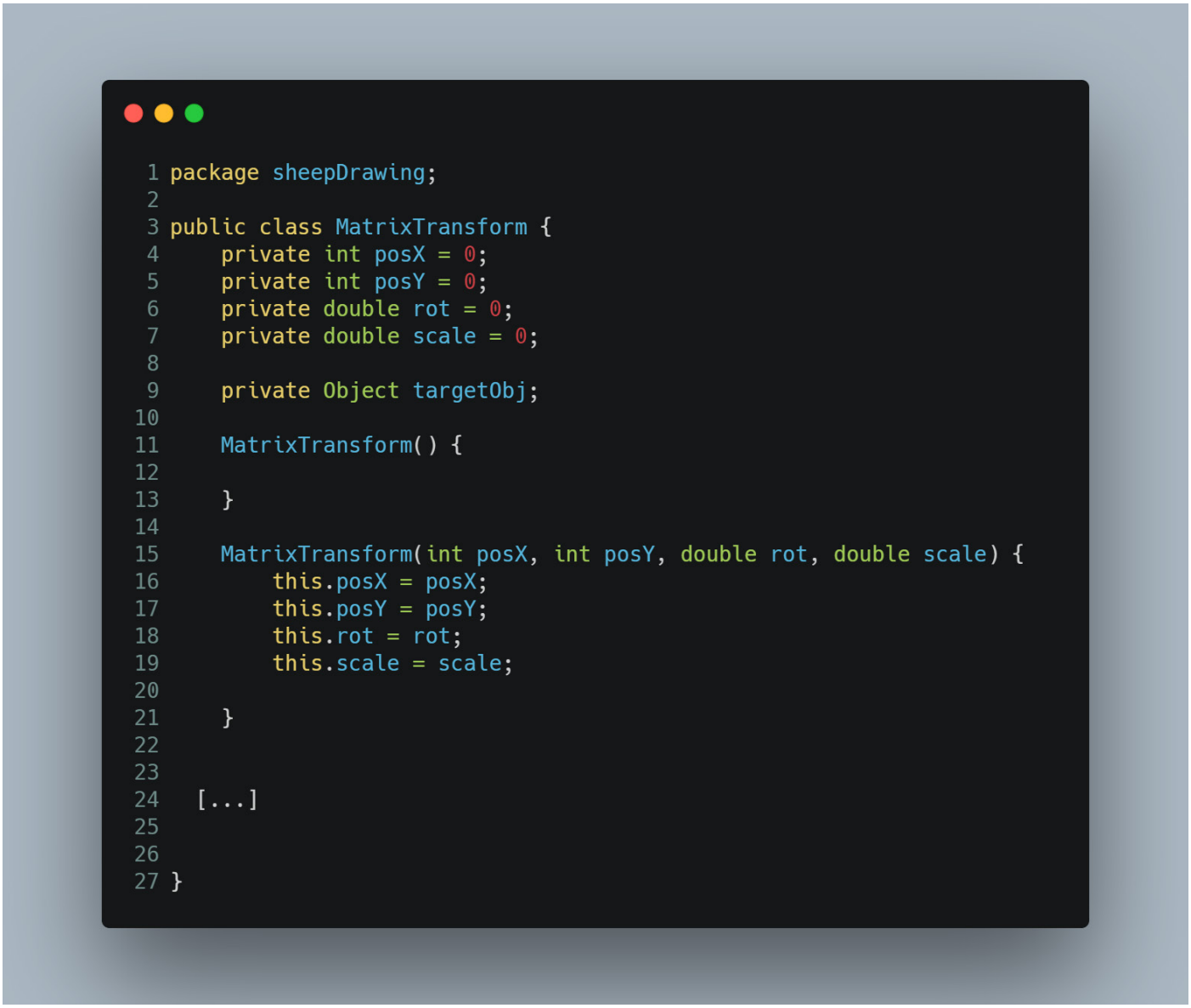
Exemple de la gestion des doublons d'images dans Dessin.java

Gestion des transformations

De loin la feature la plus difficile à implémenter.

Pour gérer l'architecture de notre librairie nous nous sommes inspiré de Processing une librairie graphique open source. Sa gestion d'instanciation d'objet graphique et manipulation des transformations nous a fait prendre conscience que, comme sur un moteur de jeux vidéo (Unity et Unreal par exemple), les logiques de transformations sont gérées dans une classe publique instancier a chaque objet et fonctionne en cascade pour actionner ces transformations sur les enfants.

C'est dans cet objet que nous gérons nos PSR ainsi que nvos calculs de symétrie.



```java
package sheepDrawing;

public class MatrixTransform {
    private int posX = 0;
    private int posY = 0;
    private double rot = 0;
    private double scale = 0;

    private Object targetObj;

    MatrixTransform() {

    }

    MatrixTransform(int posX, int posY, double rot, double scale) {
        this.posX = posX;
        this.posY = posY;
        this.rot = rot;
        this.scale = scale;

    }


  [...]


}
```

Initialisation de MatrixTransform.java

Conclusion

Un projet qui forme

Notre binome était composé d'un élève ayant deja vu beaucoup de languages différent différents, et d'un autre étant assez nouveau a la programmations. Dans les deux cas, ce projet nous a apporté beaucoup, à tous les deux. Des principes de développement pour l'un et comprendre les entrailles d'une librairie graphique pour l'autre

Aller plus loin

Ce projet est largement améliorable et ne manquerait pas d'optimasation. A commencer par la gestion des points. Chaque forme géométrique en utilise et toute les transformations pourraient s'effectuer par des mathématiques vectoriels. C'est pourquoi nous avons deja créer la class Vector2D que vous pouvez trouver dans le projet. Celle si permetterai de creer des point et d'aditionner, multiplier, normaliser etc, différents Vector2D entre eux, comme le font par exemple les librairies P5 ou Three en JavaScript

```java
public class Vector2D {
    public int x = 0;
    public int y = 0;

    Vector2D(int x, int y){
        this.x = x;
        this.y = y;
    }

    public void set(int x, int y){
        this.x = x;
        this.y = y;
    }

}
```

Démarrage de la classe Vector2D.java