« Comment devenir un Petit Prince Java ? »

Auteur	Action	Date
Lydia TERKI	Création du document	10/12/2022
Jérémy RIBES	Modification du document	15/12/2022
Lydia TERKI	Modification du document	16/12/2022
Jérémy RIBES	Modification du document	17/12/2022

Table des matières

1. Introduction 3
2. Répartition des tâches 4
3. Structure du projet 6
4. Le développement : le squelette de l'application 7
5. Un exemple d'exécution de l'application 9
6. Conclusion 10
7. Bibliographie - Webographie 10

1. Introduction

L'objectif est de créer un programme fonctionnel en Java utilisant les concepts de la P.O.O. Le principe d'encapsulation ne doit en aucun cas pouvoir être violé. L'application doit réaliser une grande fresque (patchwork) murale réalisée à partir de plusieurs dessins géométriques.
Les notions mathématiques à utiliser sont de niveau lycée et sont disponibles sur de nombreux sites. Le choix des données des classes n'est volontairement pas précisé.

- Les formes géométriques élémentaires disponibles sont les lignes, les polygones, les cercles et les ellipses ;
- Une image est composée de formes géométriques. L'itération sur les formes géométriques d'une image est à mettre en œuvre. Aucun doublon de forme géométrique ne peut exister dans une image ;
- Un dessin est composé d'images. L'itération sur les images d'un dessin est à mettre en œuvre. Aucun doublon d'image ne peut exister dans un dessin ;
- Il faut bien réfléchir aux relations entre les différentes classes du problème ;
- Les calculs du périmètre et de l'aire des formes géométriques, des images et des dessins sont requis. L'aire d'un polygone quelconque peut être calculée par triangulation. Le périmètre (l'aire) d'une image est la somme des périmètres (des aires) de ses composants. Il en est de même pour un dessin ;
- Les formes géométriques peuvent subir les transformations suivantes : homothétie, translation, rotation, symétrie centrale, symétrie axiale. Une classe matrice peut être utile. Appliquer une transformation :
  - Sur une image revient à appliquer la même transformation sur ses composants
  - Sur un dessin revient à appliquer la même transformation sur ses composants.
- La copie d'un dessin est obtenue par copie de toutes ses images ;
- Le tri des formes d'une image selon leur périmètre ;
- Le tri des images d'un dessin selon leur aire.

Afin d'assurer l'absence de doublons, il peut être judicieux de choisir une classe Collection assurant cette fonctionnalité. Il faut aussi penser à mettre en place une gestion d'exceptions lorsque cela s'avère nécessaire. L'utilisation à bon escient du mot-clé final est très vivement recommandée. Les conventions de codage Java sont à respecter scrupuleusement et les sources présentées lisiblement.

La mise en œuvre des points suivants est demandée :

Une application console créant la fresque et réalisant une batterie de tests les plus exhaustifs possibles. Plus précisément, l'ensemble des services proposé sera testé et leurs résultats affichés (méthodes de calcul de périmètre et d'aire, de transformations, de tri et de copie).
La fresque devra être affichée sous une forme textuelle la plus explicite possible.

Le groupe formé pour ce projet est composé de :

- RIBES Jérémy ;
- TERKI Lydia ;
- LABBE Luc.

2. Répartition des tâches

Pour un projet de développement d'application, il est important de répartir les tâches de manière équitable et efficace entre les différents membres de l'équipe. Cela permet de s'assurer que le projet avance de manière cohérente et que chaque membre de l'équipe est impliqué et contribue de manière significative.

Pour ce projet de « Patchwork », il est recommandé de répartir les tâches en trois grandes catégories : la conception, le développement et les tests. Chaque membre de l'équipe peut être responsable d'une de ces catégories, en fonction de ses compétences et de ses préférences.

Le premier choix qui a alors été fait en partant de ce principe est celui d'aborder la conception de manière collaborative pour démarrer efficacement le développement. Une réflexion de groupe sous forme de « brainstorming » a été réalisé pour gérer de façon efficiente la répartition des classes et de leurs fonctions.

Ensuite, Jérémy RIBES a été chargé de trois tâches :

- Celle de concrétiser les idées sélectionnées par le biais de supports visuels pour élaborer le plan du projet comme la création d'un diagramme de classe ou encore de maquettes nécessaires pour visualiser l'application finale ;
- Celle de créer un GitHub commun pour le partage d'informations et de code ;
- Et celle de réaliser le squelette de l'application en langage Java pour permettre le développement par la suite. C'est-à-dire, l'application fonctionnant avec une seule forme en ayant simplement les méthodes de périmètre et d'aire terminées pour toutes les classes.

La suite du projet a par conséquent été de compléter la mise en place du code de l'application. Pour cela, les différents membres de l'équipe ont dû utiliser les bibliothèques et outils de développement Java pour remplir la pluralité de méthodes et de fonctionnalités du sujet, en suivant les spécifications définies dans le plan fournit au début.

Lydia TERKI s'est occupée de :

- La classe Ligne ;
- La plus grande partie de la classe Ellipse et de la classe Cercle ;
- De remplir la Javadoc en fin de projet.

Luc LABBE s'est occupé de :

- La classe Polygone ;
- Compléter certaines méthodes de la classe Ellipse ;
- Compléter la classe Dessin et le Main restés au stade de squelette.

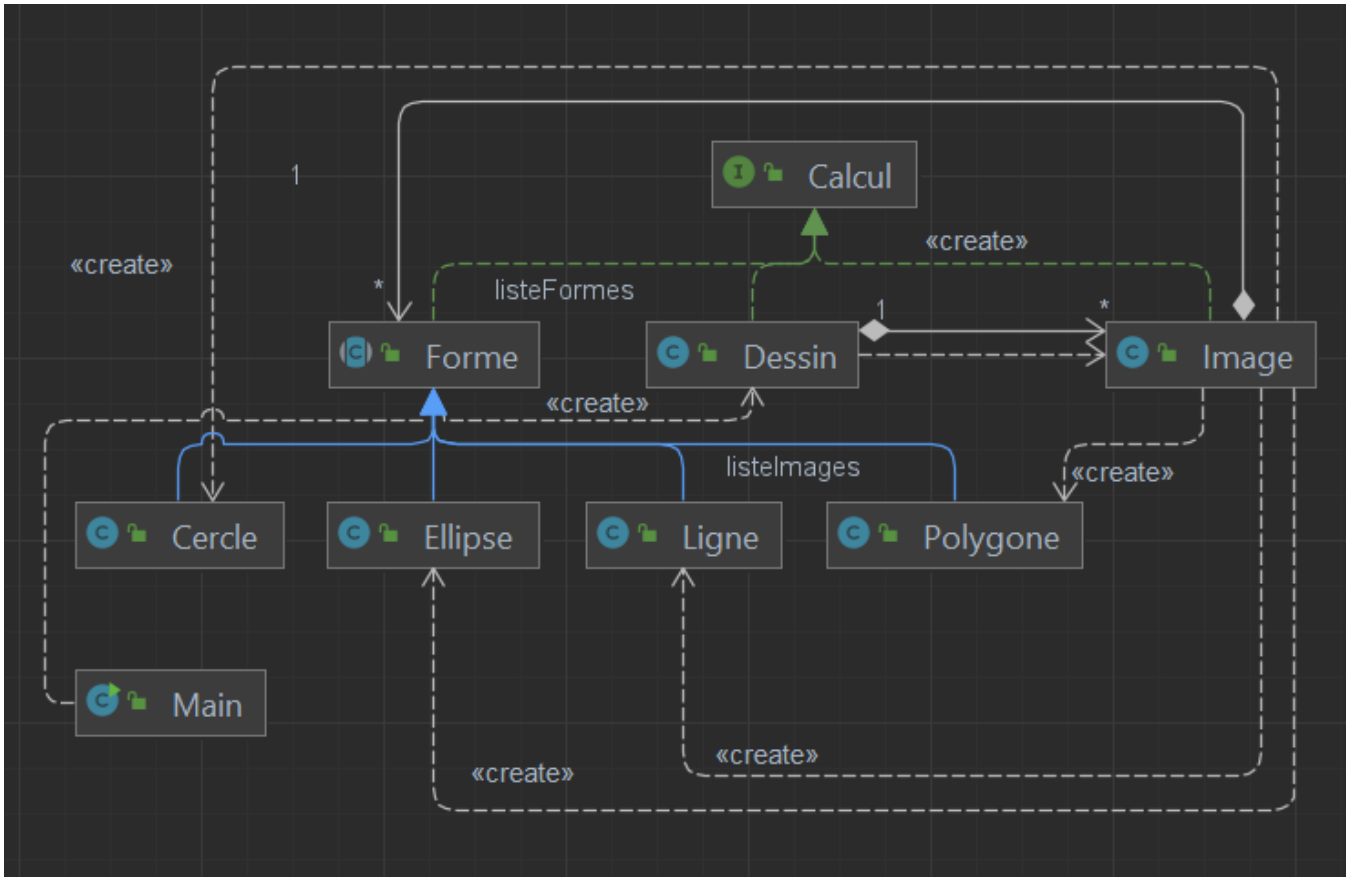
Jérémy RIBES a enfin achevé l'ensemble en :

- En réalisant les méthodes de tri par périmètre et par aire ainsi que celles de la copie d'une image et d'un dessin ;
- La gestion de certaines sorties de formes de la fenêtre ;
- S'assurant de la visibilité et du fonctionnant des méthodes sur les formes via des modifications sur le Main et la classe Dessin vers la fin de développement pour la phase de tests.

Il était important que les membres de l'équipe travaillent en étroite collaboration pour s'assurer que le projet avance de manière efficace et que les différentes tâches sont bien réparties. La communication et la coordination ont été essentielles pour réussir ce projet de développement d'application Java.

3. Structure du projet

A force de réflexion sur le sujet, voici le diagramme de classe qui a été établi en début de projet :



Les classes Cercle, Ellipse, Forme, Ligne et Polygone contiennent les constructeurs respectifs aux classes avec l'implémentation des méthodes qui permettent à l'utilisateur de modifier son dessin par la suite (rotation, translation, homothétie, symétrie centrale, symétrie axiale).
Ces classes contiennent aussi les méthodes *aire ()* et *périmètre ()* qui nous sont utiles pour implémenter les tris.

La classe abstraite Forme quant à elle définit les constructeurs, les setters et les getters. On déclare aussi les méthodes des formes en abstract afin de pouvoir les implémenter dans les classes directement.

L'interface Calcul initialise les fonctions *périmètre ()* et *aire ()* qui nous ont servis à retourner le périmètre et l'aire d'une forme.

La classe Image qui hérite de l'interface Calcul permet de générer une image (une image est composée de formes géométriques).

- Le constructeur *Image ()* de la classe Image génère deux formes avec des caractéristiques aléatoires (air, périmètre, diamètre, …). Le nombre de formes générés par image peut être modifié ;
- La méthode *paint(Graphics g)* permet de dessiner la forme dans l'interface ;
- La classe contient d'autres méthodes qu'il n'est pas utile d'expliquer dans le rapport (getters, setters, …)

4. Le développement : le squelette de l'application

Le Main

Dans le main est initialisé la fenêtre principale où est généré l'ensemble des dessins. Le panneau principal intégré dans la fenêtre est composé de deux autres panneaux :

- Un sur le dessus :
  - Instancié comme un BorderLayout pour dispatcher le titre et les boutons ;
- Et l'autre au-dessous :
  - Instancié comme un GridLayout pour afficher les différentes images du dessin généré.

Pour faire afficher un dessin ou encore appliquer des modifications, il est nécessaire de faire appel à différentes méthodes. Ces méthodes sont lancées par le clique sur un bouton :

```
btn_generer.addActionListener(e -> {
    panelPrincipal.remove(dessin.getPanelDessin());
    panelPrincipal.revalidate();
    panelPrincipal.repaint();
    panelPrincipal.add(dessin.changePanelDessin());
});
```

Lors du clique, le Main renvoie à la classe Dessin et ce qui la compose.

La classe Dessin

```
public Dessin() {
    this.listeImages = new HashSet<Image>();
    for(int i=0; i<nbImages; i++){
        listeImages.add(new Image());
    }
    this.panelDessin = new JPanel();
    this.panelDessin.setLayout(new GridLayout(2, 2));
    this.panelDessin.setBorder(this.blackline);
}
```

```
public JPanel getPanelDessin() {
    return panelDessin;
}
```

```
public JPanel getHomothetieDessin(){

    if(panelDessin != null) {
        System.out.println("\nHomothetie\n");
        for (Image image : listeImages) {
            for(Forme forme: image.getListeFormes()){
                forme.homothetie(200,200);
            }
        }
        panelDessin.revalidate();
        panelDessin.repaint();
    }

}
```

```
public void setListeImages(HashSet<Image> listeImages) {
    this.listeImages = listeImages;
}
```

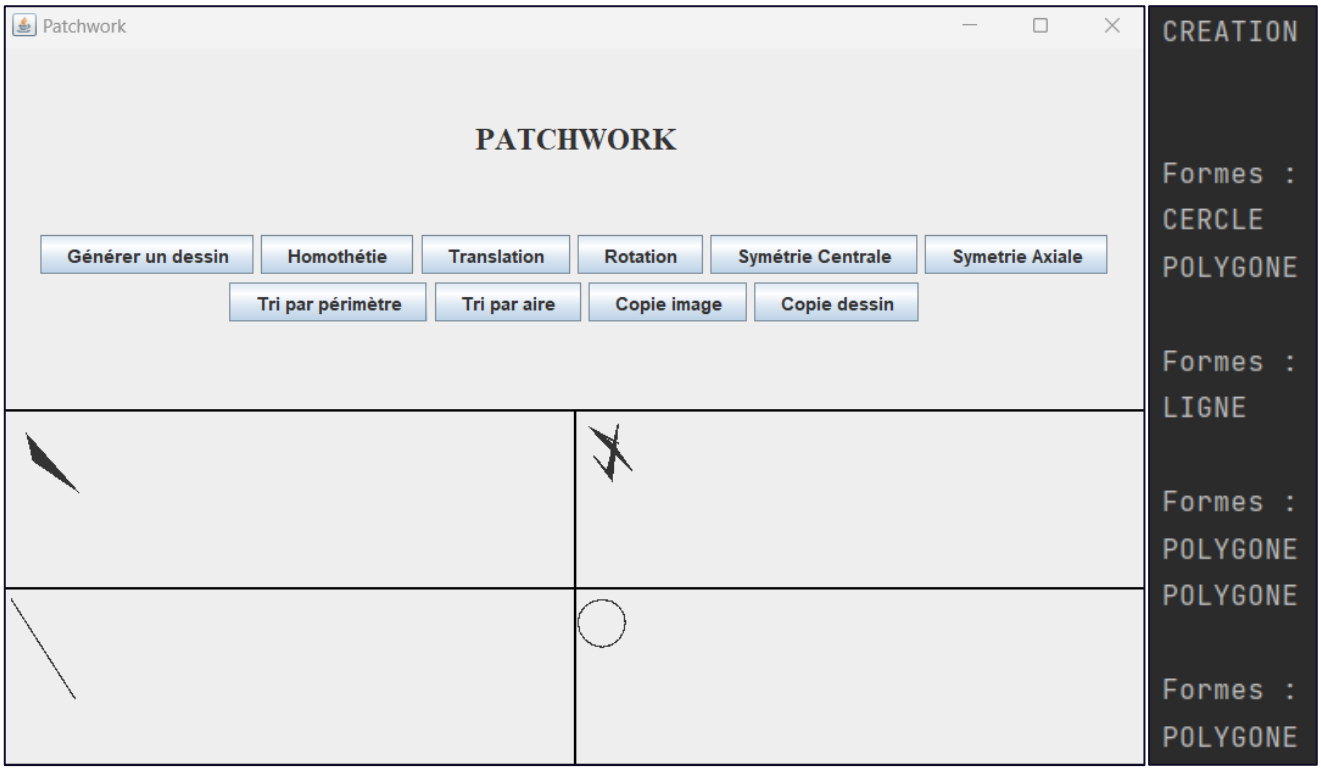
La classe Image

```
public Image() {

    this.listeFormes = new HashSet<Forme>();

    for(int i=0; i<2; i++){

        this.nbAleatoire = (int) Math.round(Math.random() * (4 - 1 + 1) + 1);
        this.x = (int) Math.round(Math.random() * (10 - 0 + 1) + 0);
        this.y = (int) Math.round(Math.random() * (10 - 0 + 1) + 0);
        this.width = (int) Math.round(Math.random() * (100 - 20 + 1) + 20);
        this.height = (int) Math.round(Math.random() * (100 - 20 + 1) + 20);

        switch (nbAleatoire){
            case 1:
                if(width!=height){
                    this.height = this.width;
                }
                listeFormes.add(new Cercle(x, y, width, height));
                break;
            case 2:
                listeFormes.add(new Ellipse(x,y,width,height));
                break;
            case 3:
                listeFormes.add(new Ligne(x,y,width,height));
                break;
            case 4:
                this.nbAleatoireTableau = (int) Math.round(Math.random() * (4 - 1 + 1) + 1);

                this.tab_x= new int[nbAleatoireTableau];
                this.tab_y= new int[nbAleatoireTableau];

                for(int j=0;j<nbAleatoireTableau; j++){
                    tab_x[j] = (int) Math.round(Math.random() * (50 - 5 + 1) + 5);
                    tab_y[j] = (int) Math.round(Math.random() * (50 - 5 + 1) + 5);
                }
                listeFormes.add(new Polygone(tab_x,tab_y));
                break;
            default:
                break;
        }
    }
}
```

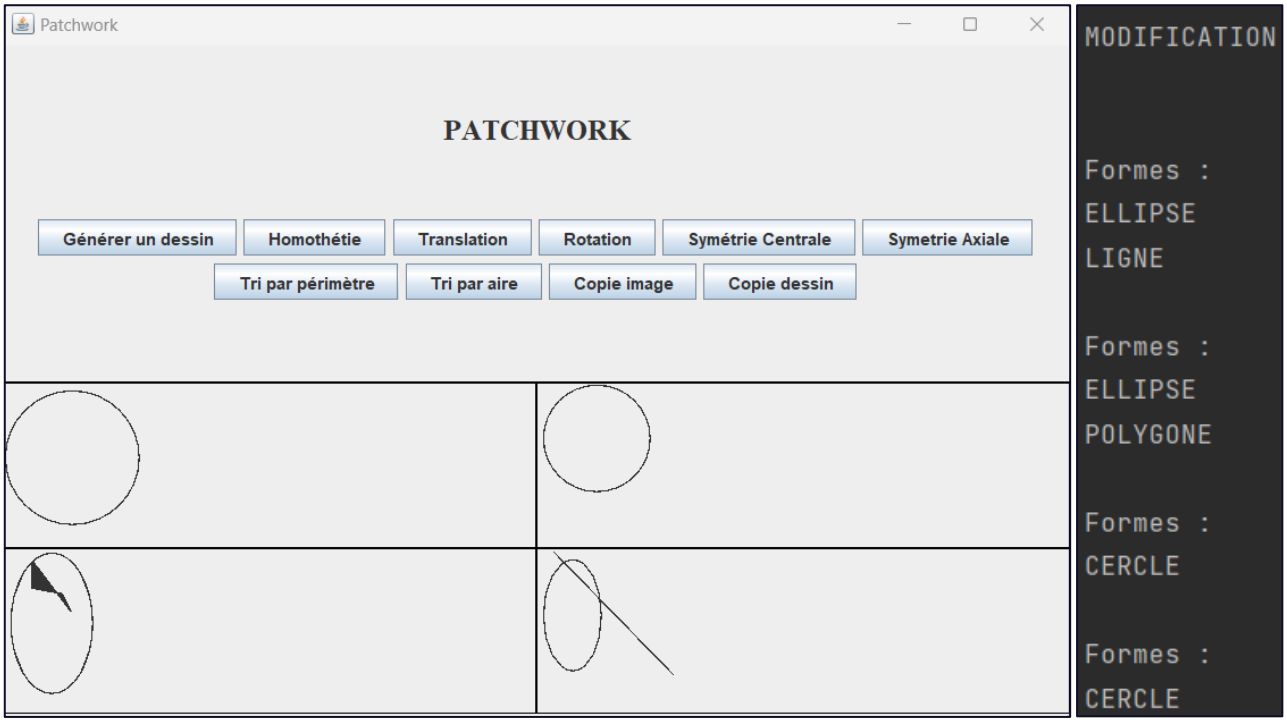
```
public void paint(Graphics g){

    System.out.println("\nFormes :");

    for(Forme forme: listeFormes) {

        if(forme instanceof Cercle) {
            g.drawOval(forme.getX(), forme.getY(), forme.getWidth(), forme.getHeight());
            System.out.println("CERCLE");
        }else if(forme instanceof Ellipse){
            g.drawOval(forme.getX(), forme.getY(), forme.getWidth(), forme.getHeight());
            System.out.println("ELLIPSE");
        }else if(forme instanceof Ligne){
            g.drawLine(forme.getX(), forme.getY(), forme.getWidth(), forme.getHeight());
            System.out.println("LIGNE");
        }else{
            g.drawPolygon(forme.getTab_x(),forme.getTab_y(),forme.getTab_x().length);
            g.fillPolygon(forme.getTab_x(),forme.getTab_y(),forme.getTab_x().length);
            System.out.println("POLYGONE");
        }
    }
}
```

5. Un exemple d'exécution de l'application

L'application en elle-même



La génération d'un dessin



6. Conclusion

En conclusion, ce projet de développement d'application Java dans le cadre de ce cours a été une expérience enrichissante. Nous avons appris de nombreuses choses sur la conception et la mise en œuvre d'une application Java avec la P.O.O., ainsi que sur la gestion d'un projet informatique. Nous avons également eu l'opportunité de mettre en pratique les connaissances théoriques acquises en cours, ce qui a été très bénéfique pour notre apprentissage.

Grâce à notre travail et notre collaboration efficace, nous avons réussi à développer une application fonctionnelle et qui répond aux besoins et aux exigences définis dans le cahier des charges. Nous sommes fiers du résultat final et nous espérons que le résultat satisfera l'évaluateur.

Ce projet a été une expérience intéressante pour nous et nous espérons que cela nous aidera dans nos prochaines matières de développement.

Par la suite, il aurait été intéressant de développer l'interface graphique tout en gérant la sortie des formes de l'écran.

7. Bibliographie - Webographie

https://duj.developpez.com/tutoriels/java/dessin/intro/
https://igm.univ-mlv.fr/~chochois/RessourcesCommunes/JAVA/Cours/coursRelationsEntreClasses.pdf
https://perso.telecom-paristech.fr/hudry/coursJava/interSwing/index.html