CC Méthodes de conception

Paysant Clémentine, Gallois Erwann, Elias Teresa, Guillou Aurelien

12 avril 2025

Sommaire

1 Introduction

Le but de ce projet est de créer une application de reproduction de formes en utilisant Java avec une architecture MVC (Modèle-Vue-Contrôleur) et d'autres méthodes vues en cours. Le jeu consiste à redessiner une image que l'on a pu observer au préalable. Cette image est constituée des différentes formes : cercle, carré et triangle. On peut y jouer de différentes façons : un ou deux joueur(s), et s'il n'y a qu'un joueur, il peut choisir s'il veut charger un fichier de dessins pré-faits ou une création avec un placement aléatoire des formes .

2 Répartitions des classes

Voici comment nous avons réparti les différentes classes et fichiers.Ainsi, le projet est plus facilement compréhensible du point de vue de l'architecture du projet et l'utilisation des méthodes de conception.

2.1 src

Le dossier src est constitué de deux dossiers, app qui comporte le code MV de l'application, qui est lui-même dans deux dossiers (model et vue). Model qui est le dossier comprenant le code du modèle avec un dossier pour les formes, un autre pour les joueurs et trois classes. Un dossier Vue, qui est l'affichage en swing du Modèle. Et util qui contient quatre classes pour le contrôleur.

2.2 dist

Le dossier dist comporte la compilation du code contenu dans le dossier src.

2.3 doc

Ce répertoire contient toute notre Javadoc.

2.4 Build

Ce fichier xml nous permet de re-générer le contenu de dist via ANT.

3 Diagramme des classes

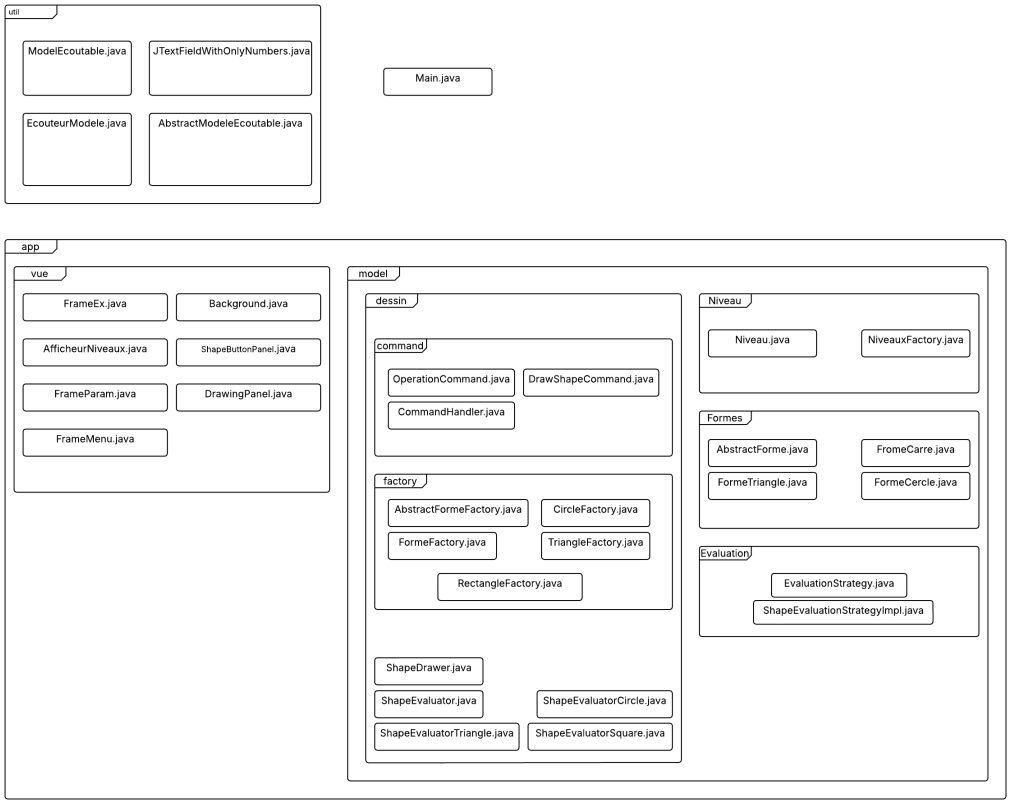

FIGURE 1 – diagramme des classes

4 MVC

4.1 Modèle

Dans le modèle MVC, le modèle représente les données et la logique du jeu. Le modèle est responsable de la création de la page de jeu, de la mise en place des formes.

Nous avons créé un dossier "formes" pour représenter des pièces de différents types(carré, cercle et triangle), qui contient des informations telles que les coordonnées et la longueur.

Nous avons un dossier "dessin" qui regroupe l'ensemble des classes utiles pour l'affichage des formes mais aussi la logique d'évaluation du score pour chaque forme lorsque le dessin est terminé il contient deux sous dossier :

- un dossier "factory" pour représenter les formes lorsqu'elle sont dessinées par l'utilisateur. Ces classes n'ont pas de constructeurs car ce sont des factory. Cela nous permets de centraliser l'ensemble des formes possibles et en rajouter pour de future mise à jour mais le dessin marchera toujours.
- un dossier "command" qui regroupe le CommandHandler pour contrôler le undo/redo. Cela va permettre à l'utilisateur de pouvoir annulée des actions ou les refaire.

Nous avons un dossier "évaluation" qui regroupe les classes pour évaluer le score de la partie

Nous avons un dernier dossier "niveau" qui gère les niveaux pré-enregistrée et serializé par Java.

4.2 Vue

Dans le modèle MVC (Modèle-Vue-Contrôleur), la vue est responsable de l'affichage des données et de la gestion des interactions utilisateur.

Dans notre projet, la vue est composée de plusieurs classes :

- MenuView : Représente l'écran principal permettant à l'utilisateur de choisir entre différents modes de jeu, comme "Joueur contre IA" ainsi que voir les règles du jeu.
- GameView : Contient les différent mode de jeu proposé dans notre application et gère l'affichage
- DrawingPanel : Est la zone centrale où l'utilisateur dessine les formes choisies à l'aide de la souris.
- Background : Importe l'image de fond affiché sur le menu principal lors du lancement du jeu

— ShapeButtonPanel : C'est la zone qui contient l'ensemble des boutons pour selectionner la forme que l'utilisateur souhaite dessinée

L'utilisateur peut cliquer sur un bouton pour choisir la forme qu'il souhaite dessiner, puis cliquer sur la zone de dessin pour placer la forme. Il peut également maintenir le clic et glisser la souris pour en ajuster la taille, à la manière de Paint. Une fois l'action réalisée, la vue transmet les coordonnées de la forme au contrôleur pour traitement.

Voici comme exemple le menu :

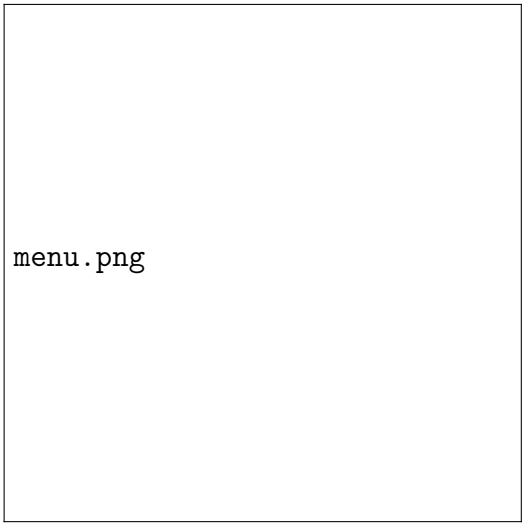

FIGURE 2 – Menu

et une partie en cours :

FIGURE 3 – Une partie en cours

4.3 Contrôleur

Dans le modèle MVC, le contrôleur est responsable de la gestion de la communication entre le modèle et la vue.

Dans notre projet, nous avons plusieurs classes qui permettent de rafraîchir et de contrôler la vue. La classe abstraite *AbstractModeleEcoutable* permet d'implémenter l'interface *EcouteurModele*. Elles permettent grâce à la fonction *fireChangement()*, de mettre à jour la vue en utilisant une fonction de mise à jour sur chaque écouteur d'une liste prédéfinie.

5 Conclusion

En utilisant l'architecture MVC, nous avons créé une application de reproduction de dessin fonctionnelle en Java. Le modèle gère les données et la logique du jeu, la vue gère l'affichage des données et l'interaction utilisateur, et le contrôleur gère la communication entre les deux. L'architecture MVC permet une séparation claire des responsabilités et facilite la maintenance et l'évolution de l'application. Nous aurions pu, dans une optique d'améliorations ajouter d'autres types de formes et implémenter d'autres design patterns.