Rapport du projet A31-MasterMind : Rendu 2

Table des matières

1. Introduction :

Dans ce 2ème jet de notre projet de jeu MasterMind en Java, nous avons terminé notre application.
Nous disposons :

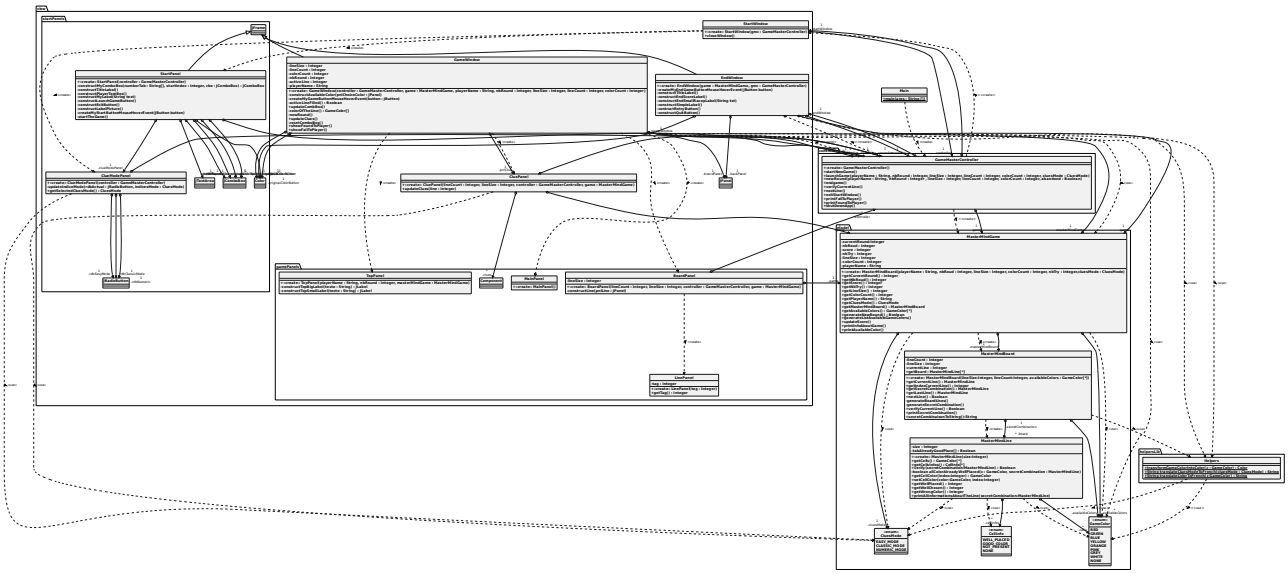
- du code source complet
- du diagramme de classe UML qui correspond à ce code
- de l'exécutable au format JAR
- d'un mode d'emploi nommé INSTALL.md

Nous allons expliquer les évolutions de nos choix de conception depuis le 1er rapport.

---

2. Nos choix et explications :

2.1 Notre DCC :



2.2 Patron de conception :

2.2.1 Contrôleur :

Notre projet utilise le patron de conception MVC (Modèle, Vue, Contrôleur). Nous avons les 3 packages pour cette architecture logiciel (*models*, *view* et *controller*). Nous n'avons qu'une seule classe de contrôle qui gère tous les contrôles de l'application. Contrairement à ce qui est écrit dans la conclusion du premier rapport, nous n'avons pas utilisé d'*Observers*, car finalement, c'est le contrôleur qui demande à la vue de se mettre à jour. Nous aurions bien sûr pu implémenter des *Oberserver* sur notre modèle afin de pouvoir tous les avertir en cas de changement.

2.2.2 Modèle :

Le modèle n'a pas beaucoup changé depuis le 1er rapport si ce n'est quelques corrections dans les règles de jeu et une réorganisation de certaines méthodes.

2.2.3 Vues :

Nous avons 3 vues dans notre application :

- fenêtre de début avec choix des options
- fenêtre de jeu
- fenêtre de fin avec un récapitulatif de la partie

Nous avons aussi éclaté en classes certains Panels de la vue de démarrage et de jeu afin de les diviser et permettre une meilleure gestion du code. Ces classes sont rangées dans des packages. Il n'y a que la vue de fin qui est en une seule classe, car celle-ci est composée d'un unique Panel.

Le fonctionnement de nos vues est le suivant : nous commençons par créer la vue de début. Lorsque le joueur clique sur démarrer une partie, la vue de début est détruite et celle de jeu est créée. Une fois qu'il a trouvé ou perdu, la vue de jeu est détruite et immédiatement recréée avec les informations de la manche suivante et ce jusqu'à la manche finale où à la place de recréer une vue de jeu, on créé la vue de fin.

Cette méthode nous permet de ne pas avoir à manipuler les Panels pour modifier les valeurs pour la manche suivante. La création d'une autre vue permet d'avoir une toute nouvelle vue et de modifier des valeurs.

Il est à noter que les fenêtres de début et de fin ont des dimensions fixes/limites pour éviter des affichages non voulus. La fenêtre de jeu a elle, une taille par défaut, mais qui peut être ajustée par le joueur en fonction de son écran.
Bien évidemment, la partie graphisme de l'application peut être améliorée, mais ce n'est pas le sujet principale du projet.

2.3 Helpers :

Nous avons aussi rajouté une classe d'aide nommé *Helpers* qui permet d'avoir accès à des méthodes d'aides, notamment la traduction des couleurs et des modes d'indices en français et la transformation de notre énumération *GameColor* en l'énumération *Color* pour JavaSwing (cela nous permet de colorier les ComboBox du jeu dans la bonne couleur afin que ce soit plus agréable à jouer). Les 3 méthodes de cette classe sont en *static*, car ceux sont des méthodes d'aides que toutes les classes peuvent utiliser et la création d'un objet de cette classe à chaque fois que l'on a besoin d'une des méthodes est très inutile.

2.4 Commentaires :

Nous avons ajouté des commentairesau début de chaque méthode pour expliquer ce qu'elle fait. Nous avons écrit les commentaires en anglais. Des commentaires sont aussi mis à certaines lignes si nous avons fait quelque chose de particulier.

2.5 Jouabilité :

Le jeu se joue avec des ComboBox. Cela est très simple à comprendre et à utiliser pour le joueur. Il ne peut pas faire autre chose. Cela permet de limiter le risque d'erreurs pour le joueur et de plantage dans le code. De plus, la gestion de ces composants dans le code est simple et facile à comprendre. Avec cette facilité, on peut toujours rajouter ou retirer des couleurs.

3. Conclusion :

Grâce à tous ces choix, notre application permet d'être très simple d'un point de vue code et d'utilisation pour l'utilisateur. Des améliorations sont toujours possibles.