Projet de Jeu, MASTER-MIND en Java

Saphir Gobbi

27 avril 2024

Table des matières

1 Introduction

Le présent rapport décrit la conception et l'implémentation d'un jeu de type « Master Mind », un jeu de logique où le joueur doit deviner une combinaison de pions de couleur, seul ou en multi-joueur. Ce projet consiste à créer une version informatisée du jeu, en permettant notamment la personnalisation des paramètres tels que le nombre de pions et de couleurs, ainsi que le nombre de tentatives autorisées. De plus, une interface graphique via Awt, Swing a été implémenté, ainsi qu'une fonction de sauvegarde et de chargement de partie en fichier Json.
Le rapport abordera la structure du jeu et son fonctionnement.

2 Structure du Jeu

La structure du jeu repose sur plusieurs classes métier, chacune ayant un rôle spécifique.

— `class GUI extends JFrame:`
S'occupe de l'affichage des classes. Possède le main du jeu.

— `abstract class GUIComponent:`
Une classe abstraite qui est étendu par toutes les classes qui affiche quelque chose. Possède comme attributstatique une liste d'options et le

profil de chaque joueurs, le nombre de joueurs, le nombre de partie et la partie actuellement jouée.

— `class Home extends GUIComponent:`
Une classe qui affiche au lancement du jeu, une page d'accueil.

— `class SetOptions extends GUIComponent:`
Une classe qui affiche une page d'options que l'utilisateur devra cocher, les traitera, initialisera les informations nécessaire, et créera les parties.

— `class Play extends GUIComponent:`
Une classe qui affiche le jeu.

— `class Score extends GUIComponent:`
Une classe qui affiche le score de chaque joueurs.

— `class Save extends GUIComponent:`
Une classe qui permet de sauvegarder la partie en cours dans n'importe quel dossier local, en le mettant dans un fichier Json.

— `class Load extends GUIComponent:`
Une classe qui permet de charger une partie enregistrer en parsant le fichier Json associer.

— `class Jeu:`
Une classe qui possède des méthodes pour tester la validité d'une tentative de l'utilisateur, et renvoie une liste selon la difficulté choisie.

— `class Traitement:`
Une classe a comme attribut toute les options, choisie par l'utilisateur. Un Traitement par joueur.

— `class Combinaison:`
Une classe qui a comme attribut une liste de Pion.

— `class Pion:`
Une classe qui a comme attribut une position et une couleur.

— `public enum Couleur:`
Une enumération de Couleur.

3 Programme principal

Le projet possède 3 dossiers : src, build, media qui contient les png utilisés, et lib qui contient la librairie Json. La compilation du jeu et le lancement se fait respectivement avec les commandes :

```
javac -cp lib/json-simple.jar -d build/ src/*.java
```

et

```
java -cp lib/json-simple.jar:build GUI
```

Pour l'affichage du jeu, le choix d'une interface graphique a été fait, via Swing, Awt et du Json pour la sauvegarde et la charge d'une partie. La classe `GUI extends JFrame`, permet l'affichage d'une fenêtre graphique, possède le `main` principal et lance la page d'accueil.

FIGURE 1 – Page d'accueil

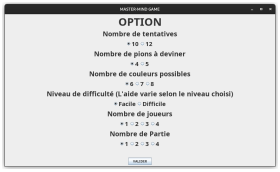

FIGURE 2 – Les options

FIGURE 3 – Le jeu

FIGURE 4 – Le tableau des scores

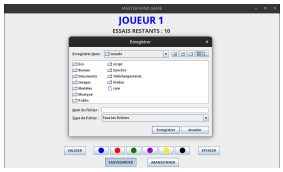

FIGURE 5 – Création d'une Sauvegarde

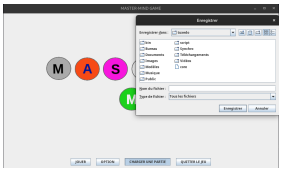

FIGURE 6 – Chargement d'une partie

4 Obstacle Rencontré

Le premier problème rencontré a été le choix de la structure à adopter, en passant d'une seul classe possédant des fonctions qui affiche les différents écrans, à une classe `GUI extend JFrame` qui affiche les écrans ainsi qu'une classe abstraite `GUIComponent` et de ses sous-classes qui re-définissent sa methode `public void setupUI()` (décrivant la structure graphique et les options de l'écran a afficher), partage ses attributs static et ont chacunes une méthode dans `GUI` accessbible pour toutes les sous-classes de `GUIComponent` qui appel `setupUI()` de chaque sous-classe de `GUIComponent`.

Dans un second temps, la fonction de Sauvegarde et de Charge a été un difficile a implémenté, d'abord par sérialisation qui n'était pas adapté. Le choix a été fait d'utilisé une librairie Json pour Sauvegarder. Convertissant toute les couleurs de type `Couleur` en `String` pendant la Sauvegarde afin que le fichier Json soit valide, puis en faisant l'opération inverse lors du chargement de la partie.

5 Conclusion

Ce projet nous a permis de mettre en pratique nos connaissances en programmation orientée objet et de relever plusieurs défis liés à la conception d'un jeu complexe. Nous avons appris à gérer la structure du jeu, l'interaction avec l'utilisateur, ainsi que la manipulation de données et les opérations de sauvegarde et restauration. Des améliorations futur pourrai être envisagé telle qu'un mode de jeu `User vs Computer`.