Rendu TP2 – INFO502

Tayeb Nemiche Moussa

8 novembre 2024

1 Introduction

Ce rapport présente l'architecture et l'organisation du mon implémentation d'un jeu de poker simplifié en Java. Le code est structuré autour de plusieurs classes qui permettent de gérer les cartes, les mains, les combinaisons de poker, le paquet de cartes et le talon de cartes. Cette architecture suit les principes de la programmation orientée objet, et offre une solution modulaire et extensible pour simuler un jeu de poker.

2 Structure du Code

Le code est composé des éléments suivants :

— **Énumérations** : pour représenter les différentes couleurs et valeurs des cartes ainsi que les combinaisons gagnantes de poker.

— **Classes** : pour représenter les objets spécifiques, tels que la carte, la main, le paquet de cartes et le talon.

2.1 Énumérations

Les énumérations sont utilisées pour représenter les valeurs et les couleurs des cartes, ainsi que les différentes combinaisons possibles au poker.

— **Couleur** : Cette énumération définit les quatre couleurs possibles des cartes au poker : COEUR, CARREAU, PIQUE et TREFLE.

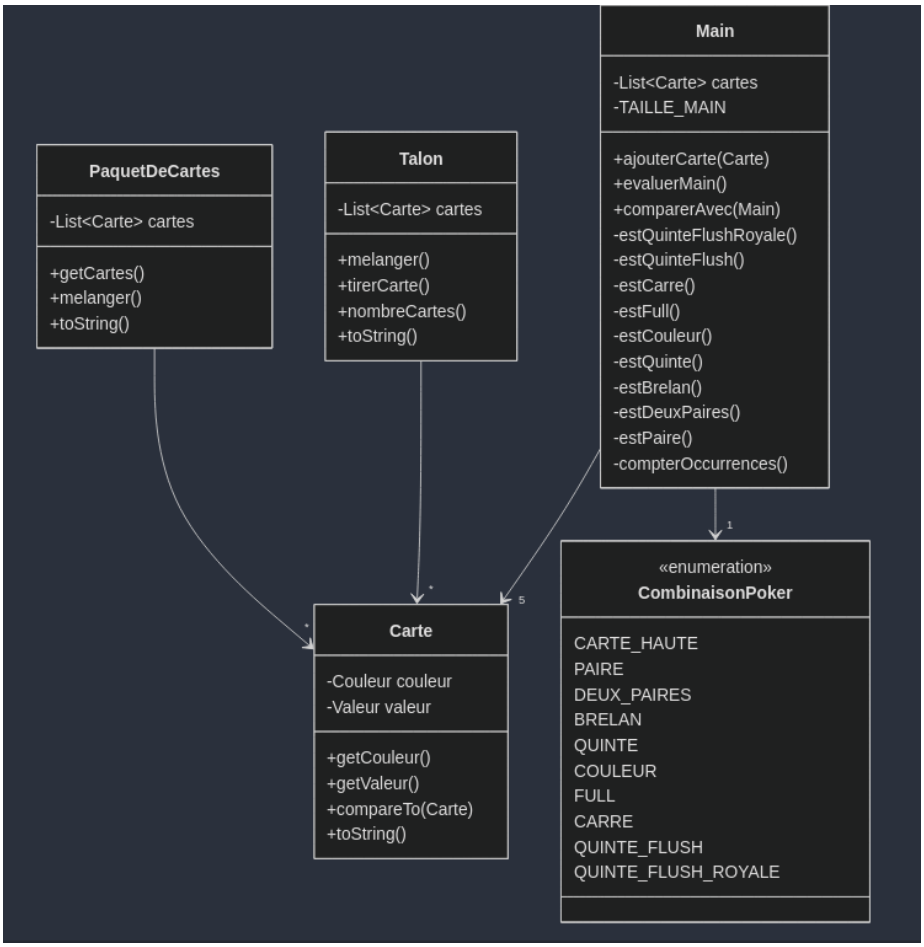
— **Valeur** : Cette énumération représente les différentes valeurs des cartes, allant de 2 à l'AS. Chaque valeur a un entier associé qui représente sa valeur numérique, ce qui permet de comparer les cartes.

— **CombinaisonPoker** : Cette énumération définit les différentes combinaisons possibles au poker, telles que la Paire, le Brelan, la Quinte, etc. Chaque combinaison est associée à une valeur numérique permettant de les comparer.

2.2 Classes

Le code est organisé en plusieurs classes qui modélisent les différentes entités du jeu de poker.

— **Carte** : Représente une carte de poker, composée d'une couleur (de type `Couleur`) et d'une valeur (de type `Valeur`). La classe implémente l'interface `Comparable` pour permettre la comparaison de cartes par leur valeur.
— **Main** : Représente une main de 5 cartes. c'est la classe la plus importante car elle contient beaucoup de fonctions utiles qui décident la résultat de la partie, elle permet d'ajouter des cartes à la main, de vérifier les combinaisons de poker présentes dans la main (comme la Paire, la Quinte, etc.), et de comparer deux mains.
— **PaquetDeCartes** : Représente un paquet de cartes. Il contient toutes les cartes possibles en fonction des couleurs et des valeurs, et permet de mélanger le paquet.
— **Talon** : Représente le talon de cartes, qui contient les cartes disponibles pour être tirées par les joueurs. Le talon peut contenir plusieurs paquets de cartes et permet de mélanger les cartes.



3 Choix de Conception

3.1 Utilisation des énumérations

Les énumérations `Couleur`, `Valeur` et `CombinaisonPoker` ont été utilisées pour assurer une gestion claire des éléments du jeu. L'utilisation des énumérations permet de centraliser la gestion des valeurs possibles, c'est un changement que j'ai fait, car au début j'ai utilisé un Liste, mais ça a causé des problèmes de genstion et un code pas trés lisible.

3.2 Classe `Carte`

La classe `Carte` est essentielle pour modéliser les cartes de poker. En utilisant un constructeur qui prend en paramètre une couleur et une valeur, cette classe permet de créer des objets représentant des cartes individuelles. La méthode `compareTo` permet de comparer les cartes entre elles, ce qui est nécessaire pour trier les mains de poker et évaluer les combinaisons.

3.3 Classe `Main`

La classe `Main` représente une main de poker composée de 5 cartes. Elle contient des méthodes pour ajouter des cartes, évaluer les combinaisons possibles, et comparer les mains entre elles. Le processus d'évaluation repose sur plusieurs méthodes privées pour vérifier les différentes combinaisons de poker, telles que la Paire, le Brelan, la Quinte Flush, etc. Le tri des cartes dans la méthode `evaluerMain` et l'utilisation de `compareTo` assurent une évaluation correcte de la main.

3.4 Classe `PaquetDeCartes` et `Talon`

La classe `PaquetDeCartes` représente un paquet de cartes complet, comprenant toutes les combinaisons possibles de couleurs et de valeurs. La méthode `melanger` permet de mélanger le paquet avant de l'utiliser.

La classe `Talon` gère plusieurs paquets de cartes et permet de tirer des cartes du talon. Elle offre une méthode `tirerCarte` pour retirer la carte en haut du talon et une méthode `nombreCartes` pour connaître le nombre de cartes restantes. Le talon peut être mélangé plusieurs fois, et il est géré de manière flexible pour permettre l'utilisation de plusieurs paquets de cartes.

3.5 Gestion des combinaisons de poker

Le code utilise une série de méthodes privées dans la classe `Main` pour déterminer les combinaisons présentes dans une main. Ces méthodes vérifient la présence de différentes combinaisons, telles que la `QuinteFlush`, le `Carre`, le `Full`, etc., en fonction des règles du poker. Le processus de comparaison de mains repose sur l'évaluation des combinaisons et, en cas d'égalité, sur la comparaison des cartes les plus élevées.

3.6 Comparaison des mains

La méthode `comparerAvec` permet de comparer deux mains de poker. Si les mains ont des combinaisons différentes, la comparaison est effectuée sur la base de la valeur des combinaisons. Si les combinaisons sont identiques, les cartes de chaque main sont comparées pour déterminer la main gagnante.

4 Jeux d'Essais

Pour tester le code, j'ai suivi les jeux d'essais proposé dans le TP permettant de vérifier :

- La mise en place et le mélange du talon.
- La création de mains et l'évaluation des combinaisons.
- La simulation d'une partie de poker simple avec 4 joueurs, sans relance ni rejet.

4.1 1. Mise en place et mélange du talon

Ce test vérifie la création d'un talon avec un ou plusieurs paquets et le mélange des cartes :

```
Talon talon = new Talon(1);
System.out.println(talon);
talon.melanger();
System.out.println(talon);
```

4.2 2. Création et évaluation de mains

Nous créons une main et évaluons sa combinaison :

```
Main main1 = new Main();
main1.ajouterCarte(new Carte(Couleur.CARREAU, Valeur.AS));
main1.ajouterCarte(new Carte(Couleur.COEUR, Valeur.ROI));
main1.ajouterCarte(new Carte(Couleur.CARREAU, Valeur.ROI));
// Ajouter 3 autres cartes..., et une autre main (main2)
System.out.println("Main: " + main);
afficherMain("Main 2", main2);
int resultat = main1.comparerAvec(main2);
```

5 Implémentation du jeu Texas Hold'em

L'implémentation du jeu de Texas Hold'em repose sur une approche orientée objet en Java. Le jeu se déroule avec un nombre de joueurs déterminé, où chaque joueur reçoit des cartes privées et participe à une évaluation de la meilleure main possible parmi les cartes communautaires et privées. Ce système utilise plusieurs classes, chacune responsable d'un aspect spécifique du jeu.

5.1 Description du processus de jeu

Le jeu commence par la création d'un objet `PokerHoldem`, où le talon est mélangé, et chaque joueur reçoit deux cartes privées. Ensuite, cinq cartes communautaires sont distribuées face visible. Chaque joueur utilise ses cartes privées et les cartes communautaires pour former la meilleure main de 5 cartes possible. La meilleure main est déterminée en générant toutes les combinaisons possibles de 5 cartes à partir des 7 cartes disponibles (les 2 cartes privées et les 5 cartes communautaires).

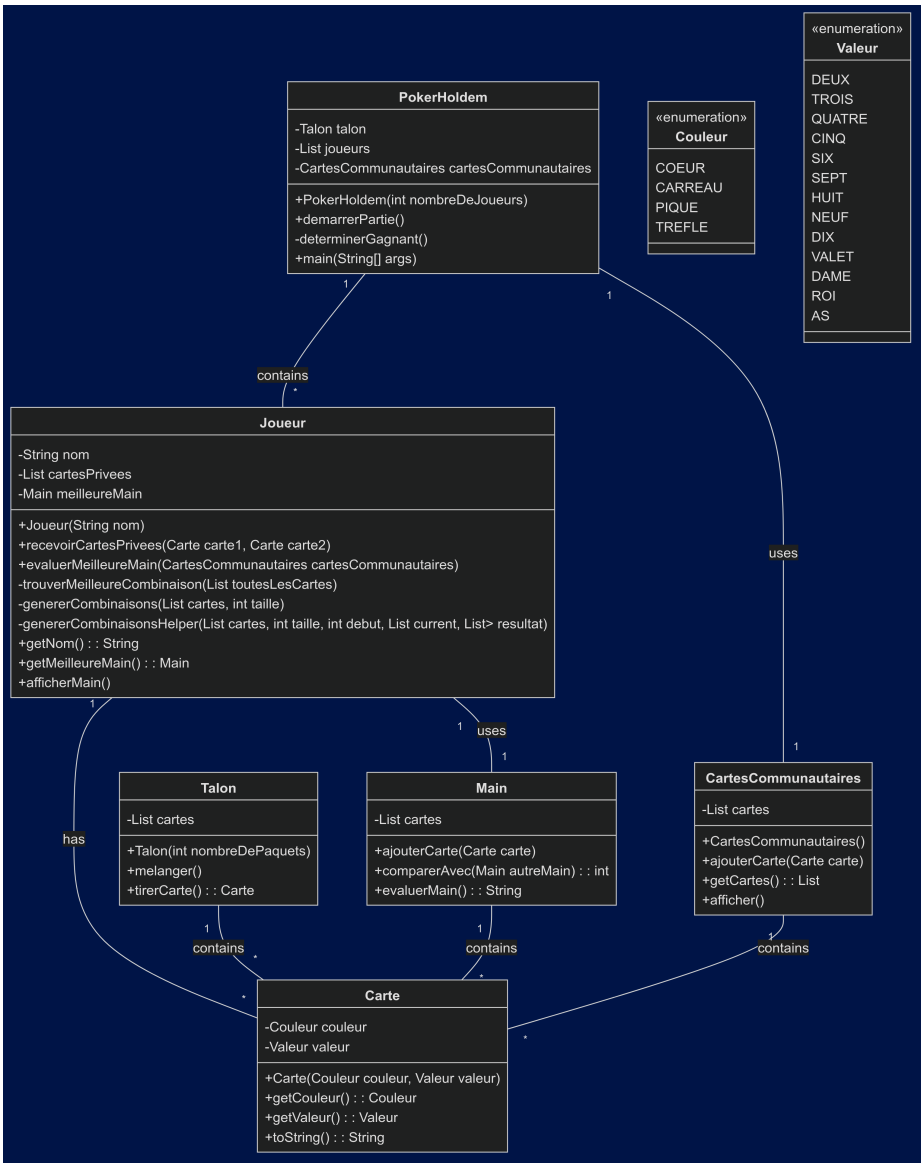
Une fois toutes les mains évaluées, le gagnant est déterminé en comparant les meilleures mains de chaque joueur. La comparaison se fait en fonction de la force des combinaisons (par exemple, une paire, une couleur, une quinte, etc.).

5.2 Démarrage et déroulement d'une partie

Voici un résumé des étapes suivies par le jeu lors de l'exécution de la méthode `demarrerPartie()` :

1. Les cartes communautaires sont affichées.
2. Chaque joueur reçoit ses cartes privées et évalue sa meilleure main en utilisant les cartes communautaires.
3. La meilleure main de chaque joueur est affichée, ainsi que la combinaison qu'il a obtenue (par exemple, `Paire`, `Quinte`, etc.).
4. Enfin, le gagnant est déterminé en comparant les mains des joueurs.

La méthode `determinerGagnant()` compare les mains de chaque joueur et annonce le joueur avec la meilleure main. Si plusieurs joueurs ont la même combinaison, la comparaison des cartes individuelles (les cartes les plus élevées) est effectuée.



5.3 Méthode d'évaluation des mains

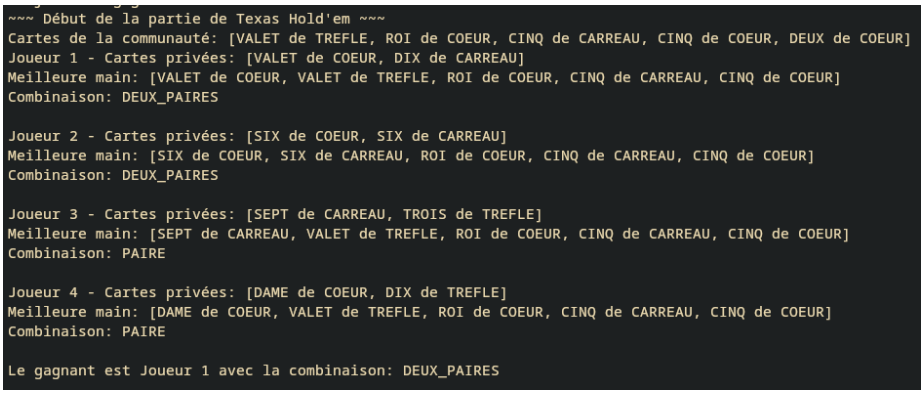
La classe `Joueur` contient la méthode `evaluerMeilleureMain`, qui prend en compte les cartes privées et les cartes communautaires. Cette méthode génère toutes les combinaisons possibles de 5 cartes parmi les 7 disponibles (cartes privées + cartes communautaires) et évalue chaque combinaison en appelant la méthode `comparerAvec()` de la classe `Main`.

Les combinaisons sont ensuite triées et la meilleure est sélectionnée. La

méthode de comparaison des mains repose sur la force des combinaisons, en suivant les règles classiques du poker (par exemple, une quinte flush royale bat une quinte flush, qui elle-même bat un carré, etc.).

5.4 Exemple d'exécution

Lors de l'exécution du jeu, la sortie console affiche les cartes de chaque joueur, les cartes communautaires, la meilleure main de chaque joueur et le gagnant de la partie. Par exemple :



6 Conclusion

Mon architecture du code est adaptée pour simuler un jeu de poker en respectant les règles de base du jeu. Elle utilise les principes de la programmation orientée objet, notamment l'encapsulation et la modularité, ce qui facilite la compréhension et l'extension du code. L'utilisation d'énumérations pour les couleurs, les valeurs et les combinaisons rend le code plus lisible et réduit les risques d'erreurs. Le système de gestion des mains et des cartes est flexible, ce qui permet d'ajouter facilement de nouvelles fonctionnalités si nécessaire.