Projet Java: UNO

TCS3-IN
Conception et programmation orientées objet et introduction au réseaux

Auteurs :
Christophe LANNOY
Tanguy PERENNOU

24 Novembre 2019

1 Lancement le jeu

- Lancer le serveur (server.ServerMain) et suivre les instruction de la console.
- Lancer les joueurs (client.PlayerMain) et suivre les instructions de la console. Si plus de joueurs que spécifié dans la console du server sont lancés, ces derniers sont refusés et reçoivent un message d'erreur.

Les adresses et ports des clients et du serveur sont modifiables dans les premières lignes de (server.ServerMain) et (client.PlayerMain) si nécessaire.

2 Extensions implémentées

- **Les 108 cartes du jeux ont été implémentées:** (classe abstraite "Card" dont toutes les cartes héritent)
  - Parmi les cartes colorées on retrouve: (classe abstraite ColoredCard)
    - Les cartes colorées de 0 à 9 modélisées par la classe "ClassicCard"
    - Les cartes +2-couleur modélisées par la classe "Plus2Card"
    - Les cartes Passer-couleur modélisées par la classe "PassCard"
    - Les cartes Inversion-couleur modélisées par la classe "InvertCard"
  - Parmi les cartes non colorées on retrouve: (classe abstraite "NonColoredCard")
    - Les cartes Joker modélisées par la classe "Joker"
    - Les cartes +4 modélisées par la classe "Plus4Card"
- **Robustesse par rapport aux entrées du joueur:** L'utilisateur communique ses choix de jeu (Poser/Piocher/Passer) par des réponses binaires (yes/no abrégé en y/n). Tant que celui-ci n'aura pas tapé une de ces deux touches la question lui sera posée à nouveau. Pour indiquer une carte qu'il souhaite jouer, il entre l'indice de cette carte. A nouveau, si l'indice n'est pas correct, la question est reposée. Il n'est donc par exemple pas possible que le joueur pose une carte qu'il n'aurait pas en main ou qu'il passe son tour sans piocher.
  De plus, une seconde vérification au niveau du serveur quant à la validité des informations envoyées par le client est également effectuée. (Cela aide au débuggage et permet de connecter à ce serveur un client de quelqu'un d'autre qui n'aurait pas implémenté cette première vérification) Si le protocole n'est pas respecté par le client ou que ce dernier joue une carte qu'il ne possède pas, un message d'erreur s'affiche sur la console du serveur.

- **Gestion de parties en plusieurs manches pour 2 à.. beaucoup de joueurs:** Ce paramètre se configure dans la console du "ServeurMain". Les résultats sont diffusés aux joueurs en fin de manches et en fin de partie (seulement sur les consoles). Par ailleurs, il possible de modifier le score maximal indiquant la fin de partie pour simuler plus vite des parties complètes (server.ServerMain ligne 7).

- **Interface graphique:** L'utilisateur peut choisir de lancer une interface graphique avant de commencer à jouer. Une combinaison d'un joueur sur console et l'autre sur GUI est possible. Une fenêtre redimensionnable s'ouvre alors affichant la main du joueur, le talon actuel ainsi que les cartes face caché des autres joueurs. Lorsqu'un Joker ou +4 est posé, la couleur choisie par le joueur est rappelée par un cadre coloré encadrant la carte. (Si on lance l'interface graphique, il est important de ne pas appeler deux joueurs par le même nom, sinon la GUI n'affiche pas correctement le nombre de carte des adversaires. Je n'ai pas eu le temps de régler ce problème) Afin de respecter le protocole de communication, les adversaires et leurs attributs s'affichent sur le GUI dès que le client en a connaissance via les messages du serveur. Ce dernier ne présentant pas l'ensemble des joueurs au début de la parties, les adversaires ne sont affichés qu'après qu'ils aient joué leur première carte (ou pioché). [1]

- **Multi-threading:** Un thread ne nécessitant pas de synchronisation est lancé au début du "ServerMain" pour pouvoir refuser des joueurs supplémentaires qui voudraient rejoindre la partie. Du coté du client des threads synchronisés sont lancés afin de pouvoir lire les entrées du clavier. Un objet synchronisé lock ainsi que les méthode notify() et wait() sont utilisés à cette fin.

- **Conception:** Le desing pattern Model View Controller a été utilisé pour pour mettre en oeuvre l'interface graphique du coté client.

- **Trieur de carte:** L'interface générique Comaparable<> a été implémentée à la classe du client "CardClient" afin de trier automatiquement les cartes du joueur.

- **Plus de carte dans le deck:** Si toutes les cartes du deck sont utilisées, on récupère les cartes du talon (sauf la dernière), on les mélange et les remet dans le deck.

[1] Si on utilise le mode graphique, à la toute fin de partie une erreur peut apparaître occasionnellement à la ligne 50 de Map.java, cette erreur ne nuit en rien au jeu puisque la partie est terminée et les scores finaux déjà envoyés. Je n'arrive cependant pas à résoudre cette erreur. (probablement un thread non synchronisé dans la méthode paint() ?)