**E) Aspect fonctionnel de la partie serveur**

1) Communication avec les clients

2) La gestion des jeux

Nous allons expliquer ici le code du serveur

4 Classes sont utilisées pour le déroulement d’une partie :

* PokerPartie : classe principale qui gère le déroulement de la partie (enchères, répartion des jetons, …).
* Jeu : classe permettant de gérer tout ce qui touche aux cartes (paquet, comparaison des mains, …).
* PokerClientThread : classe contenant les paramètres du client (jetons, jetons posés, état, …) et qui permet la communication.
* TimerChoixJoueur : classe permettant d’attendre le choix du joueur pendant 15 secondes.

La partie commence en lançant tournoi() qui lancera déroulement() avec en paramètre l’indice du clients qui doit poser la blind dans la liste clientList.

clientList est composée d’instances de PokerClientThread. Chaque instance représente un client. Cette classe contient différents paramètres, voici ceux qui nous intéresserons pour le déroulement d’une partie :

* ‘int jetons[2]’ : avec en 0 les jetons du joueur et en 1 les jetons posés.
* ‘int cartes[2]’ : avec les 2 cartes de la main du joueur.
* ‘int attente’ : pour savoir l’état du joueur (0 : en jeu, 1 : couché, 2 : tapis, -1 : perdu).
* ‘int joue’ : pour savoir si c’est son tour (0 : non, 1 : oui).
* ‘int pot’ : jetons que le joueur n’a pas le droit de toucher.

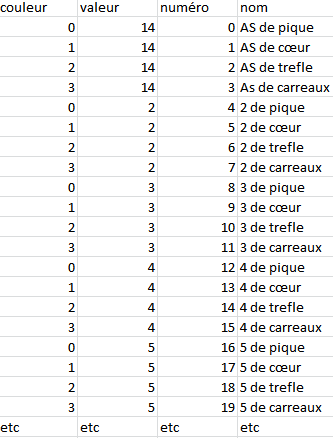
Bien sûr, ce sont des variables de classes privées et ne sont modifiables et utilisables que par des fonctions.

La classe PokerPartie est composé de différentes fonction pour permettre l’envoi des différents messages du protocole utiles au déroulement de la partie.

Au début, de tournoi(), on instancie la classe Jeu pour avoir le paquet de cartes.

Dans Jeu, le paquet de cartes est représenté par une pile d’entier contenant des entiers de 0 à 51 (‘tas’). Chaque entier représente une carte différente. Chaque carte a une valeur et une couleur qui sont des entiers allant respectivement de 2 à 14 et 0 à 3. La valeur et la couleur des cartes sont très rapide d’accès car ses données sont présente dans un tableau d’entiers à deux dimensions ‘vc’ de 52 entrées. Si on veut la couleur de la carte 12 on a donc juste à faire vc[12][1] et on obtient la couleur. De plus, ce tableau n’est initialisé qu’une seule fois par tournoi puisqu’il reste le même et a ce coup est donc négligeable. Des fonctions sont utilisées dans le code pour permettre tout changement rapide de ce système tout en conservant le code des fonctions.

Nous avons donc des cartes que l’on peut représenter comme ceci :



La pile ‘tas’ est mélanger en échangeant chaque entier avec un autre entier du tas parmi les 52 à l’aide de la fonction random() de la librairie Math ce qui nous permet d’avoir un semblant d’aléatoire dans la répartition des cartes.

A chaque fois que l’on veut tirer une carte, il suffit de dépiler un élément de ‘tas’ et on n’a donc aucun doublon de cartes possible. Cette action s’effectue par le biais de tireUneCarte().

Lorsque que deroulement() commence, il faut initialisé les variables des clients en jeu, les cases tu tableau jetonsPose[] et les différentes variables permettant le déroulement du jeu. Ce tableau permet de conservé les jetons posés par chaque client même s’il se déconnecte car c’est nécessaire pour répartir les gains correctement.

On peut remarquer qu’il y a plusieurs fonctions permettant d’effectuer les enchères. Même si elles sont assez similaires au premier coup d’œil, elles sont en fait assez différentes. premièreEnchere() permet d’effectuer la première enchère qui n’a pas de première phase contrairement aux autres et qui doit faire un tour complet de table y compris du joueur qui a mis la surblind alors que c’est le dernier à avoir relancer. Dans une enchère classique on ne retourne pas sur la dernière personne à avoir relancé.

La premierePhase() permet de regarder si quelqu’un relance lorsque l’on vient d’engager un nouveau tour d’enchère et s’il y a relance (ouverture) seulement on lancera enchere() sinon on passera au prochain tour d’enchère.

Lors d’un tour d’enchère on va parcourir les joueurs en jeu (attente==0) et envoyer un message aux clients chacun leur tour pour leur demander de faire un choix d’action. Ces choix son passer, suivre et relancer (représenté respectivement pas un l’entier 1, 2 ou 3). La relance n’est pas toujours possible (3 relance max par tour d’enchère, ouvrir n’est pas une relance), le message contient donc un booléen (‘true’ ou ‘false’) pour indiquer aux clients s’ils doivent proposer ce choix.

Le serveur va se mettre en attente de la réponse du client pendant 15 secondes en instanciant TimerChoixJoueur et en mettant un sémaphore. Si au bout des 15 secondes il n’y a pas de réponse, l’instance lance jouage() avec les paramètres pour passer et donc le joueur se couche (attente=1), ça libère le sémaphore et le programme reprend dans choix(). Si le client envoie une réponse correcte, l’instance du client lance jouage() avec le choix du joueur, ça libère le sémaphore et le programme reprend dans choix() pour appliquer le choix du joueur changer les paramètres et envoyer les modifications aux clients.

Un mauvais choix implique que le joueur se couche. Un mauvais nombre de jetons lors d’une relance est rétabli : s’il y a plus de jetons que ce qu’il possède alors il fait tapis, s’il y en moins ou le même nombre que le minimum pour suivre alors son choix devient suivre.

A la fin de tous les tours d’enchère, viens la répartition des gains. C’est là qu’intervient la variable ‘pot’ des clients. Pour savoir, combien a le droit de toucher chaque joueur, il suffit de faire jetons de la table moins le pot du client.

La fonction repartionDesGains() permet de répartir les gain entre les clients selon leur statut (gagnant, perdant, tapis).

Pour savoir qui sont les gagnants, l’instance Jeu de PokerPartie appelle la fonction gagnant() qui va comparer les mains de chaque client encore en jeu. Pour ce faire, on va regarder quel est la meilleure combinaison de chaque client en regardant dans cet ordre s’il a :

* Quinte flush royale.
* Quinte flush.
* Carré.
* Full.
* Couleur.
* Suite.
* Brelan.
* Deux paires.
* Une paire.

Pour chaque combinaison, on récupère en même temps la meilleure carte de la combinaison. Puis s’il y a égalité, on va comparer le reste des cartes (je rappelle qu’on ne prend en compte que 5 cartes).

Après la répartition des gains, il faut vérifié si la partie est terminée (plus qu’un seul joueur avec des jetons) et si ce n’est pas le cas relancer un tournoi() continue de boucler et relance le premier tour d’enchère en faisant avancer l’indice du client devant poser la blind.

3) La base de données

4) Diagramme de l’application :