Classes et structures de donnés

Player

{

String name ;

Int score ;

Int status ; // 0 = afk, 1 = ready for game

List<Card> cards ;

}

MessageInfo

{

Int code ; // code du message

String description // description du message

}

Team

{

Int id ;

String nom ;

Player player1 ;

Player player2 ;

Int score\_actuel ;

}

Bet

{

Enum Couleur  couleur; // couleur de l’atout parrié

Int point ; // nombre de point parié

Player player ;

Team team;

}

Description du protocol

Connexion :

1. Client> se connecte au serveur sur le port 42442
2. Serveur> envoi un message (class Message )de confirmation de connexion
3. Client> envoi son nom
4. Serveur> envoi confirmation nom valide
5. Client> envoi qu’il est prêt pour une nouvelle partie
6. Serveur> confirme son inscription à la stack d’attente de partie et lui précise des informations (ex : nombre de joueur déjà présent, son équipe)
7. Serveur> envoi la message la partie commence

Jeu :

# Le jeu commence

# préparation de la manche

1. Serveur> envoi liste des cartes du joueur
2. Serveur> envoi message de demande de pari et lui donne des informations (les couleurs déjà pariées par tel joueurs)
3. Client> pari sur une couleur avec infos (score de pari)
4. Serveur> confirme et envoi à tous les joueurs les informations du pari
5. Serveur> valide le paris

# la manche commence

1. Serveur> envoi liste de carte et demande de jouer une carte
2. Client> joue une carte
3. Serveur> valide carte ou rejette et demande de rejouer
4. Serveur> notifie la carte jouée
5. Serveur> notifie équipe gagne la stack et donne les points

# Quand il n’y a plus de carte le jeu fini

1. Serveur> notifie fin de la manche et donne les infos

# seconde manche

# fin du jeu 1000 points

1. Serveur> notifie la fin du jeu et annonce l’équipe gagnante

Packet C# NetworkComms

/\* Expéditeurs : serveur (et peut être client)

\*\* message d’informations sans attente de réponses \*/

Packet00Message

* String Message

/\* Expéditeur : client

\*\* client fait une demande de connexion sous le nom \*/

Packet01LoginRequest

* String name

/\* Expéditeur : serveur

\*\* réponse du serveur concernant la demande de connexion du client \*/

Packet02LoginAnswer

* Boolean accepted

/\* Expéditeur : client

\*\* demande du client de rejoindre une table de jeu\*/

Packet03WaitGameRequest

* Nothing

/\* Expéditeur : server

\*\* le serveur répond à la demande de client lorsqu’il a pu lui attribué une table \*/

Packet04WaitGameAnswer

* Boolean accepted

/\* Expéditeur : serveur

\*\* demande du serveur d’un parie en donnant le dernier parie

\*/

Packet05BetRequest

* String description // description de l’attente du serveur
* String type
* Bet last\_bet ; // si null = pas de bet précedent

/\* Expéditeur : client

\*\* Le client donne son parie

\*/

Packet06BetAnswer

* String type
* Bet newbet

/\* Expéditeur : client

\*\* lorsqu’un client reçoit un Bet Request il peut choisir de renvoyer un coinche pour coincher

\*/

Packet061CoincheAnswer

* String type
* Boolean coinche ;

/\* Expéditeur : serveur

\*\* Lorsqu’un joueur a coincher le serveur envoi un requête de surcoinche a celui-ci

\*/

Packet062SurCoincheRequest

* String description
* String player\_name

/\*Expéditeur : client

\*\* Lorsque qu’un joueur a été coincher il peut surcoincher ou non

\*/

Packet063SurCoincheAcnswer

* Boolean bool  // true = surcoinche, false = pas de surcoinche

/\* Expéditeur : serveur

\*\* Lorsqu’un bet a été accepté le serveur envoi le packet pour informer que le bet est final

\*/

Packet07BetAccepted

* String description
* Bet bet\_final

/\* Expéditeur : serveur

\*\* Le serveur demande au client de jouer une carte

\*/

Packet08PlayingCardRequest

* Deck deck ; // cartes jouées sur la table

/\* Expéditeur : client

\*\* le client joue une carte

\*/

Packet09PlayinCardAnswer

* Card card // carte à jouer