Table

1.0

Generated by Doxygen 1.8.2

Sun Apr 14 2013 15:03:19

Contents

Chapter 1

Projet L3 : Poker

∗Création de la classe Table

Author

LAGNIEZ Jean-Marc

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 Carte Class Reference

classe représentant une carte

`#include "/media/USB20FD/proker/*serveurTable/elementJeu"`

Public Member Functions

- Carte ()

  Constructeur par défaut.
- Carte (const Carte &carte)

  Constructeur par copie.
- Carte (const int &valeur, const int &couleur)

  Constructeur.
- Carte & **operator=** (const Carte &)
- void **setValeur** (const int &)
- void **setCouleur** (const int &)
- int **getValeur** ()
- int **getCouleur** ()
- void **affiche** (ostream &os)

5.1.1 Detailed Description

classe représentant une carte

-

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Carte::Carte ( )

Constructeur par défaut.

Couleur de la carte représentée par un entier Constructeur par défaut de la classe carte

5.1.2.2 Carte::Carte (const Carte & *carte*)

Constructeur par copie.

Constructeur par copie de la classe Carte prenant en paramètre une instance de la classe Carte

Parameters

carte,:	instance de la classe Carte que l'on va copier pour créer une nouvelle carte

5.1.2.3 Carte::Carte (const int & *valeur,* const int & *couleur*)

Constructeur.

Constructeur de la classe Carte prenant en paramètre une valeur et une couleur

Parameters

valeur,:	Valeur que l'on souhaite donner à la carte
couleur,:	Couleur que l'on souhaite donner à la carte

The documentation for this class was generated from the following files:

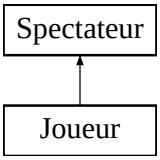
- C:/Users/jm/Desktop/projet/serveurTable/elementJeu/Carte.h
- C:/Users/jm/Desktop/projet/serveurTable/elementJeu/Carte.cpp

5.2 Joueur Class Reference

classe représentant un joueur

`#include "/media/USB20FD/proker/*serveurTable/elementJeu"`

Inheritance diagram for Joueur:



Public Member Functions

- Joueur ()

  Constructeur par défaut.
- Joueur (Spectateur, float nbJetons)

  Constructeur.
- float **getJeton** () const
- Carte **getCartes** (const int &) const
- bool **isFold** () const
- bool **isActif** () const
- bool **isAllIn** () const
- float **getJetonDebutDeManche** () const
- float **getJetonDebutDePartie** () const
- float **getPot** () const
- int **getNbJPot** () const

- void **setJeton** (const float &)
- void **setCartes** (const Carte &, const int &)
- void **setFold** (const bool &)
- void **setActif** (const bool)
- void **setAllIn** (const bool)
- void **setJetonDebutDeManche** (const float)
- void **setJetonDebutDePartie** (const float)
- void **setPot** (const float)
- void **setNbJPot** (const int)
- int **getValeurMain3** ()
- int **getValeurMain4** ()
- int **getValeurMainMax** ()
- void **setValeurMain3** (int)
- void **setValeurMain4** (int)
- void **setValeurMainMax** (int)
- Carte **getCarteMax** ()
- void **setCarteMax** (Carte)
- void **setMainMax3** (int, Carte)
- void **setMainMax4** (int, Carte)
- void **setCartesMax** (int, Carte)
- Carte **getMainMax3** (int)
- Carte **getMainMax4** (int)
- Carte **getCartesMax** (int)
- void affiche (ostream &) const

  Affichage.
- float miseDeCetteManche ()

  Mise totale du joueur sur une manche.
- void nouveauGain (float)

  Nouveau gain.
- void **augmentePot** (float)
- void resetJ ()

  Réinitialisation du Joueur.

5.2.1 Detailed Description

classe représentant un joueur

- La classe gère le déroulement de la partie du côté du joueur

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Joueur::Joueur ( )

Constructeur par défaut.

Constructeur par défaut de la classe Joueur

5.2.2.2 Joueur::Joueur (Spectateur *s*, float *nbJetons*)

Constructeur.

Constructeur à deux paramètres de la classe Joueur

Parameters

Spectateur,:	une instance de la classe Spectateur
nbJetons,:	Le nombre de jetons avec lequel le joueur rentre dans la partie

5.2.3 Member Function Documentation

5.2.3.1 void Joueur::affiche (ostream & *os*) const [virtual]

Affichage.

Permet l'affichage d'un joueur

Reimplemented from Spectateur.

5.2.3.2 float Joueur::miseDeCetteManche ( )

Mise totale du joueur sur une manche.

Somme totale qu'aura misée le joueur à la fin d'une manche

5.2.3.3 void Joueur::nouveauGain (float *f*)

Nouveau gain.

Gain du joueur à la fin d'une manche

5.2.3.4 void Joueur::resetJ ( )

Réinitialisation du Joueur.

Réinitialisation des attributs du joueur en cas de nouvelle partie

The documentation for this class was generated from the following files:

- C:/Users/jm/Desktop/projet/serveurTable/elementJeu/Joueur.h
- C:/Users/jm/Desktop/projet/serveurTable/elementJeu/Joueur.cpp

5.3 Paquet Class Reference

classe représentant un paquet de cartes

`#include "/media/USB20FD/proker/*serveurTable/elementJeu"`

Public Member Functions

- Paquet ()

  Constructeur par défaut.
- Carte sortirCarte ()

  Sortir une carte du paquet.
- void initialiserPaquet ()

  Réinitialisation du paquet.
- int **getTaille** ()

5.3.1 Detailed Description

classe représentant un paquet de cartes

-

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Paquet::Paquet ( )

Constructeur par défaut.

```
vector contenant des instances de la classe Carte
```

représentant les cartes du paquet Constructeur par défaut de la classe Paquet

5.3.3 Member Function Documentation

5.3.3.1 void Paquet::initialiserPaquet ( )

Réinitialisation du paquet.

Le paquet est réinitialisé en cas de nouvelle partie

5.3.3.2 Carte Paquet::sortirCarte ( )

Sortir une carte du paquet.

Méthode qui sort une carte déterminée aléatoirement du paquet, ladite carte est supprimé du vector cartes

The documentation for this class was generated from the following files:

- C:/Users/jm/Desktop/projet/serveurTable/elementJeu/Paquet.h
- C:/Users/jm/Desktop/projet/serveurTable/elementJeu/Paquet.cpp

5.4 Sock Class Reference

Public Member Functions

- **Sock** (int tip, int protocole)
- **Sock** (int tip, const char ∗nomServ, const char ∗protoServ, int protocole=0)
- **Sock** (int tip, short nport, int protocole)
- bool **good** ()
- int **getsDesc** ()
- int **getsRetour** ()
- int **returnPort** ()

The documentation for this class was generated from the following files:

- C:/Users/jm/Desktop/projet/serveurTable/sock/sock.h
- C:/Users/jm/Desktop/projet/serveurTable/sock/sock.cc

5.5 SockDist Class Reference

Public Member Functions

- **SockDist** (const char ∗nomHote, const char ∗nomServ, const char ∗protoServ)
- **SockDist** (const char ∗nomHote, short numPort)
- sockaddr_in ∗ **getAdrDist** ()
- int **getsLen** ()

The documentation for this class was generated from the following files:

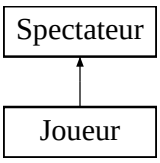
- C:/Users/jm/Desktop/projet/serveurTable/sock/sockdist.h
- C:/Users/jm/Desktop/projet/serveurTable/sock/sockdist.cc

5.6 Spectateur Class Reference

classe représentant un spectateur

`#include "/media/USB20FD/proker/∗serveurTable/elementJeu"`

Inheritance diagram for Spectateur:



Public Member Functions

- Spectateur ()

  Constructeur par défaut.
- Spectateur (int desc)

  Constructeur.
- Spectateur (int desc, string nom)

  Constructeur.
- Spectateur (const Spectateur &spectateur)

  Constructeur par copie.
- int **getDesc** () const
- string **getNom** () const
- void **setDesc** (const int &)
- void **setNom** (const string &)
- virtual void **affiche** (ostream &) const

5.6.1 Detailed Description

classe représentant un spectateur

- La classe gère le déroulement de la partie du côté du spectateur

5.6.2 Constructor & Destructor Documentation

5.6.2.1 Spectateur::Spectateur ( )

Constructeur par défaut.

Nom du spectateur Constructeur par défaut de la classe Spectateur

5.6.2.2 Spectateur::Spectateur (int *desc*)

Constructeur.

Constructeur avec un descripteur de boîte réseau en paramètre

Parameters

desc,:	Descripteur de la boîte réseau pour le spectateur que l'on veut créer

5.6.2.3 Spectateur::Spectateur (int *desc,* string *nom*)

Constructeur.

Constructeur avec un descripteur de boîte réseau en paramètre et le nom qu'on veut donner au spectateur

Parameters

desc,:	Descripteur de la boîte réseau pour le spectateur que l'on veut créer
nom,:	Nom que l'on souhaite donner au spectateur crée

5.6.2.4 Spectateur::Spectateur (const Spectateur & *spectateur*)

Constructeur par copie.

Constructeur par copie avec en paramètre une instance de la classe Spectateur que l'on copie

Parameters

spectateur,:	Instance de la classe Spectateur

The documentation for this class was generated from the following files:

- C:/Users/jm/Desktop/projet/serveurTable/elementJeu/Spectateur.h
- C:/Users/jm/Desktop/projet/serveurTable/elementJeu/Spectateur.cpp

5.7 Table Class Reference

Public Member Functions

- **Table** (int, float)
- Table (int, int, int, float)

  Le constructeur de Table ∗Permet de creer une table de jeu. Elle reçois les messages du serveur avec 'descRecu' et comunique avec le serveur avec 'descEnvoie'. La table a au maximum 'max' joueur et la mise de départ est de 'mise'.
- ∼Table ()

  Le desctructeur de la table ∗Permet de détruire une table de jeu.
- int **getDescTable** () const

- int **getNbJoueur** () const
- int **getnbJMax** () const
- int **getMiseMin** () const
- int **getPot** () const
- Carte ∗ **getMilieu** () const
- Joueur **getJoueur** (int) const
- Spectateur **getSpectateur** (int) const
- void **addSpectateur** ()
- void addJoueur (Spectateur ∗, char ∗)

  Ajoute un nouveau specatateur ∗Permet d'ajouter un Spectateur dans la table des Spectateur.
- void removeSpectateur (int)

  Supprime un spectateur ∗Permet de supprimer un spectateur en place 'place' dans le tableau des spectateurs.
- void removeJoueur (int)

  Supprime un joueur ∗Permet de supprimer un joueur en place 'place' dans le tableau des joueurs. Si le joueur est en cour de partie, il sera considéré comme inactif jusquà la fin de la partie ou jusqu'à une éventuel reconexion. A la suite de la partie, il sera supprimé de la table si il n'est pas reconnecté.
- void **ajouterJoueur** (char ∗)
- void verifierPresenceJoueur ()

  Verifie si un joueur est actif ou non. ∗Permet de supprimer les joueurs non actif pour une nouvelle partie.
- void run ()

  Lance la table.
- void lancerPartie ()

  Lance une partie de jeu.
- void preparerJoueur ()

  Prépare un joueur pour un lancement de partie ∗Permet d erendre tous les joueurs coucher en non coucher, les joueurs All In en non All In.
- void newDealer ()

  Cherche un nouveau dealer ∗Permet de trouver le premier joueur à droite du dealer et de le rendre dealer à son tour.
- int smallBlind ()

  Cherche la petite blind ∗Permet de déterminer qui sera la petite blind de la partie.
- int bigBlind (const int)

  Cherche la grosse blind ∗Permet de déterminer qui sera la grosse blind de la partie.
- void distribuer ()

  Initialise le paquet et lance la méthode donnerCartes() ∗Permet d'initialiser le paquet.
- void donnerCartes ()

  Donne les cartes pour chaque joueur en jeu ∗Permet de distribuer le jeu de chaque joueur pour cette partie.
- void joueurQuitteTable (int)

  Le Joueur sera suprimé de la table des Joueur pour une nouvelle partie ∗Permet qu'un utilisateur puisse qui son état de Joueur pour la prochaine partie qui sera lancé
- bool tourDeMise (int)

  Méthode appelé pour chaque debut de manche jusquà la fin de la manche ∗Permet de savoir quand une manche ou la partie se termine. Si le 'verificateur' est égal à 2 alors c'est la première manche de jeu. Si le 'verificateur' est égal à 1 alors c'est une manche différente de la première.
- void methodeMiser ()

  Attends que le joueur mise ∗Permet au joueur de miser. Si il est trop long, la mise correspondra à une parole ou à suivre.
- int aMiser (const int)

  Détermine le prochain joueur à miser ∗Permet de connaître le nouveau joueur à miser. Il est déterminé par rapport au joueur ayant miser juste avant lui 'previousMiseur'.
- void newManche ()

  Prépare une nouvelle manche.
- void poserFlop (int)

  Pose le flop ∗Permet de connaître le flop et lance la méthode envoyerBoard(placeNext,0)

- void poserTurnRiver (int)

  Pose la turn ou la river ∗Permet de connaitre la turn ou la river et lance la methode envoyerBoard(placeNext,i). i est la position de cette derniere dans le tableau de carte.
- int ObtenirPointMain (Carte main[])

  Fonction qui calcul le nombre de points associés à une main de 5 cartes.
- void EvaluerMain ()

  Fonction qui evalue la main des differents joueurs en essayant tte les combinaisons avec 3 cartes du boards et les deux cartes des joueurs.
- void EvaluerMain2 ()

  Fonction qui evalue la main des differents joueurs en essayant tte les combinaisons avec 4 cartes du boards et une des deux cartes de chaque joueur.
- int EssayerMain3CartesMilieu (int[], int)

  Fonction qui essaye la main donnée.
- int **EssayerMain4CartesMilieu** (int array[], int player)
- void **test** ()
- void SetClassementJoueurs ()

  Fonction qui remplis le classement des joueurs en fonction de leurs nombre de points.
- void RemplissageJeuxMax ()

  Fonction qui remplis les jeuxs maximun de chaque joueurs en fonction des différentes cartes du boards.
- float **distribuerLePot** ()
- void emporteLePot ()

  Remporte la totalité du pot quand il ne reste que un seul joueur.
- void nettoieTable ()

  Réintialise la table.
- void joueurMise (char ∗, int)

  Le joueur fait une action de mise ∗Permet de savoir si le joueur en position 'place' relance,suis ou fold selon les données stocké dans 'miseChar' et de faire une action selon.
- void joueurFold (int)

  Le joueur Fold ∗Permet de mettre le joueur en coucher en position 'place'.
- void joueurAllIn (int)

  Le joueur All In ∗Permet de mettre en tapis en position 'place'.
- void joueurMiseInactif (int)

  Applique une action si un joueur est inactif ∗Si le joueur peut suivre, il suis sinon il se couche.
- float newPotSc ()
- void messageSpec (char ∗, int)

  Envoie 'donnée' de taille 'taille' au spectateurs.
- void convertirTableEnChar (Spectateur ∗)

  Envoie les données de table au nouveau Spectateur 's'.
- void envoieQuitteTable (int)

  Envoie que le joueur en position 'place' quitte la table.
- void redonnerSonJeu (int, Spectateur ∗)

  Un Joueur se reconnecte après une déconextion, il récupère son jeu.
- void lancerPartieReseau (int, int)

  Préviens qu'une nouvelle partie est lancé ∗On envoie la place du dealer, de la petiteBlind et de la grosseBlind.
- void convertirJoueurEnCarte (int)

  Envoie les carte d'un joueur en position 'place'.
- void miserReseau (int, float)

  Envoie la mise sur le reseau ∗Permet d'envoyer la mise 'jeton' du joueur en position 'place' d'un joueur à tous les spéctateurs.
- void **gagnantEtSommeGagne** (int, float)
- void envoyerBoard (int, int)

Envoie les cartes du milieu et le prochain à miser ∗Permet d'envoyer les cartes qui sont au milieu à partir de la position 'positionPremierCarteAEnvoyer' et le prochain à miser en position 'placeNext'.

- void actionSpec (int)

  Un spectateur en position 'place' a envoyé un message.
- void actionSpecJoueur (int)

  Un Spectateur ou un joueur en position 'place' a envoyé un message.
- bool convertirCharDeJoueur (char ∗, Spectateur ∗, int &, float &)

  Un spectateur veut devenir joueur ∗On recupere la place et les jeton de 'message' qui à été envoyé par le Spectateur 's'.
- void modifBDD (int)

  Modifie le n-uplets de la base de donnée du joueur concerné. Il rajoute les jeton du joueur.
- bool ajouterJetonTable (Spectateur ∗, float &)

  Recupere les jetons demandé dans le n-uplets du Spectateur 's' concerné ∗Il vérifie si le nombre de jeton n'est pas trop élevé. Si c'est inférieur, il soustrais le nombre de jeton demandé de la base de donnée.
- void actionServeur ()

  Un message est arrivé dans le tube. On le traite ∗Si le serveur à fermé le tube alors on attends que la partie se termine et on supprime la table.
- void erreur ()

  Une erreur est arrivé. On supprime la table.

5.7.1 Member Function Documentation

5.7.1.1 void Table::addJoueur (Spectateur ∗ *s*, char ∗ *aConvertir*)

Ajoute un nouveau specatateur ∗Permet d'ajouter un Spectateur dans la table des Spectateur.

Ajoute un nouveau joueur ∗Permet de transformer un spectateur en joueur. Le spectateur communique la place et la somme misé. Cette place et cette somme est stocké dans le buffer sommePlace.

5.7.1.2 bool Table::ajouterJetonTable (Spectateur ∗ *s*, float & *jeton*)

Recupere les jetons demandé dans le n-uplets du Spectateur 's' concerné ∗Il vérifie si le nombre de jeton n'est pas trop élevé. Si c'est inférieur, il soustrais le nombre de jeton demandé de la base de donnée.

Returns

true Si la somme demandé n'est pas trop élévé
false Sinon

5.7.1.3 int Table::aMiser (const int *previousMiseur*)

Détermine le prochain joueur à miser ∗Permet de connaître le nouveau joueur à miser. Il est déterminé par rapport au joueur ayant miser juste avant lui 'previousMiseur'.

Returns

placeMiseur Renvoie le joueur devant miser

5.7.1.4 bool Table::convertirCharDeJoueur (char ∗ *aConvertir*, Spectateur ∗ *s*, int & *place*, float & *jeton*)

Un spectateur veut devenir joueur ∗On recupere la place et les jeton de 'message' qui à été envoyé par le Spectateur 's'.

Returns

true Si les jetons demandé sont pas trop élevé par rapport à l'argent de sa base de donnée et si la place n'est pas prise
false Si la place est prise
false Si la demande de jeton est supérieur à l'argent de la base d edonnée

5.7.1.5 int Table::EssayerMain3CartesMilieu (int *array[],* int *player*)

Fonction qui essaye la main donnée.

Parameters

array	tableau de valeur qui va nous permettre de choisir les differentes cartes du board
player	numero du joueur à traité

Returns

le nombre de points correspondant à la main testé.

5.7.1.6 float Table::newPotSc ( )

Returns

potSecondaire Retourne le pot parralèle d'un joueur all in

5.7.1.7 int Table::ObtenirPointMain (Carte *main[]*)

Fonction qui calcul le nombre de points associés à une main de 5 cartes.

Parameters

main	Main à traiter.

Returns

le nombre de points correspondant à la main passé en paramètre.

5.7.1.8 bool Table::tourDeMise (int *verificateur*)

Méthode appelé pour chaque debut de manche jusquà la fin de la manche ∗Permet de savoir quand une manche ou la partie se termine. Si le 'verificateur' est égal à 2 alors c'est la première manche de jeu. Si le 'verificateur' est égal à 1 alors c'est une manche différente de la première.

Returns

true Si il ne reste plus que un joueur en liste
false Sinon

The documentation for this class was generated from the following files:

- C:/Users/jm/Desktop/projet/serveurTable/Table.h
- C:/Users/jm/Desktop/projet/serveurTable/Table.cpp

Chapter 6

File Documentation

6.1 C:/Users/jm/Desktop/projet/serveurTable/elementJeu/Joueur.h File Reference

Header de la classe Carte du point de vue d'une table.

```
#include <math.h>
#include "Spectateur.h"
#include "Carte.h"
```

Classes

- class Joueur

  classe représentant un joueur

Functions

- ostream & **operator**<< (ostream &, Joueur &)

6.1.1 Detailed Description

Header de la classe Carte du point de vue d'une table. Header de la classe Joueur du point de vue d'une table.

Author

Maryan Ludwiczak

Date

10 avril 2013

6.2 C:/Users/jm/Desktop/projet/serveurTable/elementJeu/Paquet.h File Reference

Header de la classe Paquet du point de vue d'une table.

```
#include <vector>
#include <time.h>
#include <iostream>
#include <stdlib.h>
#include "Carte.h"
```

Classes

- class Paquet

  classe représentant un paquet de cartes

6.2.1 Detailed Description

Header de la classe Paquet du point de vue d'une table.

Author

Maryan Ludwiczak

Date

10 avril 2013

6.3 C:/Users/jm/Desktop/projet/serveurTable/elementJeu/Spectateur.h File Reference

Header de la classe Spectateur du point de vue d'une table.

```
#include <iostream>
#include <string.h>
#include <string>
```

Classes

- class Spectateur

  classe représentant un spectateur

Functions

- ostream & **operator**<< (ostream &, Spectateur &)

6.3.1 Detailed Description

Header de la classe Spectateur du point de vue d'une table.

Author

Maryan Ludwiczak

Date

10 avril 2013

Index