Wizard Poker

0.0.1

Généré par Doxygen 1.8.10

Table des matières

Chapitre 1

Index des espaces de nommage

1.1 Liste des espaces de nommage

Liste de tous les espaces de nommage avec une brève description :

Chapitre 2

Index hiérarchique

2.1 Hiérarchie des classes

Cette liste d'héritage est classée approximativement par ordre alphabétique :

Chapitre 3

Index des classes

3.1 Liste des classes

Liste des classes, structures, unions et interfaces avec une brève description :

Chapitre 4

Index des fichiers

4.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

Chapitre 5

Documentation des espaces de nommage

5.1 Référence de l'espace de nommage PacketManager

Fonctions

- void managePacket (Packet : :packet ∗)
- void makeLoginRequest (const char ∗, const char ∗)
- void makeRegistrationRequest (const char ∗, const char ∗)
- void sendDisconnection ()
- void requestCollection ()
- void loginResult (const Packet : :loginResultPacket ∗)
- void collectionResult (const Packet : :collectionListPacket ∗)
- void manageDisconnectRequest (Player ∗, Packet : :packet ∗)
- void manageCollectionRequest (Player ∗, Packet : :packet ∗)
- void managePacket (Player ∗, Packet : :packet ∗)
- void initGame (Player ∗, std : :string)
- void sendCard (Player ∗, Card ∗)
- void setTurn (Player ∗, std : :string)
- void sendInfoStartTurn (Player ∗, int, int)
- void sendAttack (Player ∗, std : :string, int, int, bool, bool, unsigned int)
- void askDefausse (Player ∗, int)
- void sendEndGame (Player ∗, bool)

5.1.1 Documentation des fonctions

5.1.1.1 void PacketManager : :askDefausse (Player ∗ *player,* int *amount*)

5.1.1.2 void PacketManager : :collectionResult (const Packet : :collectionListPacket ∗ *collectionPacket*)

5.1.1.3 void PacketManager : :initGame (Player ∗ *player,* std : :string *ennemyPseudo*)

5.1.1.4 void PacketManager : :loginResult (const Packet : :loginResultPacket ∗ *resultPacket*)

5.1.1.5 void PacketManager : :makeLoginRequest (const char ∗ *pseudo,* const char ∗ *password*)

5.1.1.6 void PacketManager : :makeRegistrationRequest (const char ∗ *pseudo,* const char ∗ *password*)

5.1.1.7 void PacketManager : :manageCollectionRequest (Player ∗ *player,* Packet : :packet ∗ *collectionReqPacket*)

5.1.1.8 void PacketManager : :manageDisconnectRequest (Player ∗ *player,* Packet : :packet ∗ *disconnectReqPacket*)

5.1.1.9 void PacketManager : :managePacket (Packet : :packet ∗ *customPacket*)

5.1.1.10 void PacketManager : :managePacket (Player ∗ *player,* Packet : :packet ∗ *customPacket*)

5.1.1.11 void PacketManager : :requestCollection ( )

5.1.1.12 void PacketManager : :sendAttack (Player * *player,* std : :string *pseudo,* int *cardWichAttack,* int *targetID,* bool *isEffect,* bool *newCard,* unsigned int *finalLife*)

Send attack informations

Paramètres

player	: the players who to send this packet
pseudo	: attacking player's pseudo
targetID	: ID of the target (-1 for player, other for cardMonster)
cardWichAttack	: Card which attack the other
isEffect	: if the attack is an effect (false if it is attack)
newCard	: true if it's a new card wich make attack
finalLife	: final life of the target after attack

5.1.1.13 void PacketManager : :sendCard (Player * *player,* Card * *card*)

5.1.1.14 void PacketManager : :sendDisconnection ( )

5.1.1.15 void PacketManager : :sendEndGame (Player * *player,* bool *victory*)

5.1.1.16 void PacketManager : :sendInfoStartTurn (Player * *player,* int *energy,* int *adverseCardInHand*)

Send informations when the player start he turn

Paramètres

player	: the players who to send this packet
energy	: energy that have the player (it is egal at he max energy)
adverseCardIn↩ Hand	: number of card that the adverse player have

5.1.1.17 void PacketManager : :setTurn (Player * *player,* std : :string *pseudo*)

5.2 Référence de l'espace de nommage WizardLogger

Fonctions

- void initLogger (bool, std : :string)
- void info (std : :string)
- void warning (std : :string)
- void error (std : :string)
- void error (std : :string, std : :string)
- void fatal (std : :string)
- void fatal (std : :string, std : :string)

5.2.1 Documentation des fonctions

5.2.1.1 void WizardLogger : :error (std : :string *message*)

5.2.1.2 void WizardLogger : :error (std : :string *message,* std : :string *ex*)

5.2.1.3 void WizardLogger : :fatal (std : :string *message*)

5.2.1.4 void WizardLogger : :fatal (std : :string *message,* std : :string *ex*)

5.2.1.5 void WizardLogger : :info (std : :string *message*)

5.2.1.6 void WizardLogger : :initLogger (bool *useConsole,* std : :string *logFileName*)

5.2.1.7 void WizardLogger : :warning (std : :string *message*)

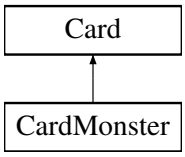
Chapitre 6

Documentation des classes

6.1 Référence de la classe Card

`#include <Card.hpp>`

Graphe d'héritage de Card :



Fonctions membres publiques

- Card (int id, bool isMonster, std : :string name, std : :string description, std : :size_t energy, int HP)
- ∼Card ()=default
- int getId ()
- bool isMonster ()
- std : :string getName ()
- std : :string getDescription ()
- int getEnergyCost ()
- int getMaxHP ()
- int getHP ()
- Card (unsigned int id, std : :string name, unsigned int energy, int effect, bool)
- Card (Card &card)=default
- Card & operator= (const Card &)=default
- virtual void applyEffect (CardMonster &cardmonster, Game &)
- virtual void applyEffect (PlayerInGame &player, Game &)
- virtual ∼Card ()
- unsigned int getId () const
- std : :string getName () const
- unsigned int getEnergyCost ()
- bool gotEffect ()
- int getEffectID ()
- virtual bool isMonster ()
- virtual bool canBeApplyOnCard ()
- virtual bool canBeApplyOnPlayer ()

6.1.1 Description détaillée

One class per card. Contain all information of the card

6.1.2 Documentation des constructeurs et destructeur

6.1.2.1 Card : :Card (int *id,* bool *isMonster,* std : :string *name,* std : :string *description,* std : :size_t *energy,* int *HP*) [inline]

6.1.2.2 Card : :~Card ( ) [default]

6.1.2.3 Card : :Card (unsigned int *id,* std : :string *name,* unsigned int *energy,* int *effect,* bool *save* = true)

Constructor

Paramètres

id	of the card
name	of the card
energy	of the card
effect	name of the specific effect
save	True if save in cache

6.1.2.4 Card : :Card (Card & *card*) [default]

6.1.2.5 virtual Card : :~Card ( ) [virtual]

6.1.3 Documentation des fonctions membres

6.1.3.1 void Card : :applyEffect (CardMonster & *cardmonster,* Game & *game*) [virtual]

Apply the effect on a monster

Paramètres

the	monster targeted
the	game where the effect will be apply

Renvoie

void

6.1.3.2 void Card : :applyEffect (PlayerInGame & *player,* Game & *game*) [virtual]

Apply the effect on a player

Paramètres

the	player targeted
the	game where the effect will be apply

Renvoie

void

6.1.3.3 bool Card : :canBeApplyOnCard ( ) [virtual]

Check if the effect can be apply on a player

Renvoie

true if yes, false if not

6.1.3.4 bool Card : :canBeApplyOnPlayer ( ) [virtual]

Check if the effect can be apply on a player

Renvoie

true if yes, false if not

6.1.3.5 std : :string Card : :getDescription ( ) [inline]

6.1.3.6 int Card : :getEffectID ( )

Return the effect id

Renvoie

-1 if no effect or the effect id

6.1.3.7 int Card : :getEnergyCost ( ) [inline]

6.1.3.8 unsigned int Card : :getEnergyCost ( ) [inline]

6.1.3.9 int Card : :getHP ( ) [inline]

6.1.3.10 int Card : :getId ( ) [inline]

6.1.3.11 unsigned int Card : :getId ( ) const [inline]

6.1.3.12 int Card : :getMaxHP ( ) [inline]

6.1.3.13 std : :string Card : :getName ( ) [inline]

6.1.3.14 std : :string Card : :getName ( ) const [inline]

6.1.3.15 bool Card : :gotEffect ( )

Check if the card got an effect

Renvoie

true if yes, false if not

6.1.3.16 bool Card : :isMonster ( ) [inline]

6.1.3.17 virtual bool Card : :isMonster ( ) [inline], [virtual]

Réimplémentée dans CardMonster.

6.1.3.18 Card& Card : :operator= (const Card &) [default]

La documentation de cette classe a été générée à partir des fichiers suivants :

— client/Card.hpp
— server/Card.cpp

6.2 Référence de la classe CardManager

`#include <CardManager.hpp>`

Fonctions membres publiques statiques

- static Card * getCardById (unsigned int id)
- static void loadAllCards ()
- static Card * chooseCardWin ()

6.2.1 Documentation des fonctions membres

6.2.1.1 Card * CardManager : :chooseCardWin ( ) [static]

Select a random card

Renvoie

Card*

6.2.1.2 Card * CardManager : :getCardById (unsigned int *id*) [static]

6.2.1.3 void CardManager : :loadAllCards ( ) [static]

Create and keep all the cards in a dictionnary witj the id as key

Renvoie

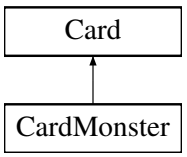
void

La documentation de cette classe a été générée à partir des fichiers suivants :
- server/CardManager.hpp
- server/CardManager.cpp

6.3 Référence de la classe CardMonster

`#include <CardMonster.hpp>`

Graphe d'héritage de CardMonster :



Fonctions membres publiques

- virtual void dealDamage (CardMonster &otherMonster)
- virtual void dealDamage (PlayerInGame &player)
- virtual void incrementTour ()
- CardMonster (unsigned int id, std : :string name, unsigned int energy, int effect, bool, unsigned int life, unsigned int attack, unsigned int nbrTour=0)
- CardMonster (CardMonster &otherMonster)
- virtual ~CardMonster ()
- unsigned int getLife ()
- unsigned int getAttack ()

- unsigned int getMaxLife ()
- unsigned int getNbrTourPose ()
- virtual bool isMonster () override
- bool isTaunt ()
- void setTaunt (bool value)
- void setLife (unsigned int newLife)
- void setAttack (unsigned int newAttack)
- void setMaxLife (unsigned int newMax)
- bool isDead ()

6.3.1 Documentation des constructeurs et destructeur

6.3.1.1 CardMonster : :CardMonster (unsigned int *id,* std : :string *name,* unsigned int *energy,* int *effect,* bool *aBool,* unsigned int *life,* unsigned int *attack,* unsigned int *nbrTour* = 0)

Constructor

Paramètres

id	of the card
name	of the card
energy	of the card
effect	id of the specific effect
save	True if save in cache
life	of the card
attack	of the card
number	of tour that the card is placed

6.3.1.2 CardMonster : :CardMonster (**CardMonster** & *otherMonster*)

6.3.1.3 CardMonster : :~CardMonster ( ) [virtual]

6.3.2 Documentation des fonctions membres

6.3.2.1 void CardMonster : :dealDamage (**CardMonster** & *otherMonster*) [virtual]

The monster attack an other monster

Paramètres

card	(other monster)

Renvoie

void

6.3.2.2 void CardMonster : :dealDamage (**PlayerInGame** & *player*) [virtual]

The monster attack a player

Paramètres

player	

Renvoie

void

6.3.2.3 unsigned int CardMonster : :getAttack ( ) [inline]

6.3.2.4 unsigned int CardMonster : :getLife ( ) [inline]

6.3.2.5 unsigned int CardMonster : :getMaxLife ( ) [inline]

6.3.2.6 unsigned int CardMonster : :getNbrTourPose ( ) [inline]

6.3.2.7 void CardMonster : :incrementTour ( ) [virtual]

Increment the tour of the monster

Renvoie

void

6.3.2.8 bool CardMonster : :isDead ( )

Check if the monster is dead

Renvoie

true if the monster is dead

6.3.2.9 virtual bool CardMonster : :isMonster ( ) [inline], [override], [virtual]

Réimplémentée à partir de Card.

6.3.2.10 bool CardMonster : :isTaunt ( ) [inline]

6.3.2.11 void CardMonster : :setAttack (unsigned int *newAttack*) [inline]

6.3.2.12 void CardMonster : :setLife (unsigned int *newLife*) [inline]

6.3.2.13 void CardMonster : :setMaxLife (unsigned int *newMax*) [inline]

6.3.2.14 void CardMonster : :setTaunt (bool *value*) [inline]

La documentation de cette classe a été générée à partir des fichiers suivants :
- server/CardMonster.hpp
- server/CardMonster.cpp

6.4 Référence de la structure Packet : :carteInfosPacket

`#include <Packet.hpp>`

Attributs publics

- int ID = CARTE_INFO_ID
- int size = sizeof(int)+sizeof(char)∗MAX_CARTE_DESCRITION_SIZE
- int carteID
- char cartesDescription [MAX_CARTE_DESCRITION_SIZE]

6.4.1 Documentation des données membres

6.4.1.1 int Packet : :carteInfosPacket : :carteID

6.4.1.2 char Packet : :carteInfosPacket : :cartesDescription[MAX_CARTE_DESCRITION_SIZE]

6.4.1.3 int Packet : :carteInfosPacket : :ID = CARTE_INFO_ID

6.4.1.4 int Packet : :carteInfosPacket : :size = sizeof(int)+sizeof(char)∗MAX_CARTE_DESCRITION_SIZE

La documentation de cette structure a été générée à partir du fichier suivant :
— common/Packet.hpp

6.5 Référence de la structure Packet : :carteRequestPacket

`#include <Packet.hpp>`

Attributs publics

— int ID = CARTE_REQ_ID
— int size = sizeof(int)
— int carteID

6.5.1 Documentation des données membres

6.5.1.1 int Packet : :carteRequestPacket : :carteID

6.5.1.2 int Packet : :carteRequestPacket : :ID = CARTE_REQ_ID

6.5.1.3 int Packet : :carteRequestPacket : :size = sizeof(int)

La documentation de cette structure a été générée à partir du fichier suivant :
— common/Packet.hpp

6.6 Référence de la classe ChatManager

`#include <ChatManager.hpp>`

Fonctions membres publiques

— sendMessage (Player player, std : :string message)

6.6.1 Documentation des fonctions membres

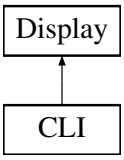
6.6.1.1 ChatManager : :sendMessage (Player *player,* std : :string *message*)

La documentation de cette classe a été générée à partir du fichier suivant :
— server/ChatManager.hpp

6.7 Référence de la classe CLI

`#include <CLI.hpp>`

Graphe d'héritage de CLI :



Fonctions membres publiques

- CLI ()
- ~CLI ()
- void displayFatalError (std : :string)
- void displayLoginPrompt ()
- void displayLoginResult (std : :string)
- void valideLogin ()
- void displayMainWindow ()
- void displayFriendsWindow ()
- void displayCollectionWindow ()
- void displayDeckWindow ()
- void displayWait ()
- void displayGame ()
- void focusTchat ()

6.7.1 Documentation des constructeurs et destructeur

6.7.1.1 CLI : :CLI ( )

6.7.1.2 CLI : :~CLI ( )

6.7.2 Documentation des fonctions membres

6.7.2.1 void CLI : :displayCollectionWindow ( ) [virtual]

Implémente Display.

6.7.2.2 void CLI : :displayDeckWindow ( ) [virtual]

Implémente Display.

6.7.2.3 void CLI : :displayFatalError (std : :string *error*) [virtual]

Implémente Display.

6.7.2.4 void CLI : :displayFriendsWindow ( ) [virtual]

Implémente Display.

6.7.2.5 void CLI : :displayGame ( ) [virtual]

Implémente Display.

6.7.2.6 void CLI : :displayLoginPrompt ( ) [virtual]

Implémente Display.

6.7.2.7 void CLI : :displayLoginResult (std : :string *errorMessage*) [virtual]

Implémente Display.

6.7.2.8 void CLI : :displayMainWindow ( ) [virtual]

Implémente Display.

6.7.2.9 void CLI : :displayWait ( ) [virtual]

Implémente Display.

6.7.2.10 void CLI : :focusTchat ( ) [virtual]

Implémente Display.

6.7.2.11 void CLI : :valideLogin ( ) [virtual]

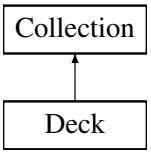
Implémente Display.

La documentation de cette classe a été générée à partir des fichiers suivants :
- client/CLI.hpp
- client/CLI.cpp

6.8 Référence de la classe Collection

`#include <Collection.hpp>`

Graphe d'héritage de Collection :



Fonctions membres publiques

- Collection ()=default
- Collection (std : :vector< Card * >)
- Collection (std : :vector< unsigned >)
- Collection (const Collection &)=default
- Collection & operator= (const Collection &)=default
- virtual Error addCard (Card *)
- virtual Error addCard (int cardId)
- void removeCard (int i)
- void removeCard (Card *)
- void removeCardId (int cardId)
- int indexOfCard (int cardId)
- int getCardIndex (Card *)
- Card * getCardOnIndex (const unsigned index)
- std : :vector< unsigned > getCardsId () const

— virtual ∼Collection ()=default

Attributs protégés

— std : :vector< Card * > _listCard

6.8.1 Documentation des constructeurs et destructeur

6.8.1.1 Collection : :Collection ( ) [default]

6.8.1.2 Collection : :Collection (std : :vector< Card * > *listCard*)

Constructor

Paramètres

listCard	list of card

6.8.1.3 Collection : :Collection (std : :vector< unsigned > *listIdCard*)

Constructor

Paramètres

listIdCard	the list of card ID

6.8.1.4 Collection : :Collection (const Collection &) [default]

6.8.1.5 virtual Collection : :∼Collection ( ) [virtual], [default]

6.8.2 Documentation des fonctions membres

6.8.2.1 Error Collection : :addCard (Card * *card*) [virtual]

Add a Card to the collection

Paramètres

card	the card to add

Renvoie

True if we can add Card (false if there is more than two cards the same)

Réimplémentée dans Deck.

6.8.2.2 Error Collection : :addCard (int *cardId*) [virtual]

Add a Card to the collection

Paramètres

cardId	the id of the card

Renvoie

True if we can add Card (false if there is more than two cards the same)

Réimplémentée dans Deck.

6.8.2.3 int Collection : :getCardIndex (Card ∗ *card*)

Get the index of one card

Paramètres

card	

Renvoie

index or -1 if not found

6.8.2.4 Card ∗ Collection : :getCardOnIndex (const unsigned *index*)

Get the card on the specific index

Paramètres

index	of the card

Renvoie

the Card at the specific index

6.8.2.5 std : :vector< unsigned > Collection : :getCardsId ( ) const

6.8.2.6 int Collection : :indexOfCard (int *cardId*)

6.8.2.7 Collection& Collection : :operator= (const Collection &) [default]

6.8.2.8 void Collection : :removeCard (int *i*)

Remove a Card on a specific position

Paramètres

i	index of the card

6.8.2.9 void Collection : :removeCard (Card ∗ *card*)

Remove a Card from the collection

Paramètres

card	the card to remove

6.8.2.10 void Collection : :removeCardId (int *cardId*)

Remove a Card from the collection

Paramètres

cardId	the id of the card

6.8.3 Documentation des données membres

6.8.3.1 std : :vector<Card*> Collection : :_listCard [protected]

La documentation de cette classe a été générée à partir des fichiers suivants :
- server/Collection.hpp
- server/Collection.cpp

6.9 Référence de la structure Packet : :collectionListPacket

`#include <Packet.hpp>`

Attributs publics

- int ID = COLLECTION_LIST_ID
- int size = sizeof(int)*MAX_CARTES
- int cartesList [MAX_CARTES]

6.9.1 Documentation des données membres

6.9.1.1 int Packet : :collectionListPacket : :cartesList[MAX_CARTES]

6.9.1.2 int Packet : :collectionListPacket : :ID = COLLECTION_LIST_ID

6.9.1.3 int Packet : :collectionListPacket : :size = sizeof(int)*MAX_CARTES

La documentation de cette structure a été générée à partir du fichier suivant :
- common/Packet.hpp

6.10 Référence de la classe Connection

`#include <Connection.hpp>`

Fonctions membres publiques

- Connection (char *)
- ~Connection ()
- void sendPacket (Packet *, size_t)
- Connection ()
- ~Connection ()
- void mainLoop ()

6.10.1 Documentation des constructeurs et destructeur

6.10.1.1 Connection : :Connection (char * *hostName*)

6.10.1.2 Connection : :~Connection ( )

6.10.1.3 Connection : :Connection ( )

6.10.1.4 Connection : :~Connection ( )

6.10.2 Documentation des fonctions membres

6.10.2.1 void Connection : :mainLoop ( )

6.10.2.2 void Connection : :sendPacket (Packet ∗ *packet*, size_t *size*)

La documentation de cette classe a été générée à partir des fichiers suivants :
- client/Connection.hpp
- client/Connection.cpp

6.11 Référence de la structure dataIGPlayer

`#include <PlayerInGame.hpp>`

Attributs publics

- int playerHeal
- int energy
- int maxEnergy
- int limitEnergy
- std : :vector< Card ∗ > cardsInHand
- std : :vector< CardMonster ∗ > cardsPlaced
- bool turn

6.11.1 Documentation des données membres

6.11.1.1 std : :vector<Card∗> dataIGPlayer : :cardsInHand

6.11.1.2 std : :vector<CardMonster∗> dataIGPlayer : :cardsPlaced

6.11.1.3 int dataIGPlayer : :energy

6.11.1.4 int dataIGPlayer : :limitEnergy

6.11.1.5 int dataIGPlayer : :maxEnergy

6.11.1.6 int dataIGPlayer : :playerHeal

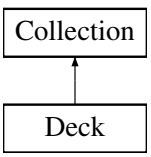
6.11.1.7 bool dataIGPlayer : :turn

La documentation de cette structure a été générée à partir du fichier suivant :
- server/PlayerInGame.hpp

6.12 Référence de la classe Deck

`#include <Deck.hpp>`

Graphe d'héritage de Deck :



Fonctions membres publiques

- Deck (std : :string name, std : :vector< Card ∗ > listCard)

- Deck (std : :string name, std : :vector< unsigned > listCard)
- Deck (const Deck &)
- Deck & operator= (const Deck &)=default
- std : :string getName () const
- bool isValide ()
- Card * pickup ()
- virtual Error addCard (Card *card) override
- virtual Error addCard (int cardId) override
- bool deleteDeck (PlayerInGame *)
- Deck * copyDeck ()
- bool operator== (const std : :string &) const

Fonctions membres publiques statiques

- static Deck * getDeck (std : :string, std : :vector< Deck * >)

Membres hérités additionnels

6.12.1 Description détaillée

A deck is a list of 20 Cards (or less)

6.12.2 Documentation des constructeurs et destructeur

6.12.2.1 Deck : :Deck (std : :string *name,* std : :vector< Card * > *listCard*)

Constructor

Paramètres

name	of the deck
listCard	the must be add on deck

6.12.2.2 Deck : :Deck (std : :string *name,* std : :vector< unsigned > *listCard*)

Constructor

Paramètres

name	of the deck
listCard	the must be add on deck

6.12.2.3 Deck : :Deck (const Deck & *deck*)

Copy Constructor

Paramètres

deck	to copy

6.12.3 Documentation des fonctions membres

6.12.3.1 Error Deck : :addCard (Card * *card*) [override], [virtual]

Adds a card in the deck

Paramètres

card	the card to add

Renvoie

Error and NoError if all is ok

Réimplémentée à partir de Collection.

6.12.3.2 Error Deck : :addCard (int *cardId*) [override], [virtual]

Adds a card in the deck

Paramètres

cardId	the id of the card to add

Renvoie

Error and NoError if all is ok

Réimplémentée à partir de Collection.

6.12.3.3 Deck ∗ Deck : :copyDeck ( )

Copy the deck AND all the card !

Renvoie

the new deck

6.12.3.4 bool Deck : :deleteDeck (PlayerInGame ∗)

6.12.3.5 Deck ∗ Deck : :getDeck (std : :string *name,* std : :vector< Deck ∗ > *listDeck*) [static]

Get the deck with a specific name

Paramètres

name	of the deck
listDeck	to find the deck with the specific name

Renvoie

the deck or nullptr

6.12.3.6 std : :string Deck : :getName ( ) const [inline]

6.12.3.7 bool Deck : :isValide ( )

Checks if the deck is valide. He must have 20 cards

Renvoie

True if the deck is valide

6.12.3.8 Deck& Deck : :operator= (const Deck &) [default]

6.12.3.9 bool Deck : :operator== (const std : :string & *deckName*) const

6.12.3.10 Card * Deck : :pickup ( )

Returns the last card of the deck

Renvoie

the id of the card or -1 if empty

La documentation de cette classe a été générée à partir des fichiers suivants :
- server/Deck.hpp
- server/Deck.cpp

6.13 Référence de la structure Packet : :deckContentPacket

`#include <Packet.hpp>`

Attributs publics

- int ID = DECK_CONT_ID
- int size = sizeof(int)*DECK_SIZE+sizeof(int)
- int deckID
- int cartesList [DECK_SIZE]

6.13.1 Documentation des données membres

6.13.1.1 int Packet : :deckContentPacket : :cartesList[DECK_SIZE]

6.13.1.2 int Packet : :deckContentPacket : :deckID

6.13.1.3 int Packet : :deckContentPacket : :ID = DECK_CONT_ID

6.13.1.4 int Packet : :deckContentPacket : :size = sizeof(int)*DECK_SIZE+sizeof(int)

La documentation de cette structure a été générée à partir du fichier suivant :
- common/Packet.hpp

6.14 Référence de la structure Packet : :deckRequestPacket

`#include <Packet.hpp>`

Attributs publics

- int ID = DECK_REQ_ID
- int size = sizeof(int)
- int deckID

6.14.1 Documentation des données membres

6.14.1.1 int Packet : :deckRequestPacket : :deckID

6.14.1.2 int Packet : :deckRequestPacket : :ID = DECK_REQ_ID

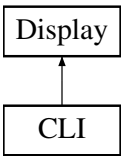
6.14.1.3 int Packet : :deckRequestPacket : :size = sizeof(int)

La documentation de cette structure a été générée à partir du fichier suivant :
— common/Packet.hpp

6.15 Référence de la classe Display

`#include <Display.hpp>`

Graphe d'héritage de Display :



Fonctions membres publiques

— Display ()=default
— virtual ∼Display ()=default
— virtual void displayFatalError (std : :string)=0
— virtual void displayLoginPrompt ()=0
— virtual void displayLoginResult (std : :string)=0
— virtual void valideLogin ()=0
— virtual void displayMainWindow ()=0
— virtual void displayFriendsWindow ()=0
— virtual void displayCollectionWindow ()=0
— virtual void displayDeckWindow ()=0
— virtual void displayWait ()=0
— virtual void displayGame ()=0
— virtual void focusTchat ()=0

6.15.1 Documentation des constructeurs et destructeur

6.15.1.1 Display : :Display ( ) [default]

6.15.1.2 virtual Display : :∼Display ( ) [virtual], [default]

6.15.2 Documentation des fonctions membres

6.15.2.1 virtual void Display : :displayCollectionWindow ( ) [pure virtual]

Implémenté dans CLI.

6.15.2.2 virtual void Display : :displayDeckWindow ( ) [pure virtual]

Implémenté dans CLI.

6.15.2.3 virtual void Display : :displayFatalError (std : :string) [pure virtual]

Implémenté dans CLI.

6.15.2.4 virtual void Display : :displayFriendsWindow ( ) [pure virtual]

Implémenté dans CLI.

6.15.2.5 virtual void Display : :displayGame ( ) [pure virtual]

Implémenté dans CLI.

6.15.2.6 virtual void Display : :displayLoginPrompt ( ) [pure virtual]

Implémenté dans CLI.

6.15.2.7 virtual void Display : :displayLoginResult (std : :string) [pure virtual]

Implémenté dans CLI.

6.15.2.8 virtual void Display : :displayMainWindow ( ) [pure virtual]

Implémenté dans CLI.

6.15.2.9 virtual void Display : :displayWait ( ) [pure virtual]

Implémenté dans CLI.

6.15.2.10 virtual void Display : :focusTchat ( ) [pure virtual]

Implémenté dans CLI.

6.15.2.11 virtual void Display : :valideLogin ( ) [pure virtual]

Implémenté dans CLI.

La documentation de cette classe a été générée à partir du fichier suivant :
— client/Display.hpp

6.16 Référence de la classe Effect

`#include <Effect.hpp>`

Fonctions membres publiques

— Effect ()
— unsigned getId ()
— virtual ∼Effect ()
— virtual void apply (CardMonster ∗, Game ∗)=0
— virtual void apply (PlayerInGame ∗player, Game ∗game)
— virtual bool isTaunt ()
— virtual bool canBeApplyOnPlayer ()
— virtual bool canBeApplyOnCard ()

Fonctions membres publiques statiques

— static void loadAllEffect ()
— static Effect ∗ getEffectByID (unsigned)

6.16.1 Documentation des constructeurs et destructeur

6.16.1.1 Effect : :Effect ( )

6.16.1.2 virtual Effect : :∼Effect ( ) [inline], [virtual]

6.16.2 Documentation des fonctions membres

6.16.2.1 virtual void Effect : :apply (CardMonster ∗ , Game ∗) [pure virtual]

6.16.2.2 virtual void Effect : :apply (PlayerInGame ∗ *player,* Game ∗ *game*) [inline], [virtual]

6.16.2.3 virtual bool Effect : :canBeApplyOnCard ( ) [inline], [virtual]

6.16.2.4 virtual bool Effect : :canBeApplyOnPlayer ( ) [inline], [virtual]

6.16.2.5 Effect ∗ Effect : :getEffectByID (unsigned *id*) [static]

Get the effect that match with the id

Paramètres

id	: the id of an effect

Renvoie

Effect∗

6.16.2.6 unsigned Effect : :getId ( ) [inline]

6.16.2.7 virtual bool Effect : :isTaunt ( ) [inline], [virtual]

6.16.2.8 void Effect : :loadAllEffect ( ) [static]

Create and save all the effects in a vector

Renvoie

void

La documentation de cette classe a été générée à partir des fichiers suivants :
— server/Effect.hpp
— server/Effect.cpp

6.17 Référence de la classe FriendsManager

`#include <FriendsManager.hpp>`

6.17.1 Description détaillée

One class per Player

La documentation de cette classe a été générée à partir du fichier suivant :
— server/FriendsManager.hpp

6.18 Référence de la classe Game

`#include <Game.hpp>`

Fonctions membres publiques

— Game ()
— Game (const Game &)
— Game & operator= (const Game &)=default
— virtual ~Game ()=default
— void checkDeckAndStart ()
— void draw ()
— void draw (PlayerInGame *)
— Error placeCardAffectPlayer (PlayerInGame *, Card *)
— Error placeCard (PlayerInGame *, Card *, CardMonster *)
— Error attackWithCard (PlayerInGame *, CardMonster *, CardMonster *)
— Error attackWithCardAffectPlayer (PlayerInGame *, CardMonster *)
— void nextPlayer ()
— void sendInfoAction (PlayerInGame *, int, int, bool, bool, unsigned)

Fonctions membres publiques statiques

— static void addPlayerWaitGame (Player player)

6.18.1 Description détaillée

One class per game. Contains the two players and all other information for the game

6.18.2 Documentation des constructeurs et destructeur

6.18.2.1 Game : :Game ( )

Default constructor

6.18.2.2 Game : :Game (const Game & *game*)

Copy constructor

Paramètres

game	who must be copied

6.18.2.3 virtual Game : :~Game ( ) `[virtual],[default]`

6.18.3 Documentation des fonctions membres

6.18.3.1 void Game : :addPlayerWaitGame (Player *player*) `[static]`

Adds a player to the waiting list If there is more than one player who is waiting, then it creates a Game

Paramètres

player	the new player waiting

6.18.3.2 Error Game : :attackWithCard (PlayerInGame ∗ *plG,* CardMonster ∗ *card,* CardMonster ∗ *targetCard*)

Funciton when player attack a card

Paramètres

plG	who play
card	which play
targetCard	card which is attack

Renvoie

Error or "NoError" if all is ok

6.18.3.3 Error Game : :attackWithCardAffectPlayer (PlayerInGame ∗ *plG,* CardMonster ∗ *card*)

Funciton when player attack a card

Paramètres

plG	who play
card	which play
targetCard	card which is attack

Renvoie

Error or "NoError" if all is ok

6.18.3.4 void Game : :checkDeckAndStart ( )

Checks if the player have set his deck If all is ok, the game starts

6.18.3.5 void Game : :draw ( )

Current player draw a card

6.18.3.6 void Game : :draw (PlayerInGame ∗ *plG*)

Specific player draw a card

Paramètres

plG	who must draw

6.18.3.7 void Game : :nextPlayer ( )

Switches player turn

6.18.3.8 Game& Game : :operator= (const Game &) [default]

6.18.3.9 Error Game : :placeCard (PlayerInGame * *plG,* Card * *cardPlaced,* CardMonster * *targetCard*)

Function when player place card

Paramètres

plG	player who place the card
cardPlaced	the card the must be place
targetCard	the card which will have the effect if the placed card have it

6.18.3.10 Error Game : :placeCardAffectPlayer (PlayerInGame * *plG,* Card * *cardPlaced*)

Function when player place card and attack player

Paramètres

plG	player who place the card
cardPlaced	the card the must be place
targetPlayer	player who will have the effect if the placed card have it

Renvoie

Error or "NoError" if all ok

6.18.3.11 void Game : :sendInfoAction (PlayerInGame * *plG,* int *cardWichAttack,* int *attackCard,* bool *isEffect,* bool *newCard,* unsigned *heal*)

Send information

Paramètres

plG	who play
cardWichAttack	
attackCard	card which is attack (-1 if player)
heal	of the attack entity

La documentation de cette classe a été générée à partir des fichiers suivants :
- server/Game.hpp
- server/Game.cpp

6.19 Référence de la structure Packet : :loginRequestPacket

`#include <Packet.hpp>`

Attributs publics

- int ID = LOGIN_REQ_ID
- int size = sizeof(char)*MAX_PSEUDO_SIZE*2
- char pseudo [MAX_PSEUDO_SIZE]
- char password [MAX_PSEUDO_SIZE]

6.19.1 Documentation des données membres

6.19.1.1 int Packet : :loginRequestPacket : :ID = LOGIN_REQ_ID

6.19.1.2 char Packet : :loginRequestPacket : :password[MAX_PSEUDO_SIZE]

6.19.1.3 char Packet : :loginRequestPacket : :pseudo[MAX_PSEUDO_SIZE]

6.19.1.4 int Packet : :loginRequestPacket : :size = sizeof(char)∗MAX_PSEUDO_SIZE∗2

La documentation de cette structure a été générée à partir du fichier suivant :
— common/Packet.hpp

6.20 Référence de la structure Packet : :loginResultPacket

`#include <Packet.hpp>`

Attributs publics

— int ID = LOGIN_RES_ID
— int size = sizeof(int)
— int resultCode

6.20.1 Documentation des données membres

6.20.1.1 int Packet : :loginResultPacket : :ID = LOGIN_RES_ID

6.20.1.2 int Packet : :loginResultPacket : :resultCode

6.20.1.3 int Packet : :loginResultPacket : :size = sizeof(int)

La documentation de cette structure a été générée à partir du fichier suivant :
— common/Packet.hpp

6.21 Référence de la classe Packet

`#include <Packet.hpp>`

Classes

— struct carteInfosPacket
— struct carteRequestPacket
— struct collectionListPacket
— struct deckContentPacket
— struct deckRequestPacket
— struct loginRequestPacket
— struct loginResultPacket
— struct packet

Types publics

— enum IDList {
LOGIN_REQ_ID = 11, REGIST_REQ_ID = 12, LOGIN_RES_ID = 13, DISCONNECT_ID = 14,
COLLECTION_REQ_ID = 21, COLLECTION_LIST_ID = 22, DECK_REQ_ID = 23, DECK_CONT_ID = 24,
CARTE_REQ_ID = 25, CARTE_INFO_ID = 26, TCHAT_CONV_REQ_ID = 31, TCHAT_NEW_CONV_ID = 32,
TCHAT_MESSAGE_ID = 33, TCHAT_END_REQ_ID = 34, TCHAT_END_CONV_ID = 35 }

Attributs publics statiques

— static const int packetSize = sizeof(int)∗2
— static const int packetMaxSize = sizeof(int[200])+packetSize

6.21.1 Documentation des énumérations membres

6.21.1.1 enum Packet : :IDList

Valeurs énumérées

LOGIN_REQ_ID

REGIST_REQ_ID

LOGIN_RES_ID

DISCONNECT_ID

COLLECTION_REQ_ID

COLLECTION_LIST_ID

DECK_REQ_ID

DECK_CONT_ID

CARTE_REQ_ID

CARTE_INFO_ID

TCHAT_CONV_REQ_ID

TCHAT_NEW_CONV_ID

TCHAT_MESSAGE_ID

TCHAT_END_REQ_ID

TCHAT_END_CONV_ID

6.21.2 Documentation des données membres

6.21.2.1 const int Packet : :packetMaxSize = sizeof(int[200])+packetSize [static]

6.21.2.2 const int Packet : :packetSize = sizeof(int)∗2 [static]

La documentation de cette classe a été générée à partir du fichier suivant :
— common/Packet.hpp

6.22 Référence de la structure Packet : :packet

`#include <Packet.hpp>`

Attributs publics

— int ID
— int size = 0

6.22.1 Documentation des données membres

6.22.1.1 int Packet : :packet : :ID

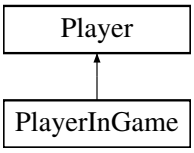
6.22.1.2 int Packet : :packet : :size = 0

La documentation de cette structure a été générée à partir du fichier suivant :
— common/Packet.hpp

6.23 Référence de la classe Player

`#include <Player.hpp>`

Graphe d'héritage de Player :



Fonctions membres publiques

- Player (nlohmann : :json &, int sockfd=0)
- void adjudicateVictory ()
- void adjudicateDefeat ()
- Error addCardCollection (Card ∗c)
- Collection ∗ getCollection ()
- void updateSockfd (int a)
- std : :string getName () const
- std : :string getPass () const
- unsigned getVictories () const
- unsigned getDefeats () const
- Deck ∗ getDeck (std : :string)
- bool removeDeck (Deck ∗)
- void sendPacket (Packet : :packet ∗, size_t)
- void recvLoop ()
- void logout ()
- std : :string serialise () const
- bool operator== (const std : :string &) const
- bool operator< (const Player &) const
- bool operator> (const Player &) const
- virtual ∼Player ()

Fonctions membres protégées

- std : :vector< Deck ∗ > getListDeck ()

Attributs protégés

- unsigned _victories
- unsigned _defeats

Amis

- std : :ostream & operator<< (std : :ostream &, const Player &)
- std : :string & operator<< (std : :string &, const Player &)

6.23.1 Description détaillée

One class per player. This object stocks the socket to communicate with player When the server starts, he must load all Players

6.23.2 Documentation des constructeurs et destructeur

6.23.2.1 Player : :Player (nlohmann : :json & *info,* int *sockfd* = 0)

6.23.2.2 virtual Player : :∼Player ( ) [inline], [virtual]

6.23.3 Documentation des fonctions membres

6.23.3.1 Error Player : :addCardCollection (Card ∗ *c*) [inline]

6.23.3.2 void Player : :adjudicateDefeat ( ) [inline]

6.23.3.3 void Player : :adjudicateVictory ( ) [inline]

6.23.3.4 Collection∗ Player : :getCollection ( ) [inline]

6.23.3.5 Deck ∗ Player : :getDeck (std : :string *deckName*)

Get the deck with this name

Paramètres

deckname	the name of the deck

Renvoie

the deck (or nullptr if not found)

6.23.3.6 unsigned Player : :getDefeats ( ) const [inline]

6.23.3.7 std : :vector<Deck∗> Player : :getListDeck ( ) [inline], [protected]

6.23.3.8 std : :string Player : :getName ( ) const [inline]

6.23.3.9 std : :string Player : :getPass ( ) const [inline]

6.23.3.10 unsigned Player : :getVictories ( ) const [inline]

6.23.3.11 void Player : :logout ( )

6.23.3.12 bool Player : :operator< (const Player & *other*) const

6.23.3.13 bool Player : :operator== (const std : :string & *other_name*) const

6.23.3.14 bool Player : :operator> (const Player & *other*) const

6.23.3.15 void Player : :recvLoop ( )

6.23.3.16 bool Player : :removeDeck (Deck ∗ *deck*)

Remove a Deck

Paramètres

deck	to remove

Renvoie

True if all is ok (cann't delete if one deck)

6.23.3.17 void Player : :sendPacket (Packet : :packet * *packet,* size_t *size*)

6.23.3.18 std : :string Player : :serialise ( ) const

6.23.3.19 void Player : :updateSockfd (int *a*) [inline]

6.23.4 Documentation des fonctions amies et associées

6.23.4.1 std : :ostream& operator<< (std : :ostream & *os,* const Player & *c*) [friend]

6.23.4.2 std : :string& operator<< (std : :string & *str,* const Player & *c*) [friend]

6.23.5 Documentation des données membres

6.23.5.1 unsigned Player : :_defeats [protected]

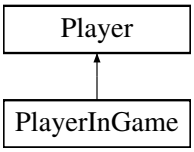
6.23.5.2 unsigned Player : :_victories [protected]

La documentation de cette classe a été générée à partir des fichiers suivants :
- server/Player.hpp
- server/Player.cpp

6.24 Référence de la classe PlayerInGame

`#include <PlayerInGame.hpp>`

Graphe d'héritage de PlayerInGame :



Fonctions membres publiques

- PlayerInGame (const PlayerInGame &)=default
- PlayerInGame & operator= (const PlayerInGame &)=default
- PlayerInGame (const Player &, Game *)
- dataIGPlayer getDataPlayer ()
- std : :vector< CardMonster * > getCardsPlaced ()
- std : :vector< Card * > getCardsInHand ()
- unsigned int nbrCardInHand ()
- void setDeck (Deck *deck)
- bool isDeckDefined ()
- Card * draw ()
- bool haveEnoughEnergy (Card *card)
- void addMaxEnergy ()
- int resetEnergy ()
- void defausseCardPlaced (CardMonster *)
- void placeCard (CardMonster *)
- void takeDamage (unsigned int)
- void getHealed (unsigned int)
- int getHeal ()
- void addDefeat ()
- void addWin ()
- bool isDead ()
- virtual ~PlayerInGame ()=default

Membres hérités additionnels

6.24.1 Documentation des constructeurs et destructeur

6.24.1.1 PlayerInGame : :PlayerInGame (const PlayerInGame &) [default]

6.24.1.2 PlayerInGame : :PlayerInGame (const Player & *player,* Game ∗ *game*)

Creates a PlayerInGame and asks to the player which Deck he would like to play with

6.24.1.3 virtual PlayerInGame : :∼PlayerInGame ( ) [virtual], [default]

6.24.2 Documentation des fonctions membres

6.24.2.1 void PlayerInGame : :addDefeat ( )

Adds a defeat to the player

6.24.2.2 void PlayerInGame : :addMaxEnergy ( )

6.24.2.3 void PlayerInGame : :addWin ( )

Adds a win to the player

6.24.2.4 void PlayerInGame : :defausseCardPlaced (CardMonster ∗ *card*)

6.24.2.5 Card ∗ PlayerInGame : :draw ( )

Gets random card of deck and place it in his hand !

Renvoie

the card or nullptr

6.24.2.6 std : :vector< Card ∗ > PlayerInGame : :getCardsInHand ( )

Returns the cards in hand

6.24.2.7 std : :vector< CardMonster ∗ > PlayerInGame : :getCardsPlaced ( )

Returns the placed cards

6.24.2.8 dataIGPlayer PlayerInGame : :getDataPlayer ( )

Gets data information from this player to send it

Paramètres

isTurn	of the current player

6.24.2.9 int PlayerInGame : :getHeal ( )

6.24.2.10 void PlayerInGame : :getHealed (unsigned int *heal*)

6.24.2.11 bool PlayerInGame : :haveEnoughEnergy (Card * *card*)

6.24.2.12 bool PlayerInGame : :isDead ( )

Checks if the player is Dead

6.24.2.13 bool PlayerInGame : :isDeckDefined ( )

Checks if the deck of this player is defined

Renvoie

True if the deck is defined

6.24.2.14 unsigned PlayerInGame : :nbrCardInHand ( )

6.24.2.15 PlayerInGame& PlayerInGame : :operator= (const PlayerInGame &) [default]

6.24.2.16 void PlayerInGame : :placeCard (CardMonster * *card*)

6.24.2.17 int PlayerInGame : :resetEnergy ( )

Permet de remettre l'énergie au maximum

Renvoie

la nouvelle valeur d'énergie

6.24.2.18 void PlayerInGame : :setDeck (Deck * *deck*)

Sets the player deck and notifies it at the game object

Paramètres

deck	The selected deck

6.24.2.19 void PlayerInGame : :takeDamage (unsigned int *damage*)

La documentation de cette classe a été générée à partir des fichiers suivants :
- server/PlayerInGame.hpp
- server/PlayerInGame.cpp

6.25 Référence de la classe PlayerManager

`#include <PlayerManager.hpp>`

Fonctions membres publiques

- PlayerManager ()=default
- std : :string getRanking ()

- void loadPlayers ()
- void savePlayers () const
- Player * signUp (std : :string, std : :string, int)
- Player * logIn (std : :string, std : :string, int)
- virtual ~PlayerManager ()=default

6.25.1 Documentation des constructeurs et destructeur

6.25.1.1 PlayerManager : :PlayerManager ( ) [default]

6.25.1.2 virtual PlayerManager : :~PlayerManager ( ) [virtual], [default]

6.25.2 Documentation des fonctions membres

6.25.2.1 std : :string PlayerManager : :getRanking ( )

6.25.2.2 void PlayerManager : :loadPlayers ( )

6.25.2.3 Player * PlayerManager : :logIn (std : :string *username,* std : :string *password,* int *sockfd*)

6.25.2.4 void PlayerManager : :savePlayers ( ) const

6.25.2.5 Player * PlayerManager : :signUp (std : :string *username,* std : :string *password,* int *sockfd*)

La documentation de cette classe a été générée à partir des fichiers suivants :

- server/PlayerManager.hpp
- server/PlayerManager.cpp

Chapitre 7

Documentation des fichiers

7.1 Référence du fichier client/Card.hpp

```
#include <string>
```

Classes

— class Card

7.2 Référence du fichier server/Card.hpp

```
#include <string>
#include <map>
#include <cstddef>
#include "Effect.hpp"
```

Classes

— class Card

7.3 Référence du fichier client/CLI.cpp

```
#include "CLI.hpp"
```

7.4 Référence du fichier client/CLI.hpp

```
#include <ncurses.h>
#include <panel.h>
#include <string>
#include <iostream>
#include "common/WizardLogger.hpp"
#include "Display.hpp"
#include "client/CLI/CLIPanel.hpp"
#include "client/CLI/LoginPanel.hpp"
#include "client/CLI/MainPanel.hpp"
#include "client/CLI/TchatPanel.hpp"
#include "client/CLI/FriendPanel.hpp"
#include "client/CLI/CollectionPanel.hpp"
```

Classes

— class CLI

Macros

— #define MAIN_LABEL "F1 : TCHAT F2 : ENVOYER (TCHAT) F3 : VALIDER F10 : QUITTER "
— #define AMIS_LABEL "F1 : TCHAT F2 : ENVOYER (TCHAT) F3 : AJOUTER F4 : RETIRER F10 : RETOUR "
— #define COLL_LABEL "F1 : TCHAT F2 : ENVOYER (TCHAT) F3 : AFFICHER DECK F10 : RETOUR "
— #define DECK_LABEL "F1 : TCHAT F2 : ENVOYER (TCHAT) F3 : CREER F4 : SUPPRIMER F10 : RETOUR "
— #define WAIT_LABEL "F1 : TCHAT F2 : ENVOYER (TCHAT) F10 : ANNULER "
— #define GAME_LABEL "F1 : TCHAT F2 : ENVOYER (TCHAT) F3 : PIOCHER F4 : JOUER F5 : JETER F6 : FIN DE TOUR F10 : ABANDONNER"
— #define DEFAULT_EMPTY_SPACE 2
— #define LINES 37
— #define COLLONNES 96
— #define COMMANDE_PANEL_HEIGTH 1
— #define GAME_HEIGTH 21
— #define GAME_WIDTH 63
— #define CARD_HEIGTH (GAME_HEIGTH%3)
— #define CARD_WIDTH (GAME_WIDTH%7)
— #define PLAYER_INFO_HEIGTH (3+2∗DEFAULT_EMPTY_SPACE)
— #define PLAYER_INFO_WIDTH GAME_WIDTH
— #define TCHAT_INPUT_HEIGTH PLAYER_INFO_HEIGTH
— #define TCHAT_INPUT_WIDTH 27
— #define TCHAT_HEIGTH 14
— #define TCHAT_WIDTH TCHAT_INPUT_WIDTH
— #define CARD_INFO_HEIGTH 5
— #define CARD_INFO_WIDTH TCHAT_INPUT_WIDTH
— #define PANEL_TOTAL_NUMBER 4

7.4.1 Documentation des macros

7.4.1.1 #define AMIS_LABEL "F1 : TCHAT F2 : ENVOYER (TCHAT) F3 : AJOUTER F4 : RETIRER F10 : RETOUR "

7.4.1.2 #define CARD_HEIGTH (GAME_HEIGTH%3)

7.4.1.3 #define CARD_INFO_HEIGTH 5

7.4.1.4 #define CARD_INFO_WIDTH TCHAT_INPUT_WIDTH

7.4.1.5 #define CARD_WIDTH (GAME_WIDTH%7)

7.4.1.6 #define COLL_LABEL "F1 : TCHAT F2 : ENVOYER (TCHAT) F3 : AFFICHER DECK F10 : RETOUR "

7.4.1.7 #define COLLONNES 96

7.4.1.8 #define COMMANDE_PANEL_HEIGTH 1

7.4.1.9 #define DECK_LABEL "F1 : TCHAT F2 : ENVOYER (TCHAT) F3 : CREER F4 : SUPPRIMER F10 : RETOUR "

7.4.1.10 #define DEFAULT_EMPTY_SPACE 2

7.4.1.11 #define GAME_HEIGTH 21

7.4.1.12 #define GAME_LABEL "F1 : TCHAT F2 : ENVOYER (TCHAT) F3 : PIOCHER F4 : JOUER F5 : JETER F6 : FIN DE TOUR F10 : ABANDONNER"

7.4.1.13 #define GAME_WIDTH 63

7.4.1.14 #define LINES 37

7.4.1.15 #define MAIN_LABEL "F1 : TCHAT F2 : ENVOYER (TCHAT) F3 : VALIDER F10 : QUITTER "

in-game layout

cardInfo

gamePanel

tchat

tchat

playersInfo # # input

commandList (key shortcut)

7.4.1.16 #define PANEL_TOTAL_NUMBER 4

7.4.1.17 #define PLAYER_INFO_HEIGTH (3+2∗DEFAULT_EMPTY_SPACE)

7.4.1.18 #define PLAYER_INFO_WIDTH GAME_WIDTH

7.4.1.19 #define TCHAT_HEIGTH 14

7.4.1.20 #define TCHAT_INPUT_HEIGTH PLAYER_INFO_HEIGTH

7.4.1.21 #define TCHAT_INPUT_WIDTH 27

7.4.1.22 #define TCHAT_WIDTH TCHAT_INPUT_WIDTH

7.4.1.23 #define WAIT_LABEL "F1 : TCHAT F2 : ENVOYER (TCHAT) F10 : ANNULER "

7.5 Référence du fichier client/Connection.cpp

```
#include "Connection.hpp"
```

7.6 Référence du fichier server/Connection.cpp

```
#include "Connection.hpp"
```

7.7 Référence du fichier client/Connection.hpp

```
#include <netdb.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/tcp.h>
#include <pthread.h>
#include "common/Packet.hpp"
#include "PacketManager.hpp"
#include "common/WizardLogger.hpp"
#include "Display.hpp"
```

Classes

— class Connection

Macros

— #define PORT 5555

Variables

— Display ∗ display

7.7.1 Documentation des macros

7.7.1.1 #define PORT 5555

7.7.2 Documentation des variables

7.7.2.1 Display∗ display

7.8 Référence du fichier server/Connection.hpp

```
#include <vector>
#include <system_error>
#include <cstdlib>
#include <unistd.h>
#include <cstring>
#include <netdb.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/tcp.h>
#include <pthread.h>
#include "common/Packet.hpp"
#include "PacketManager.hpp"
#include "Player.hpp"
#include "PlayerManager.hpp"
```

Classes

— class Connection

Macros

— #define PORT 5555
— #define BACKLOG 5 /∗ Pending connections the queue will hold ∗/

Variables

— PlayerManager ∗ pm

7.8.1 Documentation des macros

7.8.1.1 #define BACKLOG 5 /∗ Pending connections the queue will hold ∗/

7.8.1.2 #define PORT 5555

7.8.2 Documentation des variables

7.8.2.1 PlayerManager* pm

7.9 Référence du fichier client/Display.hpp

```
#include <string>
```

Classes

— class Display

7.10 Référence du fichier client/main.cpp

```
#include <cstdlib>
#include <iostream>
#include <system_error>
#include "Connection.hpp"
#include "common/WizardLogger.hpp"
#include "Display.hpp"
#include "CLI.hpp"
```

Fonctions

— int main (int argc, char **argv)

Variables

— Connection * conn
— Display * display

7.10.1 Documentation des fonctions

7.10.1.1 int main (int *argc,* char ** *argv*)

7.10.2 Documentation des variables

7.10.2.1 Connection* conn

7.10.2.2 Display* display

7.11 Référence du fichier server/main.cpp

```
#include <cstdlib>
#include <iostream>
#include <exception>
#include "Connection.hpp"
#include "CardManager.hpp"
#include "common/WizardLogger.hpp"
#include "Effect.hpp"
```

Fonctions

— int main ()

Variables

— PlayerManager * pm

7.11.1 Documentation des fonctions

7.11.1.1 int main ( )

7.11.2 Documentation des variables

7.11.2.1 PlayerManager* pm

7.12 Référence du fichier client/PacketManager.cpp

```
#include "PacketManager.hpp"
```

Variables

— Connection * conn

7.12.1 Documentation des variables

7.12.1.1 Connection* conn

7.13 Référence du fichier server/PacketManager.cpp

```
#include "PacketManager.hpp"
```

7.14 Référence du fichier client/PacketManager.hpp

```
#include <string>
#include <vector>
#include "common/Packet.hpp"
#include "common/WizardLogger.hpp"
#include "Connection.hpp"
#include "Display.hpp"
```

Espaces de nommage

— PacketManager

Macros

— #define MAX_CARTES 200

Fonctions

— void PacketManager : :managePacket (Packet : :packet ∗)
— void PacketManager : :makeLoginRequest (const char ∗, const char ∗)
— void PacketManager : :makeRegistrationRequest (const char ∗, const char ∗)
— void PacketManager : :sendDisconnection ()
— void PacketManager : :requestCollection ()
— void PacketManager : :loginResult (const Packet : :loginResultPacket ∗)
— void PacketManager : :collectionResult (const Packet : :collectionListPacket ∗)

Variables

— Display ∗ display

7.14.1 Documentation des macros

7.14.1.1 #define MAX_CARTES 200

7.14.2 Documentation des variables

7.14.2.1 Display∗ display

7.15 Référence du fichier server/PacketManager.hpp

```
#include <string>
#include "common/Packet.hpp"
#include "common/WizardLogger.hpp"
#include "Card.hpp"
#include "Collection.hpp"
#include "Player.hpp"
```

Espaces de nommage

— PacketManager

Fonctions

- void PacketManager : :manageDisconnectRequest (Player ∗, Packet : :packet ∗)
- void PacketManager : :manageCollectionRequest (Player ∗, Packet : :packet ∗)
- void PacketManager : :managePacket (Player ∗, Packet : :packet ∗)
- void PacketManager : :initGame (Player ∗, std : :string)
- void PacketManager : :sendCard (Player ∗, Card ∗)
- void PacketManager : :setTurn (Player ∗, std : :string)
- void PacketManager : :sendInfoStartTurn (Player ∗, int, int)
- void PacketManager : :sendAttack (Player ∗, std : :string, int, int, bool, bool, unsigned int)
- void PacketManager : :askDefausse (Player ∗, int)
- void PacketManager : :sendEndGame (Player ∗, bool)

7.16 Référence du fichier common/Error.hpp

Énumérations

- enum Error {
  NoError = 0, NotEnoughEnergy = 1, NotEnoughPlace = 2, NotHisTurn = 3,
  MustAttackTaunt = 4, UnknowError = 5, NotEffectForPlayer = 6, NotEffectForMonster = 7,
  DeckFull = 8, TwoSameCardMax = 9 }

7.16.1 Documentation du type de l'énumération

7.16.1.1 enum Error

Valeurs énumérées

NoError

NotEnoughEnergy

NotEnoughPlace

NotHisTurn

MustAttackTaunt

UnknowError

NotEffectForPlayer

NotEffectForMonster

DeckFull

TwoSameCardMax

7.17 Référence du fichier common/Packet.hpp

```
#include <string>
#include <list>
```

Classes

- class Packet
- struct Packet : :packet
- struct Packet : :loginRequestPacket
- struct Packet : :loginResultPacket
- struct Packet : :collectionListPacket
- struct Packet : :deckRequestPacket
- struct Packet : :deckContentPacket
- struct Packet : :carteRequestPacket
- struct Packet : :carteInfosPacket

Macros

- #define MAX_PSEUDO_SIZE 30
- #define MAX_CARTES 200
- #define DECK_SIZE 20
- #define MAX_CARTE_DESCRITION_SIZE 120

7.17.1 Documentation des macros

7.17.1.1 #define DECK_SIZE 20

7.17.1.2 #define MAX_CARTE_DESCRITION_SIZE 120

7.17.1.3 #define MAX_CARTES 200

7.17.1.4 #define MAX_PSEUDO_SIZE 30

7.18 Référence du fichier common/WizardLogger.cpp

```
#include "WizardLogger.hpp"
```

7.19 Référence du fichier common/WizardLogger.hpp

```
#include <iostream>
#include <stdexcept>
#include <string>
#include "include/spdlog/spdlog.h"
```

Espaces de nommage

- WizardLogger

Macros

- #define SERVER_LOGFILE "WizardPokerServer"
- #define CLIENT_LOGFILE "WizardPoker"
- #define LOGGER "MainLogger"

Fonctions

- void WizardLogger : :initLogger (bool, std : :string)
- void WizardLogger : :info (std : :string)
- void WizardLogger : :warning (std : :string)
- void WizardLogger : :error (std : :string)
- void WizardLogger : :error (std : :string, std : :string)
- void WizardLogger : :fatal (std : :string)
- void WizardLogger : :fatal (std : :string, std : :string)

7.19.1 Documentation des macros

7.19.1.1 #define CLIENT_LOGFILE "WizardPoker"

7.19.1.2 #define LOGGER "MainLogger"

7.19.1.3 #define SERVER_LOGFILE "WizardPokerServer"

7.20 Référence du fichier server/Card.cpp

```
#include "Card.hpp"
```

7.21 Référence du fichier server/CardManager.cpp

```
#include "CardManager.hpp"
#include "Card.hpp"
#include "CardMonster.hpp"
```

7.22 Référence du fichier server/CardManager.hpp

```
#include <map>
#include "include/json.hpp"
#include <fstream>
#include <time.h>
#include <stdlib.h>
#include "common/WizardLogger.hpp"
```

Classes

— class CardManager

Définitions de type

— using json = nlohmann : :json

7.22.1 Documentation des définitions de type

7.22.1.1 using json = nlohmann : :json

7.23 Référence du fichier server/CardMonster.cpp

```
#include "CardMonster.hpp"
```

7.24 Référence du fichier server/CardMonster.hpp

```
#include "Card.hpp"
#include "Player.hpp"
#include "PlayerInGame.hpp"
```

Classes

— class CardMonster

7.25 Référence du fichier server/ChatManager.hpp

```
#include <string>
```

Classes

— class ChatManager

7.26 Référence du fichier server/Collection.cpp

```
#include "Collection.hpp"
```

7.27 Référence du fichier server/Collection.hpp

```
#include <vector>
#include "common/Error.hpp"
#include "CardManager.hpp"
#include "Card.hpp"
```

Classes

— class Collection

7.28 Référence du fichier server/Deck.cpp

```
#include <algorithm>
#include <stdexcept>
#include "Deck.hpp"
```

7.29 Référence du fichier server/Deck.hpp

```
#include <string>
#include <vector>
#include "Collection.hpp"
#include "common/WizardLogger.hpp"
#include "common/Error.hpp"
```

Classes

— class Deck

Macros

— #define LIMITNAME 20

7.29.1 Documentation des macros

7.29.1.1 #define LIMITNAME 20

7.30 Référence du fichier server/Effect.cpp

```
#include "Effect.hpp"
#include "CardMonster.hpp"
#include "effect/Taunt.hpp"
#include "effect/Heal.hpp"
#include "effect/Damage.hpp"
#include "effect/LifeBlessing.hpp"
#include "effect/LifeCurse.hpp"
#include "effect/AttackBlessing.hpp"
#include "effect/AttackCurse.hpp"
#include "effect/Draw.hpp"
```

7.31 Référence du fichier server/Effect.hpp

```
#include <vector>
```

Classes

— class Effect

7.32 Référence du fichier server/FriendsManager.hpp

```
#include <vector>
```

Classes

— class FriendsManager

7.33 Référence du fichier server/Game.cpp

```
#include "Game.hpp"
```

7.34 Référence du fichier server/Game.hpp

```
#include <queue>
#include <vector>
#include "Player.hpp"
#include "PlayerInGame.hpp"
#include "CardMonster.hpp"
#include "common/WizardLogger.hpp"
#include "PacketManager.hpp"
#include "common/Error.hpp"
```

Classes

— class Game

Énumérations

— enum GameStatut { WAIT_DEC, IN_GAME }

7.34.1 Documentation du type de l'énumération

7.34.1.1 enum GameStatut

Valeurs énumérées

WAIT_DEC

IN_GAME

7.35 Référence du fichier server/Player.cpp

```
#include "include/json.hpp"
#include "Player.hpp"
```

Fonctions

— std : :ostream & operator<< (std : :ostream &os, const Player &c)
— std : :string & operator<< (std : :string &str, const Player &c)

7.35.1 Documentation des fonctions

7.35.1.1 std : :ostream& operator<< (std : :ostream & *os,* const Player & *c*)

7.35.1.2 std : :string& operator<< (std : :string & *str,* const Player & *c*)

7.36 Référence du fichier server/Player.hpp

```
#include <string>
#include <vector>
#include <fstream>
#include <sys/socket.h>
#include "include/json.hpp"
#include "common/Error.hpp"
#include "Collection.hpp"
#include "Deck.hpp"
#include "common/Packet.hpp"
#include "PacketManager.hpp"
#include "common/WizardLogger.hpp"
```

Classes

— class Player

7.37 Référence du fichier server/PlayerInGame.cpp

```
#include "PlayerInGame.hpp"
#include "Card.hpp"
#include "CardMonster.hpp"
```

7.38 Référence du fichier server/PlayerInGame.hpp

```
#include <vector>
#include <algorithm>
#include "Deck.hpp"
#include "Player.hpp"
#include "common/Packet.hpp"
#include "CardManager.hpp"
#include "Game.hpp"
```

Classes

— struct dataIGPlayer
— class PlayerInGame

7.39 Référence du fichier server/PlayerManager.cpp

```
#include "PlayerManager.hpp"
```

7.40 Référence du fichier server/PlayerManager.hpp

```
#include <cstdlib>
#include <iostream>
#include <algorithm>
#include <random>
#include "Player.hpp"
#include "include/json.hpp"
#include "common/Packet.hpp"
```

Classes

— class PlayerManager

Macros

— #define PLAYERS_DB "server/assets/players.json"

7.40.1 Documentation des macros

7.40.1.1 #define PLAYERS_DB "server/assets/players.json"

Index