Crazy Eights Template Classes

Generated by Doxygen 1.12.0

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

Class Documentation

2.1 Card Class Reference

Card class represents a single card when instantiated.

```
#include <template/Card.h>
```

Public Member Functions

- Card (int init_rank, string init_suit)
- int getRank ()
- string getSuit ()
- void setRank (int new_rank)
- void setSuit (string new_suit)
- string toString ()

2.1.1 Detailed Description

Card class represents a single card when instantiated.

2.1.2 Constructor Documentation

2.1.2.1 Card()

```
Card::Card (
            int init_rank,
            string init_suit)
```

Constructor to create a card. Requires an initial Rank and Suit when created.

Parameters

init_rank	An initial rank, as an integer (A=1, 2=2,... J=11, Q=12, K=13)
init_suit	An initial suit, given as a string

2.1.3 Member Function Documentation

2.1.3.1 getRank()

```
int Card::getRank ()
```

Get the card's rank.

Returns

The rank of the card as an integer.

2.1.3.2 getSuit()

```
string Card::getSuit ()
```

Get the card's suit.

Returns

The suit of the card as a string.

2.1.3.3 setRank()

```
void Card::setRank (
            int new_rank)
```

Set a new rank for the card.

Parameters

new_rank	An integer of the new card rank.

2.1.3.4 setSuit()

```
void Card::setSuit (
            string new_suit)
```

Set a new suit for the card.

Parameters

new_suit	A string of the new card suit.

2.1.3.5 toString()

```
string Card::toString ()
```

Get a string representation of the card.

Returns

String representation of the card in the following format: "[rank] of [suit]".

2.2 Deck Class Reference

Deck class represents a collection of cards in a vector format. Intended to represent a deck, however can be used for stock piles, discard piles, and other uses.

`#include <template/Deck.h>`

Public Member Functions

- Deck ()
- int getSize ()
- bool isEmpty ()
- void shuffle ()
- void clear ()
- void fillDeck ()
- void addCard (Card new_card)
- Card peekTopCard ()
- Card peekCard (int index)
- Card getTopCard ()
- Card getCard (int index)

2.2.1 Detailed Description

Deck class represents a collection of cards in a vector format. Intended to represent a deck, however can be used for stock piles, discard piles, and other uses.

2.2.2 Constructor Documentation

2.2.2.1 Deck()

```
Deck::Deck ()
```

Constructor to make a new deck. Pre-fills the deck with ordered cards by default.

2.2.3 Member Function Documentation

2.2.3.1 addCard()

```
void Deck::addCard (
            Card new_card)
```

Adds a new card to the deck. Card is placed at the back of the deck vector.

Parameters

new_card	New card object to be added to the deck.

2.2.3.2 clear()

```
void Deck::clear ()
```

Removes all cards from the deck.

2.2.3.3 fillDeck()

```
void Deck::fillDeck ()
```

Fills the deck with all 52 cards. Does not shuffle the deck.

2.2.3.4 getCard()

```
Card Deck::getCard (
            int index)
```

Removes and returns the card at the given index. Important to note is the card will be removed from the deck.

Parameters

index	The index of the desired card.

Returns

Card object of the card at the given index.

2.2.3.5 getSize()

```
int Deck::getSize ()
```

Gets the size of the deck.

Returns

The number of cards in the deck.

2.2.3.6 getTopCard()

```
Card Deck::getTopCard ()
```

Removes and returns the card at the top of the deck. Will return the last card in the vector. Important to note is the card will be removed from the deck.

Returns

Card object of the top card in the deck.

2.2.3.7 isEmpty()

```
bool Deck::isEmpty ()
```

Returns true/false depending on whether there are any cards in the deck.

Returns

boolean flag of if the deck is empty.

2.2.3.8 peekCard()

```
Card Deck::peekCard (
            int index)
```

Returns the card at the given index. Important to note is the card will not be removed from the deck.

Parameters

index	The index of the desired card.

Returns

Card object of the card at the given index.

2.2.3.9 peekTopCard()

```
Card Deck::peekTopCard ()
```

Returns the card at the top of deck. Will return the last card in the vector. Important to note is the card will not be removed from the deck.

Returns

Card object of the top card in the deck.

2.2.3.10 shuffle()

```
void Deck::shuffle ()
```

Shuffles the deck into a pseudo-random order. Uses the default_random_engine (see https://cplusplus.com/reference/random/default_random_engine/) to set the seed for shuffle algorithm.

2.3 Player Class Reference

Player class represents a single player, either human or bot. Player has a name, a hand, and a score. Class does not get input from terminal directly.

```
#include <template/Player.h>
```

Public Member Functions

- Player ()
- string getName ()
- void setName (string new_name)
- int getHandSize ()
- vector< Card > getHand ()
- void clearHand ()
- bool isEmpty ()
- Card peekCard (int index)
- Card getCard (int index)
- void addCard (Card new_card)
- int sumHand ()
- void outputHand ()
- void changeScoreBy (int toAdd)
- void setScore (int newScore)
- int getScore ()

2.3.1 Detailed Description

Player class represents a single player, either human or bot. Player has a name, a hand, and a score. Class does not get input from terminal directly.

2.3.2 Constructor Documentation

2.3.2.1 Player()

```
Player::Player ()
```

Constructor to create a Player. Sets player name to "Unamed Player", hand starts empty, and score starts at 0.

2.3.3 Member Function Documentation

2.3.3.1 addCard()

```
void Player::addCard (
            Card new_card)
```

Adds a new card to the player's hand. Card is placed at the back of the hand vector.

Parameters

new_card	New card object to be added to the player's hand.

2.3.3.2 changeScoreBy()

```
void Player::changeScoreBy (
            int toAdd)
```

Alter the player's score by given integer.

Parameters

toAdd	Score to add to player's score. Score will be reduced if toAdd is negative.

2.3.3.3 clearHand()

```
void Player::clearHand ()
```

Clears the player's hand. Removes all cards from the player's hand permanently.

2.3.3.4 getCard()

```
Card Player::getCard (
            int index)
```

Removes and returns the card at the given index. Important to note is the card will be removed from the player's hand.

Parameters

index	The index of the desired card.

Returns

Card object of the card at the given index.

2.3.3.5 getHand()

```
vector< Card > Player::getHand ()
```

Gets the entire hand of the player.

Returns

The vector containing Card objects.

2.3.3.6 getHandSize()

```
int Player::getHandSize ()
```

Gets the number of cards in the player's hand.

Returns

the size of hand.

2.3.3.7 getName()

```
string Player::getName ()
```

Gets the player's name.

Returns

The name of the player.

2.3.3.8 getScore()

```
int Player::getScore ()
```

Gets the player's score.

Returns

Player's score, given as an integer.

2.3.3.9 isEmpty()

```
bool Player::isEmpty ()
```

Returns true/false depending on whether there are any cards in the player's hand.

Returns

boolean flag of if the player's hand is empty.

2.3.3.10 outputHand()

```
void Player::outputHand ()
```

Iterates over all card objects in the player's hand and outputs each card's toString to terminal on separate lines.

2.3.3.11 peekCard()

```
Card Player::peekCard (
            int index)
```

Returns the card at the given index. Important to note is the card will not be removed from the player's hand.

Parameters

index	The index of the desired card.

Returns

Card object of the card at the given index.

2.3.3.12 setName()

```
void Player::setName (
            string new_name)
```

Sets the player's name.

Parameters

new_name	The new name of the player, as a string.

2.3.3.13 setScore()

```
void Player::setScore (
            int newScore)
```

Set a new score, overwriting the previous score.

Parameters

newScore	New player score, given as an integer.

2.3.3.14 sumHand()

```
int Player::sumHand ()
```

Calculates and returns the sum of all ranks from cards in the player's hand. Important to note is face cards are interpreted as 11, 12, and 13.

Returns

Sum of all cards in player's hand.

Chapter 3

File Documentation

3.1 Card.h

```
00001 #pragma once
00002 #include <string>
00003
00004 using namespace std;
00005
00011 class Card
00012 {
00013 private:
00017     int rank;
00021     string suit;
00022
00023 public:
00028     Card(int init_rank, string init_suit);
00029
00034     int getRank();
00040     string getSuit();
00041
00046     void setRank(int new_rank);
00052     void setSuit(string new_suit);
00053
00059     string toString();
00060 };
```

3.2 Deck.h

```
00001 #pragma once
00002 #include <vector>
00003 #include <string>
00004 #include "Card.h"
00005
00006 using namespace std;
00007
00012 class Deck
00013 {
00014 private:
00018     vector<Card> deck_vector;
00019
00020 public:
00024     Deck();
00025
00030     int getSize();
00036     bool isEmpty();
00042     void shuffle();
00047     void clear();
00052     void fillDeck();
00053
00058     void addCard(Card new_card);
00064     Card peekTopCard();
00071     Card peekCard(int index);
00077     Card getTopCard();
00084     Card getCard(int index);
00086 };
```

3.3 Player.h

```
00001 #pragma once
00002 #include <iostream>
00003 #include <string>
00004 #include <vector>
00005 #include "Card.h"
00006
00007 using namespace std;
00008
00014 class Player
00015 {
00016 private:
00020     string name;
00024     vector<Card> hand;
00028     int score;
00029
00030 public:
00034     Player();
00035
00039     string getName();
00045     void setName(string new_name);
00050     int getHandSize();
00055     vector<Card> getHand();
00059     void clearHand();
00064     bool isEmpty();
00065
00072     Card peekCard(int index);
00078     Card getCard(int index);
00083     void addCard(Card new_card);
00084
00089     int sumHand();
00093     void outputHand();
00094
00099     void changeScoreBy(int toAdd);
00104     void setScore(int newScore);
00109     int getScore();
00110 };
```