Assignment 3

Generated by Doxygen 1.8.13

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 BoardT Class Reference

Represents the state of a Forty Thieves game.

`#include <GameBoard.h>`

Public Member Functions

- BoardT (std::vector< CardT > deck)

  Constructs a new GameT instance with a given board state.
- bool is_valid_tab_mv (CategoryT c, nat n1, nat n2)

  Checks if a Tableau move is possible.
- bool is_valid_waste_mv (CategoryT c, nat n)

  Checks if its possible to move a card from Waste to Foundation or Tableau.
- bool is_valid_deck_mv ()

  Checks if a deck move of moving card from Deck to Waste is possibe.
- void tab_mv (CategoryT c, nat n1, nat n2)

  Moves Card from one position of tableau to another position of the Tableau or Foundation.
- void waste_mv (CategoryT c, nat n)

  Moves Card from Waste stack to a stack position of the Tableau or Foundation.
- void deck_mv ()

  Moves Card from Deck stack to the Waste stack.
- CardStackT get_tab (nat i)

  Gets stack of cards at position i in Tableau.
- CardStackT get_foundation (nat i)

  Gets stack of cards at position i in Foundation.
- CardStackT get_deck ()

  Gets stack of cards representing the Deck.
- CardStackT get_waste ()

  Gets stack of cards representing the waste.
- bool valid_mv_exists ()

  Checks if a valid move on the gameboard exists.
- bool is_win_state ()

  Checks if the player has won the game.

3.1.1 Detailed Description

Represents the state of a Forty Thieves game.

The board consists of card stacks and sequences of card stacks from which inidvidual cards can be moved depending on the state of other card stacks.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 BoardT()

```
BoardT::BoardT (
            std::vector< CardT > deck )
```

Constructs a new GameT instance with a given board state.

Parameters

deck	Vector sequence of type cardT repreenting the game deck.

Exceptions

invalid_argument	If there aren't two decks used in the game.

3.1.3 Member Function Documentation

3.1.3.1 deck_mv()

```
void BoardT::deck_mv ( )
```

Moves Card from Deck stack to the Waste stack.

Exceptions

invalid_argument	If a valid deck move is not possible(deck empty).

3.1.3.2 get_deck()

```
CardStackT BoardT::get_deck ( )
```

Gets stack of cards representing the Deck.

Returns

Card stack of the Deck.

3.1.3.3 get_foundation()

```
CardStackT BoardT::get_foundation (
            nat i )
```

Gets stack of cards at position i in Foundation.

Parameters

i	Position of the card stack to be returned.

Exceptions

out_of_range	If postion i is not in range of 0 to 7.

Returns

Card stack at position i of the Foundation.

3.1.3.4 get_tab()

```
CardStackT BoardT::get_tab (
            nat i )
```

Gets stack of cards at position i in Tableau.

Parameters

i	Position of the card stack to be returned.

Exceptions

out_of_range	If postion i is not in range of 0 to 9.

Returns

Card stack at position i of the Tableau.

3.1.3.5 get_waste()

```
CardStackT BoardT::get_waste ( )
```

Gets stack of cards representing the waste.

Returns

Card stack of the waste.

3.1.3.6 is_valid_deck_mv()

```
bool BoardT::is_valid_deck_mv ( )
```

Checks if a deck move of moving card from Deck to Waste is possibe.

Returns

Boolean representing if the move is valid if deck size is more than 0.

3.1.3.7 is_valid_tab_mv()

```
bool BoardT::is_valid_tab_mv (
            CategoryT c,
            nat n1,
            nat n2 )
```

Checks if a Tableau move is possible.

Parameters

c	Represents the category of the game to check such as Foundation or Tableau.
n1	Represents card stack of the card being moved.
n2	Position of the card stack to which the card is being moved to.

Exceptions

out_of_range	if position of card stack which the card is in and position of card stack which the card is being moved to is not in in the sequence of cardstacks.

Returns

Boolean representing if the move is valid.

3.1.3.8 is_valid_waste_mv()

```
bool BoardT::is_valid_waste_mv (
            CategoryT c,
            nat n )
```

Checks if its possible to move a card from Waste to Foundation or Tableau.

Parameters

c	Represents the category of the game to check such as Foundation or Tableau.
n	Position of the card stack to which the card is being moved to in Foundation or Tableau.

Exceptions

out_of_range	if position of card stack which the card is in and position of card stack which the card is being moved to is not in in the sequence of cardstacks.
invalid_argument	If Waste stack is empty

Returns

Boolean representing if the move is valid.

3.1.3.9 is_win_state()

```
bool BoardT::is_win_state ( )
```

Checks if the player has won the game.

Checks each of the stacks in Foundation to see if the stacks are more than size 0, and the top card of each stack is of rank King.

Returns

Boolean indicating if the game has been won.

3.1.3.10 tab_mv()

```
void BoardT::tab_mv (
            CategoryT c,
            nat n1,
            nat n2 )
```

Moves Card from one position of tableau to another position of the Tableau or Foundation.

Parameters

c	Represents the category of the game to check such as Foundation or Tableau.
n1	Represents card stack of the card being moved.
n2	Position of the card stack to which the card is being moved to.

Exceptions

out_of_range	if position of card stack which the card is in and position of card stack which the card is being moved to is not in in the sequence of cardstacks.
invalid_argument	If a valid Tableau moves returns False.

3.1.3.11 valid_mv_exists()

```
bool BoardT::valid_mv_exists ( )
```

Checks if a valid move on the gameboard exists.

Checks Foundation, Tableau, and Waste to see if a valid move exists of changing the states of Foundation, Waste, or Tableau.

Returns

Boolean indicating if the game has been won.

3.1.3.12 waste_mv()

```
void BoardT::waste_mv (
            CategoryT c,
            nat n )
```

Moves Card from Waste stack to a stack position of the Tableau or Foundation.

Parameters

c	Represents the category of the game to check such as Foundation or Tableau.
n	Position of Foundation or Tableau stack to which the card is being moved to.

Exceptions

out_of_range	if position of card stack which the card is in and position of card stack which the card is being moved to is not in in the sequence of cardstacks.
invalid_argument	If a valid waste move is not possible.

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

- include/GameBoard.h

3.2 CardT Struct Reference

Structure of a Card.

`#include <CardTypes.h>`

Public Attributes

- SuitT **s**
- RankT **r**

3.2.1 Detailed Description

Structure of a Card.

The documentation for this struct was generated from the following file:

- include/CardTypes.h

3.3 Stack< T > Class Template Reference

ADT for a first-in-last-out data structure using vector.

`#include <Stack.h>`

Public Member Functions

- Stack ()
  Default constructor for Stack. This Stack cannot take any parameters.
- Stack (std::vector< T > s)
  Constructs a new Stack instance with a sequence vector.
- Stack push (T e)
  Adds an item to the top of the stack.
- Stack pop ()
  Removes the element that is at the top of the stack.
- T top ()
  Retrieves the top element on the stack.
- nat size ()
  Returns number of elements in the stack sequence.
- std::vector< T > toSeq ()
  Returns a sequence of the members of this stack.

3.3.1 Detailed Description

template<class T>
class Stack< T >

ADT for a first-in-last-out data structure using vector.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Stack() [1/2]

```
template<class T>
Stack< T >::Stack ( )
```

Default constructor for Stack. This Stack cannot take any parameters.

This is a needed to inialize an empty stacks for gameboard setup.

3.3.2.2 Stack() [2/2]

```
template<class T>
Stack< T >::Stack (
            std::vector< T > s )
```

Constructs a new Stack instance with a sequence vector.

Parameters

s	A vector of type T used to represent the sequence of the stack.

3.3.3 Member Function Documentation

3.3.3.1 pop()

```
template<class T>
Stack Stack< T >::pop ( )
```

Removes the element that is at the top of the stack.

Exceptions

out_of_range	if the stack does not contain any items.

Returns

A new stack with the element at the top removed.

3.3.3.2 push()

```
template<class T>
Stack Stack< T >::push (
            T e )
```

Adds an item to the top of the stack.

Parameters

e	The item of type T to add to the top of the stack.

Returns

A new stack with the added element at the top.

3.3.3.3 size()

```
template<class T>
nat Stack< T >::size ( )
```

Returns number of elements in the stack sequence.

Returns

Size of the stack. Number of elements in the stack.

3.3.3.4 top()

```
template<class T>
T Stack< T >::top ( )
```

Retrieves the top element on the stack.

Exceptions

out_of_range	if the stack does not contain any items.

Returns

The element of type T on the top of the stack.

3.3.3.5 toSeq()

```
template<class T>
std::vector<T> Stack< T >::toSeq ( )
```

Returns a sequence of the members of this stack.

Returns

The sequence fn the stack.

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

- include/Stack.h

Chapter 4

File Documentation

4.1 include/CardStack.h File Reference

Provide definition of a card stack.

```
#include "Stack.h"
```

Typedefs

- typedef Stack< CardT > CardStackT

  Definition of CardStackT representing a stack of type CardT.

4.1.1 Detailed Description

Provide definition of a card stack.

Author

Harsh Patel

4.2 include/CardTypes.h File Reference

Provides type definition and enumerations for a game forty thieves.

Classes

- struct CardT

  Structure of a Card.

Macros

- #define ACE 1

  RankT for an Ace.
- #define JACK 11

  RankT for an Jack.
- #define QUEEN 12

  RankT for a Queen.
- #define KING 13

  RankT for a King.
- #define TOTAL_CARDS 104;

  total cards in the game.

Typedefs

- typedef unsigned short int RankT

  Describes the rank of a card.

Enumerations

- enum SuitT { **Heart**, **Diamond**, **Club**, **Spade** }

  Possible card suits.
- enum CategoryT { **Tableau**, **Foundation**, **Deck**, **Waste** }

  Describes the valid types of columns of cards on the board.

4.2.1 Detailed Description

Provides type definition and enumerations for a game forty thieves.

Author

Harsh Patel

4.3 include/GameBoard.h File Reference

Provides an ADT representing a model for the game of Forty Thieves. including game board state and its transitions.

```
#include "CardStack.h"
#include "CardTypes.h"
#include <vector>
#include <iostream>
```

Classes

- class BoardT

  Represents the state of a Forty Thieves game.

Typedefs

- typedef std::vector< CardStackT > **SeqCrdStckT**

4.3.1 Detailed Description

Provides an ADT representing a model for the game of Forty Thieves. including game board state and its transitions.

Author

Harsh Patel

4.4 include/Stack.h File Reference

Provides an abstract data type (ADT) for a first-in-last-out data structure to store a sequence of a certain type.

```
#include <vector>
#include <iostream>
#include "CardTypes.h"
```

Classes

- class Stack< T >

  ADT for a first-in-last-out data structure using vector.

Typedefs

- typedef unsigned int nat

4.4.1 Detailed Description

Provides an abstract data type (ADT) for a first-in-last-out data structure to store a sequence of a certain type.

Author

Harsh Patel

4.4.2 Typedef Documentation

4.4.2.1 nat

```
typedef unsigned int nat
```

definition of unsigned int to nat (natural number).