Project 3.1

Revision 1

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

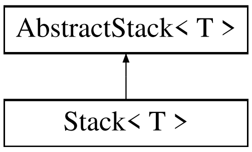
Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 Stack< T > Class Template Reference

Inheritance diagram for Stack< T >:



Public Member Functions

- Stack ()
- virtual ∼Stack ()
- Stack (const Stack &rhs)
- Stack & operator= (Stack rhs) noexcept
- std::size_t size () const
- bool isEmpty () const
- bool push (const T &newItem)
- bool pop ()
- const T & peek () const throw (std::range_error)
- T peekPop () throw (std::range_error)
- void clear ()
- void swap (Stack &lhs, Stack &rhs)

4.1.1 Constructor & Destructor Documentation

4.1.1.1 Stack() `[1/2]`

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

Stack constructor creates an empty stack

4.1.1.2 ∼Stack()

```
template<typename T >
virtual Stack< T >::∼Stack ( )  [virtual]
```

Stack destructor. Must delete any allocated memory.

4.1.1.3 Stack() [2/2]

```
template<typename T >
Stack< T >::Stack (
            const Stack< T > & rhs )
```

This is the copy constructor. It make a copy of the parameter. It is also used by the operator= in the copy-swap paradigm.

Parameters

rhs	- the Stack we are copying during construction

4.1.2 Member Function Documentation

4.1.2.1 clear()

```
template<typename T >
void Stack< T >::clear ( )
```

Deletes all entries on the stack.

Postcondition

Stack contains no items, and the size of the stack is 0.

4.1.2.2 isEmpty()

```
template<typename T >
bool Stack< T >::isEmpty ( ) const
```

Determines whether this stack is empty.

Returns

True if the stack has no items, or false if not.

4.1.2.3 operator=()

```
template<typename T >
Stack& Stack< T >::operator= (
             Stack< T > rhs )  [noexcept]
```

This is the assignment operator. It uses the copy-swap paradigm to create a copy of the parameter

Parameters

rhs	- the Stack we are assigning to this

Returns

a reference to the Stack that was copied into, a.k.a. ∗this

4.1.2.4 peek()

```
template<typename T >
const T& Stack< T >::peek ( ) const throw ( std::range_error)
```

Returns the top item off of the stack without removing it. The stack size stays the same.

Returns

Item of T that was on the top of the stack. Throws an exception of type range_error if the stack is empty.

4.1.2.5 peekPop()

```
template<typename T >
T Stack< T >::peekPop ( ) throw ( std::range_error)
```

Removes the top item of the stack and returns it.

Returns

Item of T that was on the top of the stack. Throws an exception of type range_error if the stack is empty.

4.1.2.6 pop()

```
template<typename T >
bool Stack< T >::pop ( )
```

Pops the top item off of the stack. The stack size is decreased by 1.

Returns

True if successful, or false otherwise.

4.1.2.7 push()

```
template<typename T >
bool Stack< T >::push (
             const T & newItem )
```

Pushes a new entry onto the top of the stack.

Postcondition

If successful, newItem is on the top of the stack.

Parameters

newItem	The item (of datatype T) to be pushed on top of the stack.

Returns

True if insert was successful, or false if not.

4.1.2.8 size()

```
template<typename T >
std::size_t Stack< T >::size ( ) const
```

Returns the number of items on the stack.

Returns

The integer number of items on the stack.

4.1.2.9 swap()

```
template<typename T >
void Stack< T >::swap (
            Stack< T > & lhs,
            Stack< T > & rhs )
```

This is the swap method. It will swap the internals of the two stacks. Notably it is used in the operator= to implement the copy swap paradigm. It is also used by other C++ paradigms.

Parameters

lhs	- the Stack on the left...Left Hand Side (lhs)
rhs	- the Stack on the right...Right Hand Side (rhs)

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

- Stack.hpp

Chapter 5

File Documentation

5.1 Stack.hpp File Reference

```
#include <stdexcept>
#include "abstract_stack.hpp"
#include "Node.hpp"
#include "Stack.txx"
```

Classes

- class Stack< T >

5.1.1 Detailed Description

ADT Stack implementation.

Index