PA01

Generated by Doxygen 1.8.6

Wed Sep 14 2016 20:53:54

Contents

1 Hierarchical Index

1.1 Class Hierarchy

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

2 Class Index

2.1 Class List

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

3 File Index

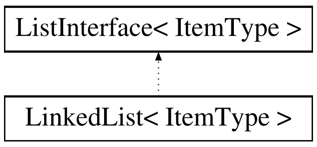
3.1 File List

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

4 Class Documentation

4.1 LinkedList< ItemType > Class Template Reference

Inheritance diagram for LinkedList< ItemType >:



Public Member Functions

- LinkedList ()
- LinkedList (const LinkedList< ItemType > &aList)
- virtual ∼LinkedList ()
- bool isEmpty () const
- int getLength () const
- bool insert (int newPosition, const ItemType &newEntry)
- bool remove (int position)
- void clear ()
- ItemType getEntry (int position) const throw (PrecondViolatedExcept)
- void replace (int position, const ItemType &newEntry) throw (PrecondViolatedExcept)

Private Member Functions

- Node< ItemType > ∗ **getNodeAt** (int position) const

Private Attributes

- Node< ItemType > ∗ **headPtr**
- int **itemCount**

4.1.1 Constructor & Destructor Documentation

4.1.1.1 template<class ItemType > LinkedList< ItemType >::LinkedList ( )

default constructor

4.1.1.2 template<class ItemType > LinkedList< ItemType >::LinkedList (const LinkedList< ItemType > & *aList*)

copy constructor

4.1.1.3 template<class ItemType > LinkedList< ItemType >::∼LinkedList ( ) `[virtual]`

destructor

4.1.2 Member Function Documentation

4.1.2.1 template<class ItemType > void LinkedList< ItemType >::clear ( ) `[virtual]`

removes all items from the list.

Precondition

None.

Postcondition

List contains no entries.

Implements ListInterface< ItemType >.

4.1.2.2 template< class ItemType > ItemType **LinkedList**< ItemType >::getEntry (int *position*) const throw **PrecondViolatedExcept)** [virtual]

Gets the entry at the given position in this list.

Precondition

1 <= position <= getLength().

Postcondition

The desired entry has been returned.

Parameters

position	The list position of the desired entry.

Returns

The entry at the given position.

Exceptions

PrecondViolatedExcept	if position < 1 or position > getLength().

Implements ListInterface< ItemType >.

4.1.2.3 template< class ItemType > int **LinkedList**< ItemType >::getLength ( ) const [virtual]

checks how many items are in the list

Returns

the integer value of how many items are contained in the list.

Implements ListInterface< ItemType >.

4.1.2.4 template< class ItemType > bool **LinkedList**< ItemType >::insert (int *newPosition,* const ItemType & *newEntry*) [virtual]

inserts an entry into the list at a given position

Precondition

None.

Postcondition

if the position is valid and insertion is possible a new entry is entered into the list.

Parameters

newPosition	the position in the list at which to insert the new entry.
newEntry	the new item to be placed in the list.

Returns

True if the item was successfully placed in the list.

Implements ListInterface< ItemType >.

4.1.2.5 template<class ItemType > bool **LinkedList**< ItemType >::isEmpty ( ) const [virtual]

checks if the list contains any items

Returns

returns true if the list is empty.

Implements ListInterface< ItemType >.

4.1.2.6 template<class ItemType > bool **LinkedList**< ItemType >::remove (int *position*) [virtual]

removes the entry at the specified position.

Precondition

None.

Postcondition

if the position is valid the item is removed from the list and the list is renumbered.

Parameters

position	the position in the list which contains the item to be removed.

Returns

True if the item was removed succesfully otherwise returns false.

Implements ListInterface< ItemType >.

4.1.2.7 template<class ItemType > void **LinkedList**< ItemType >::replace (int *position*, const ItemType & *newEntry*) throw **PrecondViolatedExcept)** [virtual]

Replaces the entry at the given position in this list.

Precondition

1 <= position <= getLength().

Postcondition

The entry at the given position is newEntry.

Parameters

position	The list position of the entry to replace.
newEntry	The replacement entry.

Exceptions

PrecondViolatedExcept	if position < 1 or position > getLength().

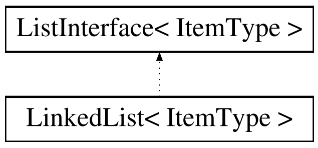
Implements ListInterface< ItemType >.

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

- LinkedList.h
- LinkedList.cpp

4.2 ListInterface< ItemType > Class Template Reference

Inheritance diagram for ListInterface< ItemType >:



Public Member Functions

- virtual bool isEmpty () const =0
- virtual int getLength () const =0
- virtual bool insert (int newPosition, const ItemType &newEntry)=0
- virtual bool remove (int position)=0
- virtual void clear ()=0
- virtual ItemType getEntry (int position) const =0
- virtual void replace (int position, const ItemType &newEntry)=0

4.2.1 Member Function Documentation

4.2.1.1 template<class ItemType > virtual void **ListInterface**< ItemType >::clear ( ) `[pure virtual]`

Removes all entries from this list.

Postcondition

List contains no entries and the count of items is 0.

Implemented in LinkedList< ItemType >.

4.2.1.2 template<class ItemType > virtual ItemType **ListInterface**< ItemType >::getEntry (int *position*) const `[pure virtual]`

Gets the entry at the given position in this list.

Precondition

1 <= position <= getLength().

Postcondition

The desired entry has been returned.

Parameters

position	The list position of the desired entry.

Returns

The entry at the given position.

Implemented in LinkedList< ItemType >.

4.2.1.3 template<class ItemType > virtual int ListInterface< ItemType >::getLength ( ) const [pure virtual]

Gets the current number of entries in this list.

Returns

The integer number of entries currently in the list.

Implemented in LinkedList< ItemType >.

4.2.1.4 template<class ItemType > virtual bool ListInterface< ItemType >::insert (int *newPosition,* const ItemType & *newEntry*) [pure virtual]

Inserts an entry into this list at a given position.

Precondition

None.

Postcondition

If 1 <= position <= getLength() + 1 and the insertion is successful, newEntry is at the given position in the list, other entries are renumbered accordingly, and the returned value is true.

Parameters

newPosition	The list position at which to insert newEntry.
newEntry	The entry to insert into the list.

Returns

True if insertion is successful, or false if not.

Implemented in LinkedList< ItemType >.

4.2.1.5 template<class ItemType > virtual bool ListInterface< ItemType >::isEmpty ( ) const [pure virtual]

Sees whether this list is empty.

Returns

True if the list is empty; otherwise returns false.

Implemented in LinkedList< ItemType >.

4.2.1.6 template<class ItemType > virtual bool ListInterface< ItemType >::remove (int *position*) [pure virtual]

Removes the entry at a given position from this list.

Precondition

None.

Postcondition

If 1 <= position <= getLength() and the removal is successful, the entry at the given position in the list is removed, other items are renumbered accordingly, and the returned value is true.

Parameters

position	The list position of the entry to remove.

Returns

True if removal is successful, or false if not.

Implemented in LinkedList< ItemType >.

4.2.1.7 template<class ItemType > virtual void **ListInterface**< ItemType >::replace (int *position,* const ItemType & *newEntry*) [pure virtual]

Replaces the entry at the given position in this list.

Precondition

1 <= position <= getLength().

Postcondition

The entry at the given position is newEntry.

Parameters

position	The list position of the entry to replace.
newEntry	The replacement entry.

Implemented in LinkedList< ItemType >.

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

- ListInterface.h

4.3 Node< ItemType > Class Template Reference

Public Member Functions

- Node ()
- Node (const ItemType &anItem)
- Node (const ItemType &anItem, Node< ItemType > ∗nextNodePtr)
- void setItem (const ItemType &anItem)
- void setNext (Node< ItemType > ∗nextNodePtr)
- ItemType getItem () const
- Node< ItemType > ∗ getNext () const

Private Attributes

- ItemType **item**
- Node< ItemType > ∗ **next**

4.3.1 Constructor & Destructor Documentation

4.3.1.1 template<class ItemType > **Node**< ItemType >::**Node** ( )

default constructor

4.3.1.2 template< class ItemType > Node< ItemType >::Node (const ItemType & *anItem*)

constructor with item value

4.3.1.3 template< class ItemType > Node< ItemType >::Node (const ItemType & *anItem,* Node< ItemType > ∗ *nextNodePtr*)

constructor with item value and next pointer

4.3.2 Member Function Documentation

4.3.2.1 template< class ItemType > ItemType Node< ItemType >::getItem ( ) const

returns the item stored in the node

4.3.2.2 template< class ItemType > Node< ItemType > ∗ Node< ItemType >::getNext ( ) const

returns the next node

4.3.2.3 template< class ItemType > void Node< ItemType >::setItem (const ItemType & *anItem*)

sets the item value of the node

4.3.2.4 template< class ItemType > void Node< ItemType >::setNext (Node< ItemType > ∗ *nextNodePtr*)

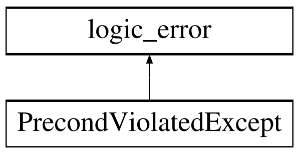
sets the pointer to the next node

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

- Node.h
- Node.cpp

4.4 PrecondViolatedExcept Class Reference

Inheritance diagram for PrecondViolatedExcept:



Public Member Functions

- **PrecondViolatedExcept** (const std::string &message="")

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

- PrecondViolatedExcept.h
- PrecondViolatedExcept.cpp

5 File Documentation

5.1 LinkedList.cpp File Reference

```
#include "LinkedList.h"
#include "Node.h"
#include "PrecondViolatedExcept.h"
```

5.1.1 Detailed Description

ADT list: Link-based implementation.

5.2 LinkedList.h File Reference

Header file for a linked list.

```
#include "ListInterface.h"
#include "Node.h"
#include "PrecondViolatedExcept.h"
#include "LinkedList.cpp"
```

Classes

- class LinkedList< ItemType >

5.2.1 Detailed Description

Header file for a linked list. ADT list: Link-based implementation. Listing 9-2.

establishes functions for list Created by Frank M. Carrano and Timothy M. Henry. Copyright (c) 2017 Pearson Education, Hoboken, New Jersey.

5.3 ListInterface.h File Reference

Interface file for the List ADT.

Classes

- class ListInterface< ItemType >

5.3.1 Detailed Description

Interface file for the List ADT.

Author

Rory Pierce

Specifies the implementation contract of the List ADT

Version

0.10

Adapted from Frank M. Carrano and Timothy M. Henry Copyright (c) 2017 Pearson Education, Hoboken, New Jersey.

5.4 Node.cpp File Reference

```
#include "Node.h"
```

5.4.1 Detailed Description

Listing 4-2

5.5 Node.h File Reference

```
#include "Node.cpp"
```

Classes

- class Node< ItemType >

5.5.1 Detailed Description

Listing 4-1

5.6 PrecondViolatedExcept.cpp File Reference

```
#include "PrecondViolatedExcept.h"
```

5.6.1 Detailed Description

Listing 7-6.

5.7 PrecondViolatedExcept.h File Reference

```
#include <stdexcept>
#include <string>
```

Classes

- class PrecondViolatedExcept

5.7.1 Detailed Description

Listing 7-5.

Index