PA01 - Linked List

Generated by Doxygen 1.8.12

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

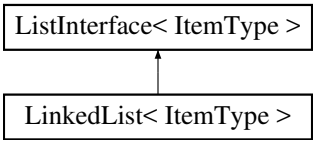
3.1 File List

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** (const LinkedList< ItemType > &aList)
- 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
- Node< ItemType > * **insertNode** (int position, Node< ItemType > *newNodePtr, Node< ItemType > *subChainPtr)

Private Attributes

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

4.1.1 Member Function Documentation

4.1.1.1 clear()

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

Removes all entries from this list.

Postcondition

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

Implements ListInterface< ItemType >.

4.1.1.2 getEntry()

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

Exceptions

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

Implements ListInterface< ItemType >.

4.1.1.3 getLength()

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

Gets the current number of entries in this list.

Returns

The integer number of entries currently in the list.

Implements ListInterface< ItemType >.

4.1.1.4 insert()

```
template<class ItemType >
bool LinkedList< ItemType >::insert (
            int newPosition,
            const ItemType & newEntry )  [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.

Implements ListInterface< ItemType >.

4.1.1.5 isEmpty()

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

Sees whether this list is empty.

Returns

True if the list is empty; otherwise returns false.

Implements ListInterface< ItemType >.

4.1.1.6 remove()

```
template<class ItemType >
bool LinkedList< ItemType >::remove (
            int position )  [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.

Implements ListInterface< ItemType >.

4.1.1.7 replace()

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

Exceptions

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

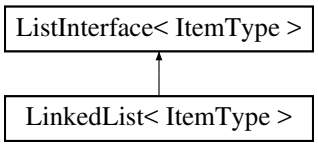
Implements ListInterface< ItemType >.

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 clear()

```
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 getEntry()

```
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 getLength()

```
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 insert()

```
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 isEmpty()

```
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 remove()

```
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 replace()

```
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 ()
  Default constructor for the class.
- Node (const ItemType &anItem)
  Copy constructor for the class.
- Node (const ItemType &anItem, Node< ItemType > *nextNodePtr)
  Constructor for the class.
- void setItem (const ItemType &anItem)
  setItem function
- void setNext (Node< ItemType > *nextNodePtr)
  setNext function
- ItemType getItem () const
  getItem function
- Node< ItemType > * getNext () const
  getNext function

Private Attributes

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

4.3.1 Constructor & Destructor Documentation

4.3.1.1 Node() [1/2]

```
template<class ItemType >
Node< ItemType >::Node ( )
```

Default constructor for the class.

constructs the class

4.3.1.2 Node() [2/2]

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

Copy constructor for the class.

constructs the class using a previously constructed reference

4.3.2 Member Function Documentation

4.3.2.1 getItem()

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

getItem function

returns current item

4.3.2.2 getNext()

```
template<class ItemType >
Node< ItemType > * Node< ItemType >::getNext ( ) const
```

getNext function

returns next item

4.3.2.3 setItem()

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

setItem function

Sets the current item to another item

4.3.2.4 setNext()

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

setNext function

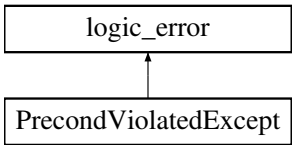
Sets the next item to an item

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

Implementation file for the Linked List ADT.

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

5.1.1 Detailed Description

Implementation file for the Linked List ADT.

Author

Someone at Pearson (I didn't code any of this)

Specifies the functions of the linked list data type

Version

0.10

5.2 LinkedList.h File Reference

Header file for the Linked List ADT.

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

Classes

- class LinkedList< ItemType >

5.2.1 Detailed Description

Header file for the Linked List ADT.

Author

Someone at Pearson (I didn't code any of this)

Specifies the members of the Linked list ADT

Version

0.10

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

Implementation file for the Node class.

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

5.4.1 Detailed Description

Implementation file for the Node class.

Author

Someone at Pearson (I didn't code any of this)

Specifies the functions of the Node class

Version

0.10

5.5 Node.h File Reference

Header file for the Node class.

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

Classes

- class Node< ItemType >

5.5.1 Detailed Description

Header file for the Node class.

Author

Someone at Pearson (I didn't code any of this)

Specifies the members of the Node class and defines function parameters

Version

0.10

5.6 PA01.cpp File Reference

Main driver for this project.

```
#include <iostream>
#include "LinkedList.h"
#include "ListInterface.h"
#include "Node.h"
#include "PrecondViolatedExcept.h"
```

Functions

- int **main** ()

5.6.1 Detailed Description

Main driver for this project.

Author

Willis T. Allstead

Runs some basic tests on the project as a whole

Version

0.10

5.7 PrecondViolatedExcept.cpp File Reference

Implementation file for the PrecondViolatedExcept class.

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

5.7.1 Detailed Description

Implementation file for the PrecondViolatedExcept class.

Author

Someone at Pearson (I didn't code any of this)

Specifies function of the class.

Version

0.10

5.8 PrecondViolatedExcept.h File Reference

Header file for the PrecondViolatedExcept class.

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

Classes

- class PrecondViolatedExcept

5.8.1 Detailed Description

Header file for the PrecondViolatedExcept class.

Author

Someone at Pearson (I didn't code any of this)

Specifies the members of the Node class and defines function parameters

Version

0.10

Index