PA01 - Linked List

Generated by Doxygen 1.8.6

Sun Sep 18 2016 18:45:01

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

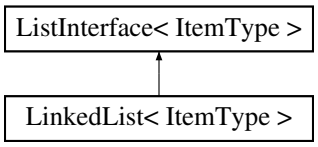
3.1 File List

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 (PrecondViolatedExcep)
- void replace (int position, const ItemType &newEntry) throw (PrecondViolatedExcep)

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 for the LinkedList class

Precondition

LinkedList object does not exist

Postcondition

The LinkedList object has been created with default values

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

Copy constructor for the LinkedList class

Precondition

Two LinkedList object exists

Postcondition

The contents of the parameter LinkedList is copied into the parent class.

Parameters

aList	The LinkedList to be copied

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

Deconstructor for the LinkedList class

Precondition

LinkedList object is in memory

Postcondition

LinkedList object is removed from memory

4.1.2 Member Function Documentation

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

Remove first object from the list until the list is empty

Precondition

LinkedList is in its present state

Postcondition

all objects in the list are removed

Implements ListInterface< ItemType >.

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

Find the entry at the point specified in the list

Precondition

The entry at a certain point in the list is unknown

Postcondition

The entry at a certain point in the list is known

Parameters

position	Where in the list the entry should be returned

Returns

The entry at the position in the LinkedList

Exceptions

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

Implements ListInterface< ItemType >.

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

Find the number of objects in the LinkedList

Precondition

Number of objects in LinkedList is unknown

Postcondition

Number of objects in LinkedList is known

Returns

An int representing number of objects in the LinkedList

Implements ListInterface< ItemType >.

4.1.2.4 template<class ItemType > Node< ItemType > * LinkedList< ItemType >::getNodeAt (int *position*) const [private]

Locates a specified node in a linked list.

Precondition

position is the number of the desired node; position >= 1 and position <= itemCount.

Postcondition

The node is found and a pointer to it is returned.

Parameters

position	The number of the node to locate.

Returns

A pointer to the node at the given position.

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

Insert an entry to the LinkedList at the specified position

Precondition

LinkedList is in its present state

Postcondition

LinkedList has an entry inserted at the position provided by the parameter

Parameters

newPosition	Where in the list the entry should be inserted
newEntry	The item to be inserted into the list

Returns

A bool specifying what if the insertion was successful

Implements ListInterface< ItemType >.

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

Checks if the LinkedList is empty

Precondition

Unknown if state of LinkedList is empty

Postcondition

State of LinkedList is known

Returns

A bool determining if LinkedList is empty

Implements ListInterface< ItemType >.

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

Remove an entry in the LinkedList at the specified position

Precondition

LinkedList is in its present state

Postcondition

Entry at the specified position has been removed from the list

Parameters

position	Where in the list the entry should be removed

Returns

A bool specifying if the removal was successful

Implements ListInterface< ItemType >.

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

Replaces the entry at a given point

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

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

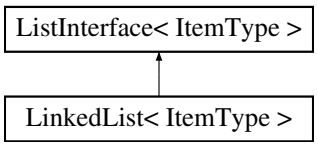
Implements ListInterface< ItemType >.

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 for the Node class

Postcondition

Node class object is created

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

Constructor for the Node class (with entry as parameter input)

Postcondition

Contents of Node are copied from node parameter

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

Constructor for the Node class (with entry and pointer to next Node as parameter input)

Postcondition

Contents of Node are copied from parameter node, including next pointer.

4.3.2 Member Function Documentation

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

Retrieve the data in the node

Returns

the data item was storing

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

Find the node the next pointer is pointing at

Returns

A pointer pointing to the next node

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

Set item in node to input

Postcondition

item in node has been set to input

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

Set the next pointer

Postcondition

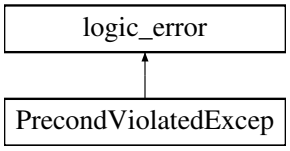
next pointer now set to parameter input pointer

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

- Node.h
- Node.cpp

4.4 PrecondViolatedExcep Class Reference

Inheritance diagram for PrecondViolatedExcep:



Public Member Functions

- **PrecondViolatedExcep** (const string &message="")

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

- PrecondViolatedExcep.h
- PrecondViolatedExcep.cpp

5 File Documentation

5.1 LinkedList.cpp File Reference

Implementation file for LinkedList creation.

```
#include "LinkedList.h"
#include <cassert>
#include <cstddef>
```

5.1.1 Detailed Description

Implementation file for LinkedList creation. Creates function prototype for the linked list used in the linked list

Version

1.00 Wei Tong (15 September 2016) Initial development of the linked list class

Note

Adapted from Frank M. Carrano and Timothy M. Henry Copyright (c) 2012 Pearson Education, Hoboken, New Jersey.

5.2 LinkedList.h File Reference

Header file for LinkedList creation.

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

Classes

- class LinkedList< ItemType >

5.2.1 Detailed Description

Header file for LinkedList creation. Creates function prototype for the linked list used in the linked list

Version

1.00 Wei Tong (15 September 2016) Initial development of the linked list class

Note

Adapted from Frank M. Carrano and Timothy M. Henry Copyright (c) 2012 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

Implementation file for Node creation.

```
#include "Node.h"
#include <cstddef>
```

5.4.1 Detailed Description

Implementation file for Node creation. Creates function prototype for the Nodes used in the linked list

Version

1.00 Wei Tong (15 September 2016) Initial development of the Node class

Note

Adapted from Frank M. Carrano and Timothy M. Henry Copyright (c) 2012 Pearson Education, Hoboken, New Jersey.

5.5 Node.h File Reference

Header file for Node creation.

Classes

- class Node< ItemType >

5.5.1 Detailed Description

Header file for Node creation. Creates function prototype for the Nodes used in the linked list

Version

1.00 Wei Tong (15 September 2016) Initial development of the Node class

Note

Adapted from Frank M. Carrano and Timothy M. Henry Copyright (c) 2012 Pearson Education, Hoboken, New Jersey.

5.6 PrecondViolatedExcep.cpp File Reference

Implementation file for precondition violated exception creation.

```
#include "PrecondViolatedExcep.h"
```

5.6.1 Detailed Description

Implementation file for precondition violated exception creation. Creates function prototype for the precondition violated exceptions used in the linked list

Version

1.00 Wei Tong (15 September 2016) Initial development of the precondition violated exception class

Note

Adapted from Frank M. Carrano and Timothy M. Henry Copyright (c) 2012 Pearson Education, Hoboken, New Jersey.

5.7 PrecondViolatedExcep.h File Reference

Header file for precondition violated exception creation.

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

Classes

- class PrecondViolatedExcep

5.7.1 Detailed Description

Header file for precondition violated exception creation. Creates function prototype for the precondition violated exceptions used in the linked list

Version

1.00 Wei Tong (15 September 2016) Initial development of the precondition violated exception class

Note

Reference Adapted from Frank M. Carrano and Timothy M. Henry Copyright (c) 2012 Pearson Education, Hoboken, New Jersey.

Index