PA01 - Linked List

Generated by Doxygen 1.8.11

Contents

1 Hierarchical Index **2**

1.1 Class Hierarchy 2

2 Class Index **2**

2.1 Class List 2

3 File Index **2**

3.1 File List 2

4 Class Documentation **3**

4.1 LinkedList< ItemType > Class Template Reference 3

4.1.1 Member Function Documentation 3

4.2 ListInterface< ItemType > Class Template Reference 5

4.2.1 Member Function Documentation 6

4.3 Node< ItemType > Class Template Reference 8

4.4 PrecondViolatedExcept Class Reference 9

5 File Documentation **9**

5.1 CS302/Projects/PA01/LinkedList.cpp File Reference 9

5.1.1 Detailed Description 9

5.2 CS302/Projects/PA01/LinkedList.h File Reference 10

5.2.1 Detailed Description 10

5.3 CS302/Projects/PA01/ListInterface.h File Reference 10

5.3.1 Detailed Description 10

5.4 CS302/Projects/PA01/Node.cpp File Reference 11

5.4.1 Detailed Description 11

5.5 CS302/Projects/PA01/Node.h File Reference 11

5.5.1 Detailed Description 11

5.6 CS302/Projects/PA01/PA01.cpp File Reference 12

5.6.1 Detailed Description 12

5.7 CS302/Projects/PA01/PrecondViolatedExcept.cpp File Reference 12

5.7.1 Detailed Description 12

5.8 CS302/Projects/PA01/PrecondViolatedExcept.h File Reference 13

5.8.1 Detailed Description 13

Index 15

1 Hierarchical Index

1.1 Class Hierarchy

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

ListInterface< ItemType > 5

LinkedList< ItemType > 3

logic_error

PrecondViolatedExcept 9

Node< ItemType > 8

2 Class Index

2.1 Class List

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

LinkedList< ItemType > 3

ListInterface< ItemType > 5

Node< ItemType > 8

PrecondViolatedExcept 9

3 File Index

3.1 File List

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

CS302/Projects/PA01/LinkedList.cpp
This is the implementation file for the linked list class 9

CS302/Projects/PA01/LinkedList.h
This is the header file for the linked list class 10

CS302/Projects/PA01/ListInterface.h
Interface file for the List ADT 10

CS302/Projects/PA01/Node.cpp
This is the implementation file for the Node class 11

CS302/Projects/PA01/Node.h
This is the header file for the Node class 11

CS302/Projects/PA01/PA01.cpp
This is the test driver for Programming Assignment 1 12

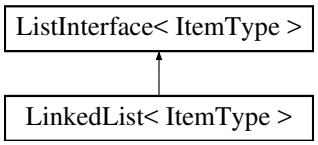
CS302/Projects/PA01/PrecondViolatedExcept.cpp
This is the himplementation file for the PrecondViolatedExcept class 12

CS302/Projects/PA01/PrecondViolatedExcept.h
This is the header file for the PrecondViolatedExcept class 13

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)

Private Member Functions

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

Private Attributes

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

4.1.1 Member Function Documentation

4.1.1.1 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 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 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 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 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 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.

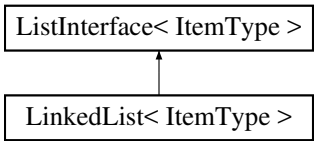
Implements ListInterface< ItemType >.

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

- CS302/Projects/PA01/LinkedList.h
- CS302/Projects/PA01/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.

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

- CS302/Projects/PA01/ListInterface.h

4.3 Node< ItemType > Class Template Reference

Public Member Functions

- **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

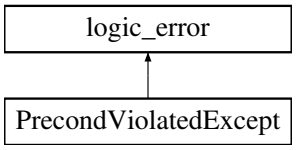
- ItemType **item**
- Node< ItemType > **next**

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

- CS302/Projects/PA01/Node.h
- CS302/Projects/PA01/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:

- CS302/Projects/PA01/PrecondViolatedExcept.h
- CS302/Projects/PA01/PrecondViolatedExcept.cpp

5 File Documentation

5.1 CS302/Projects/PA01/LinkedList.cpp File Reference

This is the implementation file for the linked list class.

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

5.1.1 Detailed Description

This is the implementation file for the linked list class.

Author

Bryce Monaco

Implements the various components of the LinkedList class

Version

1.0

Adapted from Frank M. Carrano and Timothy M. Henry Copyright (c) 2017 Pearson Education, Hoboken, New Jersey. Listing 9-2

5.2 CS302/Projects/PA01/LinkedList.h File Reference

This is the header file for the linked list class.

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

Classes

- class LinkedList< ItemType >

5.2.1 Detailed Description

This is the header file for the linked list class.

Author

Bryce Monaco

Lists the various components of the LinkedList class

Version

1.0

Adapted from Frank M. Carrano and Timothy M. Henry Copyright (c) 2017 Pearson Education, Hoboken, New Jersey. Listing 9-2

5.3 CS302/Projects/PA01/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 CS302/Projects/PA01/Node.cpp File Reference

This is the implementation file for the Node class.

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

5.4.1 Detailed Description

This is the implementation file for the Node class.

Author

Bryce Monaco

Implements the various components of the Node class

Version

1.0

Adapted from Frank M. Carrano and Timothy M. Henry Copyright (c) 2017 Pearson Education, Hoboken, New Jersey. Listing 4-2

5.5 CS302/Projects/PA01/Node.h File Reference

This is the header file for the Node class.

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

Classes

- class Node< ItemType >

5.5.1 Detailed Description

This is the header file for the Node class.

Author

Bryce Monaco

Lists the various components of the Node class

Version

1.0

Adapted from Frank M. Carrano and Timothy M. Henry Copyright (c) 2017 Pearson Education, Hoboken, New Jersey. Listing 4-1

5.6 CS302/Projects/PA01/PA01.cpp File Reference

This is the test driver for Programming Assignment 1.

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

Functions

- int **main** ()

5.6.1 Detailed Description

This is the test driver for Programming Assignment 1.

Author

Bryce Monaco

This allows the user to test the various parts of this project

Version

1.0

Not all conditions or combinations are tested

5.7 CS302/Projects/PA01/PrecondViolatedExcept.cpp File Reference

This is the himplementation file for the PrecondViolatedExcept class.

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

5.7.1 Detailed Description

This is the himplementation file for the PrecondViolatedExcept class.

Author

Bryce Monaco

Implements the various components of the PrecondViolatedExcept class

Version

1.0

Adapted from Frank M. Carrano and Timothy M. Henry Copyright (c) 2017 Pearson Education, Hoboken, New Jersey. Listing 7-6

5.8 CS302/Projects/PA01/PrecondViolatedExcept.h File Reference

This is the header file for the PrecondViolatedExcept class.

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

Classes

- class PrecondViolatedExcept

5.8.1 Detailed Description

This is the header file for the PrecondViolatedExcept class.

Author

Bryce Monaco

Lists the various components of the PrecondViolatedExcept class

Version

1.0

Copied from Frank M. Carrano and Timothy M. Henry Copyright (c) 2017 Pearson Education, Hoboken, New Jersey. Listing 7-5

Index

CS302/Projects/PA01/LinkedList.cpp, 9
CS302/Projects/PA01/LinkedList.h, 10
CS302/Projects/PA01/ListInterface.h, 10
CS302/Projects/PA01/Node.cpp, 11
CS302/Projects/PA01/Node.h, 11
CS302/Projects/PA01/PA01.cpp, 12
CS302/Projects/PA01/PrecondViolatedExcept.cpp, 12
CS302/Projects/PA01/PrecondViolatedExcept.h, 13
clear
    LinkedList, 3
    ListInterface, 6

getEntry
    LinkedList, 3
    ListInterface, 6
getLength
    LinkedList, 4
    ListInterface, 6

insert
    LinkedList, 4
    ListInterface, 6
isEmpty
    LinkedList, 4
    ListInterface, 7

LinkedList
    clear, 3
    getEntry, 3
    getLength, 4
    insert, 4
    isEmpty, 4
    remove, 4
LinkedList< ItemType >, 3
ListInterface
    clear, 6
    getEntry, 6
    getLength, 6
    insert, 6
    isEmpty, 7
    remove, 7
    replace, 8
ListInterface< ItemType >, 5

Node< ItemType >, 8

PrecondViolatedExcept, 9

remove
    LinkedList, 4
    ListInterface, 7
replace
    ListInterface, 8