

Reference Manual

Generated by Doxygen 1.5.5

Thu Mar 25 14:11:21 2010

Contents

1	Class Index	1
1.1	Class List	1
2	Class Documentation	3
2.1	List< T > Class Template Reference	3
2.2	ListIterator< T > Class Template Reference	11
2.3	NoSuchObject Class Reference	13

Chapter 1

Class Index

1.1 Class List

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

List< T >	3
ListIterator< T >	11
NoSuchObject	13

Chapter 2

Class Documentation

2.1 List< T > Class Template Reference

```
#include <list.h>
```

Public Member Functions

- [List](#) ()
- [List](#) (const [List](#)< T > &) throw (bad_alloc)
- [~List](#) ()
- void [add](#) (unsigned, const T &) throw (bad_alloc, out_of_range)
- void [addAll](#) (const [List](#)< T > &) throw (bad_alloc)
- void [addAll](#) (unsigned, const [List](#)< T > &) throw (bad_alloc, out_of_range)
- void [addFirst](#) (const T &) throw (bad_alloc)
- void [addLast](#) (const T &) throw (bad_alloc)
- void [clear](#) ()
- bool [contains](#) (const T &) const
- bool [empty](#) () const
- int [indexOf](#) (const T &) const
- T & [get](#) (unsigned) const throw (out_of_range)
- T & [getFirst](#) () const throw (NoSuchObject)
- T & [getLast](#) () const throw (NoSuchObject)
- [ListIterator](#)< T > [listIterator](#) ()
- [ListIterator](#)< T > [listIterator](#) (unsigned) throw (out_of_range)
- T [remove](#) (unsigned) throw (out_of_range)
- T [removeFirst](#) () throw (NoSuchObject)
- T [removeFirstOccurrence](#) (const T &) throw (NoSuchObject)
- T [removeLast](#) () throw (NoSuchObject)
- T [removeLastOccurrence](#) (const T &) throw (NoSuchObject)
- T [set](#) (unsigned, const T &) throw (out_of_range)
- unsigned [size](#) () const
- T * [toArray](#) () const throw (bad_alloc)
- const [List](#)< T > & [operator=](#) (const [List](#)< T > &) throw (bad_alloc)
- void [printInternal](#) (ostream &=cout)

Friends

- ostream & [operator<<](#) (ostream &, const [List< T >](#) &)

2.1.1 Detailed Description

`template<typename T> class List< T >`

Implementation of a [List](#) ADT using a doubly-linked list.

Author:

Mark Maloof

Version:

1.1, 3/23/10

2.1.2 Constructor & Destructor Documentation

2.1.2.1 `template<typename T> List< T >::List ()` [[inline](#)]

Default constructor.

2.1.2.2 `template<typename T> List< T >::List (const List< T > & list) throw (bad_alloc)`
[[inline](#)]

Copy constructor.

Exceptions:

[bad_alloc](#) if memory cannot be allocated.

2.1.2.3 `template<typename T> List< T >::~~List ()` [[inline](#)]

Class destructor.

2.1.3 Member Function Documentation

2.1.3.1 `template<typename T> void List< T >::add (unsigned index, const T & object) throw (bad_alloc, out_of_range)` [[inline](#)]

Adds the specified object to this list at the specified position.

Parameters:

index the position at which to insert the object

object the object to be inserted

Exceptions:

[bad_alloc](#) if memory cannot be allocated

2.1.3.2 `template<typename T> void List< T >::addAll (const List< T > & list) throw (bad_alloc) [inline]`

Adds all of objects in the specified list to the end of this list.

Parameters:

list containing the objects to be added

Exceptions:

bad_alloc if memory cannot be allocated

2.1.3.3 `template<typename T> void List< T >::addAll (unsigned index, const List< T > & list) throw (bad_alloc, out_of_range) [inline]`

Adds all of objects in the specified list to the specified position of this list.

Parameters:

index the position at which to insert the object

list containing the objects to be added

Exceptions:

bad_alloc if memory cannot be allocated

out_of_range if the index is out of range

2.1.3.4 `template<typename T> void List< T >::addFirst (const T & object) throw (bad_alloc) [inline]`

Adds the specified object to the front of the list.

Parameters:

object the object to be added to the front of the list

Exceptions:

bad_alloc if memory cannot be allocated

2.1.3.5 `template<typename T> void List< T >::addLast (const T & object) throw (bad_alloc) [inline]`

Adds the specified object to the end of the list.

Parameters:

object the object to be added to the back of the list

Exceptions:

bad_alloc if memory cannot be allocated

2.1.3.6 `template<typename T> void List< T >::clear () [inline]`

Removes all of the elements in the list.

2.1.3.7 `template<typename T> bool List< T >::contains (const T & object) const [inline]`

Returns true if this list contains the specified object

Parameters:

object the specified object

Returns:

true if the list contains the object; false otherwise

2.1.3.8 `template<typename T> bool List< T >::empty () const [inline]`

Returns true if this list is empty; returns false otherwise.

Returns:

true if empty; false otherwise

2.1.3.9 `template<typename T> int List< T >::indexOf (const T & object) const [inline]`

Returns the position of the specified object in this list. Returns -1 if the object is not in this list.

Parameters:

object the object to be found in the list

Returns:

the position of object in the list

2.1.3.10 `template<typename T> T & List< T >::get (unsigned index) const throw (out_of_range) [inline]`

Returns a reference to the object at the specified position in this list.

Parameters:

index the position of the object

Returns:

a reference to the object at the specified position

Exceptions:

out_of_range if the index is out of range

2.1.3.11 `template<typename T> T & List< T >::getFirst () const throw (NoSuchObject)`
[inline]

Returns a reference to the first object in this list.

Returns:

a reference to the first object

Exceptions:

NoSuchObject if no such object exists in this list

2.1.3.12 `template<typename T> T & List< T >::getLast () const throw (NoSuchObject)`
[inline]

Returns a reference to the last object in this list.

Returns:

a reference to the last object

Exceptions:

NoSuchObject if no such object exists in this list

2.1.3.13 `template<typename T> ListIterator< T > List< T >::listIterator ()` [inline]

Returns a bidirectional iterator for this list.

Returns:**2.1.3.14** `template<typename T> ListIterator< T > List< T >::listIterator (unsigned index)`
`throw (out_of_range)` [inline]

Returns a bidirectional iterator for this list that starts at the specified position.

Parameters:

index the starting position of the iterator

Returns:

a bidirectional list iterator

Exceptions:

out_of_range if the index is out of range

2.1.3.15 `template<typename T> T List< T >::remove (unsigned index) throw (out_of_range)`
[inline]

Removes and returns the object at the specified position.

Parameters:

index the position of the object

Returns:

the object at the specified position

Exceptions:

out_of_range if the index is out of range

2.1.3.16 `template<typename T> T List< T >::removeFirst () throw (NoSuchElementException)`
[inline]

Removes and returns the first object in this list.

Returns:

the object at the front of the list

Exceptions:

NoSuchObject if no such object exists in this list

2.1.3.17 `template<typename T> T List< T >::removeFirstOccurrence (const T & object) throw (NoSuchElementException)` [inline]

Removes and returns the first occurrence of the object in this list.

Parameters:

object the object in the list

Returns:

the first occurrence of object in this list

Exceptions:

NoSuchObject if no such object exists in this list

2.1.3.18 `template<typename T> T List< T >::removeLast () throw (NoSuchElementException)`
[inline]

Removes and returns the last object in this list.

Returns:

the last object in this list

Exceptions:

NoSuchObject if no such object exists in this list

2.1.3.19 `template<typename T> T List< T >::removeLastOccurrence (const T & object) throw (NoSuchObject) [inline]`

Removes and returns the last occurrence of the object in this list.

Parameters:

object the object in the list

Returns:

the last occurrence of object in this list

Exceptions:

NoSuchObject if no such object exists in this list

2.1.3.20 `template<typename T> T List< T >::set (unsigned index, const T & object) throw (out_of_range) [inline]`

Sets the object at the specified position to the specified object and returns the replaced object.

Parameters:

index the position of the object

object the object

Returns:

the replaced object

Exceptions:

out_of_range if the index is out of range

2.1.3.21 `template<typename T> unsigned List< T >::size () const [inline]`

Returns the size (i.e., number of objects) of this list.

Returns:

an unsigned integer indicating this list's size

2.1.3.22 `template<typename T> T * List< T >::toArray () const throw (bad_alloc)` [inline]

Returns an array containing the objects of this list.

Returns:

an array containing the objects of this list

Exceptions:

bad_alloc if memory cannot be allocated.

2.1.3.23 `template<typename T> const List< T > & List< T >::operator= (const List< T > & list) throw (bad_alloc)` [inline]

Returns a deep copy of the specified list.

Parameters:

list the list to be copied.

Returns:

a copy of the list.

Exceptions:

bad_alloc if memory cannot be allocated.

2.1.3.24 `template<typename T> void List< T >::printInternal (ostream & out = cout)` [inline]

A utility method that prints the internal state of this list.

2.1.4 Friends And Related Function Documentation

2.1.4.1 `template<typename T> ostream& operator<< (ostream & out, const List< T > & list)` [friend]

Overloaded stream insertion operator that outputs the specified list to the output stream in the format [e1,e2,e3,...], where e1, e2, ... are the elements of the list.

Parameters:

out the output stream

list the specified list

Returns:

the modified output stream

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

- list.h

2.2 ListIterator< T > Class Template Reference

```
#include <iterator.h>
```

Public Member Functions

- [ListIterator](#) ()
- bool [hasNext](#) () const
- bool [hasPrevious](#) () const
- T & [next](#) () throw (NoSuchElementException)
- T & [previous](#) () throw (NoSuchElementException)
- void [set](#) (const T &)
- void [printInternal](#) () const

Friends

- class [List](#)< T >

2.2.1 Detailed Description

```
template<typename T> class ListIterator< T >
```

Implements a bidirectional iterator for [List](#).

Author:

Mark Maloof

Version:

1.0, 3/25/10

2.2.2 Constructor & Destructor Documentation

2.2.2.1 `template<typename T> ListIterator< T >::ListIterator ()` `[inline]`

Default constructor.

2.2.3 Member Function Documentation

2.2.3.1 `template<typename T> bool ListIterator< T >::hasNext () const` `[inline]`

Returns true if this iterator has a next object.

Returns:

true if this iterator has a next object; false otherwise.

2.2.3.2 `template<typename T> bool ListIterator< T >::hasPrevious () const` `[inline]`

Returns true if this iterator has a previous object.

Returns:

true if this iterator has a previous object; false otherwise.

2.2.3.3 `template<typename T> T & ListIterator< T >::next () throw (NoSuchObject)`
`[inline]`

Returns a reference to the next object in this iterator.

Returns:

a reference to the next object

Exceptions:

NoSuchObject if no such object exists in this iterator

2.2.3.4 `template<typename T> T & ListIterator< T >::previous () throw (NoSuchObject)`
`[inline]`

Returns a reference to the previous object in this iterator.

Returns:

a reference to the previous object

Exceptions:

NoSuchObject if no such object exists in this iterator

2.2.3.5 `template<typename T> void ListIterator< T >::set (const T & object)` `[inline]`

Sets the object at the position of this iterator to the specified object.

Parameters:

object the object to be set

2.2.3.6 `template<typename T> void ListIterator< T >::printInternal () const` `[inline]`

A utility method that prints the internal state of this iterator.

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

- iterator.h

2.3 NoSuchObject Class Reference

```
#include <nosuchobject.h>
```

Public Member Functions

- **NoSuchObject** (const string &what)

2.3.1 Detailed Description

Implementation of the [NoSuchObject](#) exception class.

Author:

Mark Maloof

Version:

1.0, 3/23/10

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

- nosuchobject.h

Index

- ~List
 - List, 4
- add
 - List, 4
- addAll
 - List, 4, 5
- addFirst
 - List, 5
- addLast
 - List, 5
- clear
 - List, 5
- contains
 - List, 6
- empty
 - List, 6
- get
 - List, 6
- getFirst
 - List, 6
- getLast
 - List, 7
- hasNext
 - ListIterator, 11
- hasPrevious
 - ListIterator, 11
- indexOf
 - List, 6
- List, 3
 - ~List, 4
 - add, 4
 - addAll, 4, 5
 - addFirst, 5
 - addLast, 5
 - clear, 5
 - contains, 6
 - empty, 6
 - get, 6
 - getFirst, 6
 - getLast, 7
 - indexOf, 6
 - List, 4
 - listIterator, 7
 - operator<<, 10
 - operator=, 10
 - printInternal, 10
 - remove, 7
 - removeFirst, 8
 - removeFirstOccurrence, 8
 - removeLast, 8
 - removeLastOccurrence, 9
 - set, 9
 - size, 9
 - toArray, 9
- ListIterator, 11
 - hasNext, 11
 - hasPrevious, 11
 - ListIterator, 11
 - next, 12
 - previous, 12
 - printInternal, 12
 - set, 12
- listIterator
 - List, 7
- next
 - ListIterator, 12
- NoSuchObject, 13
- operator<<
 - List, 10
- operator=
 - List, 10
- previous
 - ListIterator, 12
- printInternal
 - List, 10
 - ListIterator, 12
- remove
 - List, 7
- removeFirst
 - List, 8
- removeFirstOccurrence

- List, [8](#)
- removeLast
 - List, [8](#)
- removeLastOccurrence
 - List, [9](#)
- set
 - List, [9](#)
 - ListIterator, [12](#)
- size
 - List, [9](#)
- toArray
 - List, [9](#)