

Project 5

Generated by Doxygen 1.8.12

Contents

1	Hierarchical Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Class Documentation	7
4.1	ArrayQueue< ItemType > Class Template Reference	7
4.2	Event Struct Reference	7
4.3	LinkedList< ItemType > Class Template Reference	8
4.3.1	Member Function Documentation	8
4.3.1.1	clear()	8
4.3.1.2	getEntry()	8
4.3.1.3	getLength()	9
4.3.1.4	insert()	9
4.3.1.5	isEmpty()	9
4.3.1.6	remove()	10
4.3.1.7	replace()	10
4.4	ListInterface< ItemType > Class Template Reference	11
4.4.1	Member Function Documentation	11
4.4.1.1	clear()	11
4.4.1.2	getEntry()	11

4.4.1.3	getLength()	12
4.4.1.4	insert()	12
4.4.1.5	isEmpty()	13
4.4.1.6	remove()	13
4.4.1.7	replace()	13
4.5	Merge Class Reference	14
4.6	Node< ItemType > Class Template Reference	15
4.6.1	Constructor & Destructor Documentation	15
4.6.1.1	Node() [1/2]	15
4.6.1.2	Node() [2/2]	15
4.6.2	Member Function Documentation	15
4.6.2.1	getItem()	15
4.6.2.2	getNext()	16
4.6.2.3	setItem()	16
4.6.2.4	setNext()	16
4.7	PrecondViolatedExcept Class Reference	16
4.8	PriorityQueue< ItemType > Class Template Reference	17
4.8.1	Member Function Documentation	17
4.8.1.1	peek()	17
4.9	ProcessData Struct Reference	17
4.10	SortedList< ItemType > Class Template Reference	18
4.11	Teller Struct Reference	18

5 File Documentation	19
5.1 ArrayQueue.cpp File Reference	19
5.1.1 Detailed Description	19
5.2 ArrayQueue.h File Reference	19
5.2.1 Detailed Description	19
5.3 LinkedList.cpp File Reference	19
5.3.1 Detailed Description	20
5.4 LinkedList.h File Reference	20
5.4.1 Detailed Description	20
5.5 ListInterface.h File Reference	20
5.5.1 Detailed Description	21
5.6 Merge.cpp File Reference	21
5.6.1 Detailed Description	21
5.7 Merge.h File Reference	22
5.7.1 Detailed Description	22
5.8 Node.cpp File Reference	22
5.8.1 Detailed Description	22
5.9 Node.h File Reference	23
5.9.1 Detailed Description	23
5.10 PA05.cpp File Reference	23
5.10.1 Detailed Description	24
5.11 PrecondViolatedExcept.cpp File Reference	24
5.11.1 Detailed Description	24
5.12 PrecondViolatedExcept.h File Reference	24
5.12.1 Detailed Description	25
5.13 PriorityQueue.cpp File Reference	25
5.13.1 Detailed Description	25
5.14 PriorityQueue.h File Reference	25
5.14.1 Detailed Description	26
5.15 SortedList.cpp File Reference	26
5.15.1 Detailed Description	26
5.16 SortedList.h File Reference	26
5.16.1 Detailed Description	26
Index	27

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

ArrayQueue< ItemType >	7
Event	7
ListInterface< ItemType >	11
LinkedList< ItemType >	8
SortedList< ItemType >	18
logic_error	
PrecondViolatedExcept	16
Merge	14
Node< ItemType >	15
PriorityQueue< ItemType >	17
ProcessData	17
Teller	18

Chapter 2

Class Index

2.1 Class List

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

ArrayQueue< ItemType >	7
Event	7
LinkedList< ItemType >	8
ListInterface< ItemType >	11
Merge	14
Node< ItemType >	15
PrecondViolatedExcept	16
PriorityQueue< ItemType >	17
ProcessData	17
SortedList< ItemType >	18
Teller	18

Chapter 3

File Index

3.1 File List

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

ArrayQueue.cpp	19
ArrayQueue.h	19
LinkedList.cpp	
Implementation file for the Linked List ADT	19
LinkedList.h	
Header file for the Linked List ADT	20
ListInterface.h	
Interface file for the List ADT	20
Merge.cpp	
Implementation file for the Merge sorting class	21
Merge.h	
Header file for the Merge sorting class	22
Node.cpp	
Implementation file for the Node class	22
Node.h	
Header file for the Node class	23
PA05.cpp	
Main driver for project 5	23
PrecondViolatedExcept.cpp	
Implementation file for the PrecondViolatedExcept class	24
PrecondViolatedExcept.h	
Header file for the PrecondViolatedExcept class	24
PriorityQueue.cpp	
Implementation file for the PriorityQueue class	25
PriorityQueue.h	25
SortedList.cpp	
Implementation file for the SortedList class	26
SortedList.h	26

Chapter 4

Class Documentation

4.1 `ArrayQueue< ItemType >` Class Template Reference

Public Member Functions

- bool **isEmpty** () const
- bool **enqueue** (const ItemType &newEntry)
- bool **dequeue** ()
- ItemType **peekFront** () const

Public Attributes

- int **count**

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

- [ArrayQueue.h](#)
- [ArrayQueue.cpp](#)

4.2 `Event` Struct Reference

Public Member Functions

- bool **operator>** (const [Event](#) &rhs) const
- bool **operator<** (const [Event](#) &rhs) const
- bool **operator!=** (const [Event](#) &rhs) const

Public Attributes

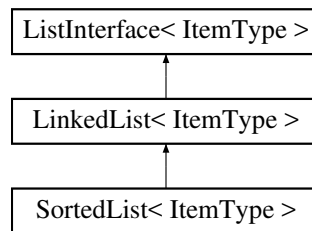
- char **type**
- int **startTime**
- int **duration**
- int **customerID**

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

- [PA05.cpp](#)

4.3 `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](#))

4.3.1 Member Function Documentation

4.3.1.1 `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 >](#).

4.3.1.2 `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 >](#).

4.3.1.3 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 >](#).

4.3.1.4 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 $1 \leq \text{position} \leq \text{getLength}() + 1$ 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

<i>newPosition</i>	The list position at which to insert <i>newEntry</i> .
<i>newEntry</i>	The entry to insert into the list.

Returns

True if insertion is successful, or false if not.

Implements [ListInterface< ItemType >](#).

4.3.1.5 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 >](#).

4.3.1.6 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 $1 \leq \text{position} \leq \text{getLength}()$ 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

<i>position</i>	The list position of the entry to remove.
-----------------	---

Returns

True if removal is successful, or false if not.

Implements [ListInterface< ItemType >](#).

4.3.1.7 replace()

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

Exceptions

PrecondViolatedExcept	if $\text{position} < 1$ or $\text{position} > \text{getLength}()$.
---------------------------------------	--

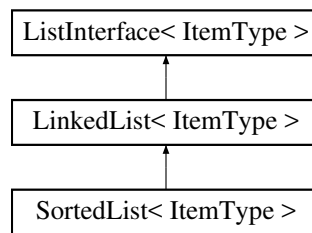
Implements [ListInterface< ItemType >](#).

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

- [LinkedList.h](#)
- [LinkedList.cpp](#)

4.4 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

4.4.1 Member Function Documentation

4.4.1.1 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 >](#).

4.4.1.2 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

<i>position</i>	The list position of the desired entry.
-----------------	---

Returns

The entry at the given position.

Implemented in [LinkedList< ItemType >](#).

4.4.1.3 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 >](#).

4.4.1.4 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 $1 \leq \text{position} \leq \text{getLength}() + 1$ 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

<i>newPosition</i>	The list position at which to insert <i>newEntry</i> .
<i>newEntry</i>	The entry to insert into the list.

Returns

True if insertion is successful, or false if not.

Implemented in [LinkedList< ItemType >](#).

4.4.1.5 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 >](#).

4.4.1.6 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 $1 \leq \text{position} \leq \text{getLength}()$ 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

<i>position</i>	The list position of the entry to remove.
-----------------	---

Returns

True if removal is successful, or false if not.

Implemented in [LinkedList< ItemType >](#).

4.4.1.7 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

<i>position</i>	The list position of the entry to replace.
<i>newEntry</i>	The replacement entry.

Implemented in [LinkedList< ItemType >](#).

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

- [ListInterface.h](#)

4.5 Merge Class Reference

Public Member Functions

- **Merge** (int toSort[], long count)
- void **merge** (long first, long mid, long last)
- void **sort** (long first, long last)

Public Attributes

- int **count**
- int * **data**
- double **elapsedTime**
- int **comparisonCount**
- int **swapCount**

Friends

- ostream & **operator**<< (ostream &out, const [Merge](#) &merge)

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

- [Merge.h](#)
- [Merge.cpp](#)

4.6 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

4.6.1 Constructor & Destructor Documentation

4.6.1.1 [Node](#)() [1/2]

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

Default constructor for the class.

constructs the class

4.6.1.2 [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

4.6.2 Member Function Documentation

4.6.2.1 [getItem](#)()

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

[getItem](#) function

returns current item

4.6.2.2 getNext()

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

getNext function

returns next item

4.6.2.3 setItem()

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

setItem function

Sets the current item to another item

4.6.2.4 setNext()

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

setNext function

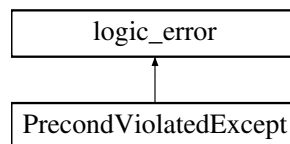
Sets the next item to an item

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

- [Node.h](#)
- [Node.cpp](#)

4.7 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](#)

4.8 PriorityQueue< ItemType > Class Template Reference

Public Member Functions

- **PriorityQueue** (const [PriorityQueue](#) &pq)
- bool **isEmpty** () const
- bool **enqueue** (const ItemType &newEntry)
- bool **dequeue** ()
- ItemType **peek** () const throw ([PrecondViolatedExcept](#))

4.8.1 Member Function Documentation

4.8.1.1 peek()

```
template<class ItemType >
ItemType PriorityQueue< ItemType >::peek ( ) const throw (PrecondViolatedExcept)
```

Exceptions

PrecondViolatedExcept	if priority queue is empty.
---------------------------------------	-----------------------------

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

- [PriorityQueue.h](#)
- [PriorityQueue.cpp](#)

4.9 ProcessData Struct Reference

Public Attributes

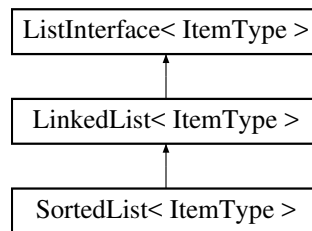
- int **cpuTime**
- int **virtualTime**
- int **maxWaitTime**
- int **avgWaitTime**
- int **maxLineLength**
- int **avgLineLength**

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

- [PA05.cpp](#)

4.10 SortedList< ItemType > Class Template Reference

Inheritance diagram for SortedList< ItemType >:



Public Member Functions

- **SortedList** (const [SortedList](#)< ItemType > &sList)
- bool **insertSorted** (const ItemType &newEntry)
- bool **removeSorted** (const ItemType &anEntry)
- int **getPosition** (const ItemType &anEntry) const

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

- [SortedList.h](#)
- [SortedList.cpp](#)

4.11 Teller Struct Reference

Public Attributes

- bool **available**

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

- [PA05.cpp](#)

Chapter 5

File Documentation

5.1 `ArrayQueue.cpp` File Reference

```
#include <iostream>
#include "ArrayQueue.h"
```

5.1.1 Detailed Description

ADT queue: Circular array-based implementation.

5.2 `ArrayQueue.h` File Reference

```
#include "PrecondViolatedExcept.h"
#include "ArrayQueue.cpp"
```

Classes

- class `ArrayQueue< ItemType >`

5.2.1 Detailed Description

ADT queue: Circular array-based implementation.

5.3 `LinkedList.cpp` File Reference

Implementation file for the Linked List ADT.

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

5.3.1 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

5.4 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 >](#)

5.4.1 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

5.5 ListInterface.h File Reference

Interface file for the List ADT.

Classes

- class [ListInterface](#)< [ItemType](#) >

5.5.1 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.

5.6 Merge.cpp File Reference

Implementation file for the [Merge](#) sorting class.

```
#include "Merge.h"
```

Functions

- ostream & **operator**<< (ostream &out, const [Merge](#) &merge)

5.6.1 Detailed Description

Implementation file for the [Merge](#) sorting class.

Author

Willis Allstead

Specifies the functions of the [Merge](#) sorting class

Version

1.0

5.7 Merge.h File Reference

Header file for the [Merge](#) sorting class.

```
#include <iostream>
#include <ctime>
```

Classes

- class [Merge](#)

5.7.1 Detailed Description

Header file for the [Merge](#) sorting class.

Author

Willis Allstead

Specifies the members of the [Merge](#) sorting class

Version

1.0

5.8 Node.cpp File Reference

Implementation file for the [Node](#) class.

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

5.8.1 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

5.9 Node.h File Reference

Header file for the [Node](#) class.

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

Classes

- class [Node](#)< [ItemType](#) >

5.9.1 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

5.10 PA05.cpp File Reference

Main driver for project 5.

```
#include <iostream>
#include <string>
#include <fstream>
#include <ctime>
#include "Merge.h"
#include "ArrayQueue.h"
#include "PriorityQueue.h"
```

Classes

- struct [Event](#)
- struct [Teller](#)
- struct [ProcessData](#)

Functions

- void **populateFile** (int, string const &)
- **ProcessData simulate** ([PriorityQueue< Event >](#) &, [ArrayQueue< Event >](#) bankLines[], int, int)
- int **processArrival** ([Event](#), [PriorityQueue< Event >](#) &, [ArrayQueue< Event >](#) &, [Teller](#) &, int)
- int **processDeparture** ([Event](#), [PriorityQueue< Event >](#) &, [ArrayQueue< Event >](#) &, [Teller](#) &, int)
- int **main** ()
- void **populateFile** (int count, std::string const &file)

5.10.1 Detailed Description

Main driver for project 5.

Author

Willis Allstead

Version

1.0

5.11 PrecondViolatedExcept.cpp File Reference

Implementation file for the [PrecondViolatedExcept](#) class.

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

5.11.1 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

5.12 PrecondViolatedExcept.h File Reference

Header file for the [PrecondViolatedExcept](#) class.

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

Classes

- class [PrecondViolatedExcept](#)

5.12.1 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

5.13 PriorityQueue.cpp File Reference

Implementation file for the [PriorityQueue](#) class.

5.13.1 Detailed Description

Implementation file for the [PriorityQueue](#) class.

Author

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

Specifies function of the class.

Version

0.10

5.14 PriorityQueue.h File Reference

```
#include "SortedList.h"
#include "PrecondViolatedExcept.h"
#include "PriorityQueue.cpp"
```

Classes

- class [PriorityQueue< ItemType >](#)

5.14.1 Detailed Description

ADT priority queue: ADT sorted list implementation.

5.15 SortedList.cpp File Reference

Implementation file for the [SortedList](#) class.

```
#include <ctime>
```

5.15.1 Detailed Description

Implementation file for the [SortedList](#) class.

Author

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

Specifies function of the class.

Version

0.10

5.16 SortedList.h File Reference

```
#include <memory>
#include "LinkedList.h"
#include "Node.h"
#include "PrecondViolatedExcept.h"
#include "SortedList.cpp"
```

Classes

- class [SortedList](#)< [ItemType](#) >

5.16.1 Detailed Description

ADT sorted list using ADT list.

Index

ArrayQueue< ItemType >, 7

ArrayQueue.cpp, 19

ArrayQueue.h, 19

clear

 LinkedList, 8

 ListInterface, 11

Event, 7

getEntry

 LinkedList, 8

 ListInterface, 11

getItem

 Node, 15

getLength

 LinkedList, 9

 ListInterface, 12

getNext

 Node, 15

insert

 LinkedList, 9

 ListInterface, 12

isEmpty

 LinkedList, 9

 ListInterface, 13

LinkedList

 clear, 8

 getEntry, 8

 getLength, 9

 insert, 9

 isEmpty, 9

 remove, 10

 replace, 10

LinkedList< ItemType >, 8

LinkedList.cpp, 19

LinkedList.h, 20

ListInterface

 clear, 11

 getEntry, 11

 getLength, 12

 insert, 12

 isEmpty, 13

 remove, 13

 replace, 13

ListInterface< ItemType >, 11

ListInterface.h, 20

Merge, 14

Merge.cpp, 21

Merge.h, 22

Node

 getItem, 15

 getNext, 15

 Node, 15

 setItem, 16

 setNext, 16

Node< ItemType >, 15

Node.cpp, 22

Node.h, 23

PA05.cpp, 23

peek

 PriorityQueue, 17

PrecondViolatedExcept, 16

PrecondViolatedExcept.cpp, 24

PrecondViolatedExcept.h, 24

PriorityQueue

 peek, 17

PriorityQueue< ItemType >, 17

PriorityQueue.cpp, 25

PriorityQueue.h, 25

ProcessData, 17

remove

 LinkedList, 10

 ListInterface, 13

replace

 LinkedList, 10

 ListInterface, 13

setItem

 Node, 16

setNext

 Node, 16

SortedList< ItemType >, 18

SortedList.cpp, 26

SortedList.h, 26

Teller, 18