

# My Project

Generated by Doxygen 1.8.9.1

Sat Apr 2 2016 16:10:45



# Contents

<b>1</b>	<b>Hierarchical Index</b>	<b>1</b>
1.1	Class Hierarchy . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	Array< type > Class Template Reference . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.2	ArrayRunner Class Reference . . . . .	5
3.2.1	Detailed Description . . . . .	6
3.3	EmptyQueueException Class Reference . . . . .	6
3.3.1	Detailed Description . . . . .	6
3.4	EmptyStackException Class Reference . . . . .	6
3.4.1	Detailed Description . . . . .	6
3.5	IArray< type > Class Template Reference . . . . .	6
3.5.1	Detailed Description . . . . .	7
3.6	ICollection< type > Class Template Reference . . . . .	7
3.6.1	Detailed Description . . . . .	7
3.6.2	Member Function Documentation . . . . .	7
3.6.2.1	Add . . . . .	7
3.6.2.2	Get . . . . .	8
3.6.2.3	IsEmpty . . . . .	8
3.6.2.4	Remove . . . . .	8
3.6.2.5	Size . . . . .	8
3.7	IndexOutOfRangeException Class Reference . . . . .	8
3.7.1	Detailed Description . . . . .	9
3.8	IQueue< type > Class Template Reference . . . . .	9
3.8.1	Detailed Description . . . . .	9
3.8.2	Member Function Documentation . . . . .	9
3.8.2.1	Dequeue . . . . .	9
3.8.2.2	Enqueue . . . . .	9

3.8.2.3	Front	10
3.8.2.4	IsEmpty	10
3.8.2.5	Size	10
3.9	IRunnable Class Reference	10
3.9.1	Detailed Description	11
3.10	IStack< type > Class Template Reference	11
3.10.1	Detailed Description	11
3.10.2	Member Function Documentation	11
3.10.2.1	IsEmpty	11
3.10.2.2	Pop	11
3.10.2.3	Push	12
3.10.2.4	Size	12
3.10.2.5	Top	12
3.11	IStoper Class Reference	12
3.11.1	Detailed Description	13
3.12	List< type > Class Template Reference	13
3.12.1	Detailed Description	13
3.12.2	Member Function Documentation	13
3.12.2.1	Add	13
3.12.2.2	Get	14
3.12.2.3	IsEmpty	14
3.12.2.4	Remove	14
3.12.2.5	Size	14
3.13	ListTest Class Reference	15
3.13.1	Detailed Description	15
3.14	Queue< type > Class Template Reference	15
3.14.1	Detailed Description	16
3.14.2	Member Function Documentation	16
3.14.2.1	Dequeue	16
3.14.2.2	Enqueue	16
3.14.2.3	Front	16
3.14.2.4	IsEmpty	17
3.14.2.5	Size	17
3.15	Stack< type > Class Template Reference	17
3.15.1	Detailed Description	18
3.15.2	Member Function Documentation	18
3.15.2.1	IsEmpty	18
3.15.2.2	Pop	18
3.15.2.3	Push	18
3.15.2.4	Size	18

<b>CONTENTS</b>	<b>v</b>
3.15.2.5 <a href="#">Top</a> . . . . .	19
3.16 <a href="#">TimeCounter Class Reference</a> . . . . .	19
3.16.1 <a href="#">Detailed Description</a> . . . . .	19
<b>Index</b>	<b>21</b>



# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

EmptyQueueException . . . . .	6
EmptyStackException . . . . .	6
IArray< type > . . . . .	6
Array< type > . . . . .	5
IArray< int > . . . . .	6
Array< int > . . . . .	5
ArrayRunner . . . . .	5
IArray< string > . . . . .	6
Array< string > . . . . .	5
ICollection< type > . . . . .	7
List< type > . . . . .	13
ICollection< string > . . . . .	7
List< string > . . . . .	13
ListTest . . . . .	15
IndexOutOfRangeException . . . . .	8
IQueue< type > . . . . .	9
Queue< type > . . . . .	15
IRunnable . . . . .	10
ArrayRunner . . . . .	5
ListTest . . . . .	15
IStack< type > . . . . .	11
Stack< type > . . . . .	17
IStoper . . . . .	12
TimeCounter . . . . .	19





## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">Array&lt; type &gt;</a>	5
<a href="#">ArrayRunner</a>	5
<a href="#">EmptyQueueException</a>	6
<a href="#">EmptyStackException</a>	6
<a href="#">IArray&lt; type &gt;</a>	6
<a href="#">IList&lt; type &gt;</a>	7
<a href="#">IndexOutOfRangeException</a>	8
<a href="#">IQueue&lt; type &gt;</a>	9
<a href="#">IRunnable</a>	10
<a href="#">IStack&lt; type &gt;</a>	11
<a href="#">IStoper</a>	12
<a href="#">List&lt; type &gt;</a>	13
<a href="#">ListTest</a>	15
<a href="#">Queue&lt; type &gt;</a>	15
<a href="#">Stack&lt; type &gt;</a>	17
<a href="#">TimeCounter</a>	19



## Chapter 3

# Class Documentation

### 3.1 `Array< type >` Class Template Reference

Inherits [IArray< type >](#).

#### Public Member Functions

- virtual void **Add** (type value)
- virtual type **Get** (int index)
- virtual void **Set** (type item, int index)
- virtual int **GetNumberOfElements** ()
- void **Add** (ExpandingType expandingType, type value)

#### Protected Attributes

- long int **arraySize**
- long int **numberOfElements**
- type \* **array**
- type \* **ptr**

#### 3.1.1 Detailed Description

```
template<typename type>class Array< type >
```

Definition at line 17 of file Array.hh.

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

- /home/lemur013/Programing/PAMSI/PAMSI/Lab3/Array.hh

### 3.2 `ArrayRunner` Class Reference

Inherits [Array< int >](#), and [IRunnable](#).

#### Public Member Functions

- virtual bool **Prepare** (int size)
- virtual bool **Run** ()

## Additional Inherited Members

### 3.2.1 Detailed Description

Definition at line 9 of file ArrayRunner.hh.

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

- /home/lemur013/Programing/PAMSI/PAMSI/Lab3/ArrayRunner.hh
- /home/lemur013/Programing/PAMSI/PAMSI/Lab3/ArrayRunner.cpp

## 3.3 EmptyQueueException Class Reference

### Public Member Functions

- void **Show** ()

### 3.3.1 Detailed Description

Definition at line 6 of file EmptyQueueException.hh.

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

- /home/lemur013/Programing/PAMSI/PAMSI/Lab3/EmptyQueueException.hh

## 3.4 EmptyStackException Class Reference

### Public Member Functions

- void **Show** ()

### 3.4.1 Detailed Description

Definition at line 6 of file EmptyStackException.hh.

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

- /home/lemur013/Programing/PAMSI/PAMSI/Lab3/EmptyStackException.hh

## 3.5 IArray< type > Class Template Reference

Inherited by [Array< type >](#).

### Public Member Functions

- virtual void **Add** (type item)=0
- virtual type **Get** (int index)=0
- virtual void **Set** (type item, int index)=0
- virtual int **GetNumberOfElements** ()=0

### 3.5.1 Detailed Description

template<typename type>class IArray< type >

Definition at line 5 of file IArray.hh.

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

- /home/lemur013/Programing/PAMSI/PAMSI/Lab3/IArray.hh

## 3.6 IList< type > Class Template Reference

#include <IList.hh>

Inherited by [List< type >](#).

### Public Member Functions

- virtual void [Add](#) (type item, int index)=0  
*Add element to list at specific place(index)*
- virtual void [Remove](#) (int index)=0  
*Add element to list at a given index.*
- virtual type [Get](#) (int index)=0  
*Return value of the element at given index.*
- virtual int [Size](#) ()=0  
*Return size of the list.*
- virtual bool [IsEmpty](#) ()=0  
*Check whether the list is empty.*

### 3.6.1 Detailed Description

template<typename type>class IList< type >

Interface describes the basic operations on the list of different type. User have to define type of the list

Definition at line 7 of file IList.hh.

### 3.6.2 Member Function Documentation

3.6.2.1 template<typename type> virtual void IList< type >::Add ( type item, int index ) [pure virtual]

Add element to list at specific place(index)

#### Parameters

in	<i>item</i>	- Item which will be added.
in	<i>index</i>	- specific place on the list for new item.

#### Exceptions

<a href="#">IndexOutOfRangeException</a>	when user tries to use this method on a non-existent index.
--	---

Implemented in [List< type >](#), and [List< string >](#).

### 3.6.2.2 `template<typename type> virtual type IList< type >::Get ( int index ) [pure virtual]`

Return value of the element at given index.

#### Returns

Value of the element at given index.

#### Exceptions

<a href="#"><code>IndexOutOfRangeException</code></a>	when user tries to use this method on a non-existent index.
---	---

Implemented in [List< type >](#), and [List< string >](#).

### 3.6.2.3 `template<typename type> virtual bool IList< type >::IsEmpty ( ) [pure virtual]`

Check whether the list is empty.

#### Returns

true - If list is empty.

false - If list is not empty.

Implemented in [List< type >](#), and [List< string >](#).

### 3.6.2.4 `template<typename type> virtual void IList< type >::Remove ( int index ) [pure virtual]`

Add element to list at a given index.

#### Parameters

<i>in</i>	<i>item</i>	- Item which will be added.
<i>in</i>	<i>index</i>	- index on the list for new item.

#### Exceptions

<a href="#"><code>IndexOutOfRangeException</code></a>	when user tries to use this method on a non-existent index.
---	---

Implemented in [List< type >](#), and [List< string >](#).

### 3.6.2.5 `template<typename type> virtual int IList< type >::Size ( ) [pure virtual]`

Return size of the list.

#### Returns

Size of the list.

Implemented in [List< type >](#), and [List< string >](#).

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

- /home/lemur013/Programing/PAMSI/PAMSI/Lab3/IList.hh

## 3.7 IndexOutOfRangeException Class Reference

### Public Member Functions

- void **Show** ()

### 3.7.1 Detailed Description

Definition at line 6 of file IndexOutOfRangeException.hh.

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

- /home/lemur013/Programing/PAMSI/PAMSI/Lab3/IndexOutOfRangeException.hh

## 3.8 IQueue< type > Class Template Reference

```
#include <IQueue.hh>
```

Inherited by [Queue< type >](#).

### Public Member Functions

- virtual void [Enqueue](#) (type item)=0  
*Add element to the end of the queue.*
- virtual type [Dequeue](#) ()=0  
*Remove element from the beginning of the queue and return it.*
- virtual type [Front](#) ()=0  
*Return element from the beginning of the queue.*
- virtual int [Size](#) ()=0  
*Return size of the queue.*
- virtual bool [IsEmpty](#) ()=0  
*Check whether the queue is empty.*

### 3.8.1 Detailed Description

```
template<typename type>class IQueue< type >
```

Interface describes the basic operations on the queue of different type. User have to define type of the queue

Definition at line 7 of file IQueue.hh.

### 3.8.2 Member Function Documentation

**3.8.2.1** `template<typename type > virtual type IQueue< type >::Dequeue ( ) [pure virtual]`

Remove element from the beginning of the queue and return it.

#### Returns

Item which will be removed.

#### Exceptions

<a href="#">EmptyQueueException</a>	when user tries to use this method on empty queue.
-------------------------------------	--

Implemented in [Queue< type >](#).

**3.8.2.2** `template<typename type > virtual void IQueue< type >::Enqueue ( type item ) [pure virtual]`

Add element to the end of the queue.

**Parameters**

<i>in</i>	<i>item</i>	- Item which will be added.
-----------	-------------	-----------------------------

Implemented in [Queue< type >](#).

**3.8.2.3** `template<typename type > virtual type IQueue< type >::Front ( ) [pure virtual]`

Return element from the beginning of the queue.

**Returns**

Item from the top of the queue.

**Exceptions**

<a href="#">EmptyQueueException</a>	when user tries to use this method on empty queue.
-------------------------------------	--

Implemented in [Queue< type >](#).

**3.8.2.4** `template<typename type > virtual bool IQueue< type >::IsEmpty ( ) [pure virtual]`

Check whether the queue is empty.

**Returns**

true - If queue is empty.  
false - If queue is not empty.

Implemented in [Queue< type >](#).

**3.8.2.5** `template<typename type > virtual int IQueue< type >::Size ( ) [pure virtual]`

Return size of the queue.

**Returns**

Size of the queue.

Implemented in [Queue< type >](#).

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

- /home/lemur013/Programing/PAMSI/PAMSI/Lab3/IQueue.hh

## 3.9 IRunnable Class Reference

Inherited by [ArrayRunner](#), and [ListTest](#).

**Public Member Functions**

- virtual bool **Prepare** (int size)=0
- virtual bool **Run** ()=0

**Protected Attributes**

- int **MySize**



### 3.9.1 Detailed Description

Definition at line 4 of file IRunnable.hh.

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

- /home/lemur013/Programing/PAMSI/PAMSI/Lab3/IRunnable.hh

## 3.10 IStack< type > Class Template Reference

```
#include <IStack.hh>
```

Inherited by [Stack< type >](#).

### Public Member Functions

- virtual void [Push](#) (type item)=0  
*Add element to the top of the stack.*
- virtual type [Pop](#) ()=0  
*Remove element from top of the stack and return it.*
- virtual type [Top](#) ()=0  
*Return item from top of the stack.*
- virtual int [Size](#) ()=0  
*Return size of the stack.*
- virtual bool [IsEmpty](#) ()=0  
*Check whether the stack is empty.*

### 3.10.1 Detailed Description

```
template<typename type>class IStack< type >
```

Interface describes the basic operations on the stack of different type. User have to define type of the stack

Definition at line 7 of file IStack.hh.

### 3.10.2 Member Function Documentation

**3.10.2.1** `template<typename type> virtual bool IStack< type >::IsEmpty ( ) [pure virtual]`

Check whether the stack is empty.

#### Returns

true - If stack is empty.  
false - If stack is not empty.

Implemented in [Stack< type >](#).

**3.10.2.2** `template<typename type> virtual type IStack< type >::Pop ( ) [pure virtual]`

Remove element from top of the stack and return it.

#### Returns

Item from top wich will be removed.

**Exceptions**

<a href="#"><i>EmptyStackException</i></a>	when user tries to use this method on empty stack.
--	--

Implemented in [Stack< type >](#).

**3.10.2.3** `template<typename type> virtual void IStack< type >::Push ( type item ) [pure virtual]`

Add element to the top of the stack.

**Parameters**

<i>in</i>	<i>item</i>	- Item which will be added.
-----------	-------------	-----------------------------

Implemented in [Stack< type >](#).

**3.10.2.4** `template<typename type> virtual int IStack< type >::Size ( ) [pure virtual]`

Return size of the stack.

**Returns**

Size of the stack.

Implemented in [Stack< type >](#).

**3.10.2.5** `template<typename type> virtual type IStack< type >::Top ( ) [pure virtual]`

Return item from top of the stack.

**Returns**

Item from top of the stack.

**Exceptions**

<a href="#"><i>EmptyStackException</i></a>	when user tries to use this method on empty stack.
--	--

Implemented in [Stack< type >](#).

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

- /home/lemur013/Programing/PAMSI/PAMSI/Lab3/IStack.hh

## 3.11 IStoper Class Reference

Inherited by [TimeCounter](#).

**Public Member Functions**

- virtual void **Start** ()=0
- virtual void **Stop** ()=0
- virtual long **GetElapsedTime** ()=0
- virtual bool **DumpToFile** ()=0

### 3.11.1 Detailed Description

Definition at line 4 of file IStoper.hh.

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

- /home/lemur013/Programing/PAMSI/PAMSI/Lab3/IStoper.hh

## 3.12 List< type > Class Template Reference

Inherits [IList< type >](#).

### Public Member Functions

- virtual void [Add](#) (type item, int index)  
*Add element to list at specific place(index)*
- virtual void [Remove](#) (int index)  
*Add element to list at a given index.*
- virtual type [Get](#) (int index)  
*Return value of the element at given index.*
- virtual int [Size](#) ()  
*Return size of the list.*
- virtual bool [IsEmpty](#) ()  
*Check whether the list is empty.*
- void **ShowList** ()

### Protected Attributes

- [Array](#)< type > \* **listArray**
- int **size**

### 3.12.1 Detailed Description

```
template<typename type>class List< type >
```

Definition at line 9 of file List.hh.

### 3.12.2 Member Function Documentation

3.12.2.1 `template<typename type> virtual void List< type >::Add ( type item, int index ) [inline],[virtual]`

Add element to list at specific place(index)

Parameters

<code>in</code>	<code>item</code>	- Item which will be added.
<code>in</code>	<code>index</code>	- specific place on the list for new item.

## Exceptions

<a href="#"><i>IndexOutOfRangeException</i></a>	when user tries to use this method on a non-existent index.
---	---

Implements [IList< type >](#).

Definition at line 27 of file List.hh.

**3.12.2.2** `template<typename type> virtual type List< type >::Get ( int index )` `[inline],[virtual]`

Return value of the element at given index.

## Returns

Value of the element at given index.

## Exceptions

<a href="#"><i>IndexOutOfRangeException</i></a>	when user tries to use this method on a non-existent index.
---	---

Implements [IList< type >](#).

Definition at line 97 of file List.hh.

**3.12.2.3** `template<typename type> virtual bool List< type >::IsEmpty ( )` `[inline],[virtual]`

Check whether the list is empty.

## Returns

true - If list is empty.  
false - If list is not empty.

Implements [IList< type >](#).

Definition at line 107 of file List.hh.

**3.12.2.4** `template<typename type> virtual void List< type >::Remove ( int index )` `[inline],[virtual]`

Add element to list at a given index.

## Parameters

<i>in</i>	<i>item</i>	- Item which will be added.
<i>in</i>	<i>index</i>	- index on the list for new item.

## Exceptions

<a href="#"><i>IndexOutOfRangeException</i></a>	when user tries to use this method on a non-existent index.
---	---

Implements [IList< type >](#).

Definition at line 68 of file List.hh.

**3.12.2.5** `template<typename type> virtual int List< type >::Size ( )` `[inline],[virtual]`

Return size of the list.

#### Returns

Size of the list.

Implements [IList< type >](#).

Definition at line 102 of file List.hh.

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

- /home/lemur013/Programing/PAMSI/PAMSI/Lab3/List.hh

## 3.13 ListTest Class Reference

Inherits [List< string >](#), and [IRunnable](#).

#### Public Member Functions

- **ListTest** (string allWords[])
- virtual bool **Prepare** (int size)
- virtual bool **Run** ()
- bool **Run** (string word)
- int **Find** (string word)

#### Additional Inherited Members

##### 3.13.1 Detailed Description

Definition at line 11 of file ListTest.hh.

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

- /home/lemur013/Programing/PAMSI/PAMSI/Lab3/ListTest.hh

## 3.14 Queue< type > Class Template Reference

Inherits [IQueue< type >](#).

#### Public Member Functions

- virtual void [Enqueue](#) (type item)  
*Add element to the end of the queue.*
- virtual type [Dequeue](#) ()  
*Remove element from the beginning of the queue and return it.*
- virtual type [Front](#) ()  
*Return element from the beginning of the queue.*
- virtual int [Size](#) ()  
*Return size of the queue.*
- virtual bool [IsEmpty](#) ()  
*Check whether the queue is empty.*
- void **ShowQueue** ()

## Protected Attributes

- [Array](#)< type > \* **queueArray**
- int **f**
- int **r**

### 3.14.1 Detailed Description

template<typename type>class Queue< type >

Definition at line 9 of file Queue.hh.

### 3.14.2 Member Function Documentation

3.14.2.1 template<typename type > virtual type Queue< type >::Deque ( ) [inline],[virtual]

Remove element from the beginning of the queue and return it.

#### Returns

Item which will be removed.

#### Exceptions

<a href="#">EmptyQueueException</a>	when user tries to use this method on empty queue.
-------------------------------------	--

Implements [IQueue< type >](#).

Definition at line 35 of file Queue.hh.

3.14.2.2 template<typename type > virtual void Queue< type >::Enqueue ( type *item* ) [inline],[virtual]

Add element to the end of the queue.

#### Parameters

in	<i>item</i>	- Item which will be added.
----	-------------	-----------------------------

Implements [IQueue< type >](#).

Definition at line 29 of file Queue.hh.

3.14.2.3 template<typename type > virtual type Queue< type >::Front ( ) [inline],[virtual]

Return element from the beginning of the queue.

#### Returns

Item from the top of the queue.

#### Exceptions

<a href="#">EmptyQueueException</a>	when user tries to use this method on empty queue.
-------------------------------------	--

Implements [IQueue< type >](#).

Definition at line 49 of file Queue.hh.

3.14.2.4 `template<typename type> virtual bool Queue< type >::IsEmpty ( ) [inline], [virtual]`

Check whether the queue is empty.

Returns

true - If queue is empty.  
false - If queue is not empty.

Implements [IQueue< type >](#).

Definition at line 66 of file Queue.hh.

3.14.2.5 `template<typename type> virtual int Queue< type >::Size ( ) [inline], [virtual]`

Return size of the queue.

Returns

Size of the queue.

Implements [IQueue< type >](#).

Definition at line 61 of file Queue.hh.

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

- /home/lemur013/Programing/PAMSI/PAMSI/Lab3/Queue.hh

## 3.15 Stack< type > Class Template Reference

Inherits [IStack< type >](#).

### Public Member Functions

- virtual void [Push](#) (type item)  
*Add element to the top of the stack.*
- virtual type [Pop](#) () throw (EmptyStackException)  
*Remove element from top of the stack and return it.*
- virtual type [Top](#) () throw (EmptyStackException)  
*Return item from top of the stack.*
- virtual int [Size](#) ()  
*Return size of the stack.*
- virtual bool [IsEmpty](#) ()  
*Check whether the stack is empty.*
- void [ShowStack](#) ()

### Protected Attributes

- [Array](#)< type > \* [stackArray](#)
- int [actualIndex](#)

### 3.15.1 Detailed Description

`template<typename type>class Stack< type >`

Definition at line 9 of file Stack.hh.

### 3.15.2 Member Function Documentation

**3.15.2.1** `template<typename type > virtual bool Stack< type >::IsEmpty ( ) [inline], [virtual]`

Check whether the stack is empty.

#### Returns

true - If stack is empty.  
false - If stack is not empty.

Implements [IStack< type >](#).

Definition at line 73 of file Stack.hh.

**3.15.2.2** `template<typename type > virtual type Stack< type >::Pop ( ) throw EmptyStackException) [inline], [virtual]`

Remove element from top of the stack and return it.

#### Returns

Item from top which will be removed.

#### Exceptions

<a href="#">EmptyStackException</a>	when user tries to use this method on empty stack.
-------------------------------------	--

Implements [IStack< type >](#).

Definition at line 41 of file Stack.hh.

**3.15.2.3** `template<typename type > virtual void Stack< type >::Push ( type item ) [inline], [virtual]`

Add element to the top of the stack.

#### Parameters

<i>in</i>	<i>item</i>	- Item which will be added.
-----------	-------------	-----------------------------

Implements [IStack< type >](#).

Definition at line 27 of file Stack.hh.

**3.15.2.4** `template<typename type > virtual int Stack< type >::Size ( ) [inline], [virtual]`

Return size of the stack.

#### Returns

Size of the stack.

Implements [IStack< type >](#).

Definition at line 68 of file Stack.hh.



3.15.2.5 `template<typename type > virtual type Stack< type >::Top ( ) throw EmptyStackException) [inline],  
[virtual]`

Return item from top of the stack.

#### Returns

Item from top of the stack.

#### Exceptions

<a href="#">EmptyStackException</a>	when user tries to use this method on empty stack.
-------------------------------------	--

Implements [IStack< type >](#).

Definition at line 55 of file Stack.hh.

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

- /home/lemur013/Programing/PAMSI/PAMSI/Lab3/Stack.hh

## 3.16 TimeCounter Class Reference

Inherits [IStoper](#).

### Public Member Functions

- virtual void **Start** ()
- virtual void **Stop** ()
- virtual long **GetElapsedTime** ()
- virtual bool **DumpToFile** ()

### 3.16.1 Detailed Description

Definition at line 11 of file TimeCounter.hh.

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

- /home/lemur013/Programing/PAMSI/PAMSI/Lab3/TimeCounter.hh



# Index

Add  
    IList, 7  
    List, 13  
Array< type >, 5  
ArrayRunner, 5

Dequeue  
    IQueue, 9  
    Queue, 16

EmptyQueueException, 6  
EmptyStackException, 6

Enqueue  
    IQueue, 9  
    Queue, 16

Front  
    IQueue, 10  
    Queue, 16

Get  
    IList, 7  
    List, 14

IArray< type >, 6  
IList  
    Add, 7  
    Get, 7  
    IsEmpty, 8  
    Remove, 8  
    Size, 8  
IList< type >, 7  
IQueue  
    Dequeue, 9  
    Enqueue, 9  
    Front, 10  
    IsEmpty, 10  
    Size, 10  
IQueue< type >, 9  
IRunnable, 10  
IStack  
    IsEmpty, 11  
    Pop, 11  
    Push, 12  
    Size, 12  
    Top, 12  
IStack< type >, 11  
IStoper, 12  
IndexOutOfRangeException, 8  
IsEmpty  
    IList, 8  
    IQueue, 10  
    IStack, 11  
    List, 14  
    Queue, 16  
    Stack, 18

List  
    Add, 13  
    Get, 14  
    IsEmpty, 14  
    Remove, 14  
    Size, 14  
List< type >, 13  
ListTest, 15

Pop  
    IStack, 11  
    Stack, 18

Push  
    IStack, 12  
    Stack, 18

Queue  
    Dequeue, 16  
    Enqueue, 16  
    Front, 16  
    IsEmpty, 16  
    Size, 17  
Queue< type >, 15

Remove  
    IList, 8  
    List, 14

Size  
    IList, 8  
    IQueue, 10  
    IStack, 12  
    List, 14  
    Queue, 17  
    Stack, 18

Stack  
    IsEmpty, 18  
    Pop, 18  
    Push, 18  
    Size, 18  
    Top, 18  
Stack< type >, 17

TimeCounter, 19  
Top

IStack, [12](#)

Stack, [18](#)