JavaScript is disabled on your browser.

Frame Alert

This document is designed to be viewed using the frames feature. If you see this message, you are using a non-frame-capable web client. Link to Non-frame version.

JavaScript is disabled on your browser.
Skip navigation links

- Package
- Class
- Tree
- Deprecated
- Index
- Help

- Prev Class
- Next Class

- Frames
- No Frames

- All Classes

- Summary:
- Nested |
- Field |
- Constr |
- Method

- Detail:
- Field |
- Constr |
- Method

Interface BoundedDeque<E>

- ---

```
public interface BoundedDeque<E>
```

A BoundedDeque is a sequential structure with restricted access and limited capacity.

Access is available only at the ends of the structure: `addFront(E)`, `E removeFront()`, and `E peekFront()` operate on the front of the list; `addBack(E)`, `E removeBack()`, and `E peekBack()` operate on the back of the list.

(A sequential structure which, like BoundedDeque, permits access and modification only at the ends is sometimes called a "deque", pronounced "deck", which is short for "double-ended queue.")

An implementation of BoundedDeque must allow duplicate elements, but must not permit `null` elements, since some of the methods use `null` as a signaling return value.

In addition to the methods required in the definition of this interface, a class implementing this interface should provide a public constructor with a single argument of type `int`, which specifies the capacity of the BoundedDeque. The constructor should throw an IllegalArgumentException if the specified capacity is negative.

- ♦
  ## Method Summary

  All Methods Instance Methods Abstract Methods

Modifier and Type	Method and Description
`boolean`	`addBack(E e)` Adds the specified element to the back of this BoundedDeque.
`boolean`	`addFront(E e)` Adds the specified element to the front of this BoundedDeque.
`int`	`capacity()` Returns the capacity of this BoundedDeque, that is, the maximum number of elements it can hold.
`E`	`peekBack()` Returns the element at the back of this BoundedDeque, or `null` if there was no such element.
`E`	`peekFront()` Returns the element at the front of this BoundedDeque, or `null` if there was no such element.
`E`	`removeBack()` Removes the element at the back of this BoundedDeque.
`E`	`removeFront()` Removes the element at the front of this BoundedDeque.
`int`	`size()` Returns the number of elements in this BoundedDeque.

- ♦
  ## Method Detail

  ### ◊ capacity

```
int capacity()
```

Returns the capacity of this BoundedDeque, that is, the maximum number of elements it can hold.
PRECONDITION: none
POSTCONDITION: the BoundedDeque is unchanged.
Returns:
the capacity of this BoundedDeque

◊ size

```
int size()
```

Returns the number of elements in this BoundedDeque.
PRECONDITION: none
POSTCONDITION: the BoundedDeque is unchanged.
Returns:
the number of elements in this BoundedDeque

◊ addFront

```
boolean addFront(E e)
```

Adds the specified element to the front of this BoundedDeque. Returns true if the operation succeeded, else false.
PRECONDITION: the BoundedDeque's size is less than its capacity.
POSTCONDITION: the element is now the front element in this BoundedDeque, none of the other elements have been changed, and the size is increased by 1.
Parameters:
`e` - the element to add to the front of the list
Returns:
`true` if the element was added, else `false`.
Throws:
`java.lang.NullPointerException` - if the specified element is null, and size is less than capacity

◊ addBack

```
boolean addBack(E e)
```

Adds the specified element to the back of this BoundedDeque. Returns true if the operation succeeded, else false.
PRECONDITION: the BoundedDeque's size is less than its capacity.
POSTCONDITION: the element is now the back element in this BoundedDeque, none of the other elements have been changed, and the size is increased by 1.
Parameters:
`e` - the element to add to the back of the list
Returns:
`true` if the element was added, else `false`.
Throws:
`java.lang.NullPointerException` - if the specified element is null, and size is less than capacity

◊ removeFront

```
E removeFront()
```

Removes the element at the front of this BoundedDeque. Returns the element removed, or `null` if there was no such element.
PRECONDITION: the BoundedDeque's size is greater than zero.
POSTCONDITION: the front element in this BoundedDeque has been removed,

none of the other elements have been changed, and the size is decreased by 1.
Returns:
    the element removed, or `null` if the size was zero.

◊ removeBack

```
E removeBack()
```

Removes the element at the back of this BoundedDeque. Returns the element removed, or `null` if there was no such element.
PRECONDITION: the BoundedDeque's size is greater than zero.
POSTCONDITION: the back element in this BoundedDeque has been removed, none of the other elements have been changed, and the size is decreased by 1.
Returns:
    the element removed, or `null` if the size was zero.

◊ peekFront

```
E peekFront()
```

Returns the element at the front of this BoundedDeque, or `null` if there was no such element.
PRECONDITION: the BoundedDeque's size is greater than zero.
POSTCONDITION: The BoundedDeque is unchanged.
Returns:
    the element at the front, or `null` if the size was zero.

◊ peekBack

```
E peekBack()
```

Returns the element at the back of this BoundedDeque, or `null` if there was no such element.
PRECONDITION: the BoundedDeque's size is greater than zero.
POSTCONDITION: The BoundedDeque is unchanged.
Returns:
    the element at the back, or `null` if the size was zero.

Skip navigation links

- Package
- Class
- Tree
- Deprecated
- Index
- Help

- Prev Class
- Next Class

- Frames
- No Frames

- All Classes

- Summary:
- Nested |
- Field |
- Constr |
- Method

- Detail:
- Field |
- Constr |
- Method

JavaScript is disabled on your browser.
Skip navigation links

- Package
- Class
- Tree
- Deprecated
- Index
- Help

- Prev Class
- Next Class

- Frames
- No Frames

- All Classes

- Summary:
- Nested |
- Field |
- Constr |
- Method

- Detail:
- Field |
- Constr |
- Method

Interface BoundedQueue<E>

-

```
public interface BoundedQueue<E>
```

An interface that specifies the familiar queue abstraction, but with limited capacity.

In addition to the methods required in the definition of this interface, a class implementing this interface should provide a public constructor with a single argument of type `int`, which specifies the capacity of the BoundedQueue. The constructor should throw an IllegalArgumentException if the specified capacity is negative.

- ♦
  ### Method Summary

  All Methods Instance Methods Abstract Methods

  Modifier and **Method and Description**

Type	
`int`	`capacity()` Returns the capacity of this BoundedQueue, that is, the maximum number of elements it can hold.
`E`	`dequeue()` Removes the element at the head of this BoundedQueue.
`boolean`	`enqueue(E e)` Adds the specified element to the tail of this BoundedQueue.
`E`	`peek()` Returns the element at the head of this BoundedQueue, or `null` if there was no such element.
`int`	`size()` Returns the number of elements in this BoundedQueue.

- ♦
  ## Method Detail

◊ capacity

```
int capacity()
```

Returns the capacity of this BoundedQueue, that is, the maximum number of elements it can hold.
PRECONDITION: none
POSTCONDITION: the BoundedQueue is unchanged.
Returns:
    the capacity of this BoundedQueue

◊ size

```
int size()
```

Returns the number of elements in this BoundedQueue.
PRECONDITION: none
POSTCONDITION: the BoundedQueue is unchanged.
Returns:
    the number of elements in this BoundedQueue

◊ enqueue

```
boolean enqueue(E e)
```

Adds the specified element to the tail of this BoundedQueue. Returns true if the operation succeeded, else false.
PRECONDITION: the BoundedQueue's size is less than its capacity.
POSTCONDITION: the element is now the tail element in this BoundedQueue, none of the other elements have been changed, and the size is increased by 1.
Parameters:
    `e` - the element to add to the queue

Returns:
: `true` if the element was added, else `false`.

Throws:
: `java.lang.NullPointerException` - if the specified element is null, and size is less than capacity

◊ dequeue

```
E dequeue()
```

Removes the element at the head of this BoundedQueue. Returns the element removed, or `null` if there was no such element.
PRECONDITION: the BoundedQueue's size is greater than zero.
POSTCONDITION: the head element in this BoundedQueue has been removed, none of the other elements have been changed, and the size is decreased by 1.

Returns:
: the element removed, or `null` if the size was zero.

◊ peek

```
E peek()
```

Returns the element at the head of this BoundedQueue, or `null` if there was no such element.
PRECONDITION: the BoundedQueue's size is greater than zero.
POSTCONDITION: The BoundedQueue is unchanged.

Returns:
: the element at the head, or `null` if the size was zero.

- Detail:
- Field |
- Constr |
- Method

JavaScript is disabled on your browser.
Skip navigation links

- Package
- Class
- Tree
- Deprecated
- Index
- Help

- Prev Class
- Next Class

- Frames
- No Frames

- All Classes

- Summary:
- Nested |
- Field |
- Constr |
- Method

- Detail:
- Field |
- Constr |
- Method

Interface BoundedStack<E>

- ---

```
public interface BoundedStack<E>
```

An interface that specifies the familiar stack abstraction, but with limited capacity.

In addition to the methods required in the definition of this interface, a class implementing this interface should provide a public constructor with a single argument of type `int`, which specifies the capacity of the BoundedStack. The constructor should throw an IllegalArgumentException if the specified capacity is negative.

- ♦
  ### Method Summary

  All Methods Instance Methods Abstract Methods

  Modifier and **Method and Description**

Type	
`int`	`capacity()` Returns the capacity of this BoundedStack, that is, the maximum number of elements it can hold.
`E`	`peek()` Returns the element at the top of this BoundedStack, or `null` if there was no such element.
`E`	`pop()` Removes the element at the top of this BoundedStack.
`boolean`	`push(E e)` Adds the specified element to the top of this BoundedStack.
`int`	`size()` Returns the number of elements in this BoundedStack.

Method Detail

◊ capacity

```
int capacity()
```

Returns the capacity of this BoundedStack, that is, the maximum number of elements it can hold.
PRECONDITION: none
POSTCONDITION: the BoundedStack is unchanged.
Returns:
the capacity of this BoundedStack

◊ size

```
int size()
```

Returns the number of elements in this BoundedStack.
PRECONDITION: none
POSTCONDITION: the BoundedStack is unchanged.
Returns:
the number of elements in this BoundedStack

◊ push

```
boolean push(E e)
```

Adds the specified element to the top of this BoundedStack. Returns true if the operation succeeded, else false.
PRECONDITION: the BoundedStack's size is less than its capacity.
POSTCONDITION: the element is now the top element in this BoundedStack, none of the other elements have been changed, and the size is increased by 1.
Parameters:
`e` - the element to add to the stack

Returns:
: `true` if the element was added, else `false`.

Throws:
: `java.lang.NullPointerException` - if the specified element is null, and size is less than capacity

◊ pop

```
E pop()
```

Removes the element at the top of this BoundedStack. Returns the element removed, or `null` if there was no such element.
PRECONDITION: the BoundedStack's size is greater than zero.
POSTCONDITION: the top element in this BoundedStack has been removed, none of the other elements have been changed, and the size is decreased by 1.

Returns:
: the element removed, or `null` if the size was zero.

◊ peek

```
E peek()
```

Returns the element at the top of this BoundedStack, or `null` if there was no such element.
PRECONDITION: the BoundedStack's size is greater than zero.
POSTCONDITION: The BoundedStack is unchanged.

Returns:
: the element at the top, or `null` if the size was zero.

- Detail:
- Field |
- Constr |
- Method

JavaScript is disabled on your browser.
Skip navigation links

- Package
- Class
- Tree
- Deprecated
- Index
- Help

- Prev
- Next

- Frames
- No Frames

- All Classes

Hierarchy For Package <Unnamed>

Interface Hierarchy

- BoundedDeque<E>
- BoundedQueue<E>
- BoundedStack<E>

Skip navigation links

- Package
- Class
- Tree
- Deprecated
- Index
- Help

- Prev
- Next

- Frames
- No Frames

- All Classes