Class Heap12<E extends java.lang.Comparable<? super E>>

```
java.lang.Object
    java.util.AbstractCollection<E>
        java.util.AbstractQueue<E>
            Heap12<E>
```

All Implemented Interfaces:

`java.lang.Iterable<E>`, `java.util.Collection<E>`, `java.util.Queue<E>`

```
public class Heap12<E extends java.lang.Comparable<? super E>>
extends java.util.AbstractQueue<E>
```

Heap12 class that implements an unbounded array-backed heap structure and is an extension of the Java Collections AbstractQueue class

The elements of the heap are ordered according to their natural ordering, Heap12 does not permit null elements. The top of this heap is the minimal or maximal element (called min/max) with respect to the specified natural ordering. If multiple elements are tied for min/max value, the top is one of those elements -- ties are broken arbitrarily. The queue retrieval operations poll, remove, peek, and element access the element at the top of the heap.

A Heap12 is unbounded, but has an internal capacity governing the size of an array used to store the elements on the queue. It is always at least as large as the queue size. As elements are added to a Heap12, its capacity grows automatically. The details of the growth policy are not specified.

This class and its iterator implements the optional methods of the Iterator interface (including remove()). The Iterator provided in method iterator() is not guaranteed to traverse the elements of the Heap12 in any particular order.

Note that this implementation is not synchronized. Multiple threads should not access a Heap12 instance concurrently if any of the threads modifies the Heap12.

Field Summary

Fields

Modifier and Type	Field	Description
`int`	`FIVE`	
`int`	`NEGATIVEINDEX`	

Constructor Summary

Constructors

Constructor	Description
`Heap12()`	0-argument constructor.
`Heap12(boolean isMaxHeap)`	Constructor to build a min or max heap
`Heap12(int capacity, boolean isMaxHeap)`	Constructor to build a heap with specified initial capacity min or max heap
`Heap12(Heap12<E> toCopy)`	Copy constructor.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
`java.util.Iterator<E>`	`iterator()`	
`boolean`	`offer(E e)`	insert an element in the heap PRECONDITION: element is comparable to other elements in the heap POSTCONDITION: size is increased by 1
`E`	`peek()`	Retrieve, but not remove, the element at top of heap.
`E`	`poll()`	remove and return the highest priority element PRECONDITION: size is greater than zero POSTCONDITION: size is decreased by 1, highest priority element is removed.
`int`	`size()`	Size of the heap

Methods inherited from class java.util.AbstractCollection

contains, containsAll, isEmpty, remove, removeAll, retainAll, toArray, toArray, toString

Methods inherited from class java.util.AbstractQueue

add, addAll, clear, element, remove

Methods inherited from interface java.util.Collection

contains, containsAll, equals, hashCode, isEmpty, parallelStream, remove, removeAll, removeIf, retainAll, spliterator, stream, toArray, toArray

Methods inherited from interface java.lang.Iterable

forEach

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

FIVE

```
public final int FIVE
```

See Also:

Constant Field Values

NEGATIVEINDEX

```
public final int NEGATIVEINDEX
```

See Also:

Constant Field Values

Constructor Detail

Heap12

```
public Heap12()
```

0-argument constructor. Creates an empty Heap12 with capacity of 5 elements, and is a min-heap

Heap12

```
public Heap12(boolean isMaxHeap)
```

Constructor to build a min or max heap

Parameters:

`isMaxHeap` - if true, this is a max-heap, else a min-heap. Initial capacity of the heap should be 5.

Heap12

```
public Heap12(int capacity,
              boolean isMaxHeap)
```

Constructor to build a heap with specified initial capacity min or max heap

Parameters:

`capacity` - initial capacity of the heap.

`isMaxHeap` - if true, this is a max-heap, else a min-heap

Heap12

```
public Heap12(Heap12<E> toCopy)
```

Copy constructor. Creates Heap12 with a deep copy of the argument

Parameters:

`toCopy` - the heap that should be copied

Method Detail

size

```
public int size()
```

Size of the heap

Specified by:

`size` in interface `java.util.Collection<E extends java.lang.Comparable<? super E>>`

Specified by:

`size` in class `java.util.AbstractCollection<E extends java.lang.Comparable<? super E>>`

Returns:

the number of elements stored in the heap

iterator

```
public java.util.Iterator<E> iterator()
```

Specified by:

`iterator` in interface `java.util.Collection<E extends java.lang.Comparable<? super E>>`

Specified by:

`iterator` in interface `java.lang.Iterable<E extends java.lang.Comparable<? super E>>`

Specified by:

`iterator` in class `java.util.AbstractCollection<E extends java.lang.Comparable<? super E>>`

Returns:

an Iterator for the heap

peek

```
public E peek()
```

Retrieve, but not remove, the element at top of heap.

Returns:

Element at top of heap. Do not remove return `null` if the heap is empty

poll

```
public E poll()
```

remove and return the highest priority element
PRECONDITION: size is greater than zero
POSTCONDITION: size is decreased by 1, highest priority element is removed.

Returns:

Element at top of heap. And remove it from the heap. return `null` if the heap is empty

offer

```
public boolean offer(E e)
```

insert an element in the heap
PRECONDITION: element is comparable to other elements in the heap
POSTCONDITION: size is increased by 1

Returns:

true

Throws:

`java.lang.ClassCastException` - if the class of the element prevents it from being added

`java.lang.NullPointerException` - if the specified element is null

`java.lang.IllegalArgumentException` - if some property of the element keeps it from being added.