AbstractQueue<E> class.

AbstractQueue<E> class provides skeletal implementation so some methods of Queue<E> interface. To learn about Queue<E> interface click here.

**Code for this class is referred and take from Javadocs.**

AbstractQueue<E> class definition

So AbstractQueue<E> class extends [AbstractCollection<E>](http://data-structure-learning.blogspot.com/2015/05/java-collection-part-10.html) and it also implements [Queue<E>](http://data-structure-learning.blogspot.com/2015/05/queue-interface.html) interface.

**public** **abstract** **class** AbstractQueue<E> **extends** AbstractCollection<E> **implements** Queue<E>

Constructor of AbstractQueue<E> class

Constructor of this class is protected by definition. And why is that? The reason is that the invocation can be done by subclass constructors only.

**protected** AbstractQueue() {

}

add(E e)

This method is used to add element e that is specified in parameter. It uses offer(E e) method of Queue<E> interface to add element in Queue.

Offer method return true for successful insertion in queue and false if element is not inserted in queue.

Add() method throws IllegalStateException(“Queue full”) if we try to insert element in Queue which is full.

**public** **boolean** add(E e) {

**if** (offer(e))

**return** **true**;

**else**

**throw** **new** IllegalStateException("Queue full");

}

remove()

remove() method is used to remove and retrieve the head of this queue. Well, poll() method also does the same but this both method are different in one way.

If there are no elements in Queue then **remove()** will **throw** an exception **NoSuchElementException,** but poll() method will return null.

**public** E remove() {

E x = poll();

**if** (x != **null**)

**return** x;

**else**

**throw** **new** NoSuchElementException();

}

element()

element() method is used to retrieve the peek value from the Queue without removing it. peek() method also does the same but there is one difference.

element() throws NoSuchElementException if there are no elements in Queue. On the contrary, peek() returns null if there are no elements in Queue.

**public** E element() {

E x = peek();

**if** (x != **null**)

**return** x;

**else**

**throw** **new** NoSuchElementException();

}

clear()

removes all the elements from this Queue. Queue will be empty after this method is called. clear() uses poll() to retrieve and remove the peek element from Queue.

**public** **void** clear() {

**while** (poll() != **null**)

;

}

addAll(Collection<? extends E> c)

This method is used to add all the elements of collection into Queue. If the specified collection is null then NullPointerException is thrown. If the Queue is modified than the modification flag is set to true. At the end the modification flag is returned to check whether Queue is changed or not.

**public** **boolean** addAll(Collection<? **extends** E> c) {

**if** (c == **null**)

**throw** **new** NullPointerException();

**if** (c == **this**)

**throw** **new** IllegalArgumentException();

**boolean** modified = **false**;

**for** (E e : c)

**if** (add(e))

modified = **true**;

**return** modified;

}

That’s all on AbstractQueue<E> class. Next we will see AbstractSet<E> class.