Deque interface in Java with Example

The java.util.Deque interface is a subtype of the java.util.Queue interface. The Deque is related to the double-ended queue that supports addition or removal of elements from either end of the data structure, it can be used as a queue (first-in-first-out/FIFO) or as a stack (last-in-first-out/LIFO).

Methods of deque:

1. **add(element):** Adds an element to the tail.
2. **addFirst(element):** Adds an element to the head.
3. **addLast(element):** Adds an element to the tail.
4. **offer(element):** Adds an element to the tail and returns a boolean to explain if the insertion was successful.
5. **offerFirst(element):** Adds an element to the head and returns a boolean to explain if the insertion was successful.
6. **offerLast(element):** Adds an element to the tail and returns a boolean to explain if the insertion was successful.
7. **iterator():** Returna an iterator for this deque.
8. **descendingIterator():** Returns an iterator that has the reverse order for this deque.
9. **push(element):** Adds an element to the head.
10. **pop(element):** Removes an element from the head and returns it.
11. **removeFirst():** Removes the element at the head.
12. **removeLast():** Removes the element at the tail.

```
// Java program to demonstrate working of
// Deque in Java
import java.util.*;

public class DequeExample
{
    public static void main(String[] args)
    {
```

```
        Deque deque = new LinkedList<>();

        // We can add elements to the queue in various ways
        deque.add("Element 1 (Tail)"); // add to tail
        deque.addFirst("Element 2 (Head)");
        deque.addLast("Element 3 (Tail)");
        deque.push("Element 4 (Head)"); //add to head
        deque.offer("Element 5 (Tail)");
        deque.offerFirst("Element 6 (Head)");
        deque.offerLast("Element 7 (Tail)");

        System.out.println(deque + "\n");

        // Iterate through the queue elements.
        System.out.println("Standard Iterator");
        Iterator iterator = deque.iterator();
        while (iterator.hasNext())
            System.out.println("\t" + iterator.next());


        // Reverse order iterator
        Iterator reverse = deque.descendingIterator();
        System.out.println("Reverse Iterator");
        while (reverse.hasNext())
            System.out.println("\t" + reverse.next());

        // Peek returns the head, without deleting
        // it from the deque
        System.out.println("Peek " + deque.peek());
        System.out.println("After peek: " + deque);

        // Pop returns the head, and removes it from
        // the deque
        System.out.println("Pop " + deque.pop());
        System.out.println("After pop: " + deque);

        // We can check if a specific element exists
        // in the deque
        System.out.println("Contains element 3: " +
                        deque.contains("Element 3 (Tail)"));

        // We can remove the first / last element.
        deque.removeFirst();
        deque.removeLast();
        System.out.println("Deque after removing " +
                           "first and last: " + deque);

    }
}
```

Run on IDE

Output:

```
[Element 6 (Head), Element 4 (Head), Element 2 (Head), Element 1 (Tail), Element 3 (Tail),
 Element 5 (Tail), Element 7 (Tail)]

Standard Iterator
        Element 6 (Head)
        Element 4 (Head)
        Element 2 (Head)
        Element 1 (Tail)
        Element 3 (Tail)
        Element 5 (Tail)
```

```
        Element 7 (Tail)
Reverse Iterator
        Element 7 (Tail)
        Element 5 (Tail)
        Element 3 (Tail)
        Element 1 (Tail)
        Element 2 (Head)
        Element 4 (Head)
        Element 6 (Head)
Peek Element 6 (Head)
After peek: [Element 6 (Head), Element 4 (Head), Element 2 (Head), Element 1 (Tail),
 Element 3 (Tail), Element 5 (Tail), Element 7 (Tail)]
Pop Element 6 (Head)
After pop: [Element 4 (Head), Element 2 (Head), Element 1 (Tail), Element 3 (Tail),
 Element 5 (Tail), Element 7 (Tail)]
Contains element 3: true
Deque after removing first and last: [Element 2 (Head), Element 1 (Tail), Element 3 (Tail),
 Element 5 (Tail)]
```

Deque in Collection Hierarchy

collectionjava

Image Source : https://www.ntu.edu.sg/home/ehchua/programming/java/J5c_Collection.html

This article is contributed by **Pratik Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GATE CS Corner Company Wise Coding Practice

Java Java-Collections

Recommended Posts:

NavigableMap Interface in Java with Example
SortedMap Interface in Java with Examples
Queue Interface In Java
PriorityQueue Class in Java
Map interface in Java with examples
Java.lang.String.compareTo()