**Java Deque Interface**

The Deque interface of the Java collections framework provides the functionality of a double-ended queue. It extends the Queue interface.

## Working of Deque

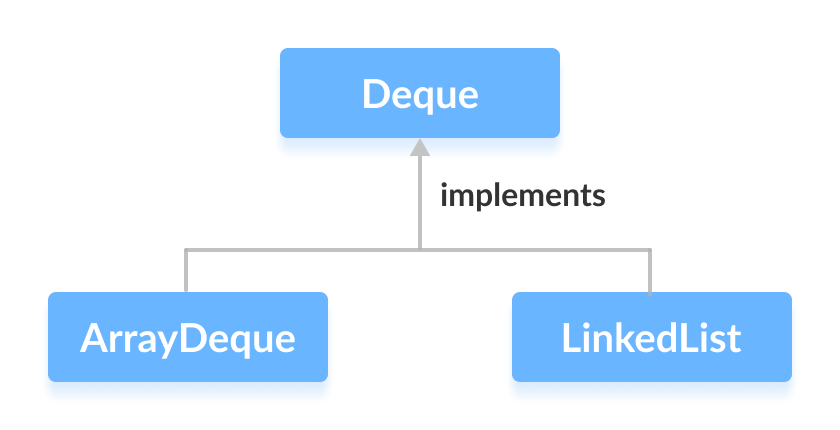
In a regular queue, elements are added from the rear and removed from the front. However, in a deque, we can **insert and remove elements from both front and rear**.



## Classes that implement Deque

In order to use the functionalities of the Deque interface, we need to use classes that implement it:

* [ArrayDeque](https://www.programiz.com/java-programming/arraydeque)
* [LinkedList](https://www.programiz.com/java-programming/linkedlist)



## How to use Deque?

In Java, we must import the java.util.Deque package to use Deque.

// Array implementation of Deque

Deque<String> animal1 = new ArrayDeque<>();

// LinkedList implementation of Deque

Deque<String> animal2 = new LinkedList<>();

Here, we have created objects animal1 and animal2 of classes ArrayDeque and LinkedList, respectively. These objects can use the functionalities of the Deque interface.

## Methods of Deque

Since Deque extends the Queue interface, it inherits all the methods of the Queue interface.

Besides methods available in the Queue interface, the Deque interface also includes the following methods:

* **addFirst()** - Adds the specified element at the beginning of the deque. Throws an exception if the deque is full.
* **addLast()** - Adds the specified element at the end of the deque. Throws an exception if the deque is full.
* **offerFirst()** - Adds the specified element at the beginning of the deque. Returns false if the deque is full.
* **offerLast()** - Adds the specified element at the end of the deque. Returns false if the deque is full.
* **getFirst()** - Returns the first element of the deque. Throws an exception if the deque is empty.
* **getLast()** - Returns the last element of the deque. Throws an exception if the deque is empty.
* **peekFirst()** - Returns the first element of the deque. Returns null if the deque is empty.
* **peekLast()** - Returns the last element of the deque. Returns null if the deque is empty.
* **removeFirst()** - Returns and removes the first element of the deque. Throws an exception if the deque is empty.
* **removeLast()** - Returns and removes the last element of the deque. Throws an exception if the deque is empty.
* **pollFirst()** - Returns and removes the first element of the deque. Returns null if the deque is empty.
* **pollLast()** - Returns and removes the last element of the deque. Returns null if the deque is empty.

## Deque as Stack Data Structure

The Stack class of the Java Collections framework provides the implementation of the stack.

However, it is recommended to use Deque as a stack instead of the Stack class. It is because methods of Stack are synchronized.

Here are the methods the Deque interface provides to implement stack:

* push() - adds an element at the beginning of deque
* pop() - removes an element from the beginning of deque
* peek() - returns an element from the beginning of deque

## Implementation of Deque in ArrayDeque Class

import java.util.Deque;

import java.util.ArrayDeque;

class Main {

public static void main(String[] args) {

// Creating Deque using the ArrayDeque class

Deque<Integer> numbers = new ArrayDeque<>();

// add elements to the Deque

numbers.offer(1);

numbers.offerLast(2);

numbers.offerFirst(3);

System.out.println("Deque: " + numbers);

// Access elements of the Deque

int firstElement = numbers.peekFirst();

System.out.println("First Element: " + firstElement);

int lastElement = numbers.peekLast();

System.out.println("Last Element: " + lastElement);

// Remove elements from the Deque

int removedNumber1 = numbers.pollFirst();

System.out.println("Removed First Element: " + removedNumber1);

int removedNumber2 = numbers.pollLast();

System.out.println("Removed Last Element: " + removedNumber2);

System.out.println("Updated Deque: " + numbers);

}

}

**Output**

Deque: [3, 1, 2]

First Element: 3

Last Element: 2

Removed First Element: 3

Removed Last Element: 2

Updated Deque: [1]

To learn more about ArrayDeque, visit Java ArrayDeque.