Preferred: do this in Java. The code of `DLList<E>` and the target `Deque<E>` are provided in Java.
If you want to write in a different language, you must (i) clear it with the TAs and me, (ii) translate all the code provided to that language (there is not much), and (iii) be prepared to provide a running demo with translations of our tests, either on a laptop at school or on Zoom or Google chat with screen sharing.

For the ADAPTEE. you are given a class `DLlist<E>`. It implements a doubly-linked list. The methods you are given are these and you DO NOT have to implement them:

`public void delete(int index)`	Deletes the element at the specified `index` position in this doubly-linked list. Does nothing if the element is not present in the doubly-linked list. Parameter: `index` - the index of the element to be removed Throws: `IndexOutOfBoundsException` if the `index` is out of range (`index < 0 || index >= length()`)
`public E get(int index)`	Returns the element at the specified position in this doubly-linked list. Parameters: `index` - index of the element to return Returns: the element at the specified position in this doubly-linked list Throws: `IndexOutOfBoundsException` if the `index` is out of range (`index < 0 || index >= length()`)
`public void insertList(E e, int index)`	Inserts the specified element at the specified `index` position in this doubly-linked list. If the `index` is `list.length()`, then the insertion is at the end of the doubly-linked list. Parameters: `e` element to be inserted `index` index at which the specified element is to be inserted. Throws: `IndexOutOfBoundsException` if the `index` is out of range (`index < 0 || index > length()`)
`public int length()`	Returns the number of elements in this doubly-linked list

You are also given the *partial* interface for `Deque<E>`, which is the TARGET interface for the ADAPTER

`void addFirst(E e)`	Inserts the specified element at the head (the front) of this deque. Parameter: `e` - the element to be inserted at the start of the deque
`void addLast(E e)`	Inserts the specified element after the final element of this deque. Parameter: `e` - the element to be inserted at the end of the deque
`E peek()`	Retrieves, but does not remove, the head of the deque (the first element of this deque) or returns null if this deque is empty.
`E poll()`	Retrieves *and removes*, the first element of the deque (the first element of this deque) or returns null if this deque is empty.
`E peekLast()`	Retrieves, but does not remove, the final element of the deque (the last element of this deque) or returns null if this deque is empty.
`E pollLast()`	Retrieves *and removes*, the final element of the deque (the last element of this deque) or returns null if this deque is empty.
`public int size()`	Returns the number of elements in this deque

Question 1 Adapt the adaptee class `DLlist<E>` to make a `Deque<E>` with the given interface using an **Object Adapter**:
`public class DequeImpl1<E> implements Deque<E>`
Remember all the methods of an interface must be public in the implementation.
Question 2 Adapt the adaptee `DLlist<E>` to make a `Deque<E>` with the given interface using a **Class Adapter**:
`public class DequeImpl2<E> extends DLlist<E> implements Deque<E>`
Remember all the methods of an interface must be public in the implementation.