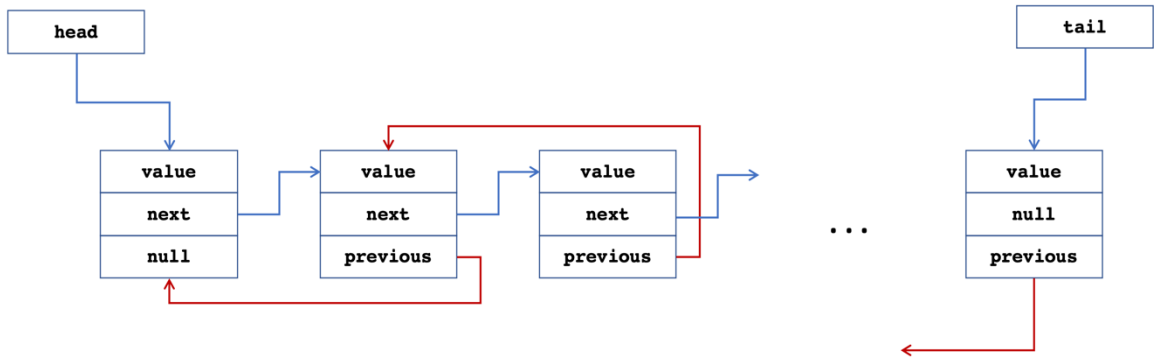
Programming and Data Structures
Assignment 6: Linked and Doubly Linked Lists

Objectives of the assignment

Students should demonstrate the following abilities:

1. Implement a generic doubly linked list data structure using the linked list data structure
2. Determine the time complexity of the doubly linked list operations
3. Define and use generic methods that manipulate doubly linked lists
4. Test their doubly linked list class and generic methods

A doubly linked list is a list where the nodes are linked forward and backward as illustrated in the figure below.

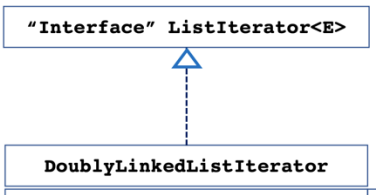


You are asked to implement the doubly linked list data structure and test it. To implement the new data structure, follow the steps below.

1. Use the implementation of the class **`LinkedList`** used in ALA #8.
2. Copy the class **`LinkedList`** to create a new generic class **`DoublyLinkedList`** that has the same public interface (methods) as the class **`LinkedList`**.
3. Modify the inner class **`Node`** by adding a data member **`previous`** of type **`Node`** as illustrated above. Modify the constructor of the class **`Node`** to initialize the data member **`previous`** to **`null`** as well.
4. Make the necessary modifications to all the methods in the class **`DoublyLinkedList`** to maintain the forward and backward links between the nodes. Make sure you take advantage of the backward access to reduce the time complexity of the method **`removeLast()`**.

5. Add a private method **`int indexOf(Node node)`** in the class **`DoublyLinkedList`** that returns the index of **`node`** in the list or **`-1`** if **`node`** is not found in the list.

6. Replace the inner class **`LinkedListIterator`** with the class **`DoublyLinkedListIterator`** that implements the interface **`ListIterator<E>`** and has the following UML diagram. Provide a definition for each method in the class as described below.



-current: Node	Holds the reference to the current element in the list
+DoublyLinkedListIterator()	Initialize **current** to head
+DoubleLinkedListIterator(int index)	Initialize **current** to the element at position **index.** It throws an **IndexOutOfBoundsException** if **index**<0 or **index**>size
+hasNext(): boolean	returns **true** if **current** is not equal to **null**
+hasPrevious(): boolean	returns **true** if **current** is not equal to **null**
+next()(): E	returns the value of **current** and moves to the next element or throws a **NoSuchElementException** if **current** is **null**
+previous(): E	returns the value of **current** and moves to the previous element or throws a **NoSuchElementException** if **current** is **null**
+nextIndex(): int	returns the index of **current** in the list or **-1** if **current** is equal to **null**
+previousIndex(): int	returns the index of **current** in the list or **-1** if **current** is equal to **null**
+set(E e): void	sets the value of **current** to **e** or throws **NoSuchElementException** if **current** is **null**
+add(E e): void	throws an **UnsupportedOperationException**
+remove(): void	throws an **UnsupportedOperationException**

7. Overload the method **`iterator()`** using the two following headers:

   `public ListIterator<E> iterator()` : returns an instance of **`DoublyLinkedListIterator`** with its data member **`current`** pointing to the head of the list.

   `public listIterator<E> iterator(int index)`: returns an instance of **`DoublyLinkedListIterator`** with its data member **`current`** pointing to the element at position **`index`**. For example, if **`index`** is equal to **`size()-1`**, the iterator starts from the end of the list (**`tail`**).

8. Determine the time complexity of all the methods in class **`DoublyLinkedList`** including the methods inside the inner class **`DoublyLinkedListIterator`**. Include the **big-O notation** of each method as a comment before the method header.

Test your doubly linked list data structure in a main method by doing the following:

1. Create an instance of the class **`DoublyLinkedList`** for the type **`String`** and add the country names, from the file `countries.txt`, to the doubly linked list .
2. In the main class, define two generic methods **`printListForward`** and **`printListBackward`** with the following headers. The first should use a forward iterator and the second a backward iterator to print the elements of **`list`**.

```
public static <E> void printForward(DoublyLinkedList<E> list)
public static<E> void printBackward(DoublyLinkedList<E> list)
```

3. In the main method, call the two methods, **`printListForward`** and **`printListBackward`**, to print the elements of the doubly linked list in forward and backward order respectively.

Submit your Java files **`DoublyLinkedList.java`**, and **`Test.java`** on courseSite.