# Doubly Linked List

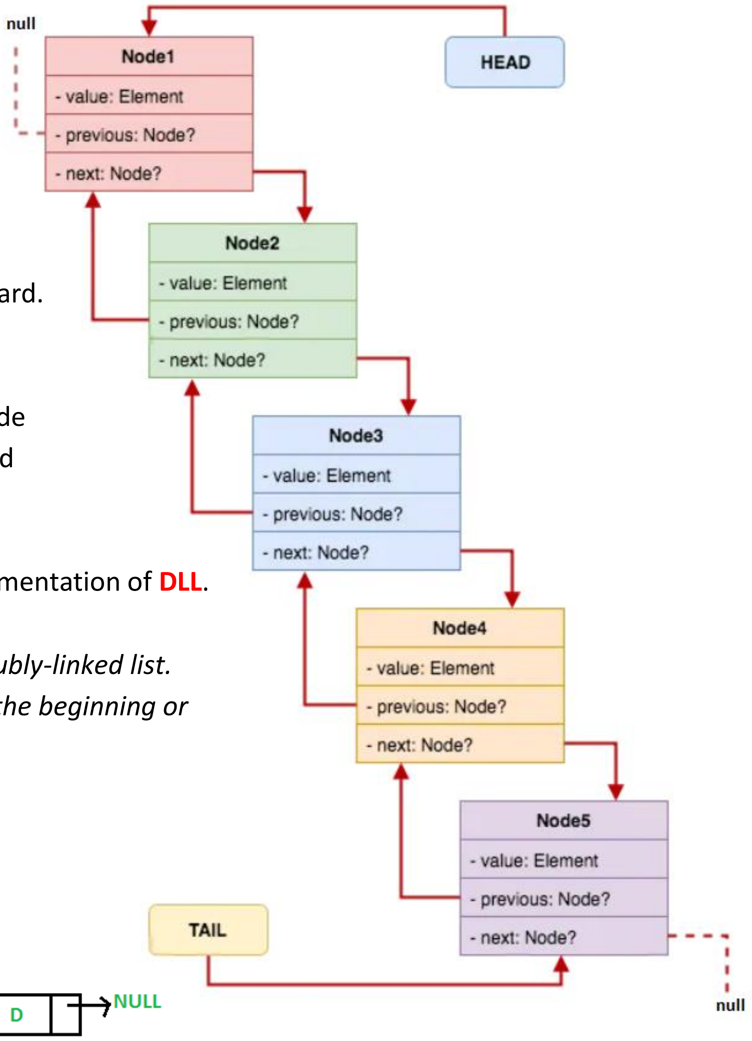
## 1. Motivation

The **singly** liked list (**SLL**) has the following disadvantages:

- It can be traversed only forward.
- Inserting a node **before** another node requires traversing from the beginning, even if a pointer to the node is given.
- Deleting a node requires traversing from the beginning as well.

## 2. Doubly Linked List

A node in a **doubly linked list** contains pointers to both next and previous nodes ->



**Advantages**:

- It can be traversed in both directions efficiently.
- Inserting a node before a given node is straightforward.
- Deleting a node is also straightforward.

**Disadvantages**:

- Significant memory overhead – two pointers per node
- All operations require an extra pointer to be updated

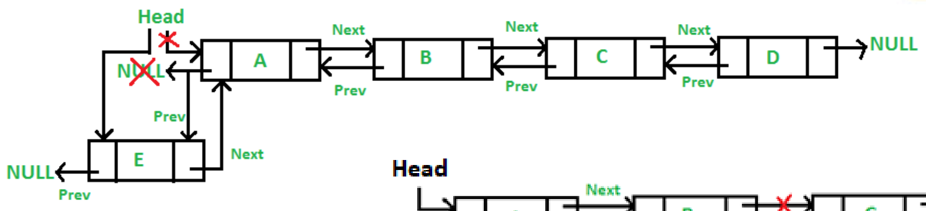
## 3. LinkedList in Java

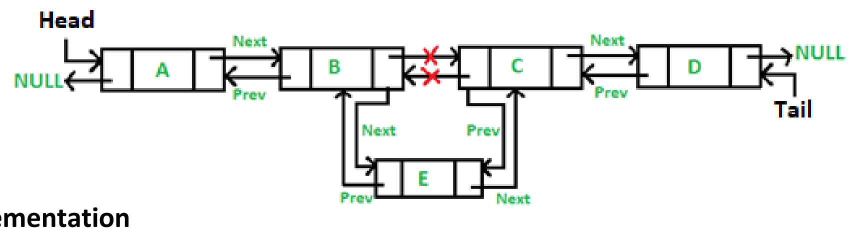
The Java implementation of `LinkedList` is actually implementation of **DLL**.

*"All of the operations perform as could be expected for a doubly-linked list. Operations that index into the list will traverse the list from the beginning or the end, whichever is closer to the specified index."*

## 4. Important Operations with DLL

- Delete a node
- Insert node at the beginning or in the middle:





## 5. Custom DLL Implementation

In the provided template the **`Node`** class is implemented as inner class to DLL. This is because nodes do not exist by themselves, but only as elements of DLL.

You have to optimize the private **`getNode()`** method and the **`add()`** method that inserts a node.

You also need to implement the remove methods.