CMSC 132: Object-Oriented Programming II

Doubly Linked List
private class Node<E>{
    private E data;
    private Node previous;
    private Node next;
    Node(E item){
        data = item;
    }
}

Node<String> n1 = new Node<>("alice");
Doubly Linked List

```java
Node<String> n1 = new Node("Alice");
Node<String> n2 = new Node("Bob");
Node<String> n3 = new Node("Cathy");
n1.next = n2;
n2.previous = n1;
n2.next = n3;
n3.previous = n2;
```
Insert a Node

```
Insert a Node

n1
alice

n2
bob

n3
cat

null

null
```
Insert a Node

Step 1:
\[ t.\text{next} = n2; \]
\[ t.\text{prev} = n1; \]
**Insert a Node**

**Step 1:**
\[
\begin{align*}
t.\text{next} &= n2; \\
t.\text{prev} &= n1;
\end{align*}
\]

**Step 2:**
\[
\begin{align*}
n1.\text{next} &= t; \\
n2.\text{prev} &= t;
\end{align*}
\]
Delete a Node

- We update two references to delete a node:
  - Next
  - Previous
Delete a node n2

1. \( n1.next = n2.next \)
2. \( n3.prev = n2.prev \)
3. n2 is garbage now
public class DoublyLinkedList<E> implements Iterable<E> {
    private int N;  // number of nodes
    private Node head;  // sentinel before the first node
    private Node tail;  // sentinel after the last node;
    DoublyLinkedList() {
        head = new Node();
        tail = new Node();
        head.next = tail;
        tail.previous = head;
    }

    private class Node {
        // Node class body here
    }
}
public void insert(E item){
    Node last = tail.previous;
    Node t = new Node(item);
    t.next = tail;
    t.previous = last;
    tail.previous = t;
    last.next = t;
    N++;
}
public void insert(E item) {
    Node last = tail.previous;
    Node t = new Node(item);
    t.next = tail;
    t.previous = last;
    tail.previous = t;
    last.next = t;
    N++;
}