

CMSC 132: Object-Oriented Programming II

Doubly Linked List

Doubly Linked List Node

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

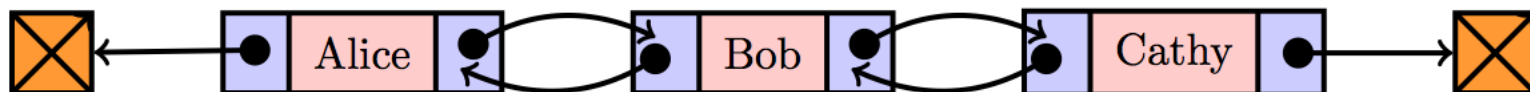


```
Node<String> n1 = new Node<>("alice");
```

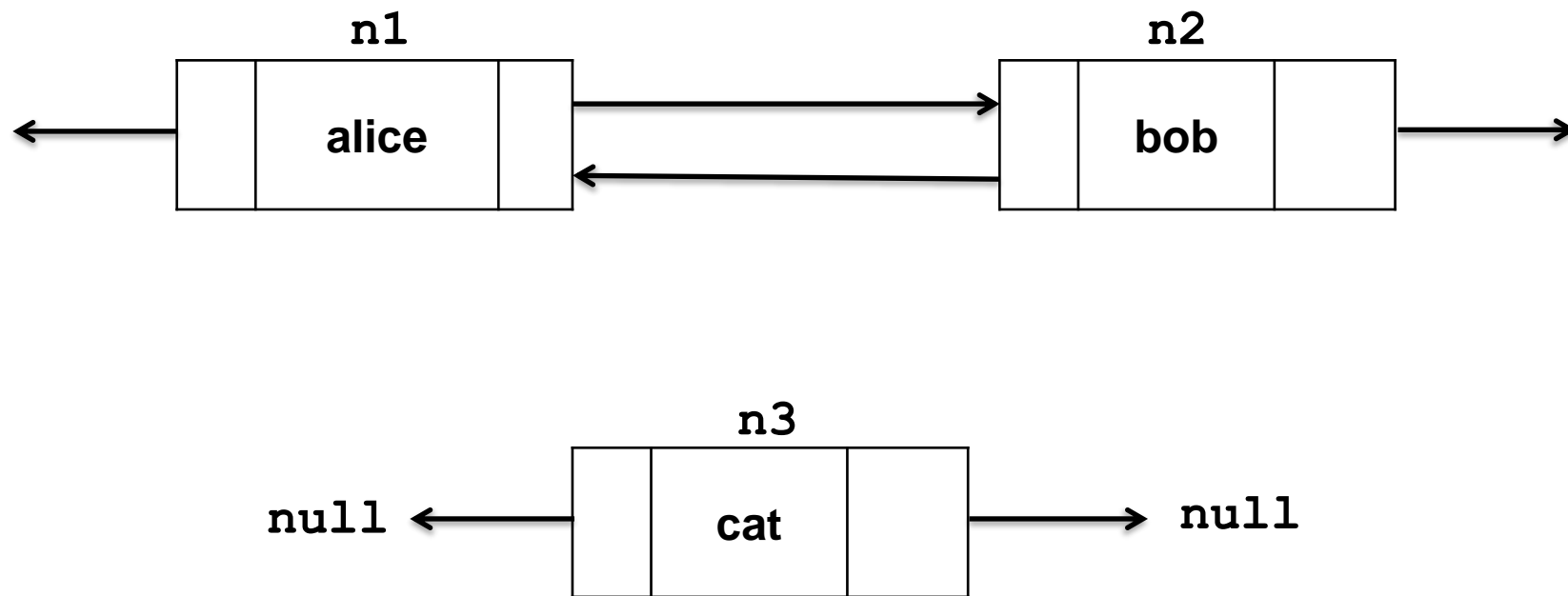


Doubly Linked List

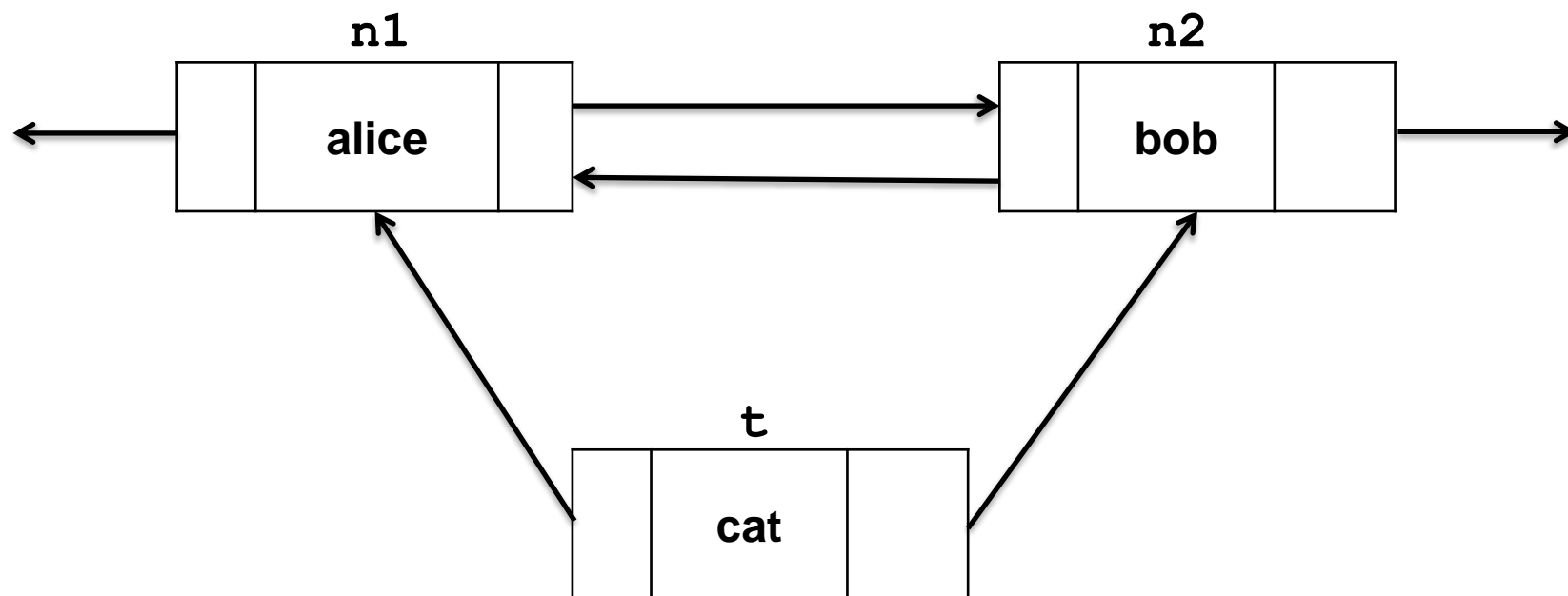
```
Node<String> n1 = new Node(" Alice" );  
Node<String> n2 = new Node(" Bob" );  
Node<String> n3 = new Node(" Cathy" );  
n1.next = n2;  
n2.previous = n1;  
n2.next = n3;  
n3.previous = n2;
```



Insert a Node



Insert a Node

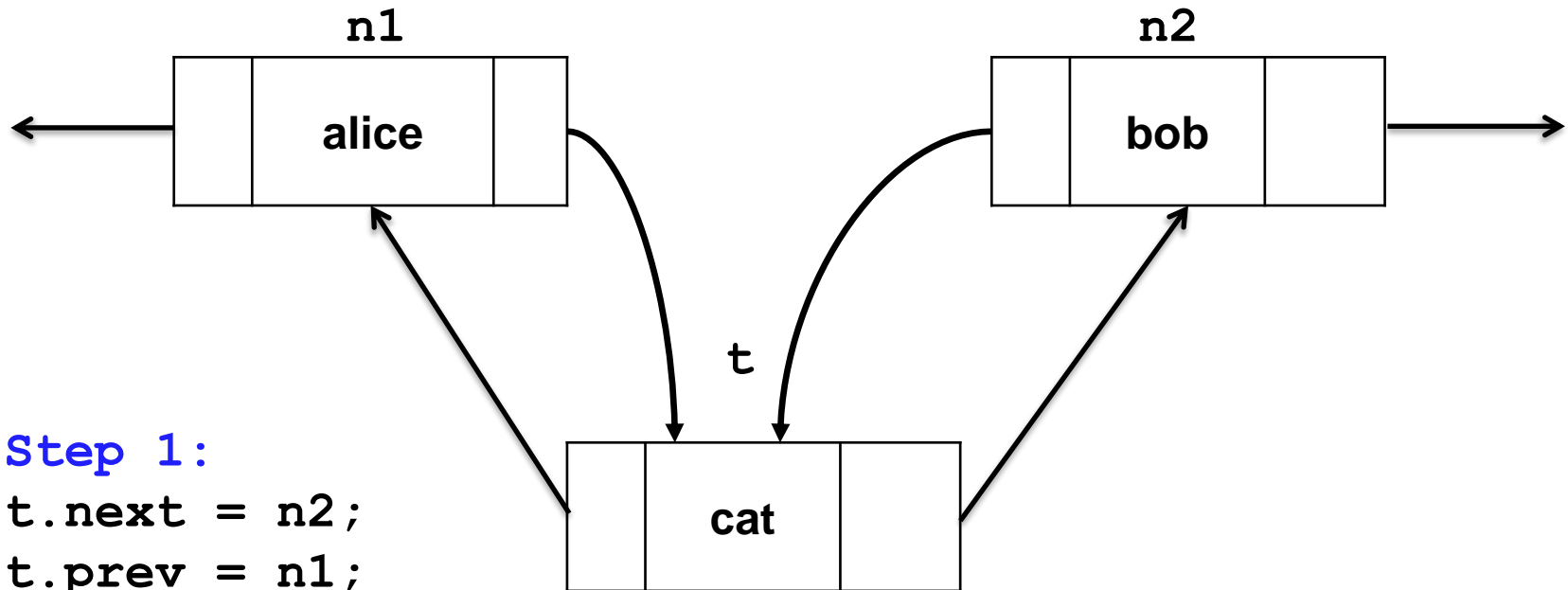


Step 1:

```
t.next = n2;
```

```
t.prev = n1;
```

Insert a Node



Step 1:

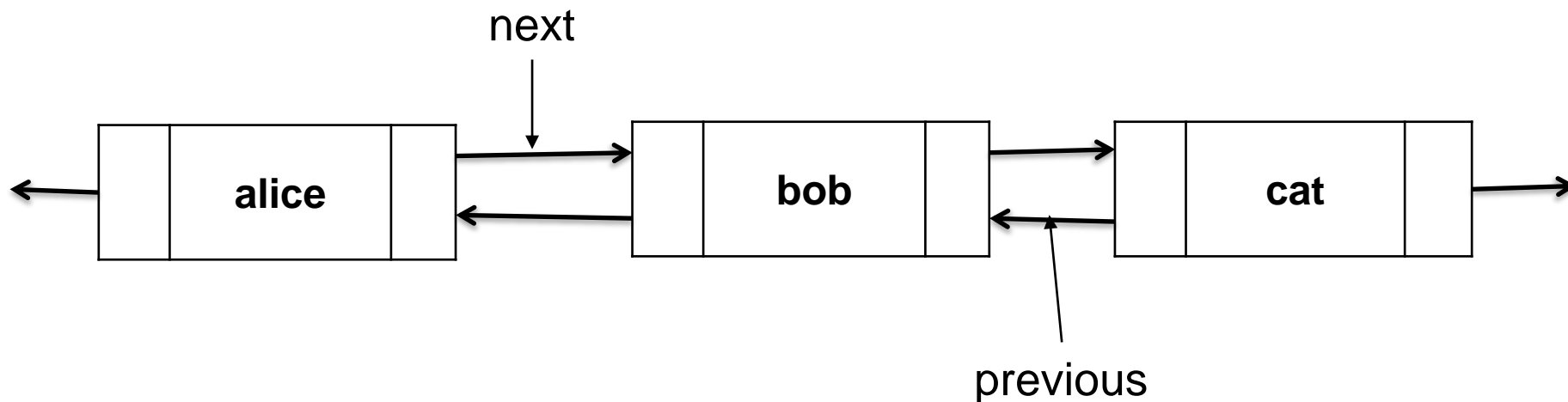
```
t.next = n2;  
t.prev = n1;
```

Step 2:

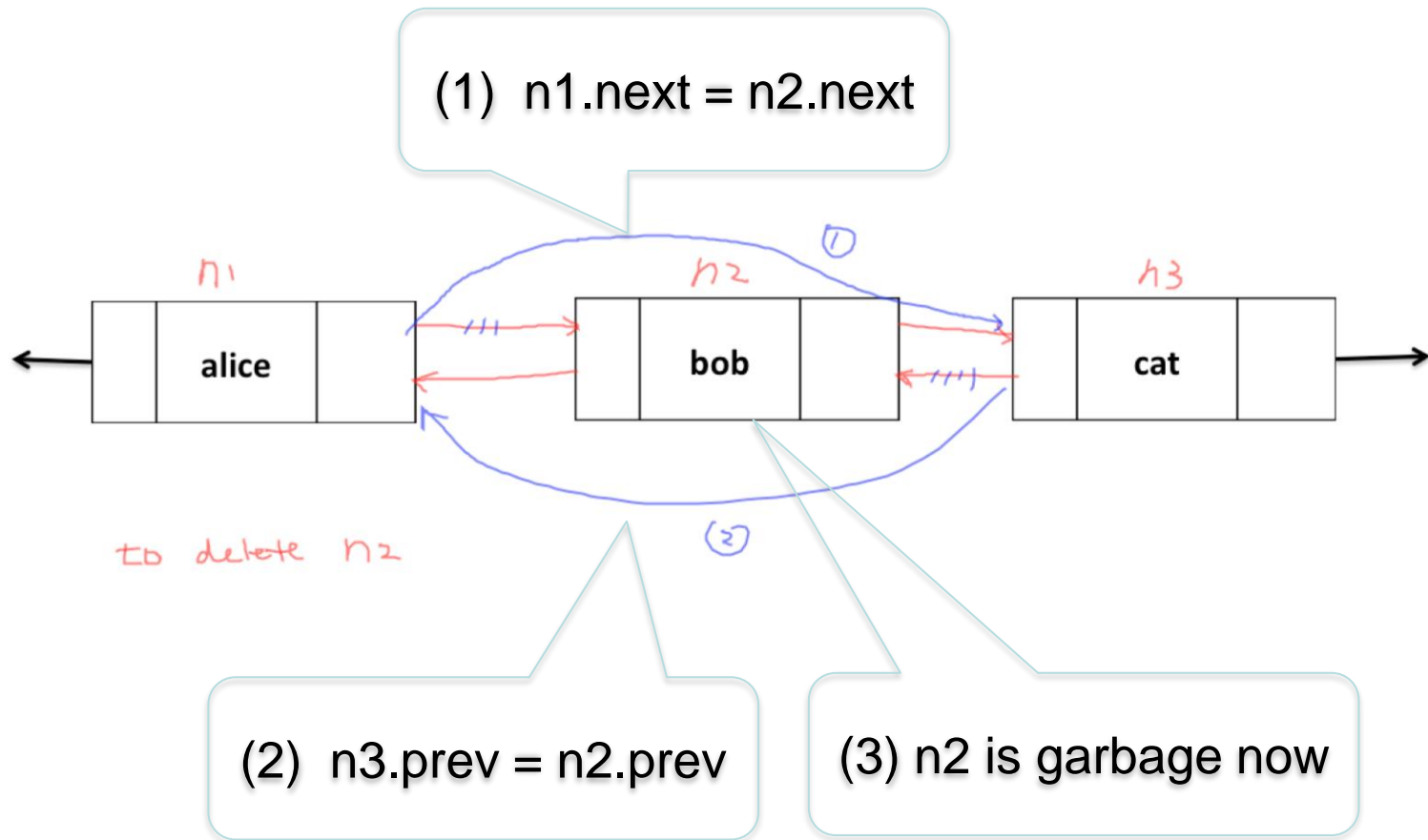
```
n1.next = t;  
n2.prev = t;
```

Delete a Node

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



Delete a node n2

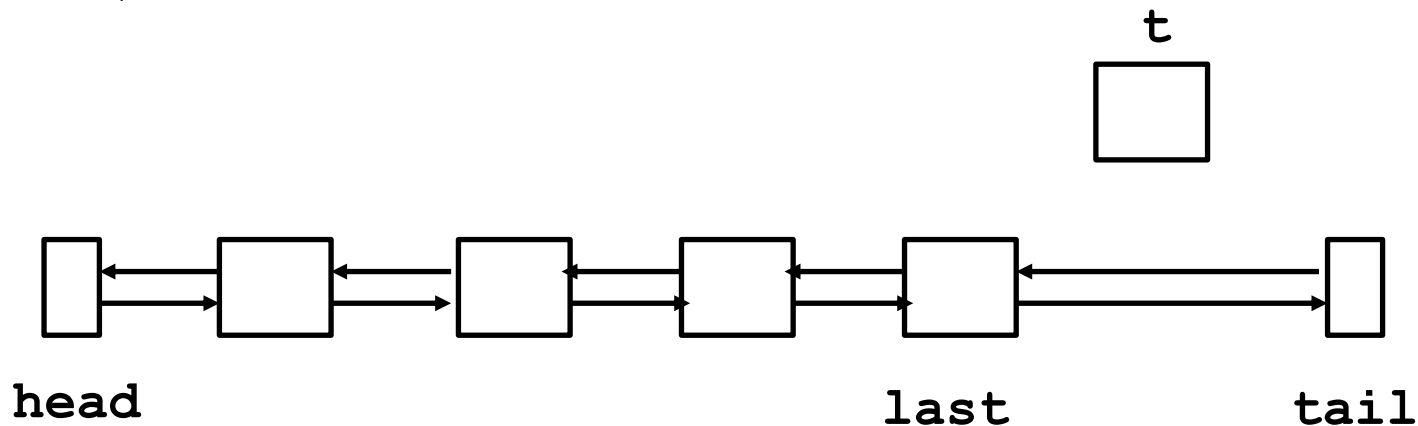


Double Linked List Class

```
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
    }
}
```

Insert a Node

```
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++;  
}
```



Insert a Node

```
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++;  
}
```

