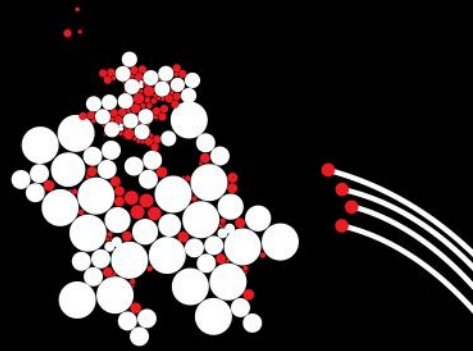
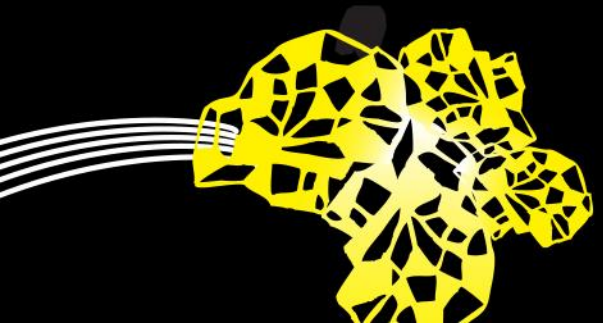
UNIVERSITY OF TWENTE.



List Implementation with LinkedList

Topic of Software Systems (TCS module 2)

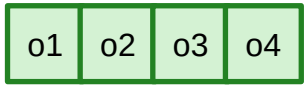
Lecturer: Faizan Ahmed



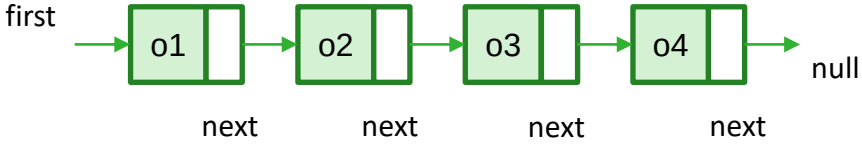
LINKED LIST

ALTERNATIVE STRUCTURE TO STORE LIST ELEMENTS

- List implementation stores information in a certain order
- We can store the information in contiguous areas (array) or in nodes, so that each node points to where the next piece of information is stored



Array list



Linked list

SIMPLE LINKED LIST

```
public class SimpleLinkedList<E> implements List<E>{

    private class Node {
        E element;
        Node next;

        public Node (E element) {
            this.element = element;
            this.next = null;
        }
    }

    private int size;
    private Node first;

    public SimpleLinkedList() {
        this.size = 0;
        this.first = null;
    }
```

nested Node class only used by SimpleLinkedList (private)

properties: size and first Node

SIMPLE LINKED LIST

QUERIES

private help method

```
private Node getNode (int pos) {
    Node p = first;
    for (int i = 0; i != pos; i++ )
        p = p.next;
    return p;
}
```

```
public int size() {
    return this.size;
}
```

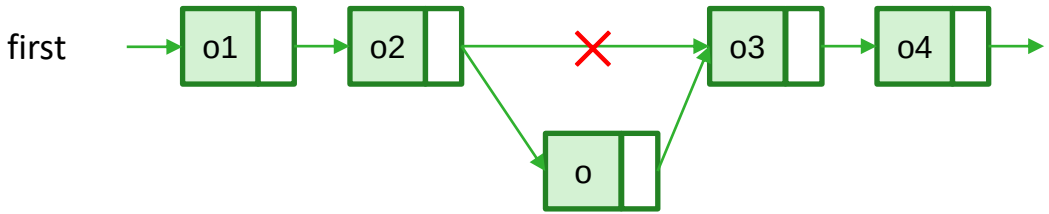
```
public boolean isEmpty() {
    return this.size == 0;
}
```

```
public boolean contains(Object o) {
    boolean result = false;
    for (Node p = first; p != null && !result; p=p.next) {
        if (p.element.equals(o))
            result = true;
    }
    return result;
}
```

SIMPLE LINKED LIST

COMMANDS

```
public void add(int index, E element) {
    Node newNode = new Node(element);
    if (index == 0) {
        newNode.next = first;
        first = newNode;
    } else {
        Node p = getNode(index - 1);
        newNode.next = p.next;
        p.next = newNode;
    }
    this.size++;
}
```

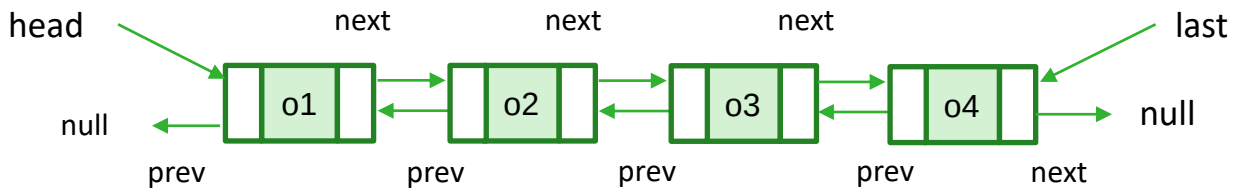


DOUBLE LINKED LIST

MOTIVATION

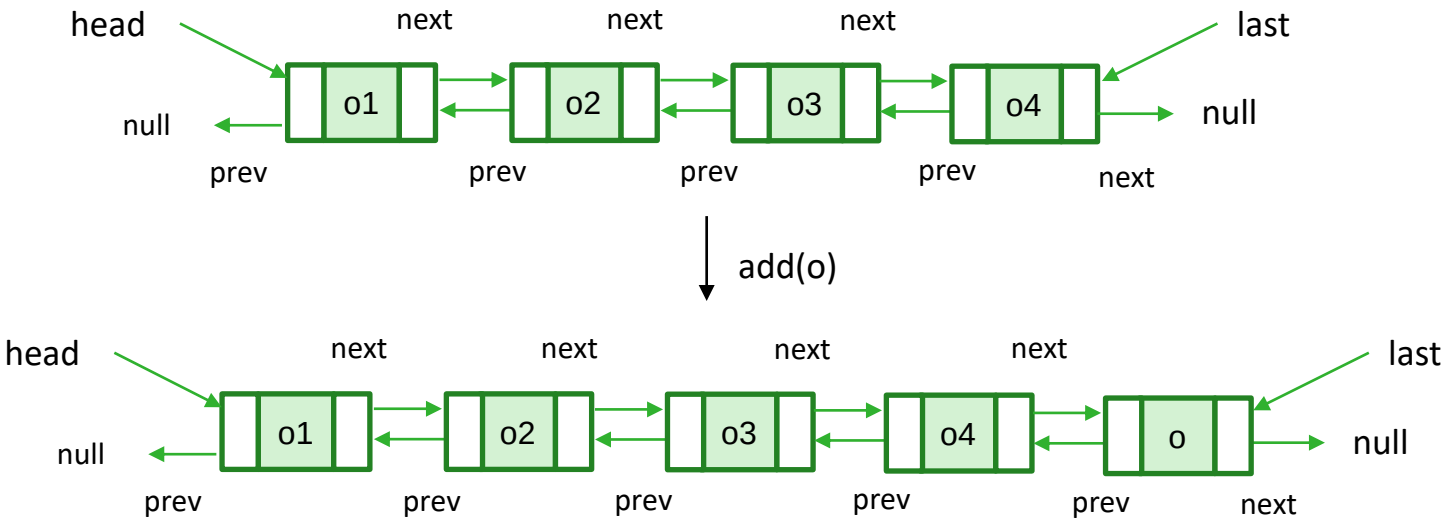
- In a linked list adding an element to the end of the list implies traversing the whole list
- To speed up this, double linked lists can be defined
- Node with pointers to both previous and next nodes

LinkedList<E> is defined like this!



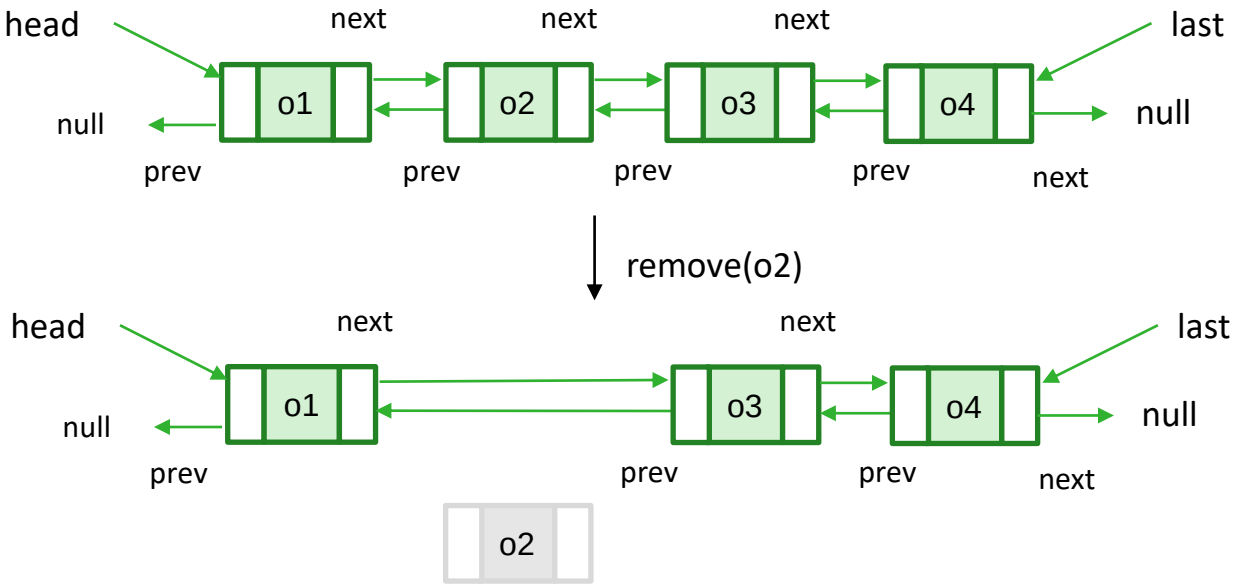
DOUBLE LINKED LIST

ADD ELEMENT (TO THE END)



DOUBLE LINKED LIST

REMOVE ELEMENT



ARRAY (DOUBLE) LINKED LISTS

COMPARISON

- In arrays it is easy to find an element, but difficult to insert or remove elements → a lot of copying!

- In (double) linked lists it is easy to insert or remove elements, but it is difficult to find elements → visit a lot of nodes before finding!

TAKE HOME MESSAGES

- Many different implementations of the List<E> interface are possible
- Implementations based on arrays, such as the ArrayList<E>, are more suitable when the elements are not often added and removed, and a lot of searching is done
- Implementations based on linked lists, such as the LinkedList<E>, are more suitable when elements are often added and removed, but searching is limited
- In practice, for your programs the differences are negligible!