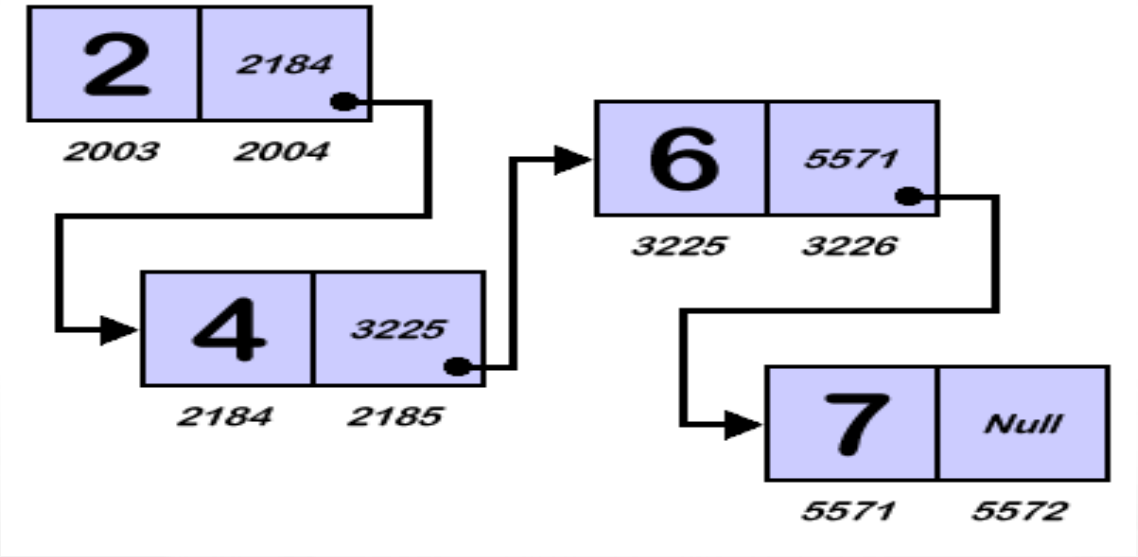
Udemy

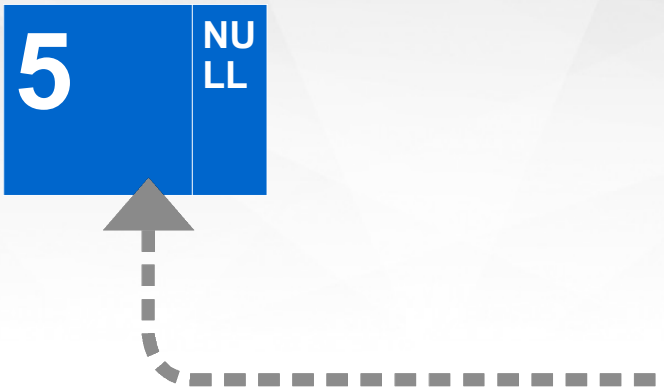
Algorithms and Data Structures in Java

Lecture: List

Instructors:
George Katsilidis
Nikos Katsilidis
Christos Topalidis



The List Node



Node :

public class Node {

private int data;

Node nextNode; }

Node :

public class Node<T> {

private T data;

Node<T> nextNode;

}

Insert At Front
case without Node


```
Node firstNode, lastNode;
public void insertAtFront(int 5){
if (isEmpty())
      firstNode = lastNode = new Node (5);
}
public Node ( int data ) {
       this.data = data;
       nextNode = null;  }
```

In Class List

In Class Node

Insert At Front case with Node


```
public void insertAtFront(int 12){
else
      firstNode = new Node (12 , firstNode);   <-- In Class List
}
public Node ( Tint data , Node next ) {
       this.data = data;                       <-- In class Node
       nextNode = next; }
```

Insert At Back
case without Node


```
Node firstNode, lastNode;
public void insertAtBack(int 5){
if (isEmpty())
      firstNode = lastNode = new Node (5);
}
public Node ( int data ) {
       this.data = data;
       nextNode = null;  }
```

In Class List

In Class Node

Insert At Front case with Node


```
public void insertAtBack(int 12){
else                                                    In Class List
    lastNode.nextNode = new Node (12);
    lastNode = lastNode.nextNode;      }
public Node ( Tint data , Node next ) {
       this.data = data;                          In class Node
       nextNode = next; }
```

Delete At-Back Front case without Node

```
public T removeFromFront() throws EmptyListException{
        if (isEmpty())
            throw new EmptyListException(name);
}


class EmptyListException extends Exception {
    public EmptyListException(){
        super("list");    }
    public EmptyListException(String name){
        super(name +" is empty");    }
}
```

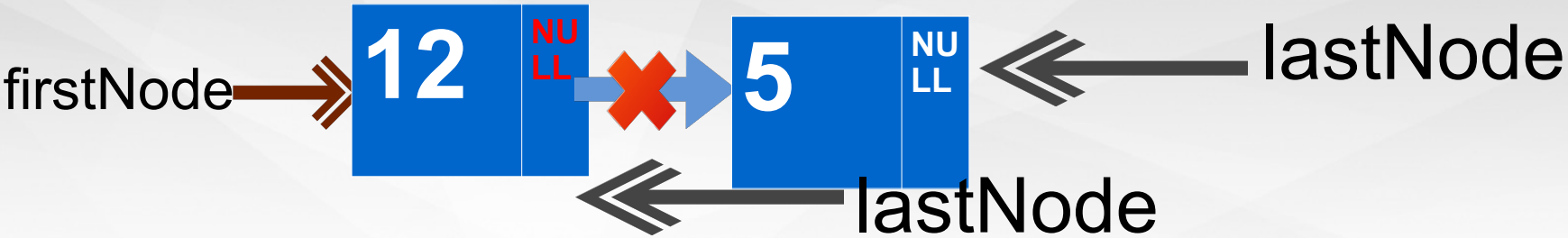
Delete At Front-Back case with one Node


```
public T removeFromFront() throws EmptyListException{
if(firstNode == lastNode)
      firstNode = lastNode = null;
}
```

Delete At Front
case of multiple Nodes


```
public T removeFromFront() throws EmptyListException{
if(firstNode == lastNode)
      firstNode = lastNode = null;
else
            firstNode = firstNode.getNext();
}
```

Delete At Back
case of multiple Nodes




```
public T removeFromBack() throws EmptyListException{
else{
     Node current = firstNode;
     while (current.getNext() != lastNode)
          current = current.getNext();
      lastNode = current;
      current.setNext(null); }
}
```