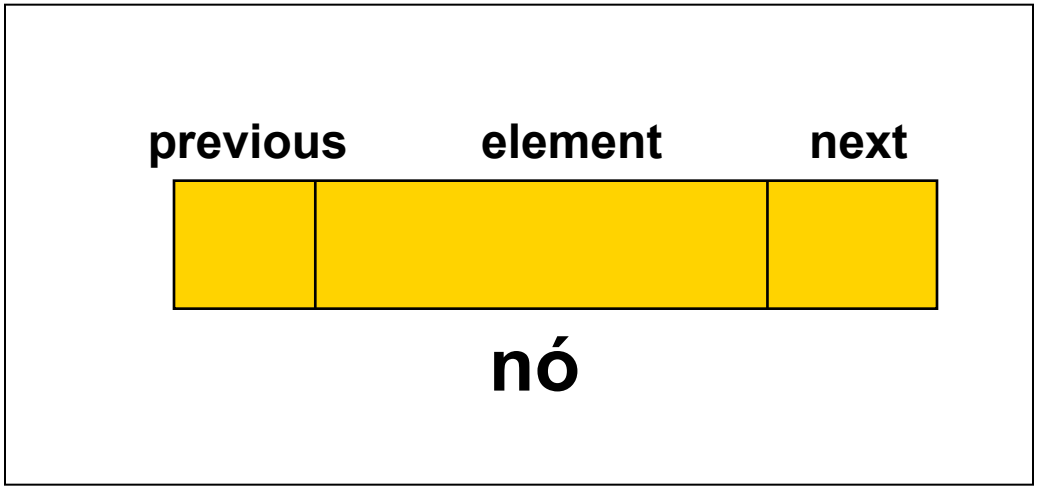
Listas Duplamente Encadeadas

Professor Mateus Raeder

UNISINOS
UNIVERSIDADE DO VALE DO RIO DOS SINOS

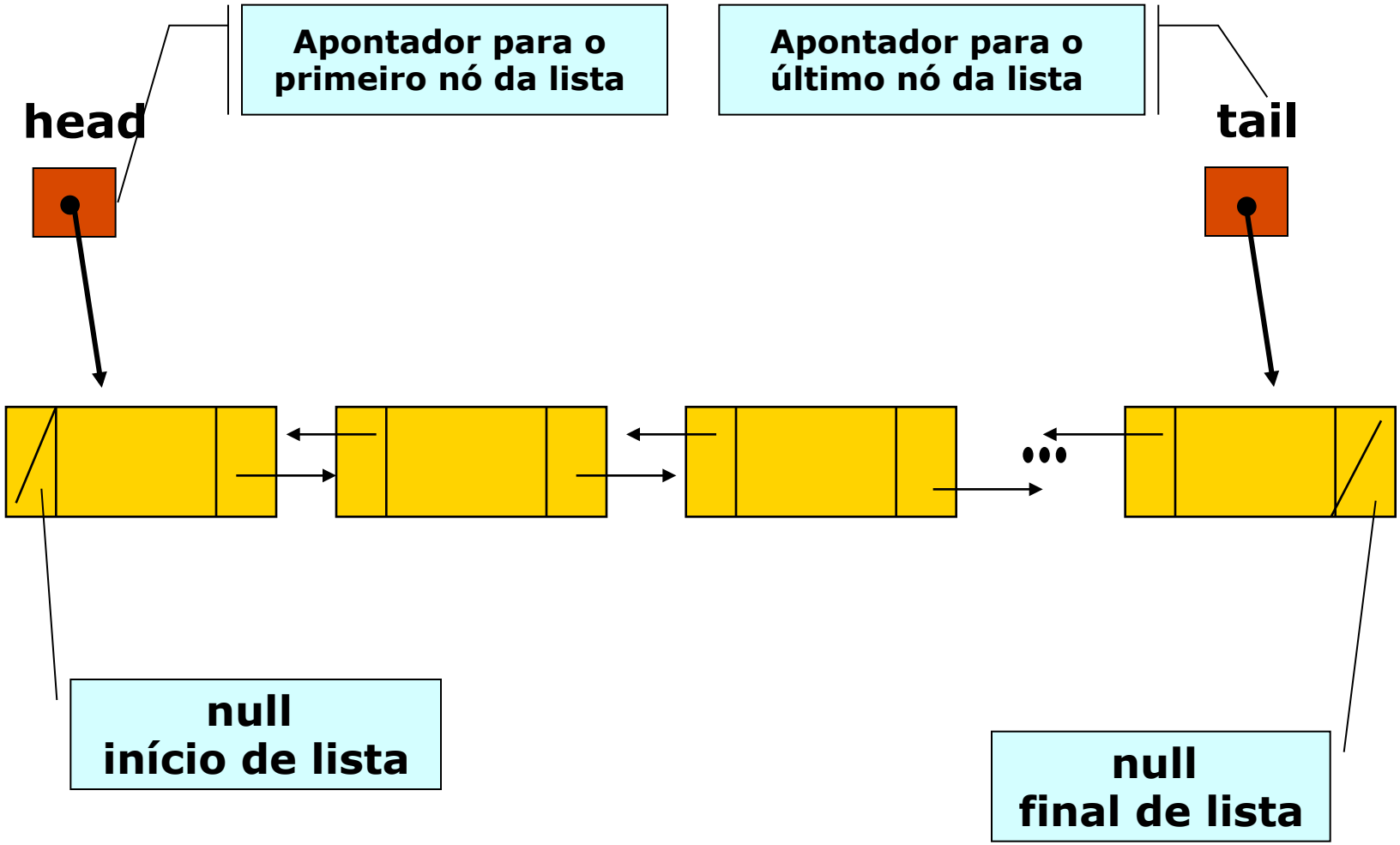
Listas Duplamente Encadeadas

- Cada nó possui dois ponteiros:



- Vantagem: simplifica certas operações e permite percorrer a lista nos dois sentidos.
- Desvantagem: gasta mais espaço do que a simplesmente encadeada (mais um ponteiro em cada nó) e pode tornar mais complexas certas operações.

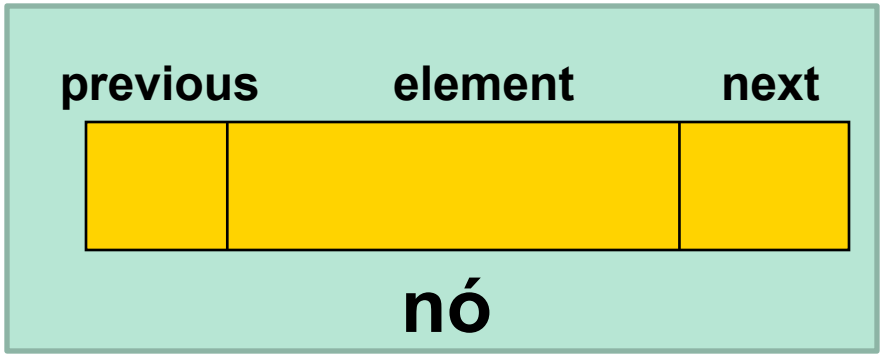
Lista encadeada com referência ao ultimo elemento da lista



classe DNode

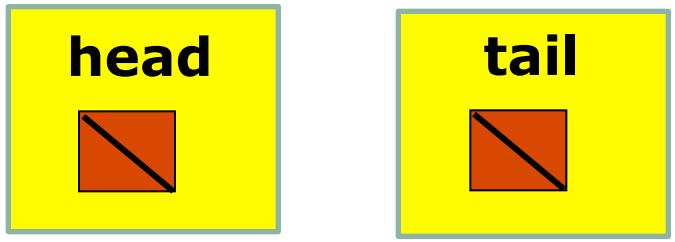
```
public class DNode<E> {
     private E element;
     private DNode<E> next;
     private DNode<E> previous;

public DNode(E element){
    this.element = element;
}

public E getElement() {

}
public void setElement(E element) {

}
```

```
public DNode<E> getNext() {

}
public void setNext(DNode<E> next) {

}
public DNode<E> getPrevious() {

}
public void setPrevious(DNode<E> previous)
{

}

}
```



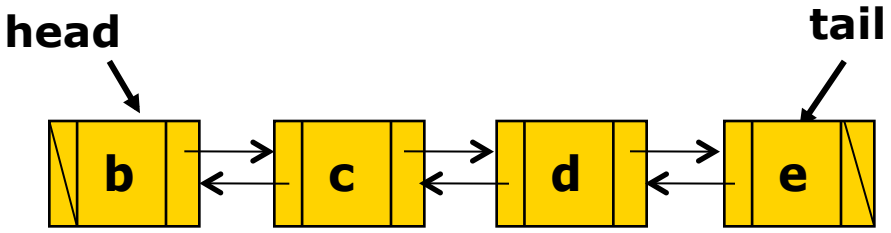
class DoublyLinkedList

```
public class DoublyLinkedList<E> implements List<E> {
    protected DNode<E> head; //nodo cabeça da lista
    protected DNode<E> tail; //nodo cauda da lista
    protected int numElements; //número de nodos da lista

    public DoublyLinkedList() {
        numElements = 0;
        head = tail = null;
    }

    public boolean isEmpty() {

    }

    public boolean isFull() { return false; }

    public int numElements() {

    }
```



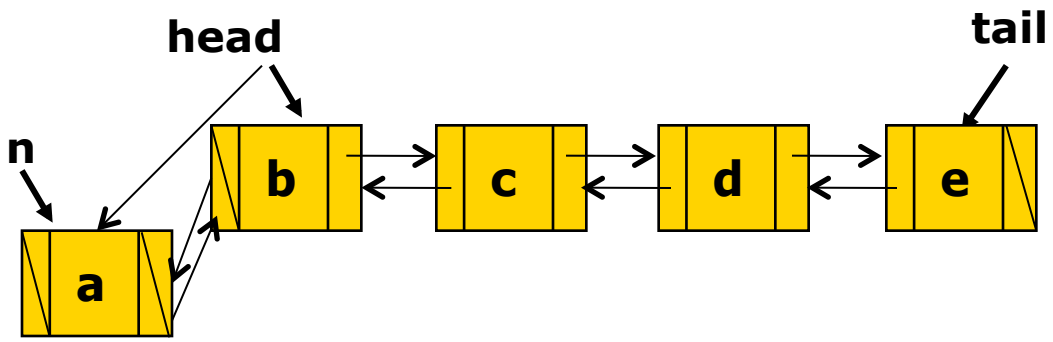
class DoublyLinkedList

```
public E get(int pos) {




}
```



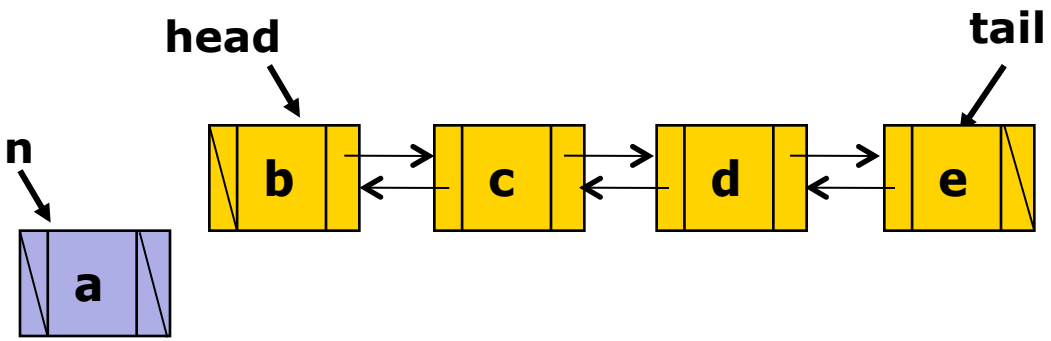
class DoublyLinkedList

```
public void insertFirst(E insertItem) {



}
```



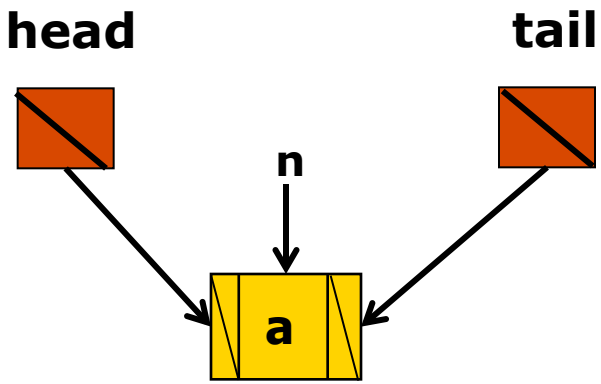
class DoublyLinkedList

```
public void insertFirst(E insertItem) {

}
```



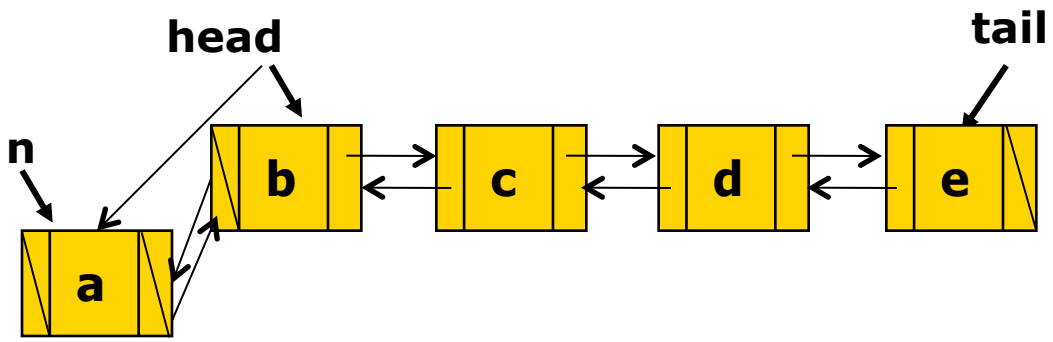
class DoublyLinkedList

```
public void insertFirst(E insertItem) {

}
```



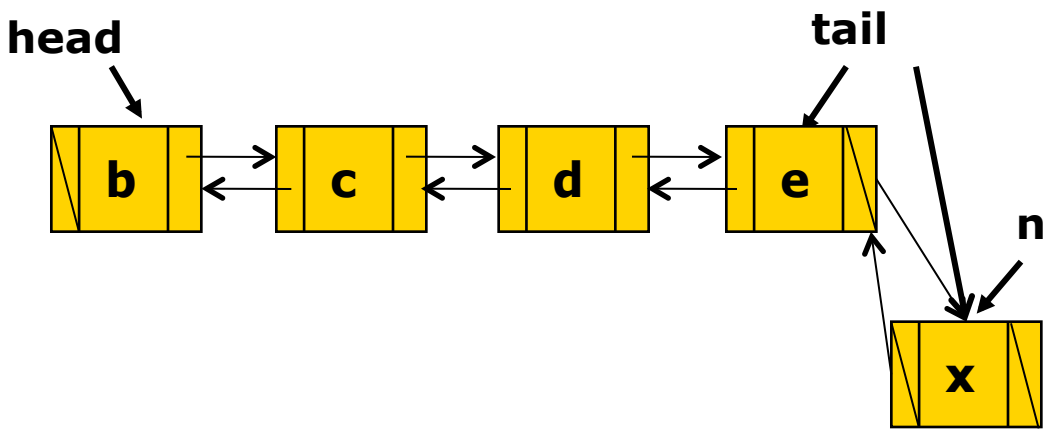
class DoublyLinkedList

```
public void insertFirst(E insertItem) {

}
```



class DoublyLinkedList

```
public void insertLast(E insertItem) {

}
```

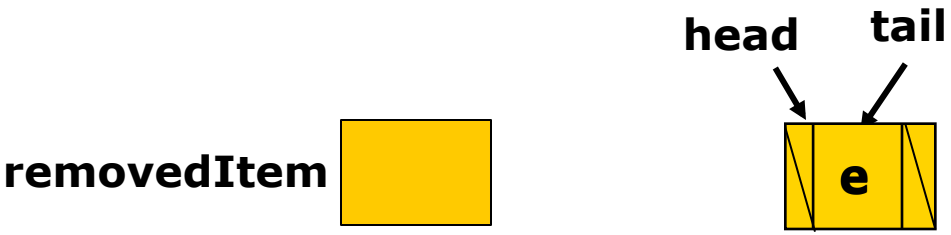


class DoublyLinkedList

```
public void insert(E insertItem, int pos) {



}
```

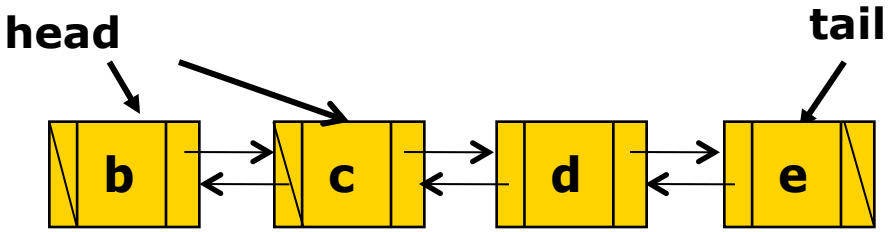
class DoublyLinkedList

```
public E removeFirst() throws UnderflowException {

}
```



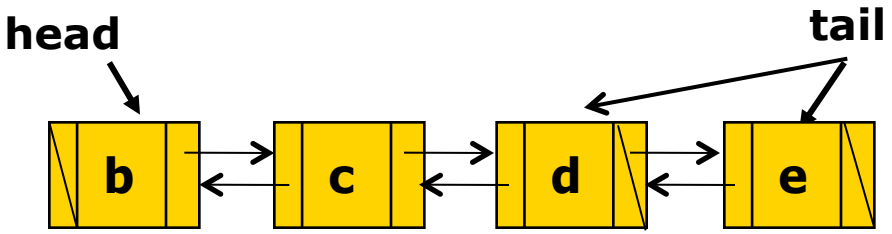
class DoublyLinkedList

```
public E removeFirst() throws UnderflowException {

}
```



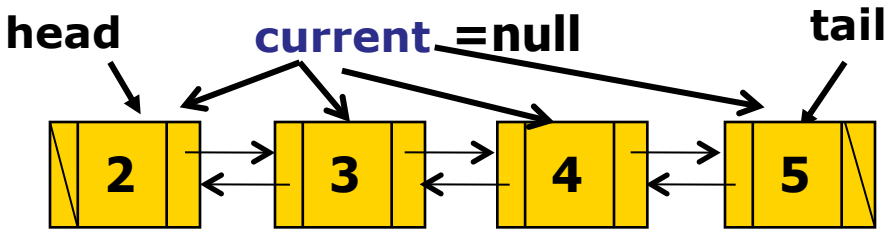
class DoublyLinkedList

```
public E removeLast() throws UnderflowException {




}
```



class DoublyLinkedList

```
public void print() {
    DNode<E> current = head;
    while (current != null) {
        System.out.println(current.getElement());
        current = current.getNext();
    }
}
```



class DoublyLinkedList

```
public int search(E element) {

        ?

}


public E remove(int pos){

        ?

}
```

Testando a lista

```
public static void main(String args[]) {
    DoublyLinkedList<Integer> list = new DoublyLinkedList<Integer>();
    list.insertLast(2);
    list.insertLast(4);
    list.insertLast(6);
    list.insertLast(1);
    list.insertLast(8);
    list.insertLast(9);
    list.print();
    try {
      list.removeFirst();
    } catch (UnderflowException e) {
      e.printStackTrace();
    }
    list.print();
}
```