Institute of
Pervasive Computing

LINKED LISTS

Algorithms and Data Structures 1
Exercise – 2023S

Markus Jäger (Computer Science)
Florian Beck (Artificial Intelligence)
Raja Zafar (Artificial Intelligence)

Institute of Pervasive Computing
Johannes Kepler University Linz

markus.jaeger@jku.at
florian.beck@jku.at
raja.zafar@pervasive.jku.at

**JOHANNES KEPLER
UNIVERSITÄT LINZ**
Altenberger Straße 69
4040 Linz, Österreich
jku.at

OVERVIEW

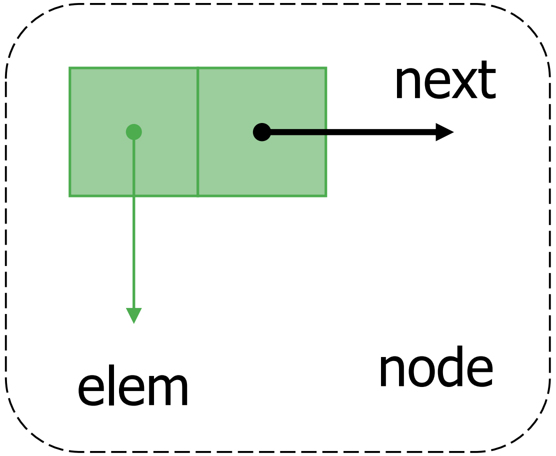
Short summary about the structure of **linked lists**

- Singly linked list
- Doubly linked list
- List structure comparison

LINKED LISTS

A linked list $L = (a_1, a_2, ..., a_n)$

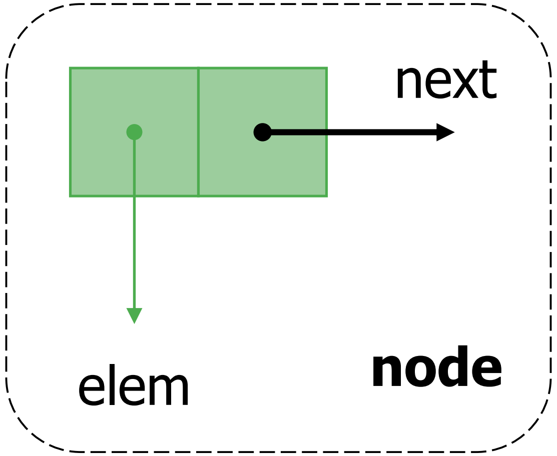
- is a collection of nodes that collectively form a linear sequence
- each node stores a reference to an object that is an element of the sequence, as well as a reference to the next node of the list

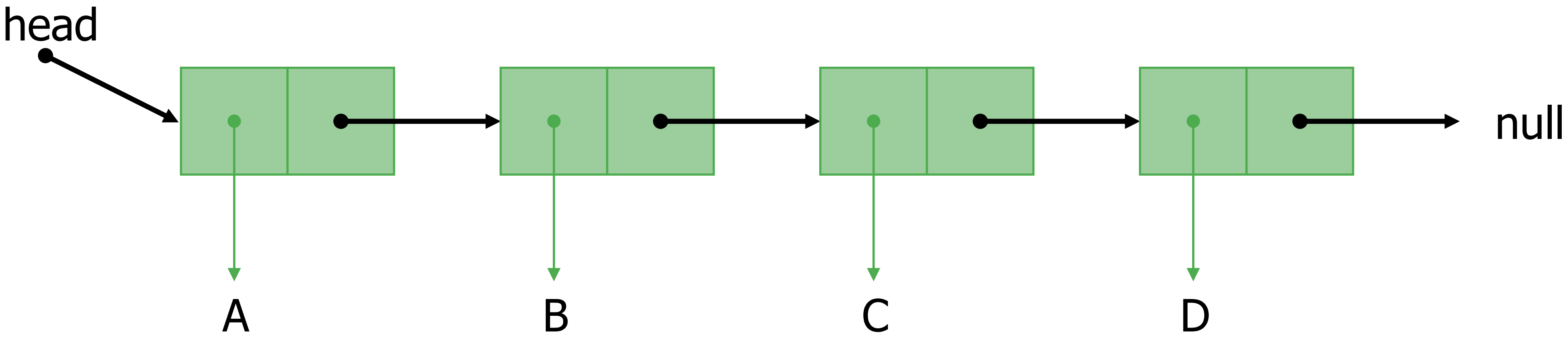


SINGLY LINKED LIST

Each **node** stores

- **element**
- link to the **next** node

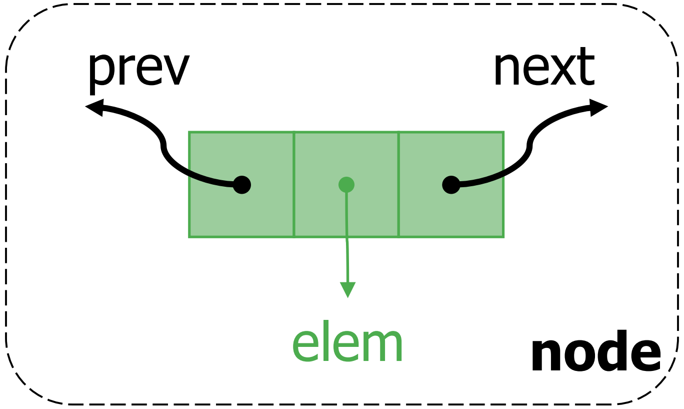


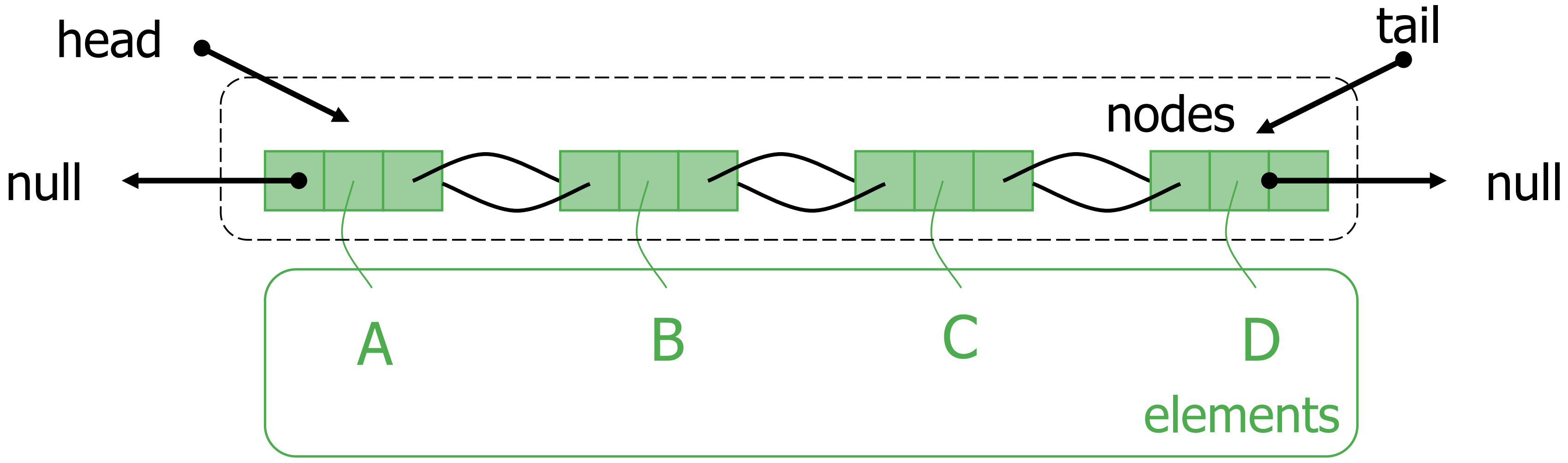


DOUBLY LINKED LIST

Nodes store

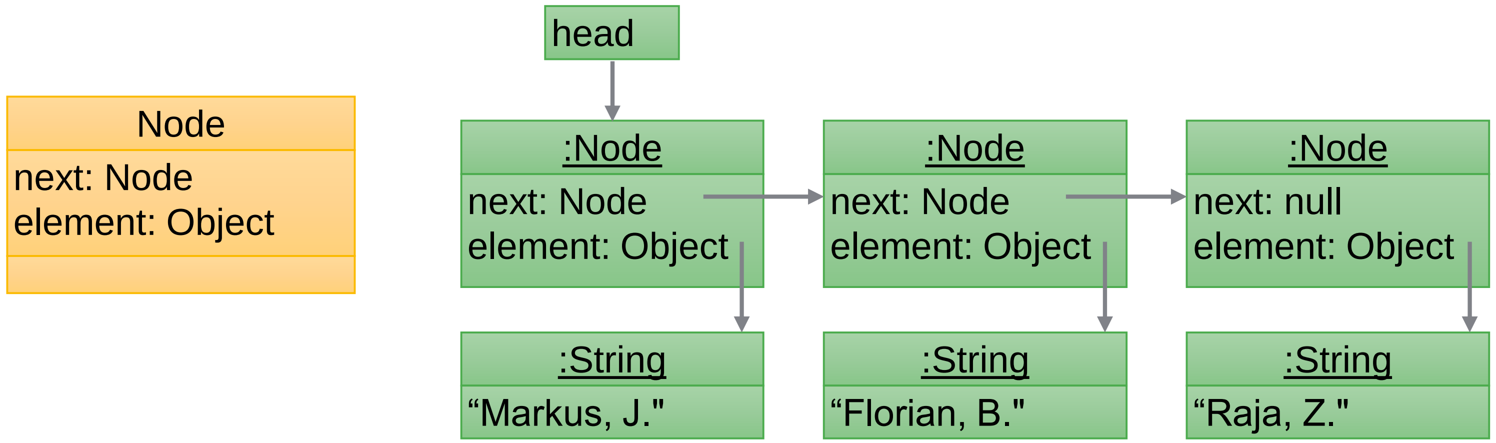
- **element**
- link to the **previous** node
- link to the **next** node





IMPLEMENTATION OF LISTS

- **Singly linked list:**



Node
next: Node
element: Object

head

:Node
next: Node
element: Object

:String
“Markus, J."

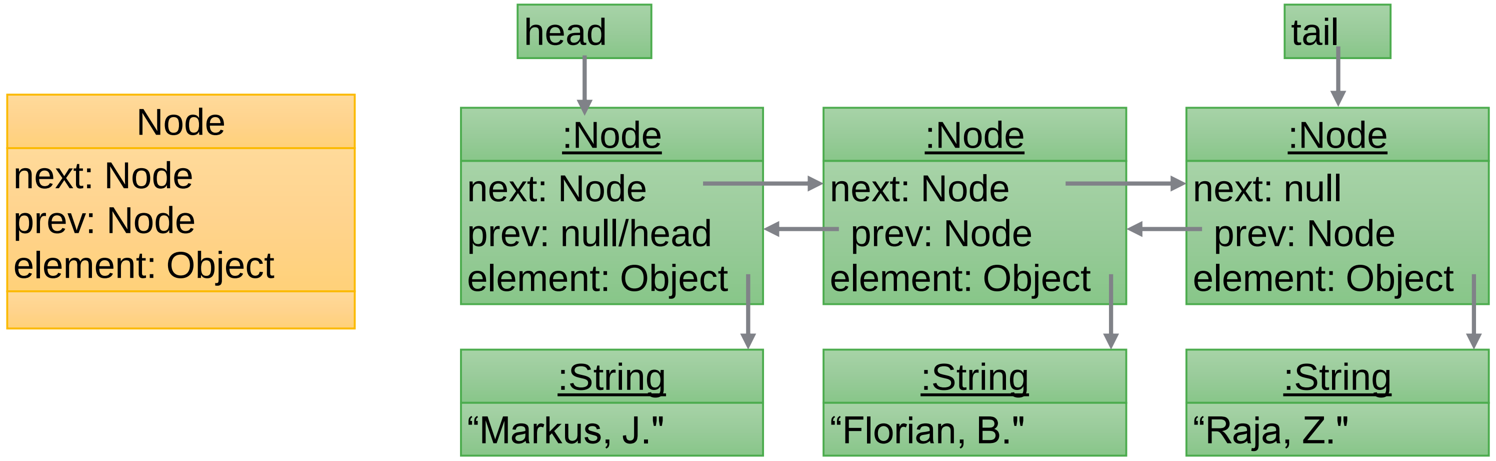
:Node
next: Node
element: Object

:String
“Florian, B."

:Node
next: null
element: Object

:String
“Raja, Z."

- **Doubly linked list:**



Node
next: Node
prev: Node
element: Object

head

tail

:Node
next: Node
prev: null/head
element: Object

:String
“Markus, J."

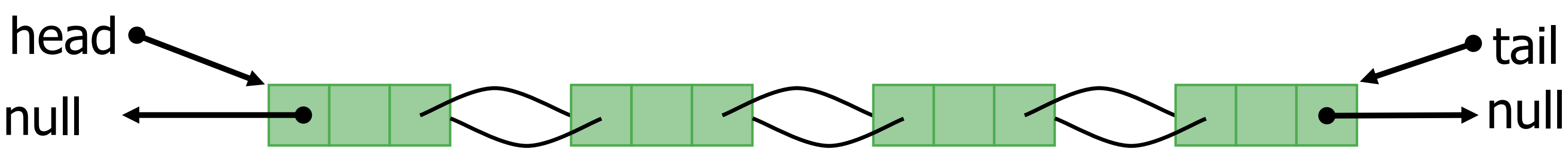
:Node
next: Node
prev: Node
element: Object

:String
“Florian, B."

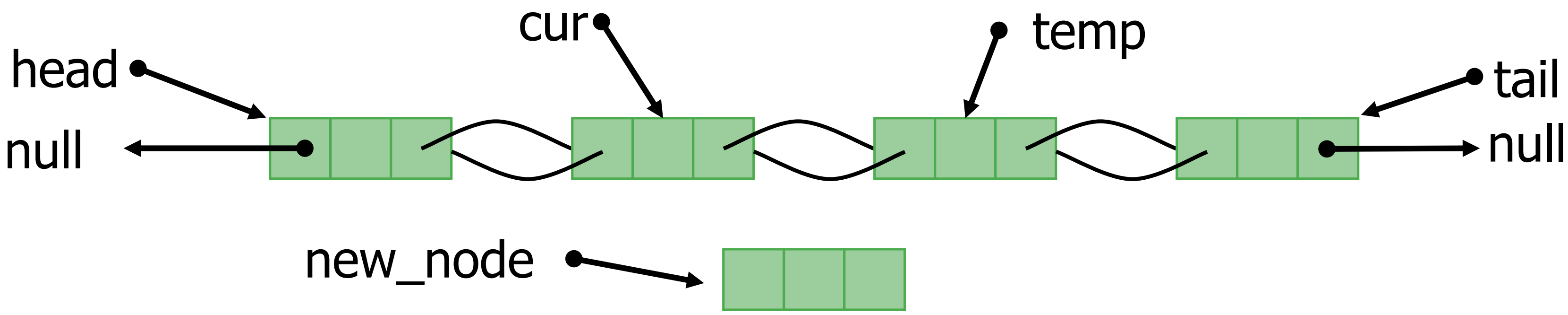
:Node
next: null
prev: Node
element: Object

:String
“Raja, Z."

DOUBLY LINKED LIST :: INSERTION

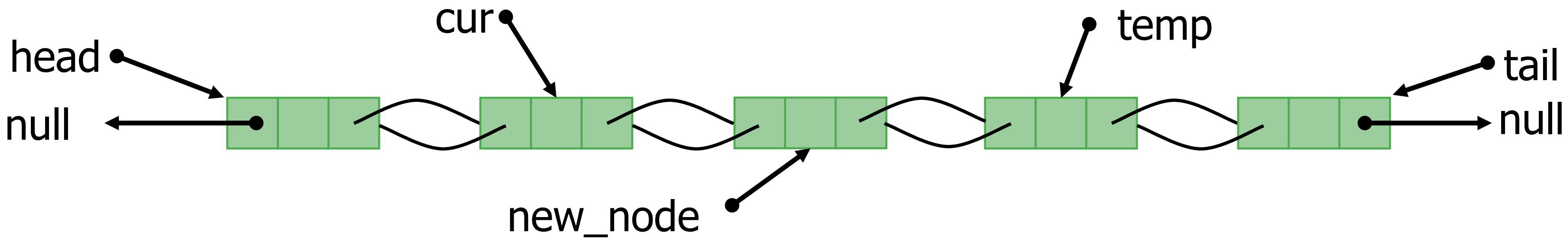


Iterate list with „**cur**“, until position before designated insertion position is reached.
Mark position after designated insertion position with „**temp**“.
Set designated new-to-insert node to „**new_node**“



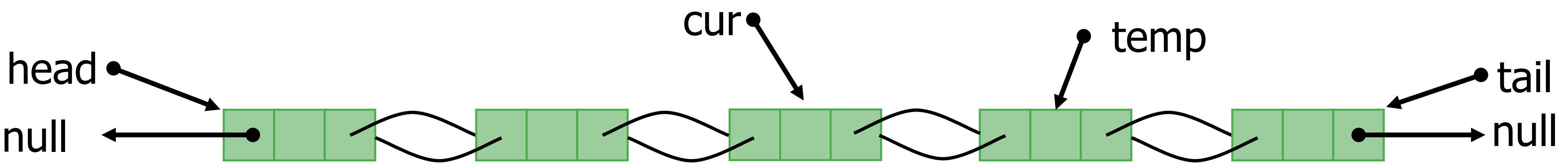
Relink prev and next „pointer“ accordingly:

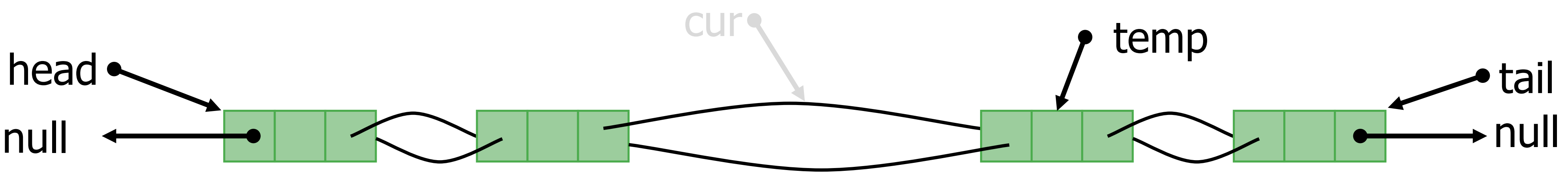
cur.next = new_node
new_node.prev = cur
new_node.next = temp
temp.prev = new_node



DOUBLY LINKED LIST :: REMOVAL

Iterate list with „**cur**“, until position of designated deletion position is reached. (Also possible if cur points to one element earlier – but then the relinking has another order!)
Mark position after designated deletion position with „**temp**“.





cur.prev.next = temp
temp.prev = cur.prev
cur.prev = null
cur.next = null
cur.elem = null

Note: always consider the different possibilities/ cases on inserting and deleting nodes:

- (before) head
- in the list
- (after) tail

Institute of
Pervasive Computing

LINKED LISTS

Algorithms and Data Structures 1
Exercise – 2023S

Markus Jäger (Computer Science)
Florian Beck (Artificial Intelligence)
Raja Zafar (Artificial Intelligence)

Institute of Pervasive Computing
Johannes Kepler University Linz

markus.jaeger@jku.at
florian.beck@jku.at
raja.zafar@pervasive.jku.at

JOHANNES KEPLER
UNIVERSITÄT LINZ
Altenberger Straße 69
4040 Linz, Österreich
jku.at