

Praktische Informatik 2

SS 2008

Mag. Reinhard Wolfinger

Email: reinhard.wolfinger@jku.at

Telefon: 0732/2468-7134

Sprechstunde: Dienstag, 10:00-11:00

Institut für Systemsoftware

Johannes Kepler Universität Linz

Hochschulfondsgebäude, 3. Stock, Raum 304

Organisatorisches

• www.ssw.uni-linz.ac.at/Teaching/Lectures/PID/2008/

• Übungen

- 8 Übungen (werden von Tutoren korrigiert)
- Übungshinweise im WWW beachten
- Keine Toleranz für Abschreiben. Wer 1x abschreibt kann nicht positiv beurteilt werden.
- Abgabe elektronisch (Upload ZIP/RAR) und in Papierform (Postkästen Hochschulfondsgebäude, 3. Stock) jeweils bis Dienstag 12:00
- Passwort für Upload geht an EMail-Adresse in KUSSS
- Korrigierte Übungen eine Woche später Dienstag 12:00 retour

• Übungstest

- Bewertung: Punktemittelwert
(Übungspunkte / Übungsanzahl + Testpunkte) / 2
Test muss positiv sein

• Fragen?

Datum	Nr	Übung	Übungsabgabe
4.3.	1	Übung 1 (12 Punkte): Verkettete Liste	
11.3.	2	Übung 2 (24 Punkte): Listen	Übung 1
18.3.			
25.3.			
1.4.	3		Übung 2
8.4.	4	Übung 3 (24 Punkte): Bin. Suchbäume	
15.4.	5		
22.4.	6	Übung 4 (24 Punkte): Heaps	Übung 3
29.4.	7	Übung 5 (24 Punkte): Graphen	
6.5.	8		Übung 4
13.5.			
20.5.	9	Übung 6 (24 Punkte): Hashen	Übung 5
27.5.	10	Übung 7 (12 Punkte): Stringsuche	
3.6.	11	Übung 8 (12 Punkte): Sortieren	Übung 6
10.6.	12		Übung 7
17.6.	13		Übung 8
24.6.	14	Test	

Gesamt 156 Punkte

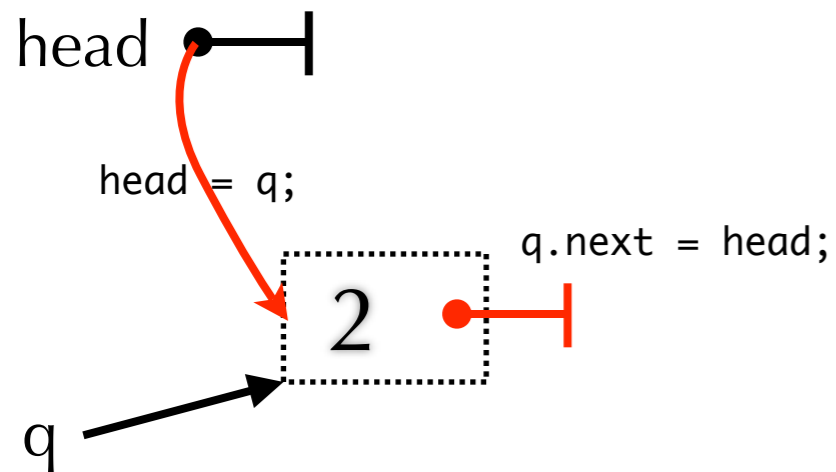
Lineare Liste

```
class Node {
    int val;
    Node next = null;
    Node(int val) { this.val = val; }
}

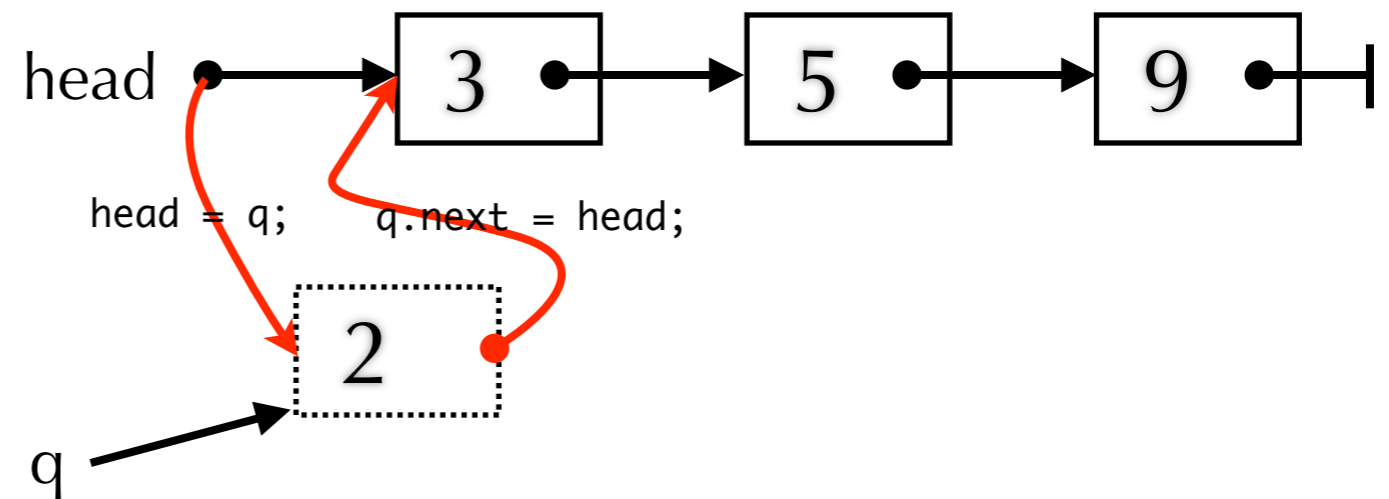
public class List {
    private Node head = null;
    public void prepend(int val) { ... } // vorne anfügen
    public void append(int val) { ... } // hinten anfügen
    public Node find(int val) { ... } // suchen
    public void insert(int val) { ... } // (aufsteigend) sortiert einfügen
    public void remove(int val) { ... } // entfernen
}
```

prepend

Fall 1: leere Liste



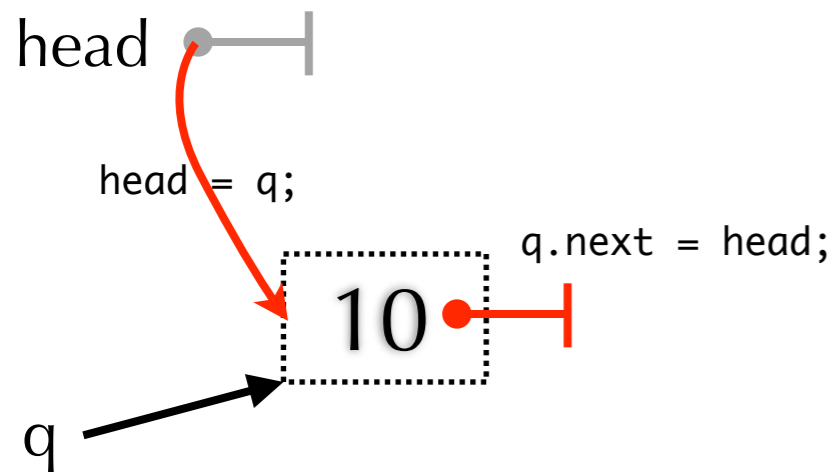
Fall 2: Liste mit Knoten



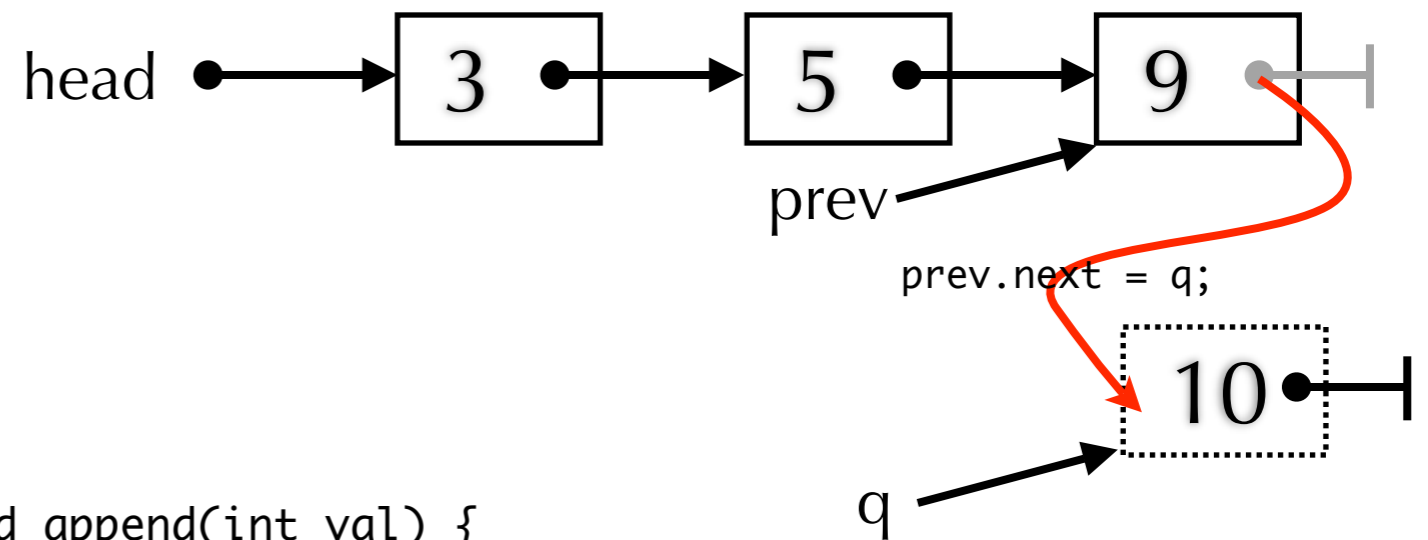
```
public void prepend(int val) {  
    Node q = new Node(val);  
    q.next = head;  
    head = q;  
}
```

append

Fall 1: leere Liste



Fall 2: Liste mit Knoten

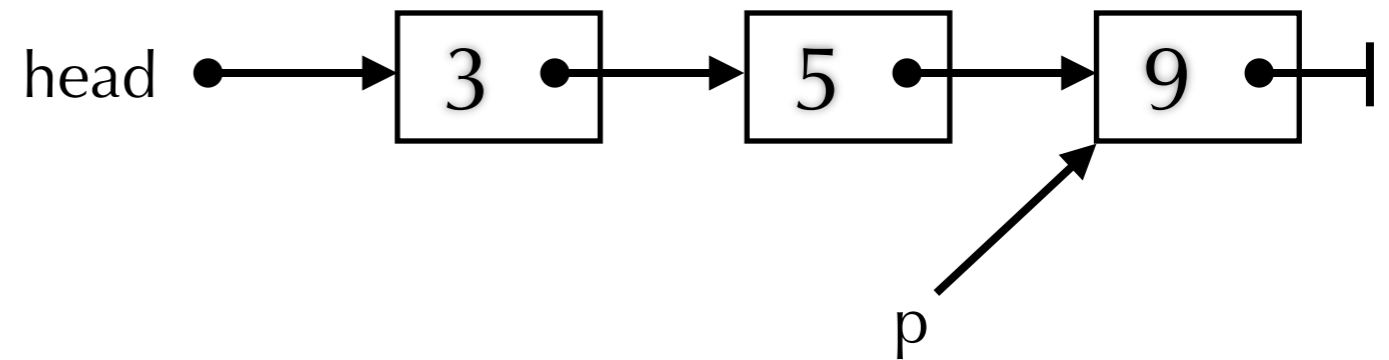
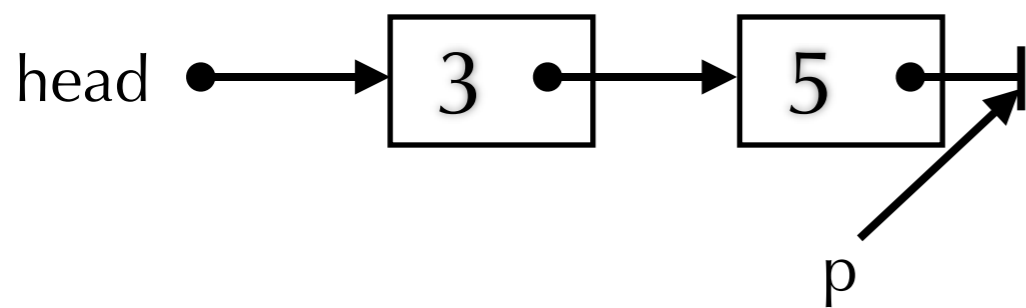


```
public void append(int val) {  
    Node q = new Node(val);  
    if(head == null) {  
        // prepend  
        head = q;  
    } else {  
        Node prev = head;  
        while(prev.next != null)  
            prev = prev.next;  
        prev.next = q;  
    }  
}
```

find

Fall 1: nicht gefunden

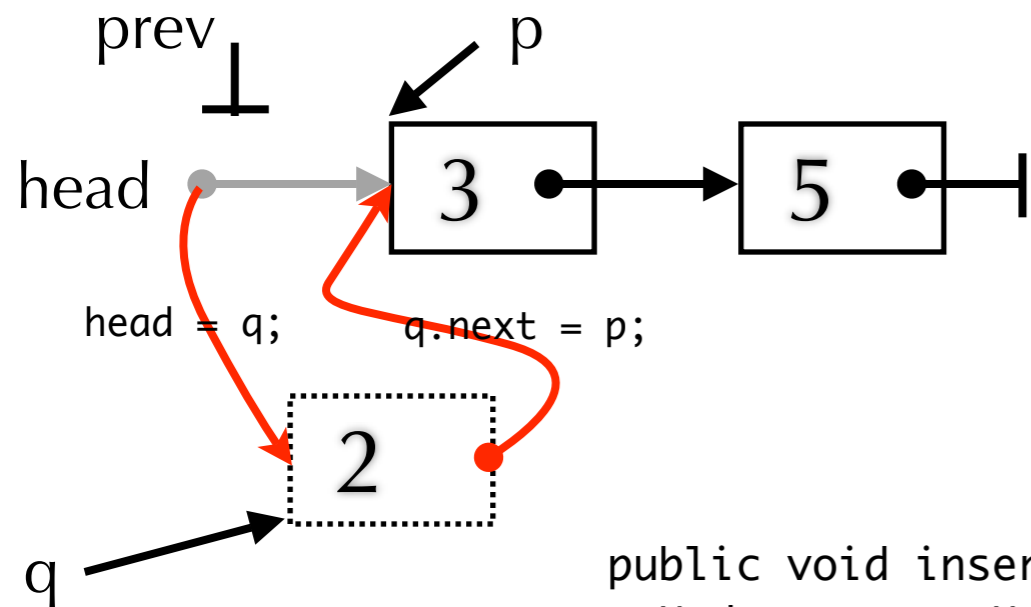
Fall 2: gefunden



```
public Node find(int val) {  
    Node p=head;  
    while(p != null && p.val != val)  
        p = p.next;  
    return p;  
}
```

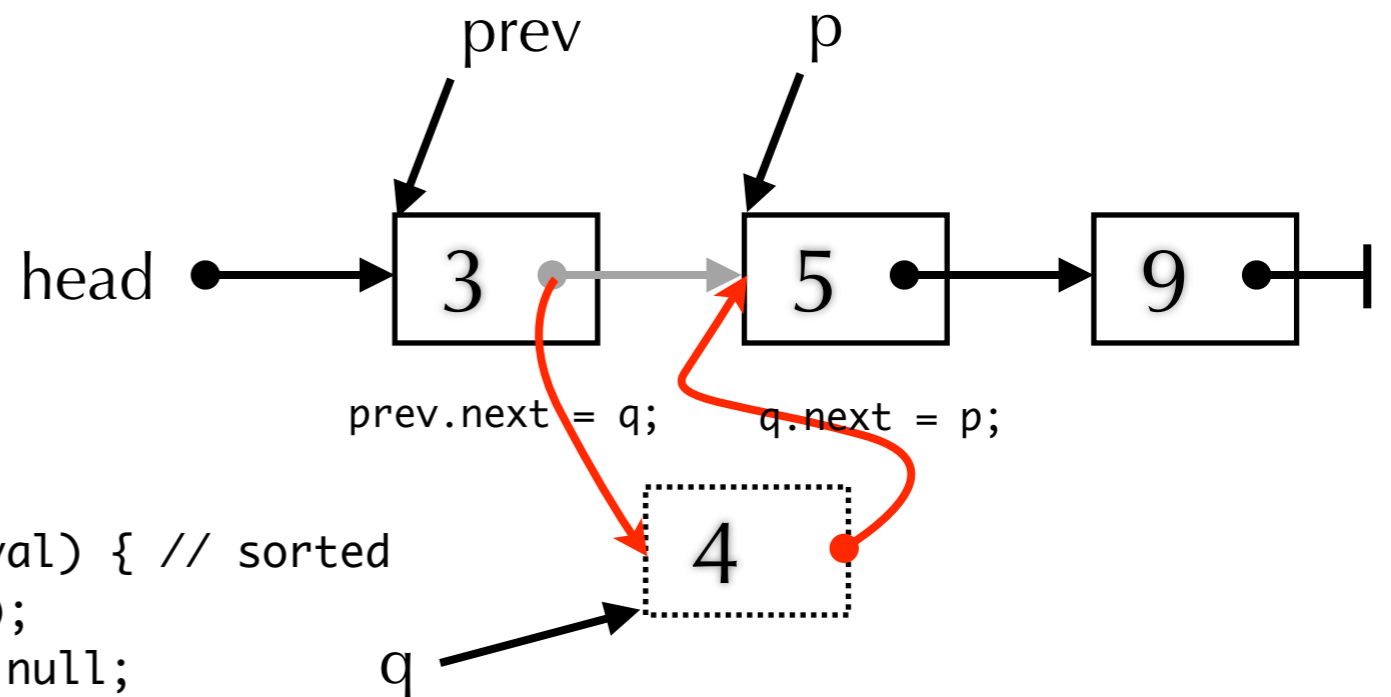
insert

Fall 1: am Anfang



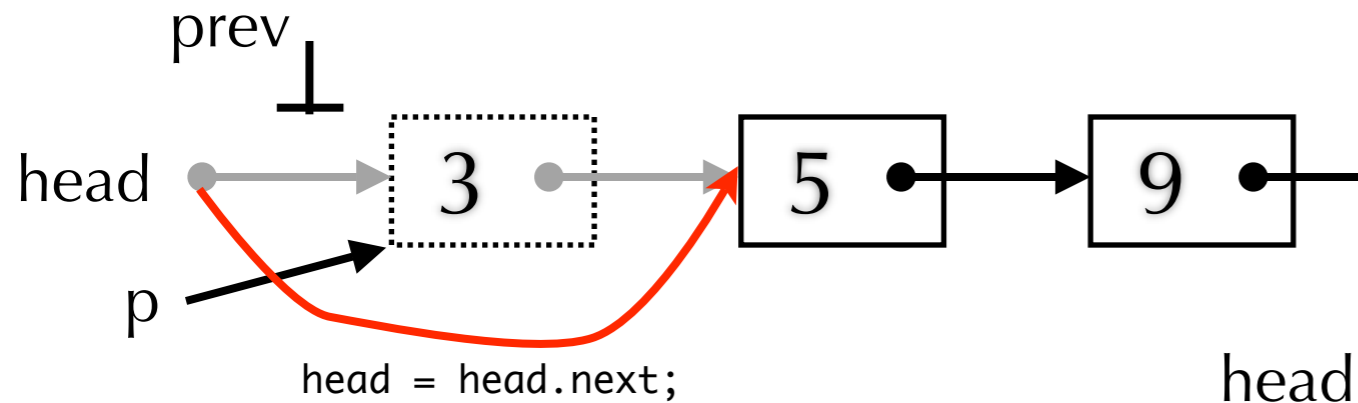
```
public void insert(int val) { // sorted
    Node q = new Node(val);
    Node p = head, prev = null;
    while(p != null && p.val < val) {
        prev = p;
        p = p.next;
    }
    if(prev == null)
        head = q;
    else
        prev.next = q;
    q.next = p;
}
```

Fall 2: in der Mitte

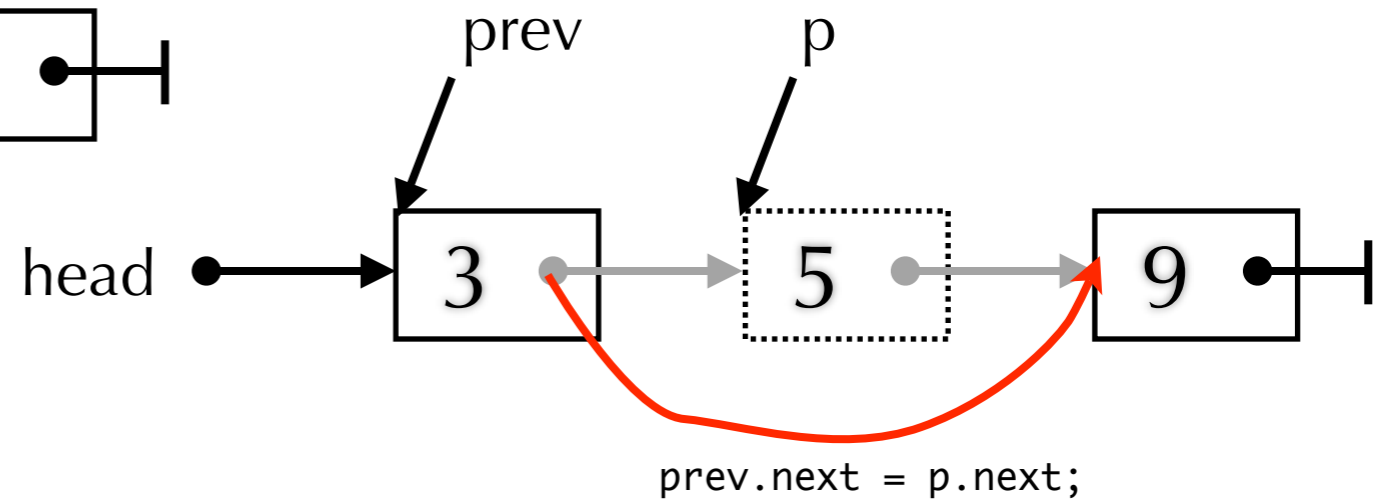


remove

Fall 1: am Anfang



Fall 2: in der Mitte



```
Node remove(int val) {  
    Node p=head, prev = null;  
    while(p != null && p.val != val) {  
        prev = p;  
        p = p.next;  
    }  
    if(p != null) {  
        if(p == head)  
            head = head.next;  
        else  
            prev.next = p.next;  
    }  
    return p;  
}
```