Praktische Informatik 2

Dr. Reinhard Wolfinger

Email: reinhard.wolfinger@jku.at
Telefon: 0732/2468-7134
Sprechstunde: Dienstag, 14:00-15:00

Institut für Systemsoftware
Hochschulfondgebäude
3. Stock, Raum HF301

Organisatorisches

www.ssw.uni-linz.ac.at/Teaching/Lectures/PI2/2010/

- Übungen
  - 8 Übungen, von Tutoren korrigiert
  - Übungshinweise im Web beachten
  - Keine Toleranz für Abschreiben.
    Wer 1x abschreibt wird nicht beurteilt.
  - Abgabe elektronisch (Upload ZIP/RAR)
    und in Papierform
    Postkästen in Hochschulfondgebäude
    3. Stock, hinter Aufzug
  - Abgabe Dienstag 10:15
  - Passwort per Email
  - Korrigierte Übungen eine Woche später Dienstag 12:00 zurück
- Test
  - Bewertung: (Übungspunkte / Übungsanzahl + Testpunkte) / 2
  - Test muss positiv sein

Datum	Nr	Übung	Übungsabgabe
2.3.	1	Übung 1 (12 Punkte)	
9.3.	2	Übung 2 (24 Punkte)	Übung 1
16.3.	3		
23.3.	4	Übung 3 (24 Punkte)	Übung 2
30.3.		*Osterferien*	
6.4.		*Osterferien*	
13.4.	5		
20.4.	6	Übung 4 (24 Punkte)	Übung 3
27.4.	7	Übung 5 (24 Punkte)	
4.5.		*Tag des Landespatrons*	
11.5.	8		Übung 4
18.5.	9	Übung 6 (24 Punkte)	
25.5.		*Pfingsten*	
1.6.	10	Übung 7 (12 Punkte)	Übung 5
8.6.	11	Übung 8 (12 Punkte)	
15.6.	12		Übung 6
22.6.	13	Test	Übung 7
29.6.	14	Nachbesprechung Übungstest	Übung 8

Fragen?

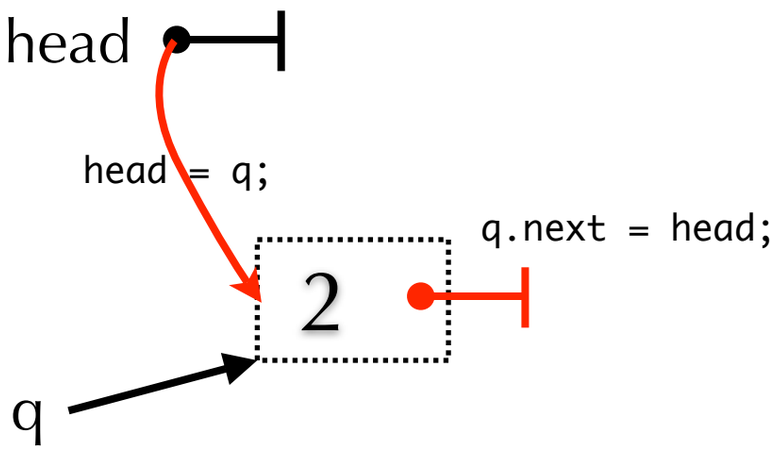
Lineare Liste

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

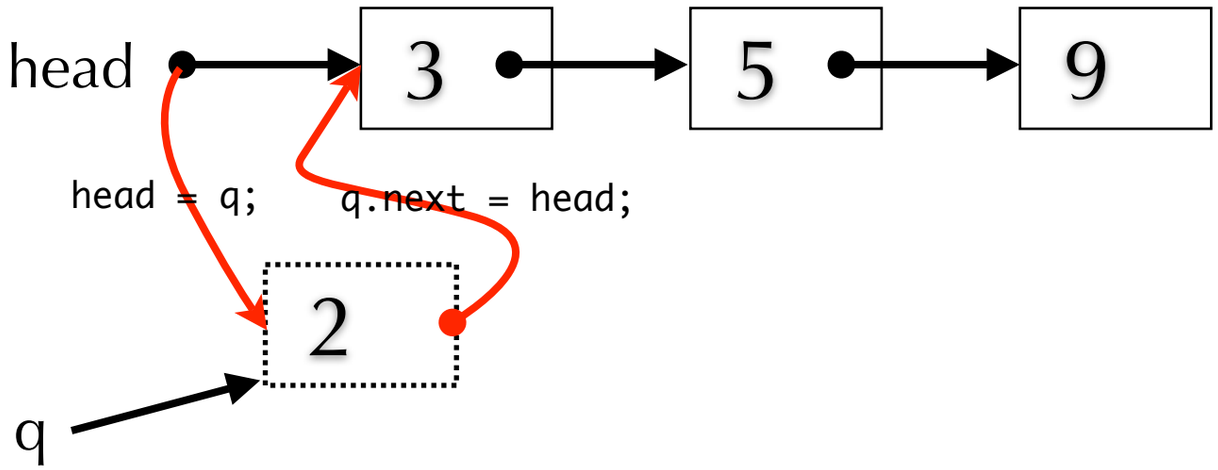
```
public class List {
  private Node head = null;
  public void prepend(int val) { ... }        // vorne anfügen
  public void append(int val) { ... }         // hinten anfügen
  public Node find(int val) { ... }           // suchen
  public void insert(int val) { ... }         // (aufsteigend) sortiert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

prepend

Fall 1: leere Liste

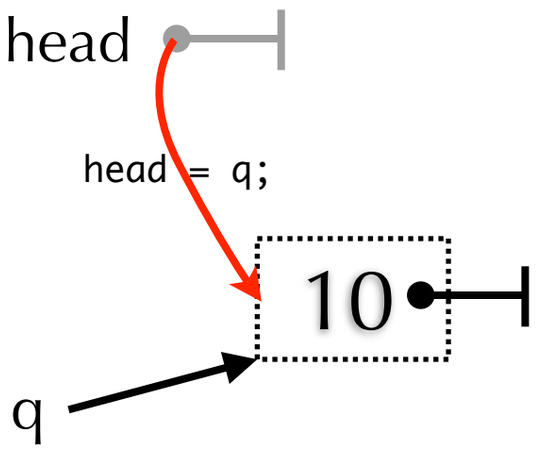


Fall 2: Liste mit Knoten



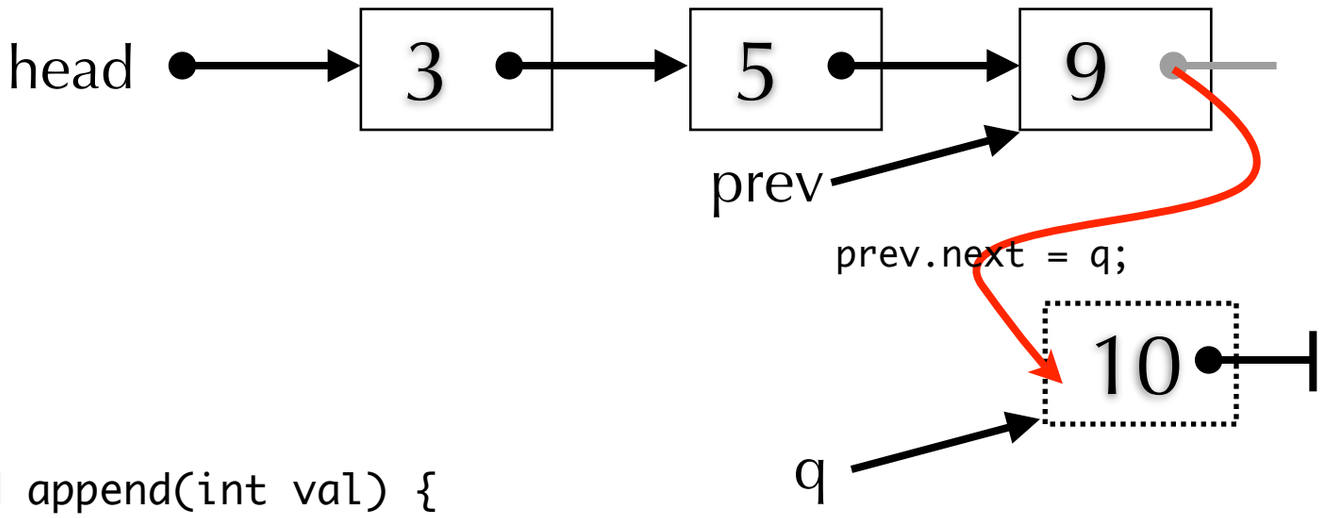

```
public void prepend(int val) {
  Node q = new Node(val);
  q.next = head;
  head = q;
}
```

append



Fall 1: leere Liste

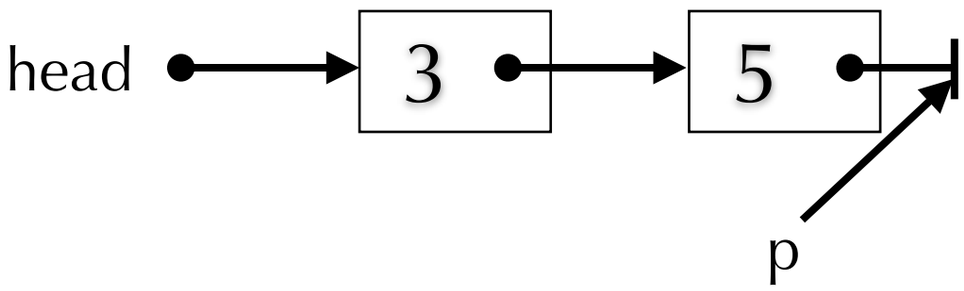
Fall 2: Liste mit Knoten



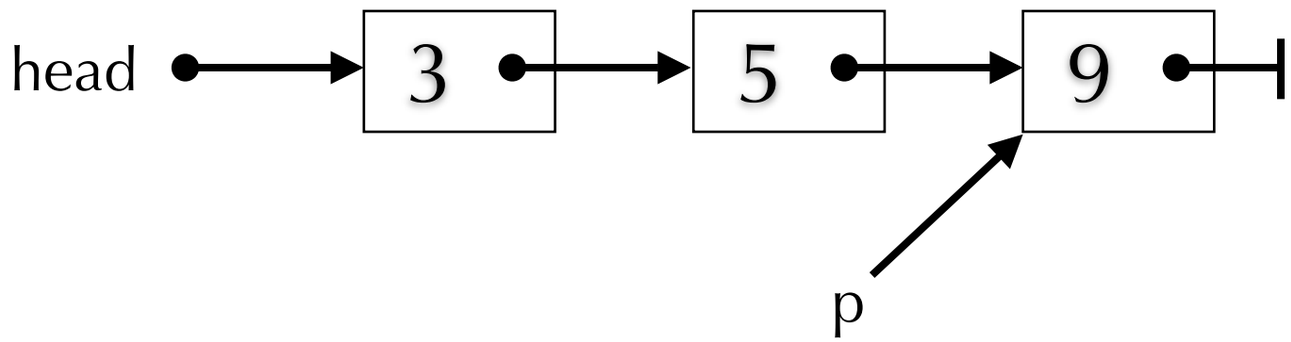

```
public void append(int val) {
  Node q = new Node(val);
  if(head == null) {
    // prepend
    head = q;
  } else {
    Node prev = head;
    while(prev.next != null)
      prev = prev.next;
    prev.next = q;
  }
}
```

find

Fall 1: nicht gefunden



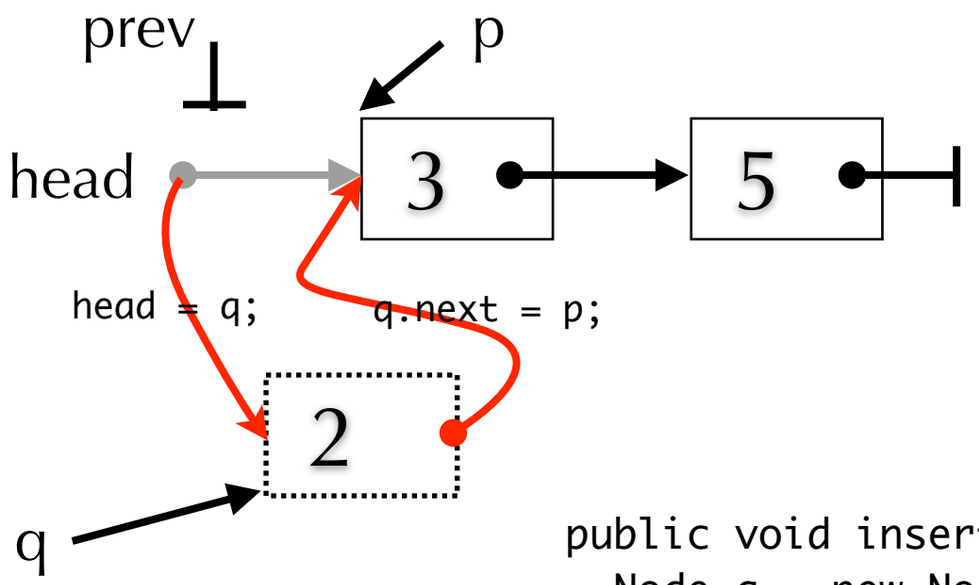
Fall 2: gefunden



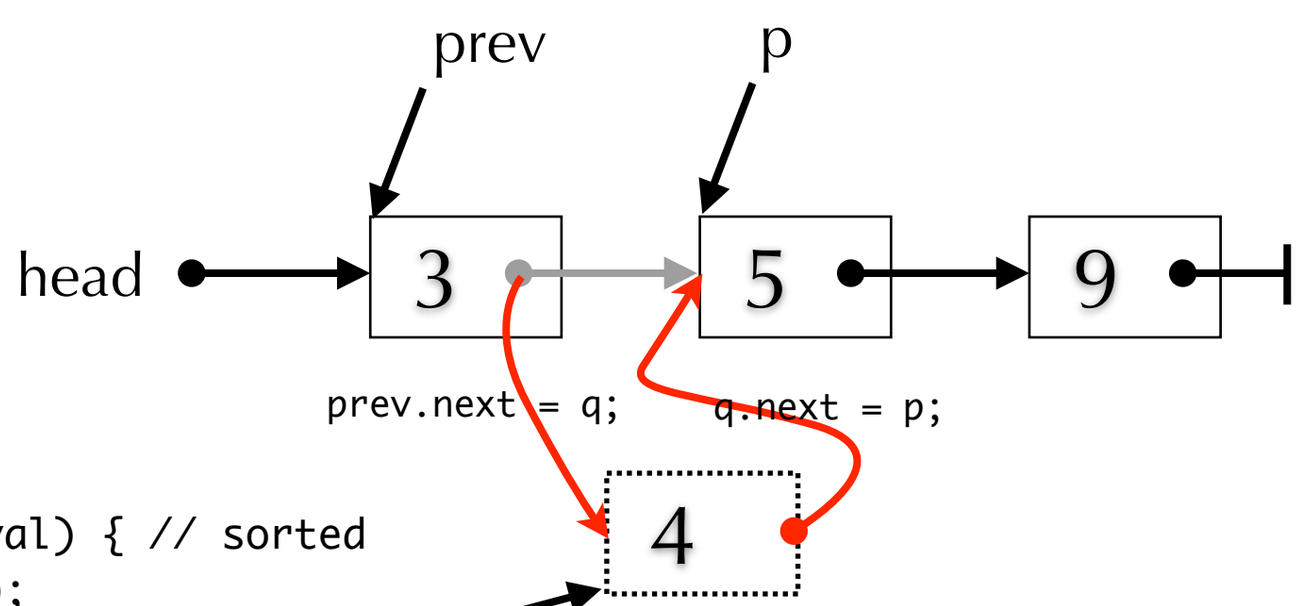

```
public Node find(int val) {
  Node p=head;
  while(p != null && p.val != val)
    p = p.next;
  return p;
}
```

insert

Fall 1: am Anfang



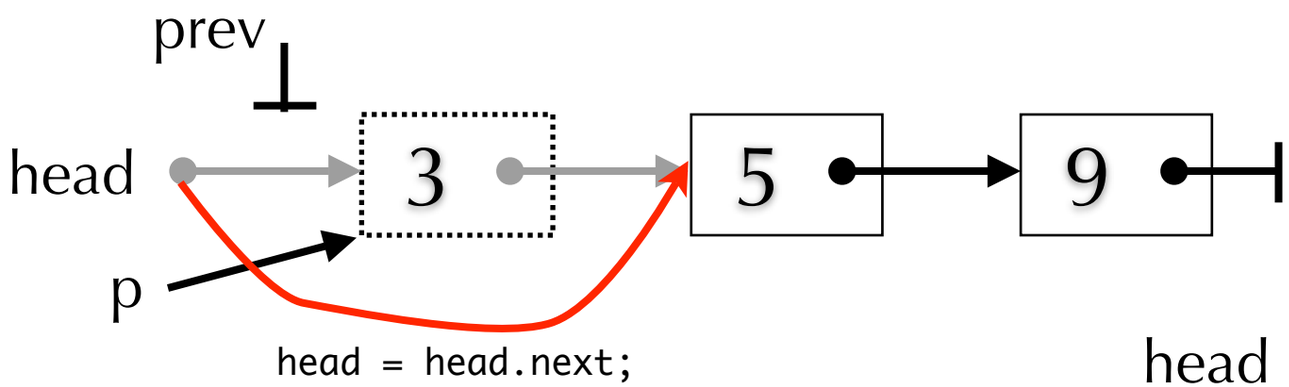
Fall 2: in der Mitte



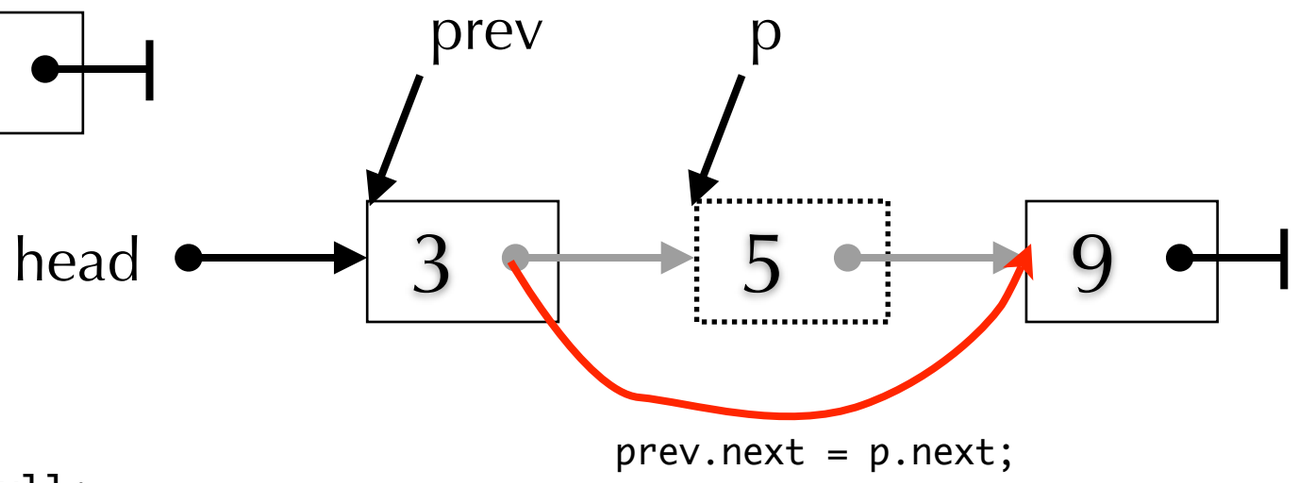

```
public void insert(int val) { // sorted
  Node q = new Node(val);
  Node p = head, prev = null;
  while(p != null && p.val < val) {
    prev = p;
    p = p.next;
  }
  if(prev == null)
    head = q;
  else
    prev.next = q;
  q.next = p;
}
```

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;
}
```