

Wiederholung

## 1: Testen

wann? testet? immer !!

was testen? Alles !!

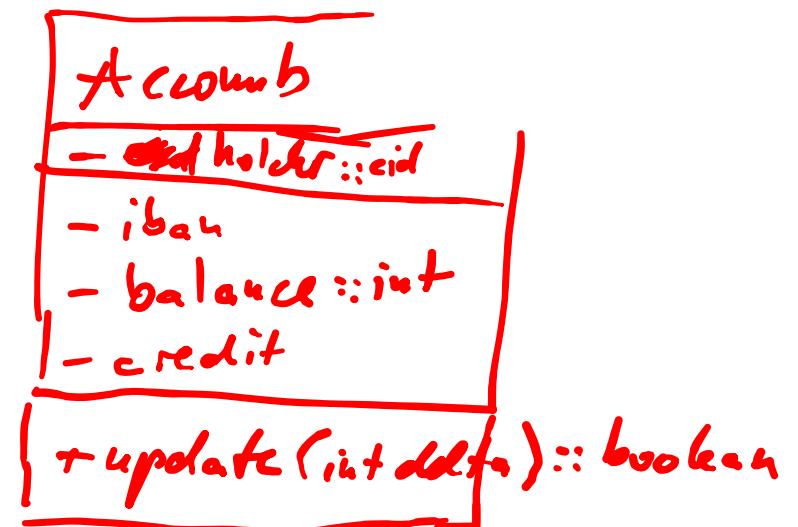
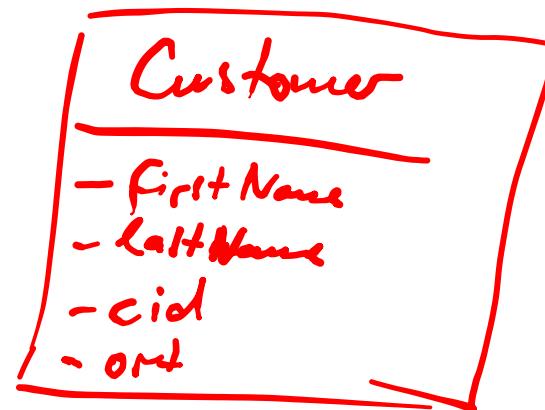
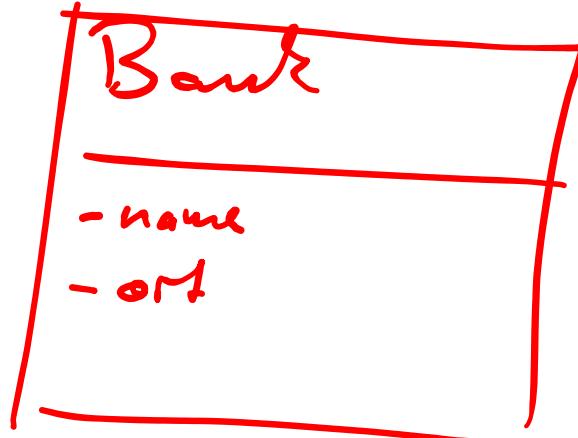
wie testen? gradlew clean build  
(incl. test)

wie Testfälle bauen? Grenzwerte, etc. Wert

# Beispiel Bank

## Customer

## Account



Aufgabe: 3 Klassen  
update testen  
Dummy to play

## 2) Sortieren

Bubblesort

Selectionsort

Insortionsort

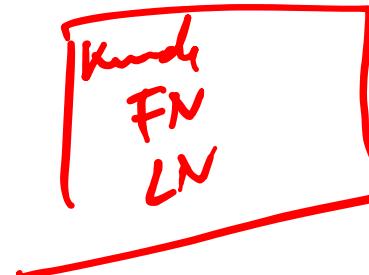
Quicksort

Mergesort}

Trefferliste

Reise

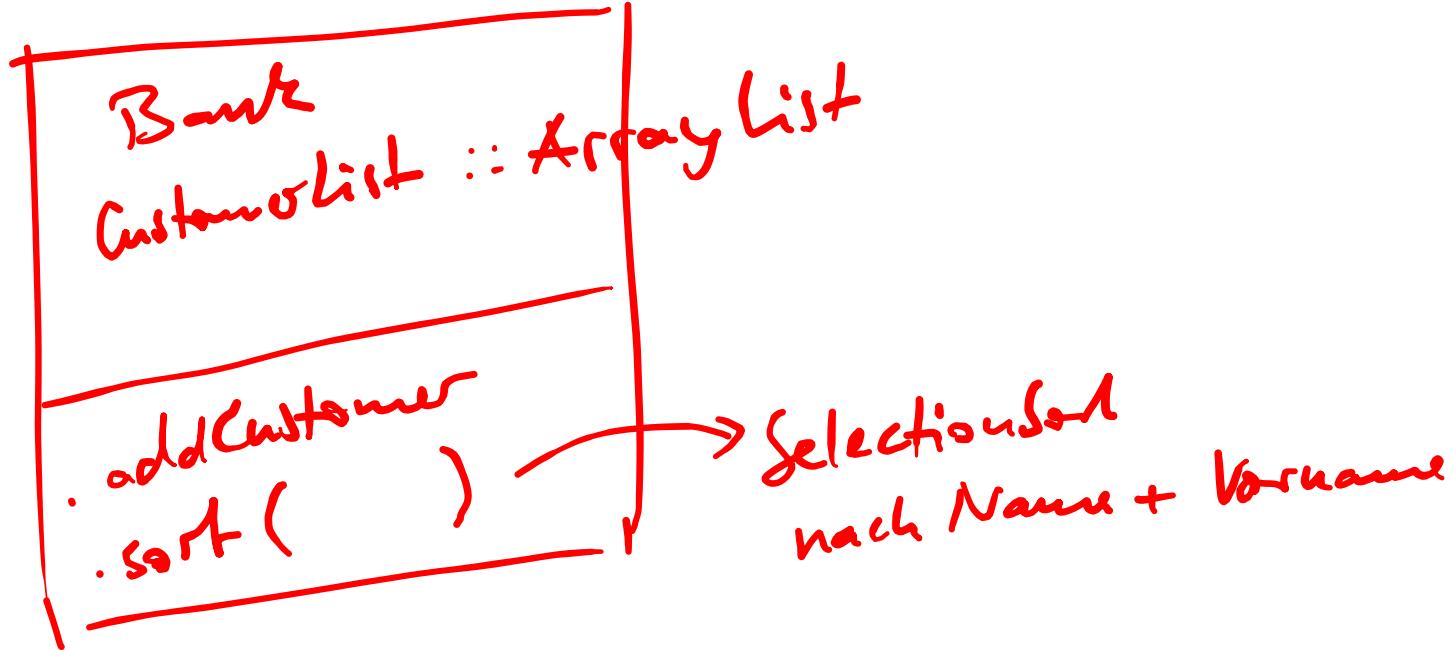
stabil



Komplexität:  $O(n^2)$

$O(n \cdot \log n)$

# Aufgabe:



Sortieren  
(Sort)

Suchen  
(Search)

Select.  
(Select)

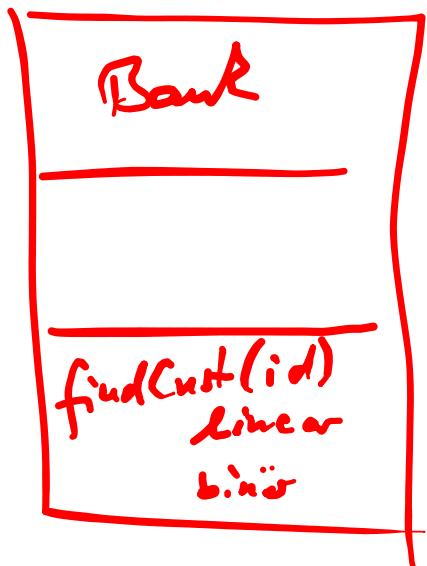
Lineare      Binär      Hashing

langsam      - Sortiert      - Hashfunktion  
 $O(n)$       - schneller      - superschnell  
                 $O(\log n)$        $O(1)$

Aufgabe:

a) Sortiere Kunden (v)

b) Finde zu einem Account den Kunden  
b.1) linear      B1  
b.2) binär



# Selection



Findet die 3 Kunden von höchster Bankbalance

### 3) spezielle Methoden

Überschreiben Object.

- a) warum?      .  
                      . toString für Logging  
                      . equals / hashCode: Nur ich weiß es.

- b) welche?      — toString()  
                      — equals()  
                      — hashCode()!

- c) wie sieht equals?
- null?  
    instanceof?  
    enthalten?

Aufgabe Bank `toString` v. schon gemacht  
equals → gleiche Name  
hashCode → Anzahl Zeichen im  
Name

Customer `toString`  
equals → Customer ID  
hashCode → Customer ID modulo 100

## 4) Spezielle Klasse

### 4 a) Wrapper

int → Integer

Integer ~~int~~ age = 7;  
                  ^  
                  autom. boxing

Integer age = new Integer(7);

Warum?

## 45) String

```
String s = new String ("Auto");  
          = "Auto";
```

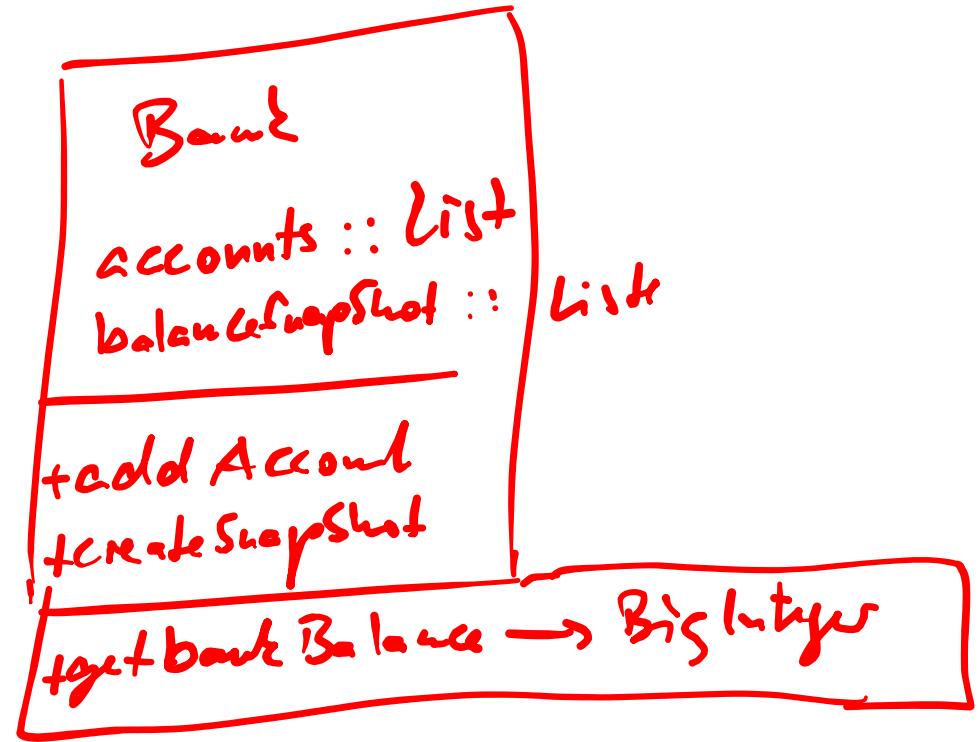
immutable

```
s = "a";  
s = "b";
```

" == " ↴

StringBuilder, StringBuffer

Aufgabe:



Dummie und nutzen

## 5) Containers

Abstrakte Klasse

implementieren Teil von Interface vor

List l = new ArrayList

Interface

no public methods signatures / Köpfe

warum definieren?

Von außen  
sieht nur  
Interface



Always implements  
against the  
interface.

Richtige Wahl des Containers für  
Personenanzahl erledigt.

LinkedList

ArrayList

HashMap

Vergleich, compare

impl. Comparable  
extends Comparable

$a > b \Rightarrow$  pos.

$a == b \Rightarrow$  0

$a < b \Rightarrow$  neg.

Account impl. Comp.

a > b

a.balance >  
b.balance

Customer impl. Comp.

c > d

c. give Some val  
Accounts >

d. give Some  
val Accounts

# 6) Exceptions

Error  $\longleftrightarrow$  Exception

unvermeidbar      "erk" vermeidbar

checked

kontroll.

unchecked

nicht kontrolliert

Beispiele

```
try{ File f = "...";  
    f.open();  
} catch(FileNotFoundException e){}
```

Array Index/Not Found  
a[15]  
NullPointerException

Warum sind nicht alle Exceptions checked?

myObject.read(); will Exception

```
try {  
    myObject.read()  
}  
catch (Exception e) {  
    ...  
}
```

void myMethod() throws Exception  
myObject.read();



try {

}

catch (

)

catch (

)



Classen werden gewählt

Aufgabe: Exception werfen, wenn keine  
Account zum Kunden  
gefunden werden.

Wir wollen Exception jeweils reduzieren,  
wählen?

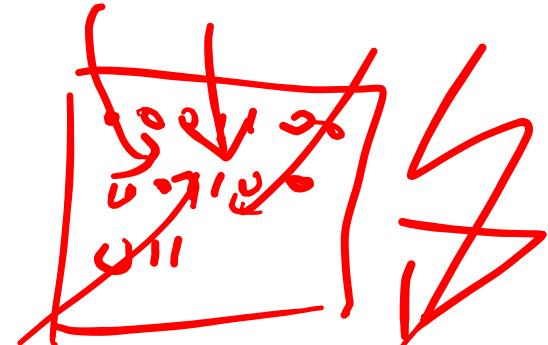
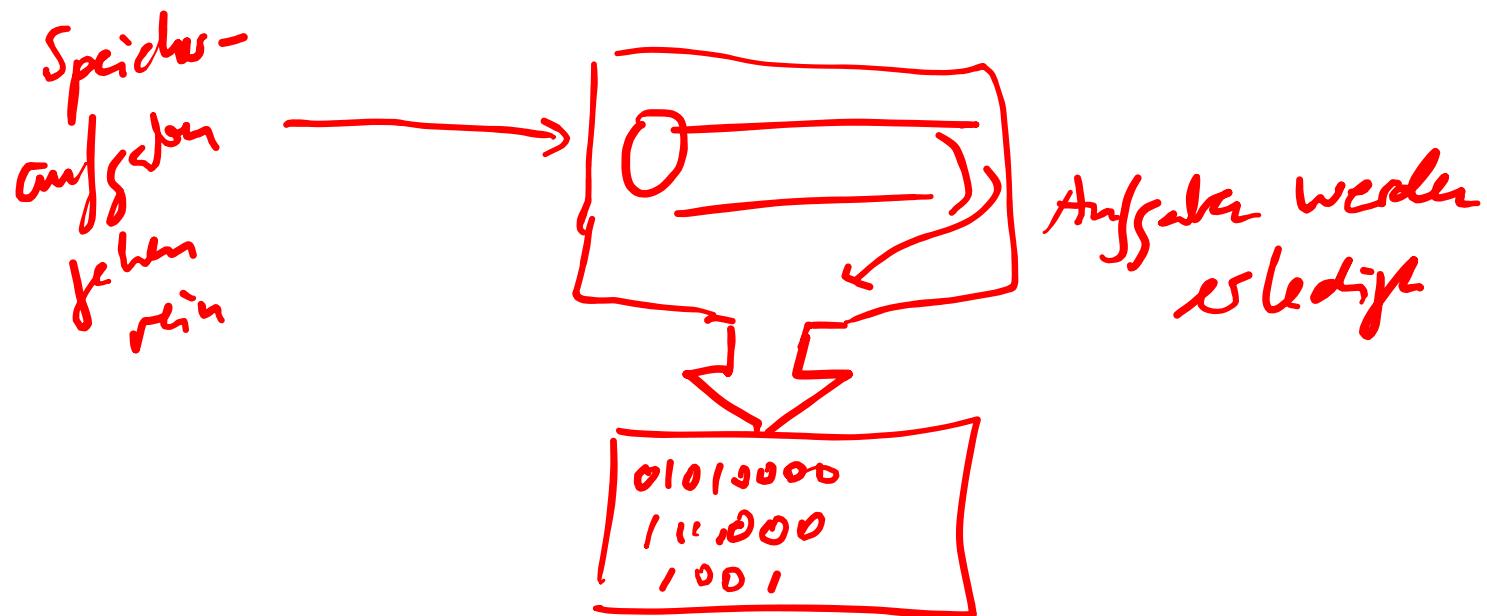
Exception sind "heavy"

besser Optionals (ab Java 8)

7) ~~DAO~~ DAO

Manager Interface } Singleton  
nur 1 Instanz

warum? Ressourcenverwaltung

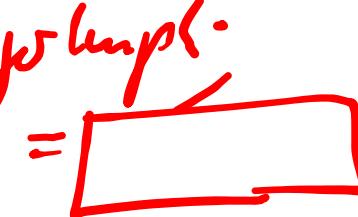


## 2 Versionen

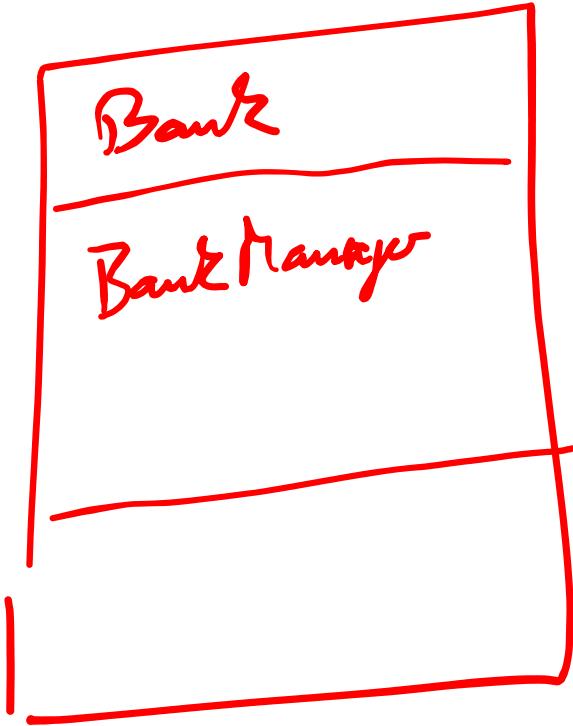
thread sicher

nicht threadsicher

```
class PostgresManagerImpl {  
    static public private PostgresManagerImpl postgresManagerImpl;  
    private PostgresManagerImpl()  
    static public getPostgresManagerImpl()  
    {  
        2  check null  
        3  return  
    }  
}
```



- 1). Constructor private
- 2). static Variable für Instance
- 3) public static getMethode
- 4) erzeuge instant



Singleton

## 9) Threads

extends Thread

implements Runnable

run();

aber man startet mit start()

Thread.join();

blockiert und wartet

.setPriority(..)

ist thread fertig ist

1..10

zusätzliche Datensstrukturen

Synchronized(    ) {



Monitor

}

Aufgabe: Starte einen Hintergrund-thread,  
der <sup>von</sup> allen Daten immer ein  
Backup macht.

# 1.) Iterator

ein Objekt, welches einen Container durchlaufen kann

Iterable interface

- hasNext() :: Boolean
- next() :: Element

Generics

ArrayList<Customer>

Iterator<Customer>

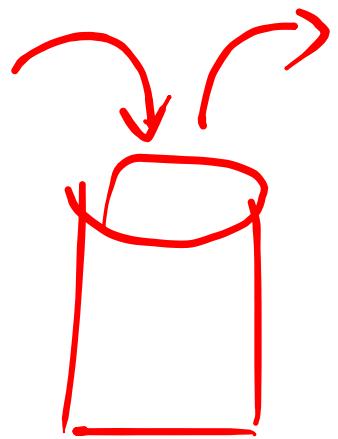
Warum muss ich den Typ angeben?  
→ Lautet ein Gasterminotiv  
→ Lautet ein Typfelder

Aufgabe a) Bank implements Iterable<Customer>  
(Account)

b) Safe im Dummy but  
for (Customer c : bank)  
Geld nun!

# 10) Datenstrukturen

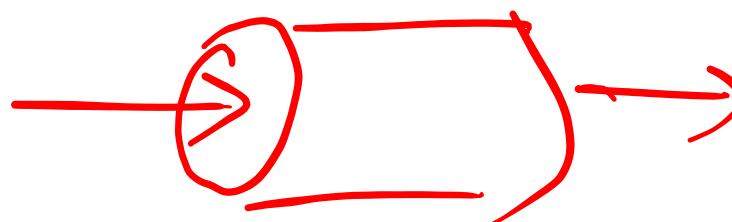
Stack



FILO

put / push  
get / remove

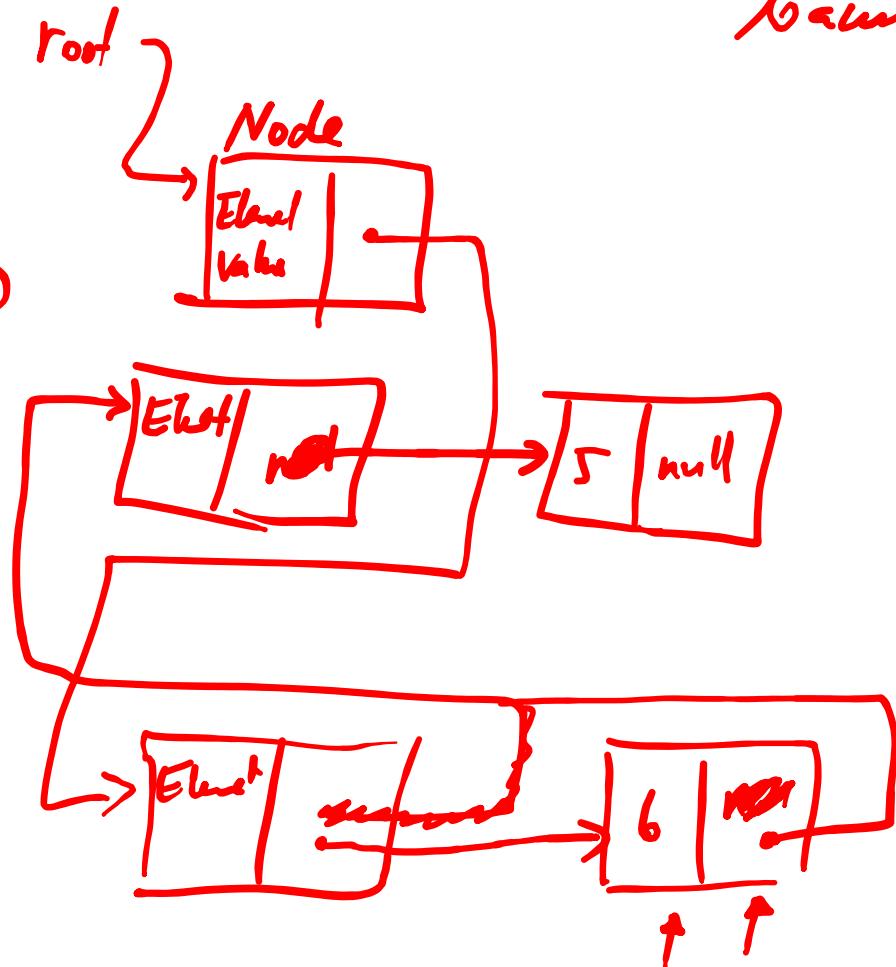
Queue



FIFO

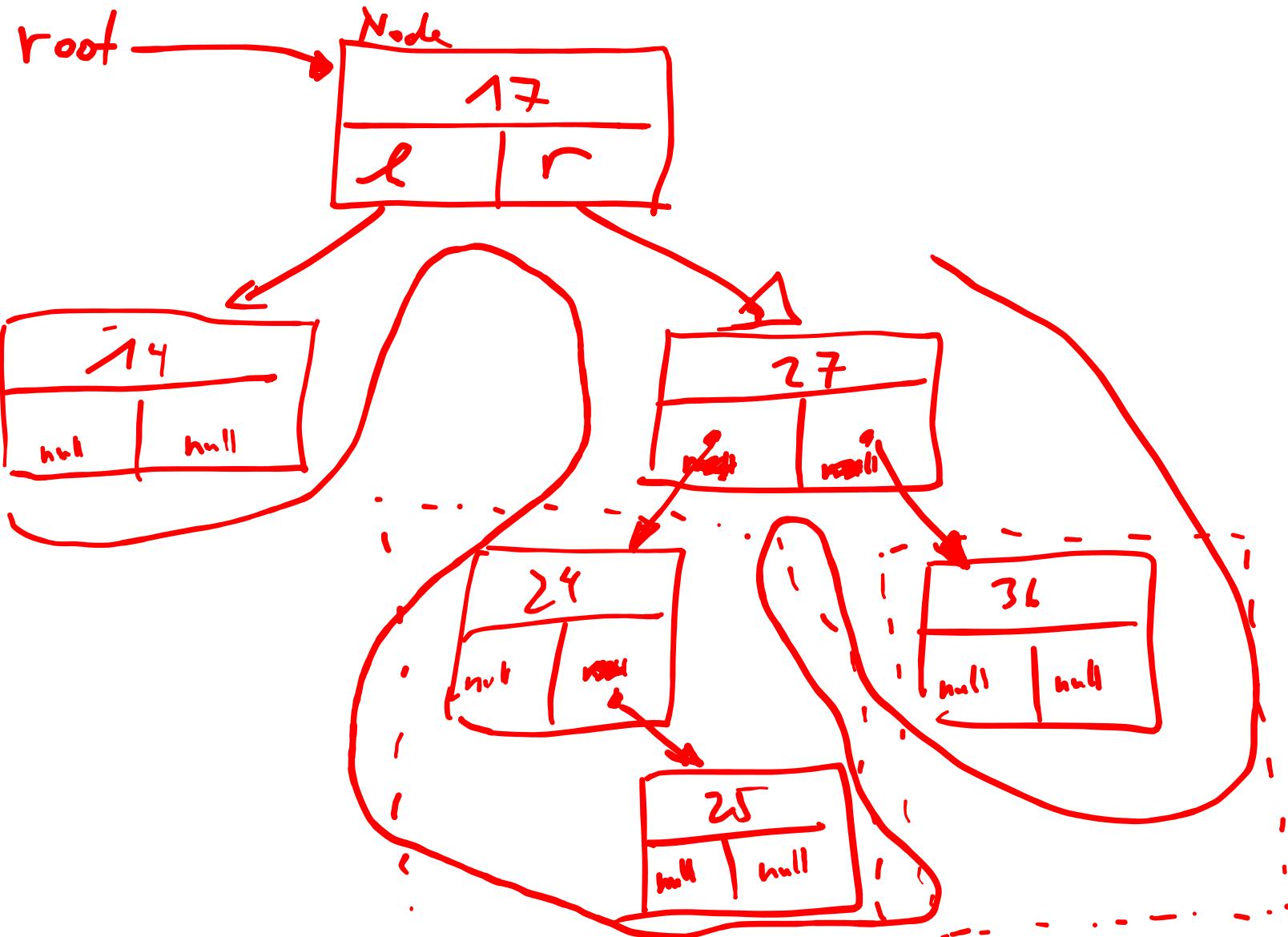
remove  
idx[?]  
get[?]

Linked List



Baum  
(Entscheidungs-  
baum)

# EntscheidungsBäume



- insert
- delete
- iterate()  
Durchlauf

```
class Node {  
    int element;  
    Node next; // Zeiger
```

```
    void delete(5) {  
        a = root  
        while a.element != 5 a = a.next;  
    }  
}  
?  
! ! !
```

Java ist  
cool.