Mosbach – Java 2

VL 03

Klasse

class...

Implement. starten

Abstr. Klasse

~~Interface~~

erfüllt einen Teil des Interfaces, nicht alles

Interface

interface ...

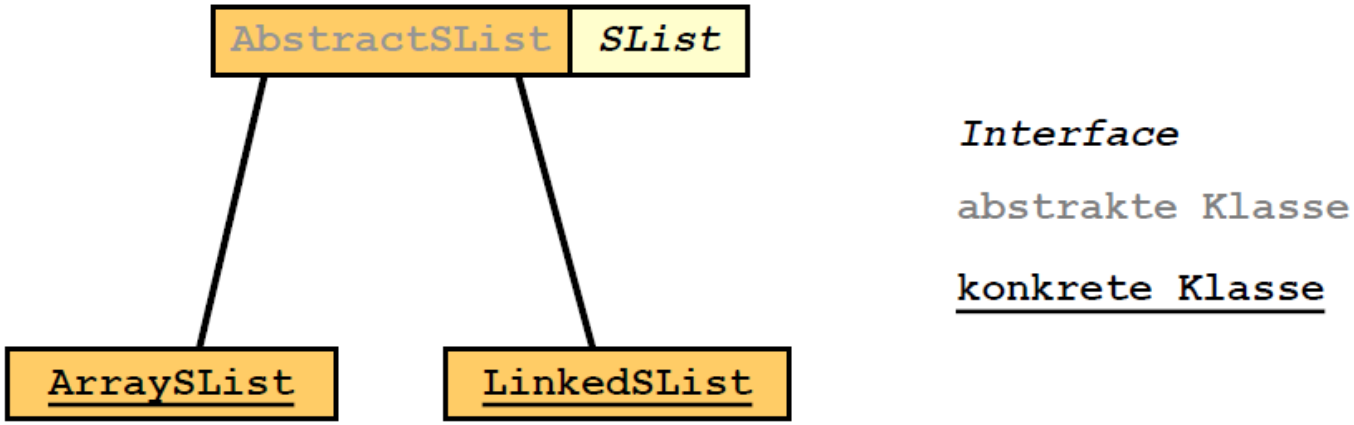
Signaturen (Methköpfe)

- Vertrag/Schnitt.
- Typ

Abstrakte Klassen

Abstrakte Klassen

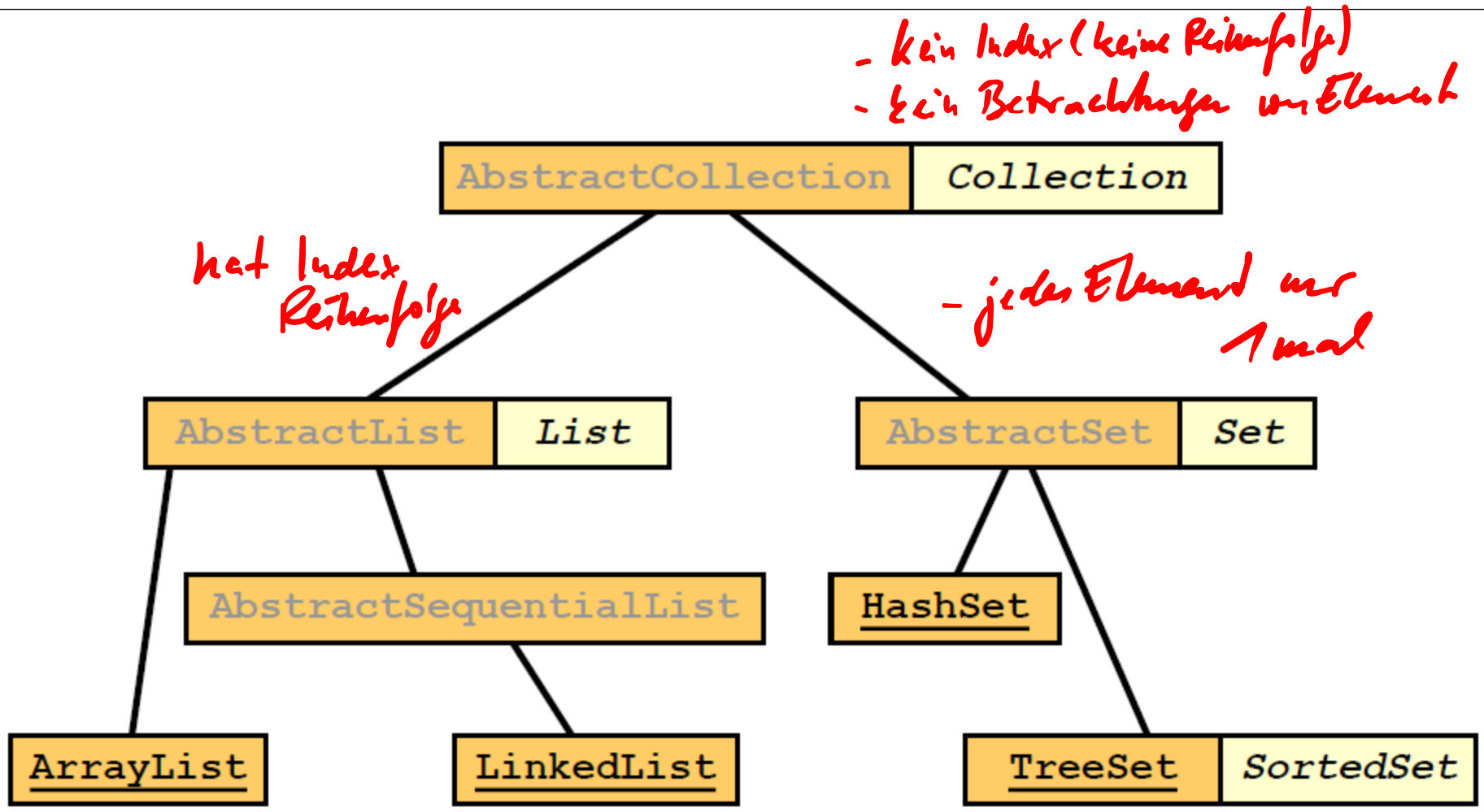
- Dem Programmierer viele Möglichkeiten bereits vorgeben, wie alles oder nur Teile eines interfaces implementiert sein könnten.

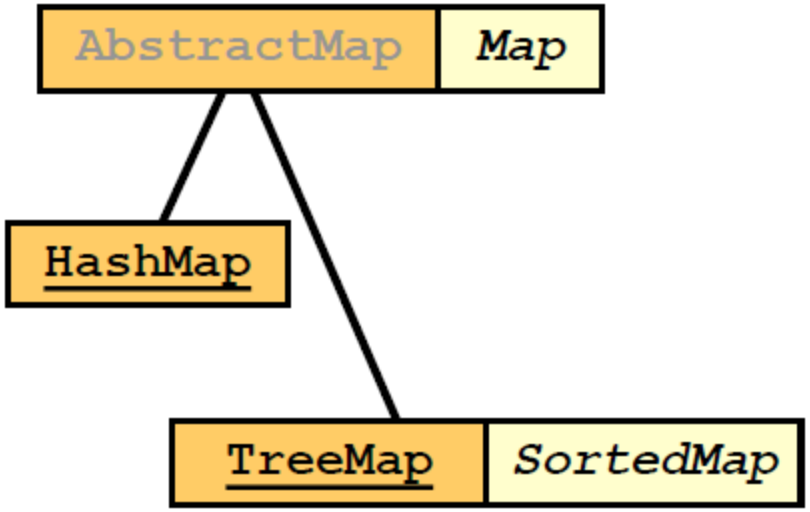


Container-Klassen

Collector Framework

- Collection: Container, der Dinge enthält
- List: Container mit Reihenfolge
  - get
  - contains
  - remove
  - clear
  - size


- kein Index (keine Reihenfolge)
- kein Betrachtungen von Element
AbstractCollection
Collection
hat Index
Reihenfolge
- jedes Element nur
1 mal
AbstractList
List
AbstractSet
Set
AbstractSequentialList
HashSet
ArrayList
LinkedList
TreeSet
SortedSet



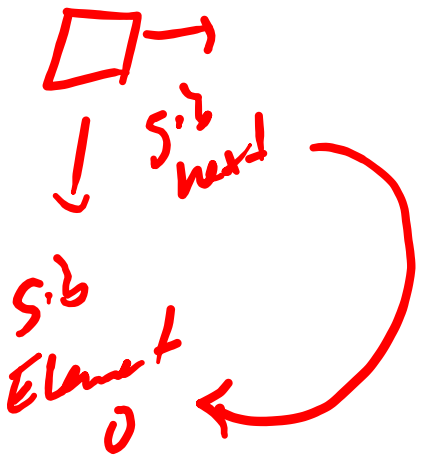
List : [2, 8, 16, 4, 7]

Set : {2, 8, 16, 4, 7}

Map : {[Name: Michal], [Age : 18], ... }

Idee des Durchlaufens durch einen Container

Idee des Vorgehens



Iterator

↳ alle erfüllen das

↳ forEach stellt überall

Collector Framework, II

- Iterator: etwas, was durch alle Elemente eines Containers durchläuft
  - next()
  - hasNext()
- Collection .. has only add, get
- List .. has also
  - iterator() ... it gives an iterator
  - sort() ... sorts but needs an comparator

Generics

ArrayList<Staff>

Ohne Generics

- ArrayList<Object>
- get(0) → Object
- add(Object)

list.add(Staff)
get(0) → Object
Cast!!

Mit Generics

ArrayList<Staff>

- get(0) → Staff
- add(Staff)

list.add(Staff)
get(0) → Staff
Typsicher!!

Generics Example, ArrayList

- ArrayList<T>
- denn der ArrayList-Programmierer kann nicht wissen, welchen Typ ich verwenden möchte

- ArrayList<Object>
- dann müsste ich später ständig casten
- und kann auch falsch casten

- ArrayList<MyClass>
- effizient und typsicher

Aufgaben

- Liste von Zufallszahlen
- Ausgabe über den Iterator

- Erweitere unsere SortedLinkedList um einen Iterator
- Zeige, dass dann auch forEach anwendbar ist

Zusatz

- CollectionUtils

- Nutze aber Lambda!

```
List<Integer> primes = Arrays.asList(3, 5, 7, 11, 13)
CollectionUtils.exists(primes, even); //false

better use lambda
Predicate even = new Predicate() {
   public boolean evaluate(Object object) {
        return ((Integer)object) % 2 == 0;
   }
}
```

Wiederholung

- Konzepte
  - Abstrakte Klasse
- Richtige Klasse wählen
- Collection
- List
- Iterator

- `Arrays.asList`
- `forEach`
- `iterator()`
- `next()`
- `hasNext()`

Wiederholung

- Generics
- Slides Neuendorf Generics ansehen