05-Collections-1

Objektorientierte Programmierung | Matthias Tichy

Software Engineering
Programming Languages

universität ulm

Lernziele

- Queues
- Sets
- Maps

Gute und schlechte Nachrichten

Die gute Nachricht

- Sie müssen vermutlich niemals mehr eine verkettete Liste programmieren

Die schlechte Nachricht

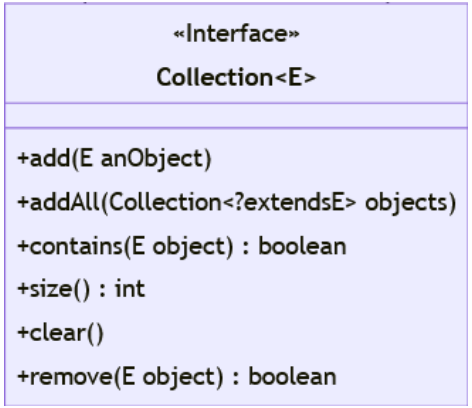
- Sie müssen die Datenstrukturbibliotheken verstehen und anwenden können

Queues

Queue<E>

Interface

```
public interface Queue<E> extends Collection <E> {
  boolean add(E e);
  boolean offer (E e);
  E peek();
  E remove();
  E poll();
  E remove();
}
```



- „Queues typically, but do not necessarily, order elements in a FIFO (first-in-first-out) manner.”

https://docs.oracle.com/en/java/javase/20/docs/api/java.base/java/util/Queue.html

Queue<E>

Interface

Aktion	Exception	spezieller Wert
Einfügen	add(e)	offer(e)
Löschen	remove()	poll()
Betrachten	element()	peek()

false bzw. null

```
var q = new LinkedBlockingQueue<Integer>();
q.offer(12);
q.offer(42);
var element = q.poll();
while (element != null) {
  System.out.println(element);
  element = q.poll();
}
```

Deque<E>

Interface

Aktion	erstes Element		letztes Element	
	Exception	spezieller Wert	Exception	spezieller Wert
Einfügen	addFirst(e)	offerFirst(e)	addLast(e)	offerLast(e)
Löschen	removeFirst()	pollFirst()	removeLast()	pollLast()
Betrachten	getFirst()	peekFirst()	getLast()	peekLast()

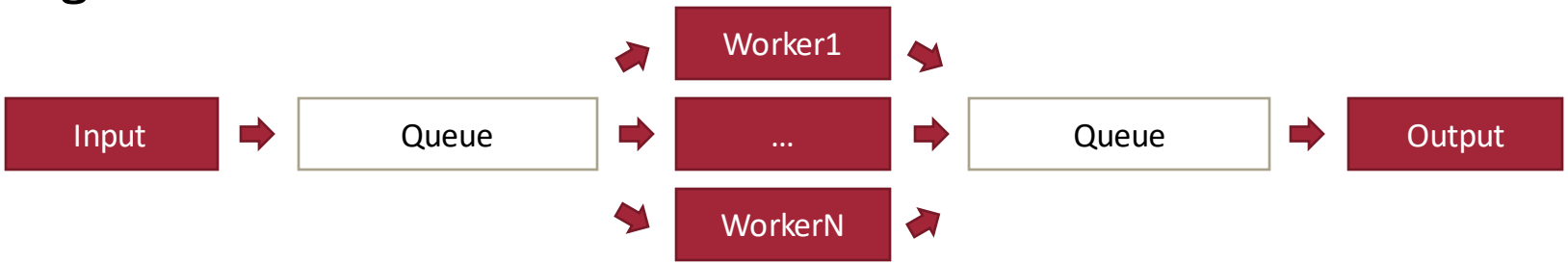
Stack Methode	Deque-Methode
push(e)	addFirst(e)
pop()	removeFirst()
peek()	peekFirst()

BlockingQueue<E>

Interface

Aktion	Exception	spezieller Wert	Blocks	Timeout
Einfügen	add(e)	offer(e)	put(e)	offer(e, time, unit)
Löschen	remove()	poll()	take()	poll(time,unit)
Betrachten	element()	peek()	n/a	n/a

- Insb. zur Kommunikation zwischen Threads in nebenläufigen Programmen



Exceptions

Queue<E>

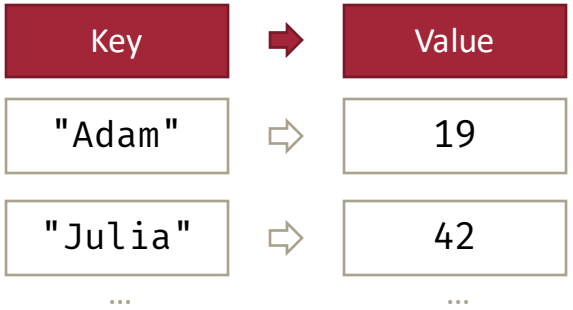
Aktion	Exception	spezieller Wert
Einfügen	add(e)	offer(e)
Löschen	remove()	poll()
Betrachten	element()	peek()

false bzw. null

```
try {
  inputQueue.add(12);
  // ... do many other things
  var e = outputQueue.remove();
  // ... do many other things
} catch (NoSuchElementException ex) {
  System.out.println("Queue empty: Send 503 error to requesting client.");
  // ...
} catch (IllegalStateException ex) {
  System.out.println("Queue full: Send 503 error to requesting client.");
  // ...
}
```

Maps und Sets

Hashtabelle



- Zugriff über einen berechneten Wert („key“)
- keine Ordnung mehr (Reihenfolge unbekannt)
- sehr schnell: ∅ O(1)
- Hashkollisionen
- ➔ "Algorithmen und Datenstrukturen"
- Anwendung:
  assoziatives Array (*HashMap*, *Map*, *Dictionary*, *Lookup Table*)

HashMap

- Hash (und damit Position) wird von einem Schlüsselwert (Key) berechnet
- gespeichert wird aber ein anderer Wert (Value)

- Typischerweise werden Keys auch mitgespeichert ➔ Key/Value-Paare

Map<K,V>

Interface

```
public interface Map<K, V> {
  ...
  V get(Object key);
  V put(K key, V value);
  V remove(Object key);
  boolean containsKey(Object key);
  boolean containsValue(Object value);
  ...
}
```

- Was passiert, wenn sich ein Key-Objekt ändert?

  → Problem (außer `key.equals(key2)`)

- Wieso wird `K` nur bei `put` verwendet?

  → Mehr Freiheit (ist aber Teils umstritten)

https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/Map.html

HashMap<K,V>

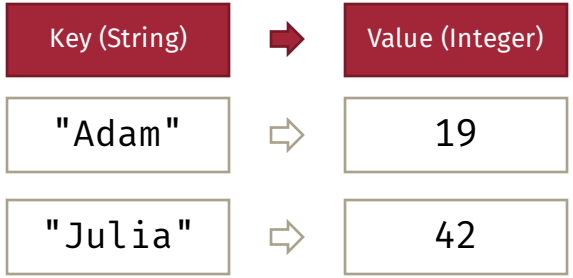
Interface

```
public class HashMap<K, V> extends AbstractMap<K,V> implements Map<K,V> {
  static final int DEFAULT_INITIAL_CAPACITY = 1 << 4; // aka 16
  static final float DEFAULT_LOAD_FACTOR = 0.75f;
  static final int TREEIFY_THRESHOLD = 8;
  transient Node<K,V>[] table;
  final Node<K,V>[] resize() {...}
  ...
}
```

- Keine garantierte Ordnung
- Was passiert bei Hashkollisionen, also `!key1.equals(key2)` und `hash(key1) == hash(key2)`?

https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/HashMap.html

Ein Hash-Map-Beispiel!




```java
import java.util.HashMap;

class HashMapHeaven {
    public static void main(String[] args) {
        var map = new HashMap<String, Integer>();

        map.put("Adam", 19);
        map.put("Julia", 42);

        System.out.println(map.get("Adam"));
    }
}
```

```
19
```

Set<E>, TreeSet<E>, HashSet<E>

Interface

```
public interface Set<E> extends Collection<E> {
  ...
}
```

- Eine *Collection*, die keine doppelten Elemente enthält.
- Formaler: Mengen enthalten keine Elemente `e1` und `e2` mit `e1.equals(e2)` und haben maximal ein `null`-Element

TreeSet	HashSet
Verwendet Baum-Struktur	Verwendet Hash-Map
Durch Sortierung geordnet	Ungeordnet

https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/Set.html

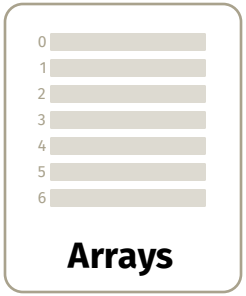
Generische Algorithmen

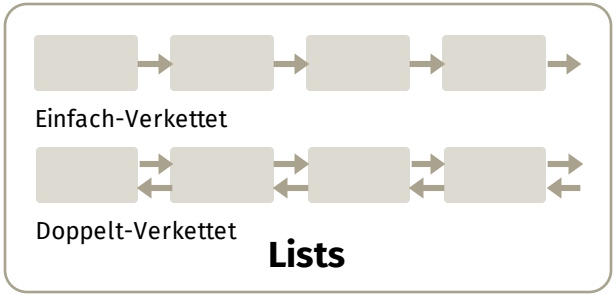
- Einige werden in der Collections-Klasse gesammelt
- Beispielsweise für Listen:
  - `shuffle`
  - `reverse`
  - `rotate`
  - `sort`
    (Merge-Sort bzw. Quicksort für primitive Typen)
  - `swap`
  - `replaceAll`
  - `fill`
  - `copy`
  - `binarySearch`
- Darüber hinaus bietet sie noch vieles mehr…
  z.B. Konvertierungsfunktionen

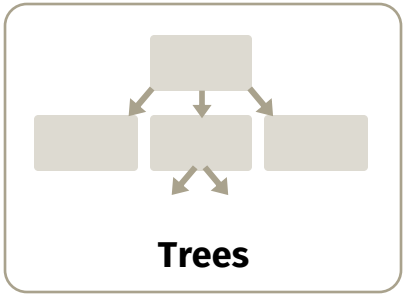
https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/Collections.html

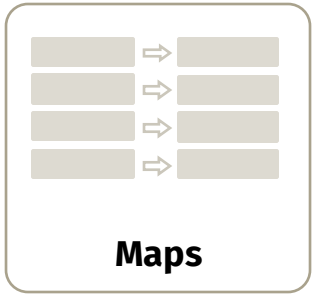
Auswahl von Datenstrukturen

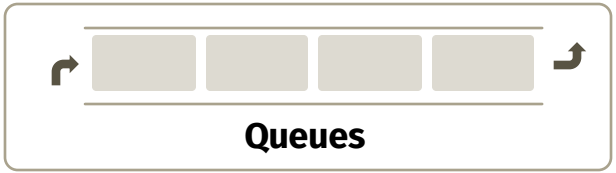
Wir kennen...

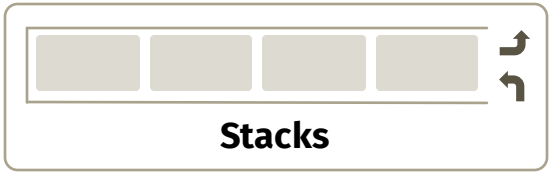

0
1
2
3
4
5
6
Arrays


Einfach-Verkettet
Doppelt-Verkettet
Lists


Trees


Maps


Queues

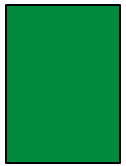

Stacks


Sets

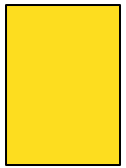
Welche Datenstruktur?

- Jemand gibt eine unbekannte Anzahl von Zahlen ein und Sie sollen den Mittelwert berechnen.

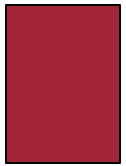
- … Liste



- … Array



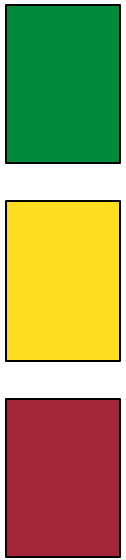
- … Hashtable



Welche Datenstruktur?

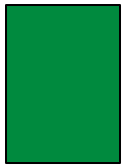
- Implementieren Sie ein Telefonbuch!

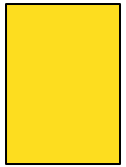
- … Hashtable
- … Baum
- … Queue

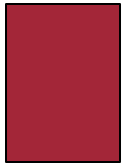


Welche Datenstruktur?

- Implementieren Sie eine Browser-Historie mit der Möglichkeit zurück und wieder vorwärts zu gelangen!
- … Doppelt verkettete Liste
- … Stack
- … Baum

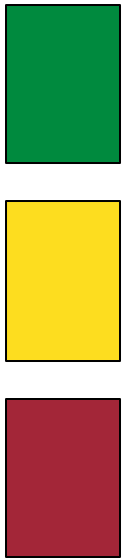






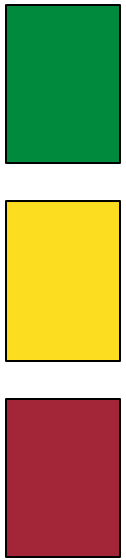
Welche Datenstruktur?

- Ermitteln Sie aus Klausurteilnehmern die 5 Studierenden mit den meisten und den wenigsten Punkten!
- … einfach verkettete Liste
- … Array
- … Baum



Welche Datenstruktur?

- Bestimmen Sie den Notenspiegel der vorher genannten Klausur!
- … Hashtable
- … Queue
- … Set



Lernziele

- Queues
- Sets
- Maps