

Java Collections Framework

André Matutat & Carsten Gips (HSBI)

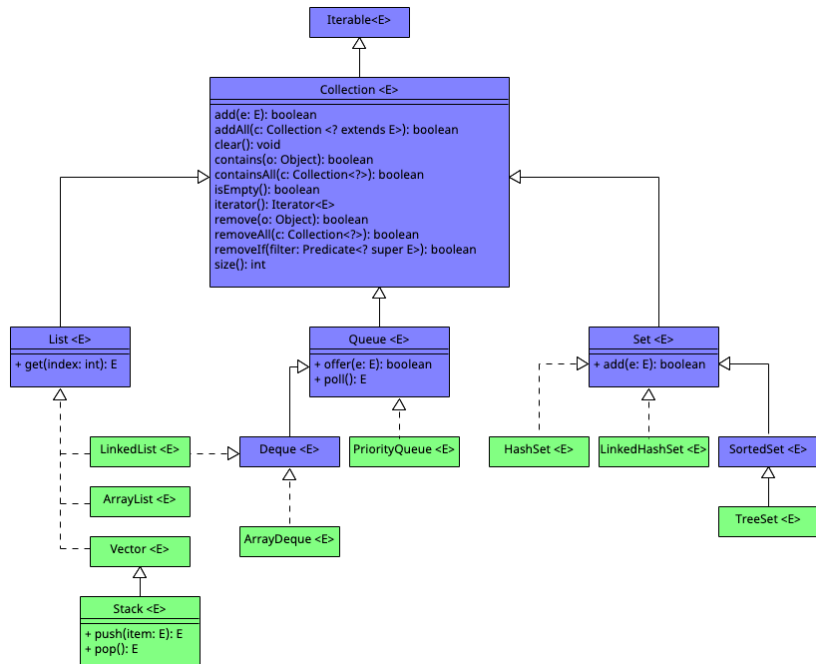
Unless otherwise noted, this work is licensed under CC BY-SA 4.0.

Motivation: Snippet aus einer Klasse im PM-Dungeon

```
private List<Entity> entities = new ArrayList<>();

public void add(Entity e){
    if (!entities.contains(e)) entities.add(e);
}
```

Collection-API in Java

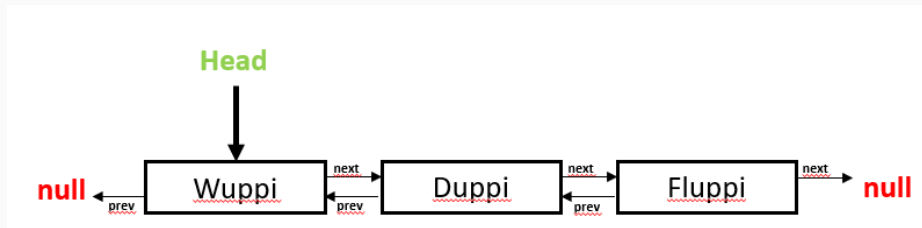


Listen: *ArrayList*

```
private List<Entity> entities = new ArrayList<>();
```



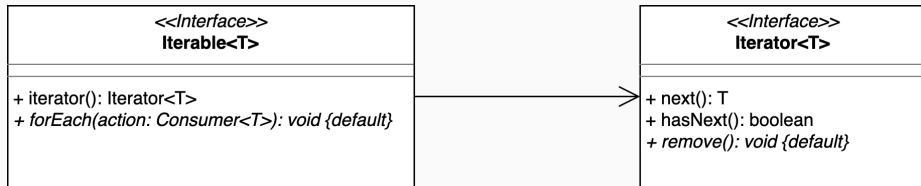
Listen: *LinkedList*



- `Vector<T>`:
 - Ein `Vector<T>` ähnelt einer `ArrayList<T>`
 - Das Array eines Vector wird jedoch verdoppelt, wenn es vergrößert wird
 - Die Methoden von `Vector<T>` sind `synchronized`
- `Stack<T>`:
 - Schnittstelle: "last in first out"-Prinzip
 - `push(T)`: Pushe Element oben auf den Stack
 - `pop(): T`: Hole oberstes Element vom Stack
 - Tatsächlich aber: `class Stack<E> extends Vector<E>`

Iterierbarkeit: *Iterable* und *Iterator*

```
private List <Entity> entities = new ArrayList<>();  
  
for (Entity e : entities) { ... }  
entities.forEach(x -> ...);
```



Collections

`addAll(c: Collection<? super T>, elements: T...): boolean`

`replaceAll(list: List<T>, oldVal: T, newVal: T): boolean`

`rotate(list: List<?>, distance: int): void`

`shuffle(list: List<?>): void`

`sort(list: List<?>, c: Comparator<? super T>): void`

`swap(list: List<?>, i: int, j: int): void`

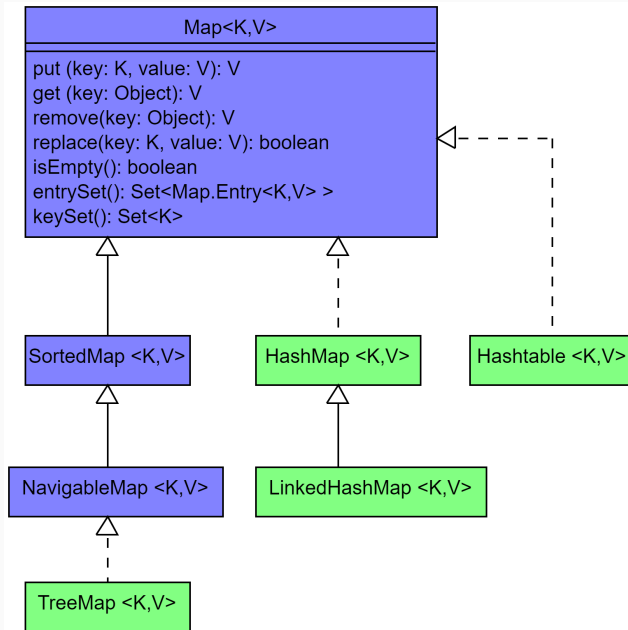
`frequency(c: Collection<?>, o: Object): int`

`fill(list: List<? super T>, obj: T): void`

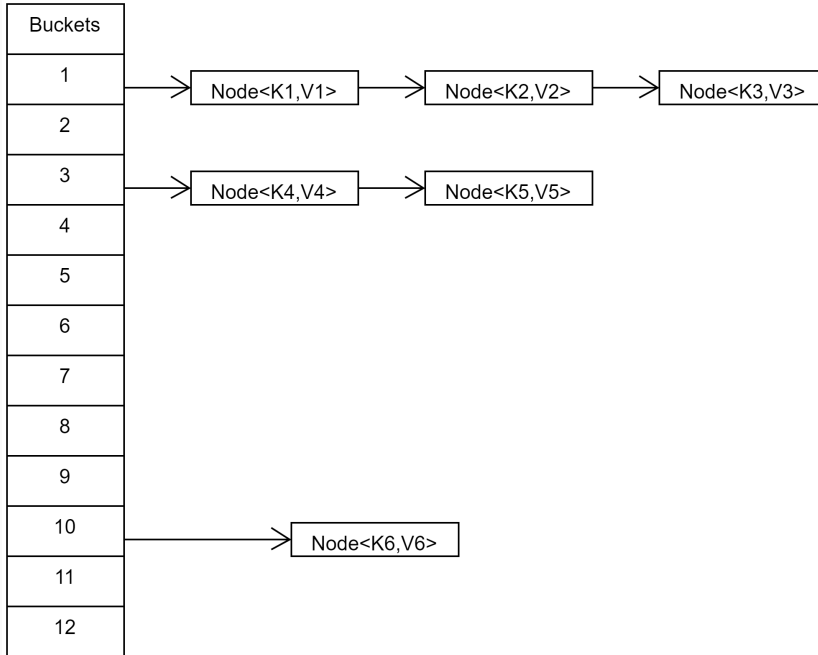
`disjoint(c1: Collection<?>, c2: Collection<?>): boolean`

`copy(dest: List<? super T>, src: List<? extends T>): void`

Map



HashMap



- Nicht zu verwechseln mit der Datenstruktur: Hash-Tabellen (!)
- `Hashtable<K,V>` ist vergleichbar mit einer `HashMap<K,V>`
- `Hashtable<K,V>`-Methoden sind `synchronized`
- Kein Key oder Value darf `null` sein

Der `equals()`-`hashCode()`-`compareTo()`-Vertrag

Wird `equals()` überschrieben, sollte auch `hashCode()` (passend) überschrieben werden.

1. Wenn `x.equals(y) == true`, dann *muss* auch `x.hashCode() == y.hashCode()`
2. Wenn `x.equals(y) == false`, *sollte* `x.hashCode() != y.hashCode()` sein
3. `x.compareTo(y) == 0` gdw. `x.equals(y) == true`

- Interface `Collection<T>`: Schnittstelle für Datenstrukturen/Sammlungen
- Klasse `Collections`: Statische Hilfs-Methoden
- `Iterable<T>` liefert einen `Iterator<T>` zur Iteration über eine `Collection<T>`
- Interface `Map<K,V>`: Speichern von Key/Value-Paaren
- `equals()` - `hashCode()` - `compareTo()` - Vertrag beachten

LICENSE



Unless otherwise noted, this work is licensed under CC BY-SA 4.0.