

## Inhaltsverzeichnis

1. Kurstag 2 - Basics 2/2
  - 1.1. Recap
  - 1.2. Überladen & Überschreiben
  - 1.3. Collections Framework
    - 1.3.1. Java Collections API Interfaces
    - 1.3.2. Special Java Collections Classes
    - 1.3.3. Synchronized Wrappers
    - 1.3.4. Unmodifiable Wrappers
  - 1.4. Assoziation
    - 1.4.1. One-to-One-Assoziation
    - 1.4.2. One-to-Many-Assoziation
    - 1.4.3. Many-to-Many-Assoziation
  - 1.5. Übungen
  - 1.6. Tipps, Patterns & Best Practices

## 1. Kurstag 2 - Basics 2/2

### Allgemeine Inhalte

- ☐ Methoden überladen, überschreiben/übersteuern
- ☐ Collections API
- ☐ Assoziationen (Beziehungsart "Besitz" im Code, sh. auch Kurstag 1)

### Fachlicher Kontext

Mögliche Assoziationen/Relationen mit Kardinalitäten

```
+ Zug      1:n    Wagon
+ Zug      1:1    Lokomotive
+ Zug      n:m    Soll-Fahrplan
(=> Zug 1:n Fahrt 1:n Soll-Fahrplan)

+ Zug      n:m    Strecke
+ Zug      1:n    Fahrt
+ Strecke  1:n    Fahrt
(=> Zug 1:n Fahrt n:1 Strecke)

+ Fahrt    1:n    Reisende
+ Zug      1:2..3  Flügel
+ Streckennetz 1:n  Strecke
+ Strecke  n:m    Abschnitte
```

### 1.1. Recap

Was bisher geschah ...

- ☐ Klassen & Objekte
- ☐ Objektvertrag (hashCode, equals)
- ☐ Vererbung & Interfaces

- Abstrakte Klassen
- Access Modifier (Sichtbarkeiten, Scopes)
- Beziehungsarten (Besitz, Aufruf, Vererbung, ...), Aggregation vs. Komposition etc.

## 1.2. Überladen & Überschreiben

### Beispiel Überladen von Methoden

```
public class PlatformDisplay {

    // multiple methods can be used to
    // update a platforms' display

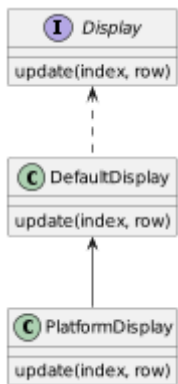
    public void update() {}

    public void update(String line) {}

    public void update(List<String> lines) {}

}
```

JAVA



### Beispiel Übersteuern von Methoden

```
public interface Display {

    void update(int index, String row);

}
```

JAVA

Die zugehörige Annotation im Code ist

```
@Override
```

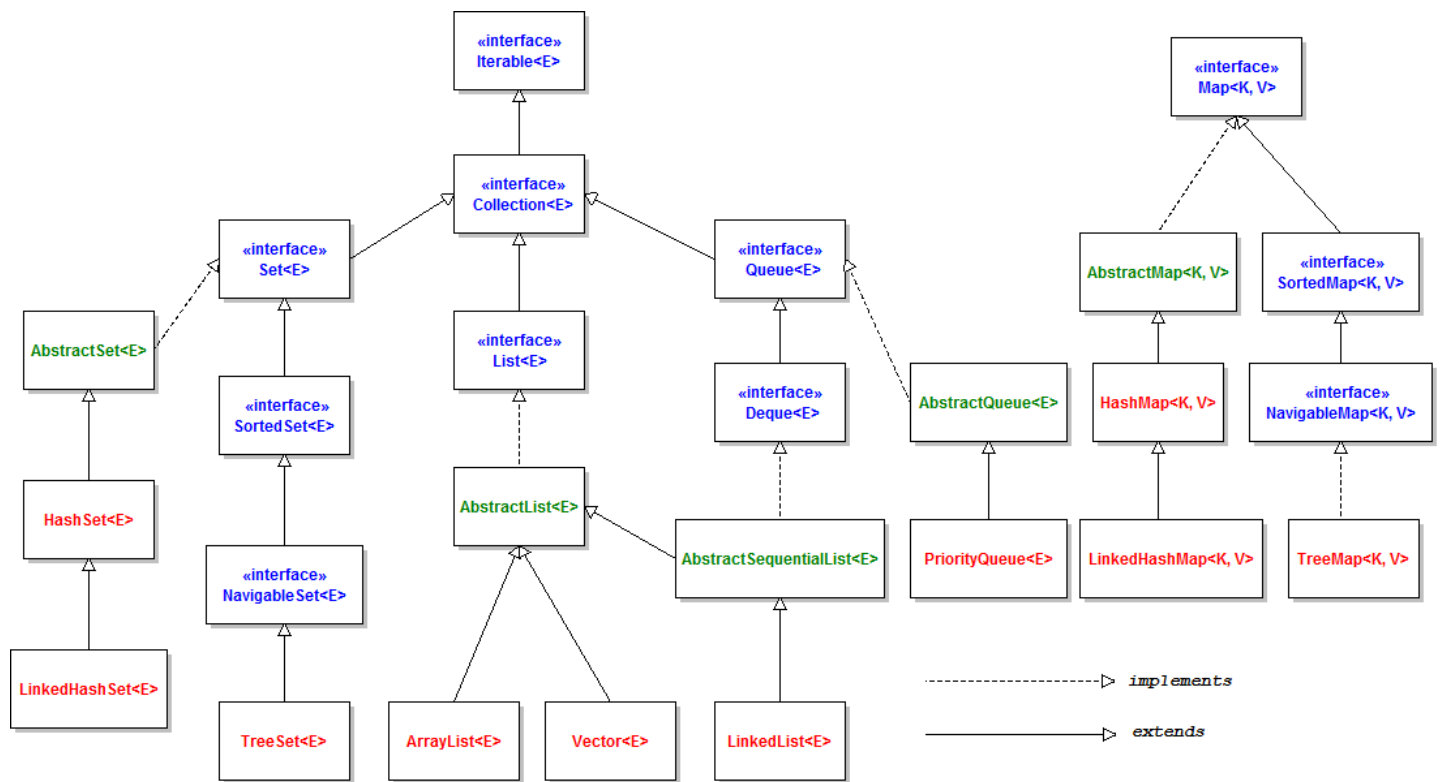
Sie sollte in jedem Fall genutzt werden, zudem wird sie auch von der IDE vorgeschlagen

## 1.3. Collections Framework

Siehe z.B. → [Java Collections Framework auf Wikipedia](https://en.wikipedia.org/wiki/Java_collections_framework) ([https://en.wikipedia.org/wiki/Java\\_collections\\_framework](https://en.wikipedia.org/wiki/Java_collections_framework))

### Die wichtigsten Auswahlkriterien:

- Position von Elementen (*numerischer index*)
- Möglichkeit zur Ordnung von Elementen (z.B. *insertion order*)
- Möglichkeit von `null` Elementen oder Duplikaten (*Set* vs. *List*)
- Zugriff auf Elemente anhand eines Schlüssels (*List* vs. *Map*)



### 1.3.1. Java Collections API Interfaces

Java collection interfaces are the foundation of the Java Collections Framework. All core collection interfaces are generic. For example `public interface Collection<E>`. The `<E>` syntax is for Generics and when we declare Collection (später mehr zum Thema 'Generics')

- Collection interface.** This is the root of the collection hierarchy. A collection represents a group of objects known as its elements. Some basic operations are provided.
- Iterator Interface.** Iterator interface provides methods to iterate over the elements of the Collection. Iterators allow the caller to remove elements from the underlying collection during the iteration.
- Set Interface** Set is a collection that cannot contain duplicate elements. The Java platform contains three general-purpose Set implementations: HashSet, TreeSet, and LinkedHashSet.
- List Interface** List is an ordered collection and can contain duplicate elements. You can access any element by its index. List has a dynamic length. ArrayList and LinkedList are implementation classes of List interface.
- Queue Interface** Queue is a collection used to hold multiple elements prior to processing. Besides basic Collection operations, a Queue provides additional insertion, extraction, and inspection operations. Queues typically, but do not necessarily, order elements in a FIFO (first-in-first-out) manner.
- Deque Interface** A linear collection that supports element insertion and removal at both ends. The name deque is short for "double-ended queue"
- Map Interface** Java Map is an object that maps keys to values. A map cannot contain duplicate keys, each key can map to at most one value. The Java platform contains three general-purpose Map implementations: HashMap, TreeMap, and LinkedHashMap.
- SortedSet Interface** SortedSet is a Set that maintains its elements in ascending order. Sorted sets are used for naturally ordered sets.
- SortedMap Interface** A map that maintains its mappings in ascending key order. This is the Map analog of SortedSet. Sorted maps are used for naturally ordered collections of key/value pairs.

### 1.3.2. Special Java Collections Classes

Java Collections framework comes with many implementation classes for the interfaces. Most common implementations are:

1. `HashSet` Class
2. `TreeSet` Class
3. `ArrayList` Class
4. `LinkedList` Class
5. `HashMap` Class
6. `TreeMap` Class

### 1.3.3. Synchronized Wrappers

The **synchronization** wrappers add automatic synchronization (thread-safety) to an arbitrary collection. Each of the six core collection interfaces - `Collection`, `Set`, `List`, `Map`, `SortedSet`, and `SortedMap` - has one static factory method, which return a synchronized (thread-safe) collection backed up by the specified collection.

### 1.3.4. Unmodifiable Wrappers

**Unmodifiable** wrappers take away the ability to modify the collection by intercepting all the operations that would modify the collection and throwing an `UnsupportedOperationException`.

Its main usage are;

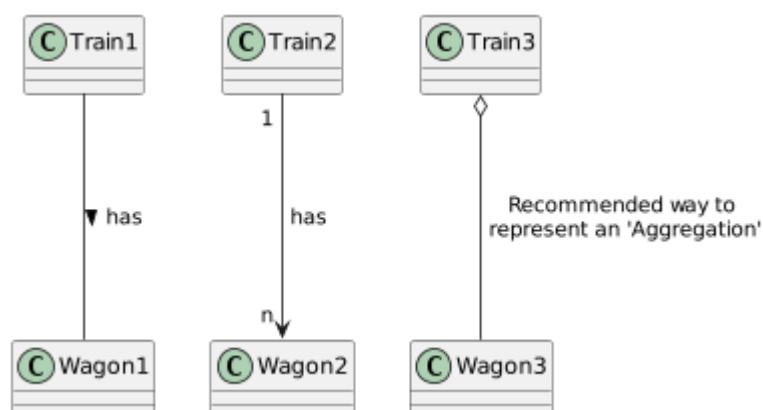
- To make a collection immutable once it has been built
- To allow certain clients read-only access to data structures (keep a reference to the backing collection but hand out a reference to the wrapper)
- To avoid `ConcurrentModificationException`

## 1.4. Assoziation

### 1.4.1. One-to-One-Assoziation

### 1.4.2. One-to-Many-Assoziation

Ein Beispiel für eine Eins-zu-Viele Beziehung:

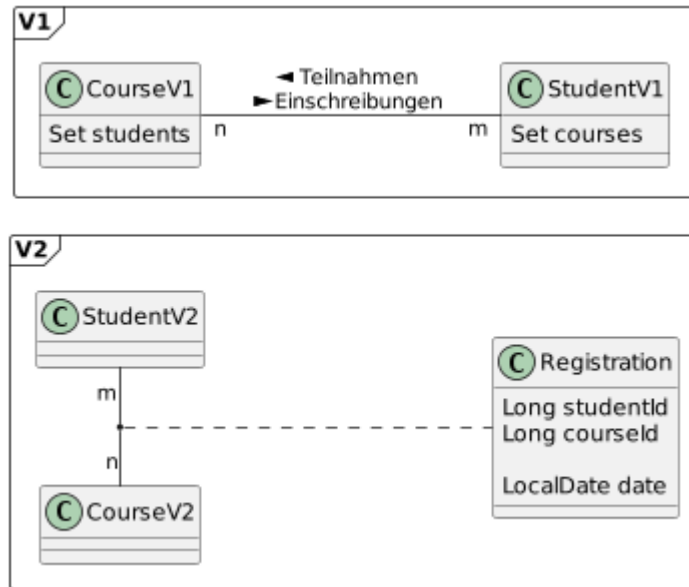


Siehe Implementierungen für V1 und V2:

```
de.dhbw.course2.basics.collections.n
```

### 1.4.3. Many-to-Many-Assoziation

Die folgenden zwei Varianten kommen am häufigsten vor:



Implementierungsbeispiele:

```
de.dhbw.course2.basics.collections.nm
```

Mehr zu Datenbanken & SQL siehe → Kurstag 7.

## 1.5. Übungen

Die Übungsaufgaben können hier (innerhalb von Kurs 2) implementiert werden:

```
/src/test/java/de/dhbw/course2/study/MyStudyTests.java
```

### Übung 1

Erstelle eine Klasse **News**, die eine Aussage als String enthalten können. Mache die News durch Nutzung der **equals()** Methode vergleichbar, sodass du prüfen kannst, ob 2 News (= 2 Instanzen der Klasse News) mit verschiedenen - vielleicht *ähnlichen* Aussagen - inhaltlich gleich oder sogar identisch sind.

### Übung 2

Schreibe einen Test, in dem eine **SortedMap** benutzt wird, befülle diese mit mindestens 5 Einträgen, nutze dazu **String** sowohl für den Schlüssel (K) also auch für den Wert (V). Beschreibe die Haupt-Charakteristik den dieses Map-Typus und prüfe, ob die Haupteigenschaft gegeben ist.

### Übungsfragen

Hier findest du sich kleine Übungsfragen für die Inhalte des Kurses 2:

```
/exam/src/test/java/de/dhbw/exam/course2/ExamTests.java
```

## 1.6. Tipps, Patterns & Best Practices