STREAM-API

STREAM BEISPIEL

```
Owner crazyCatLady = new Owner("Crazy Cat Lady");
Owner horst = new Owner("Horst");
Owner gertrude = new Owner("Gertrude");

List<Animal> animals = List.of(
        new Dog(horst),
        new Cat(crazyCatLady),
        new Cat(gertrude),
        new Dog(gertrude),
        new Cat(crazyCatLady));

List<String> ownerNames = new ArrayList<>();
for (Animal animal : animals) {
    if (animal instanceof Cat) {
        Owner owner = animal.getOwner();
        String name = owner.getName();
        ownerNames.add(name);
    }
}
```

```
List<String> ownerNames = animals.stream()    // Stream<Animal>
        .filter(animal -> animal instanceof Cat)    // Stream<Animal>
        .map(Animal::getOwner)    // Stream<Owner>
        .map(Owner::getName)    // Stream<String>
        .toList();    // List<String>
```

- API, um verschachtelte Schleifen elegant auszudrücken
- Nutzt viele funktionale Interfaces (=> Lambdas / Methodenreferenzen)

- Zwischenschritte: Liefern abgewandelten Stream
- Meist lazy: Tun nur etwas wenn nötig und so spät wie möglich
- Ausnahme: „stateful intermediate Operations“ wie distinct und sorted

Methode	Parameter	Rückgabe	Beschreibung
`map`	`Function<T, R>`	`Stream<R>`	Wandelt Inhalt des Streams um
`filter`	`Predicate<T>`	`Stream<T>`	Filtert Objekte aus Stream
`flatMap`	`Function<T, Stream<R>>`	`Stream<R>`	Löst Verschachtelung auf
`distinct`	-	`Stream<T>`	Entfernt Duplikate
`sorted`	`Optional: Comparator<T>`	`Stream<T>`	Sortiert Stream

- Letzter Aufruf: Liefern Ergebnis aus Stream
- Aufruf löst die Abarbeitung von intermediate Operators aus

Methode	Parameter	Rückgabe	Beschreibung
`collect`	`Meist: Collector`	`*`	Wandelt Inhalt des Streams um
`toList`	`-`	`List<T>`	Shortcut für collect(Collectors.toList())
`findFirst / findAny`	`-`	`Optional<T>`	Liefert erstes / irgendein Element aus Stream gewrappt in einem Optional
`forEach`	`Consumer<T>`	`-`	Führt Consumer für jedes Element durch
`anyMatch`	`Predicate<T>`	`boolean`	Evaluiert, ob mindestens ein Element das Predicate erfüllt

- Klasse mit vielen Methoden zur Collector-Erzeugung

Methode	Beschreibung
`toList`	Erstellt Liste aus Stream-Elementen
`toSet`	Erstellt Set aus Stream-Elementen
`joining`	Fügt Strings aus Stream<String> zusammen
`counting`	Zählt Elemente
`groupingBy`	Gruppiert Elemente anhand eines Kriteriums zu einer Map

Aus Collections:

```
animalList.stream();

animalSet.stream();

animalCollection.stream();
```

Aus Arrays:

```
Arrays.stream(animalArray);

Stream.of(animalArray);
```

Aus Varargs:

```
Stream.of(animal1, animal2, animal3);
```