**Streams**

**Was ist ein Stream?**

* **Einmalige Verwendung**: Ein Stream kann nur einmal verwendet werden.
* **Lazy Evaluation**: Operationen werden erst ausgeführt, wenn eine terminale Operation aufgerufen wird

**Intermediate Operations**

* **Definieren die Verarbeitung des Streams**
* **Beispiele**:
  + filter( Predicate ): Filtert Elemente basierend auf einem Predicate
  + map( Function ): Transformiert jedes Element im Stream
  + distinct(): Entfernt Duplikate
  + sorted( Comparator ): Sortiert die Elemente
* **Eigenschaften**:
  + Sind lazy (erst bei terminal Operations ausgeführt)
  + Rückgabe ist immer ein neuer Stream
  + Unveränderlich, da Streams nicht das Original verändern

**Terminal Operations**

* **Lösen die Berechnung des Streams aus**
* **Beispiele**:
  + forEach( Consumer ): Führt eine Aktion für jedes Element aus
  + collect( Collector ): Wandelt in eine bestimmmte Collection um
  + reduce( BinaryOperation ): Reduziert die Elemente auf einen einzigen Wert
  + count(): Gibt die Anzahl der Elemente zurück
  + ...Match( Predicate ): Prüfen, ob Elemente bestimmten Kriterien entsprechen
* **Eigenschaften**:
  + Stream wird nach der terminalen Operation geschlossen
  + Alle bevorigen Intermediate Operations werden bei Ausführung der Terminal Operation durchlaufen

**Optional**

* **Beschreibung**: Ein Container-Objekt, das entweder einen Wert enthält oder leer ist
* **Verwendung**: Oft Ergebnis von terminalen Operationen wie findFirst(), findAny()
* **Methoden**:
  + isPresent(): Prüft, ob ein Wert da ist
  + ifPresent(): Führt eine Aktion aus, wenn ein Wert da ist
  + orElse(), orElseGet( Supplier ), orElseThrow(): Gibt den Wert oder führt andere Aktion aus
* Hilft, **NullPointerExceptions** zu **vermeiden**

**Erstellen eines Streams**

* **Aus einer Collection**: stream(), parallelStream()
* **Aus einem Array**: Arrays.stream(array)
* **Aus Werten**: Stream.of(1, 2, 3)
* **Aus einem File**: Files.lines(Path)
* **Primitiv Streams**: IntStream, LongStream, DoubleStream

**Fluent Programming**

* Methodenaufrufe werden in eigene Zeilen geschrieben, um es lesbarer zu machen
* **Beispiel:**

List<String> strings = Arrays.asList("abc", "def", "aaaa");

strings.stream()

.filter(s -> s.length() == 3)

.map(String::toUpperCase)

.forEach(System.out::println);

* **Code ist leserlicher und verständlicher**