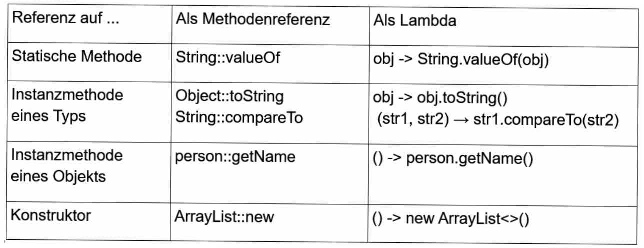


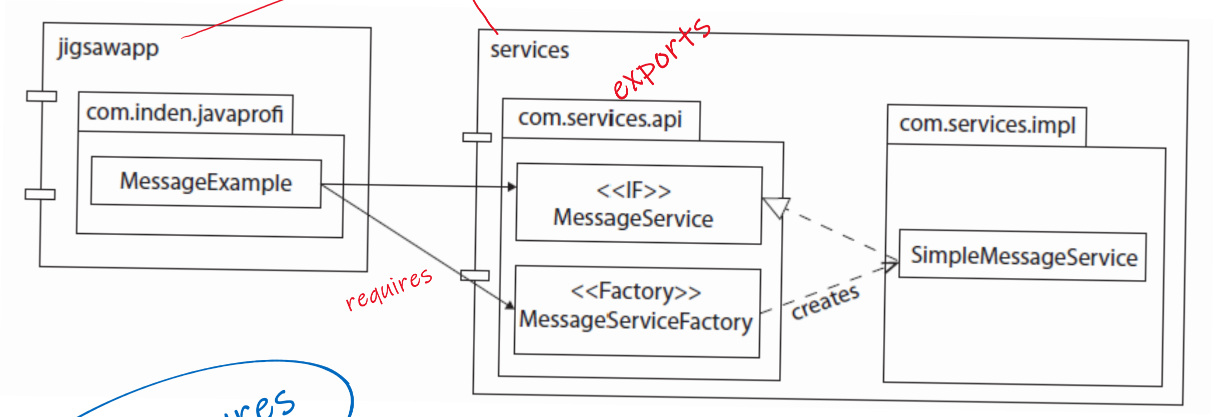
Referenz auf ...	Als Methodenreferenz	Als Lambda
Statische Methode	String::valueOf	obj -> String.valueOf(obj)
Instanzmethode eines Typs	Object::toString String::compareTo	obj -> obj.toString() (str1, str2) → str1.compareTo(str2)
Instanzmethode eines Objekts	person::getName	() -> person.getName()
Konstruktor	ArrayList::new	() -> new ArrayList<>()

Lambda

```
MeinFuncInterface f = (a, b) -> a - b;
int ergebnis = f.tuWas(7, 3);
```

Jigsaw

Modules



New Features

Streams

```
List<String> kurzeTage = TAGE.stream()          Erzeugung / Create
    .map(s -> s.substring(0, 2))  Verarbeitung / Intermediate
    .collect(Collectors.toList());  Ergebnis / Terminal
```

Optional

```
var opt = Optional.ofNullable(x);
 opt.filter(str ->
      str.contains("ab")).ifPresent(System.out::println);
// oder
opt.map((String::toUpperCase)).
          filter(s -> s.contains("AB")).
          ifPresent(System.out::println);
```

Create
----
a.stream()
Arrays.stream(data)
Stream.of(1,2,3,4)

Intermediate

.filter()
.sorted()
.map()
.peek()
distinct()
...

Geben wieder einen Stream zurück.

Terminal

.forEach()
.toArray
.collect
.reduce()
.xxMatch()
.findXx()
...

Date/Time

```
LocalDate heute = LocalDate.now();
LocalDate weihnachten = LocalDate.of(2019, 12, 24);

LocalDateTime dateTime = LocalDateTime.now()
   .minusHours(5).minusMinutes(30);

String datumAlsText = LocalDate.now().format(
   DateTimeFormatter.ofPattern("dd.MM.yyyy"));
```