

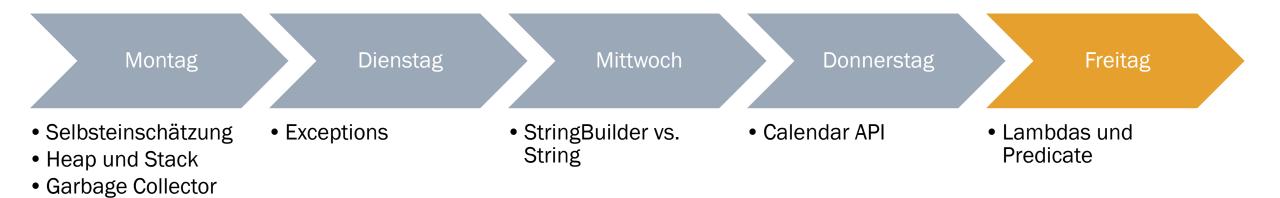
Neue Themen für die 808

Februar 2025



Plan für die Woche

und Object Lifecycle





Plan für heute

- Predicate
- Was sind Lambdas?
- Lambda-Ausdruck
- Wiederholung ArrayLists
- ArrayLists und Lambdas



Codevorführung Interfaces Rechner



Was sind Lambdas?

BUCHSEITE S.208-213

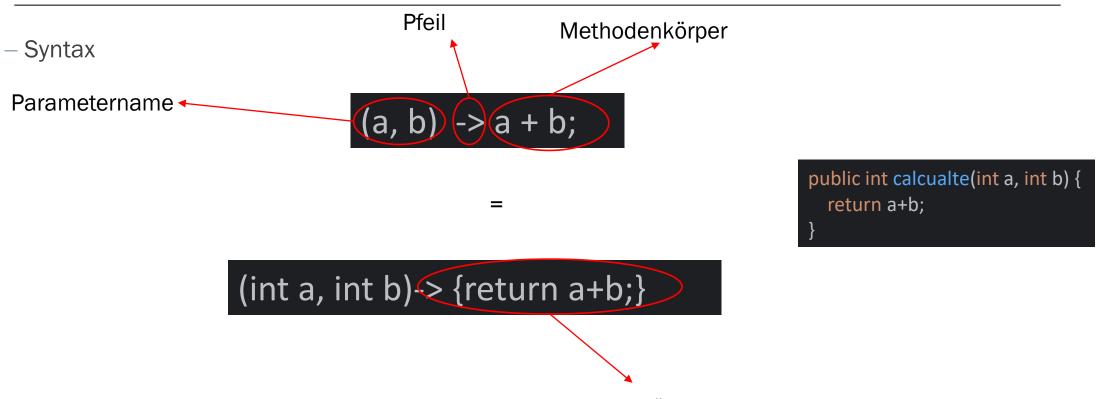


Was sind Lambdas?

- ist ein anderer Programmierstil
- seit Java 8
 - Bei switch-cases erst ab Java 14
- soll ermöglichen, Funktionen als Parameter einer Methode zu übergeben
 - Funktion: für einen Input gibt es einen Output
- führt zu kürzerem Code
 - Anonyme Klasse wird erzeugt, welche das funktionale Interface implementiert
 - Dieser Vorgang wird unter dem Lambdaausdruck versteckt



Lambdas



Mittels {} möglich Multi-line Code zu schreiben. Dann ist return und Semikolon aber notwendig!!



Lambdas

weitere Syntaxmöglichkeiten

() -> true

(a, b) -> a.startsWith("Hello")

(String a, String b) -> a.startsWith("test")



Mehrere oder kein Parameter möglich (je nach Methodenkopf im Interface)



Lambdas

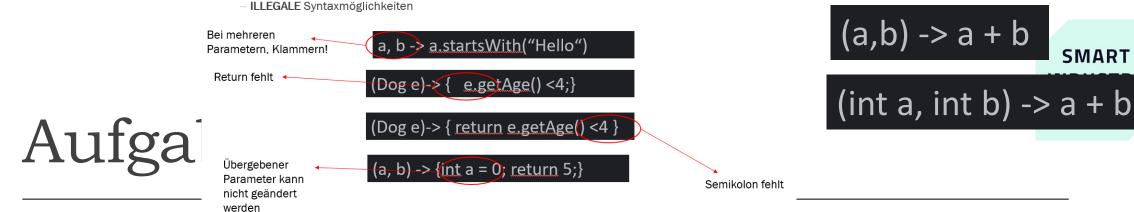
ILLEGALE Syntaxmöglichkeiten

Bei mehreren Parametern, Klammern! $a, b \rightarrow a.startsWith("Hello")$ Return fehlt $(Dog e) \rightarrow \{ e.getAge() < 4; \}$

(Dog e)-> { return e.getAge() <4 }

Übergebener Parameter kann nicht geändert werden $(a, b) \rightarrow \{(int a = 0); return 5;\}$

Semikolon fehlt



- 1. Erstelle ein Interface Calculator, welches eine einzige abstrakte Methode calculate(int a, int b) haben soll, die einen int zurückgibt.
- 2. Erstelle eine Klasse Plus, Minus, Teilen, Multiplizieren, die das Interface implementiert und die calculate-Methode überschreibt
- 3. Erstelle eine Test-Klasse, in welcher du die Klassen Plus, Minus, Teilen, Multiplizieren ausprobierst.
- 4. Erstelle nun in der Test-Klasse ein Objekt des Interfaces Calculator und schreibe einen Lambdaausdruck, wodurch die calculate-Methode des Interfaces direkt implementiert wird.



SMART INDUSTRY CAMPUS

Predicate

BUCHSEITEN S.214-215



Predicate

Lambdas funktionieren nur mit funktionalen Interfaces!

T steht für das Objekt, welches wir übergeben vgl. ArrayLists

- ist ein funktionales Interface
 - Funktionales Interface: ein Interface, welches eine einzige abstrakte Methode enthält

Die Methode kann auch andere Datentypen zurückgeben und mehr Argumente akzeptieren



Was sind Lambdas?

- ist ein anderer Programmiers
- seit Java 8
- soll ermöglichen, Funktionen
 - Funktion: für einen Input gibt e
- führt zu kürzerem Code
 - Anonyme Klasse wird erzeug for (Dog d: getSpecificDogs(dogs, Doge -> e.getAge() < 4))</p>
 - Dieser Vorgang wird unter dem Lambdaausdruck versteckt

```
for(Dog d: getSpecificDogs(dogs, new Predicate<Dog>() {
    @Override
    public boolean test(Dog e) {
        return e.getAge() < 4;
    }
})){</pre>
```

SMART INDUSTRY CAMPUS

Codevorführung

e -> e.getAge() < 4



Aufgabe

(Dog e)-> {return e.getAge() <4;}

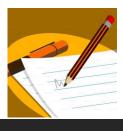
Schreibe einen Predicate<Integer>, der true zurückgibt, wenn die Zahl gerade ist, sonst false

Das könnte folgendermaßen aussehen:

Predicate<Integer> isEven = LAMBDAAUSDRUCK;

Teste deine Lösung, indem du folgendes eingibst:

System.out.println(isEven.test(15));



```
public interface Calculator {
  int calcualte(int a, int b);
}
Calculator add = (int a, int b) -> a + b;
```



Schreibe einen Predicate<Integer>, der true zurückgibt, wenn die Zahl größer als 20 ist, sonst false

Das könnte folgendermaßen aussehen:

Predicate<Integer> biggerThan20 = LAMBDAAUSDRUCK;

Teste deine Lösung, indem du folgendes eingibst:

System.out.println(biggerThan20.test(15));

T ist in unserem Fall ein Integer

e -> e.getAge() < 4



Aufgabe

(Dog e)-> {return e.getAge() <4;}

Schreibe einen Predicate<String>, der true zurückgibt, wenn ein String mit dem Buchstaben "A" beginnt.

Das könnte folgendermaßen aussehen:

Predicate < String > beginsWithA = LAMBDAAUSDRUCK;

Teste deine Lösung.



(a, b) -> a.startsWith("Hello")



Aufgabe

(String a, String b) -> a.startsWith("test")

Schreibe einen BiPredicate<String, Integer>, der true zurückgibt, wenn die Länge des Strings gleich der angegebenen Zahl ist.

Das könnte folgendermaßen aussehen:

BiPredicate<String, Integer> lengthMatches = LAMBDAAUSDRUCK;

Teste deine Lösung.





Plan für die Woche

Dienstag

Montag

Wrapper-Klassen

- ArrayLists

Predicate

 ArrayLists und Lambdas

Wiederholung

Lambdas und

 Block an Quizfragen (20)

Mittwoch

- Java class structure
- Command line
- Selbständiges Vorbereiten zur Prüfung anhand Lernplans

Donnerstag

- Statische Variablen und Methoden
- Selbständiges Vorbereiten zur Prüfung anhand Lernplans

Freitag

- Features vergleichen
- Block an Quizfragen (20)



Codevorführung für Aufgabe



Erzeuge ein Java-Programm:

- 1. Erstelle eine statische Methode boolean checkForVoteRight(int age, Predicate<Integer> predicate)
 - Die Methode soll das Ergebnis des Tests zurückgeben. Nutze dafür die test() Methode

return predicate.test(age);

- In der Main-Methode:
 - Schreibe eine Benutzereingabe, die das Alter abfragt
 - Rufe die Methode checkForVoteRight auf und überprüfe, ob der Benutzer über 16 ist. Speichere das Ergebnis in einer Variablen.
 - Wenn ja, dann hat er das Recht wählen zu gehen. Wenn nicht, dann muss er leider zu Hause bleiben. Mache Konsolenausgaben.



List<String> worte = Arrays.asList("Hallo", "Java", "Algorithmus", "Entwicklung", "Objekt", "Union", "Code", "Umbau");



Aufgabe

- 1. Erzeuge eine Methode filterWords(List<String> words, Predicate<String> predicate), die eine Liste an gefilterten Worten zurückgibt.
 - 1. Es soll eine neue ArrayList erzeugt werden, wo die gefilterten Worte abgespeichert werden sollen
 - 2. Eine Schleife soll über die übergebene Liste gehen
 - 3. Sobald ein String den Test besteht, landet es in der neuen ArrayList
- 2. Teste deine Methode mit folgenden Beispielen:

welche Wörter der Liste sind mindestens 5 Zeichen lang?

welche Wörter beginnen mit dem Buchstaben J?

welche Wörter beginnen mit einem Vokal?

welche Wörter sind mindestens 6 Zeichen lang und beginnen mit einem Vokal?

Nutze hierfür Lambdas, Predicate und die dazugehörige test()-Methode.





- 1. Erstelle ein funktionales Interface, welches eine abstrakte Methode transform(String s) hat, die einen String zurückgibt
- 2. Erstelle eine statische Methode transformStrings, die als Parameter eine Liste an Strings und eine Instanz des Interfaces aus 1) entgegennimmt und die transformierten Strings in einer Liste zurückgibt.
- 3. Nun soll eine Liste an Strings jeweils wie folgt transformiert und danach ausgegeben werden:
 - 1. Originalstrings soll ausgegeben werden
 - 2. Der String soll in Großbuchstaben ausgegeben werden
 - 3. Alle Vokale sollen mit einem Y ersetzt werden



SMART INDUSTRY CAMPUS

ArrayLists

BUCHSEITEN S.129-138



Grundlagen

- ist im Package java.util.* enthalten
- Syntax:

```
Alle möglichen Objekte
```

ArrayList a = new ArrayList();
ArrayList<String> a1 = new ArrayList<>();
ArrayList<Integer> a2 = new ArrayList<>(20);
ArrayList<Integer> a3 = new ArrayList<>(a2);

Kapazität von 20

Wrapper-Klassen



- boolean add(E element)
 - Fügt ein Element hinzu
- add(int index, E element)
 - Fügt ein Element an Index index hinzu

```
ArrayList<String> a1 = new ArrayList<>();
a1.add("Hallo");
a1.add(0, "Tschüss");
a1.add(3, "Oh oh");
```



- boolean remove(Object object)
 - Löscht das Objekt
- E remove(int index)
 - löscht das Element an Index index

```
a1.add("Hallo");
a1.add(0, "Tschüss");
a1.remove(0);
a1.remove(0);
a1.remove("Hallo");
```

```
a1.add("Hallo");
a1.add(0, "Tschüss");
a1.remove(0);
a1.remove("Hallo");
```

```
a1.add("Hallo");
a1.add(0, "Tschüss");
a1.remove(0);
a1.remove(0);
a1.remove(0);
a1.remove("Hallo");
```



- E set(int index, E newElement)
 - Ersetzt das Element an Index index

```
a1.add("Hallo");
a1.add(0, "Tschüss");
a1.set(0, "ersetze Tschüss");
a1.set(1, "ersetze Hallo");
```

[ersetze Tschüss, ersetze Hallo]



- boolean isEmpty()
 - Überprüft, ob die ArrayList leer ist
- int size()
 - Gibt die Größe der ArrayList zurück
 - Wichtig für Schleifen



- clear()
 - Löscht alle Elemente in einer ArrayList
- boolean contains(Object object)
 - Überprüft, ob ein Objekt object in der ArrayList enthalten ist



- boolean equals(Object o)
 - Damit kann man die Elemente zweier ArrayLists auf Gleichheit überprüfen

```
ArrayList<Integer> a2 = new ArrayList<>();
ArrayList<Integer> a2_2 = new ArrayList<>();
ArrayList<Integer> a3 = new ArrayList<>(a2);

System.out.println(a2.equals(a2_2));

a2.add(12);
System.out.println(a2.equals(a3));
```



Konvertieren von Array in List

– Arrays.asList(array)

```
String[] s = {"Hallo", "mein", "Name", "ist"};
List<String> arrToL = Arrays.asList(s);
```



In die andere Richtung mittels list.toArray();



Sorting ArrayList

Collections.sort(List list)

```
String[] s = {"Hallo", "mein", "Name", "ist"};
List<String> arrToArrL = Arrays.asList(s);
Collections.sort(arrToArrL);
System.out.println(arrToArrL);
```



ArrayLists und Lambdas

BUCHSEITEN S.215



- es gibt ein paar Methoden in der ArrayList-Klasse, denen man Lambdas übergeben kann
- removelf(Predicate<ElementVonArrayList> predicate)

```
List<String> aL = new ArrayList<>();
aL.add("Hallo");
aL.add("mein");
aL.add("Name");
aL.add("ist");
aL.removelf(str -> str.charAt(0) != 'm');
System.out.println(aL);
```

[mein]



- es gibt ein paar Methoden in der ArrayList-Klasse, denen man Lambdas übergeben kann
- forEach(Consumer<ElementVonArrayList> consumer)

```
List<String> aL = new ArrayList<>();
aL.add("Hallo");
aL.add("mein");
aL.add("Name");
aL.add("ist");
aL.forEach(str -> str.concat("a"));
System.out.println(aL);
```

[Hallo, mein, Name, ist]

```
for(String str: aL){
    str.concat("a");
}
```



- es gibt ein paar Methoden in der ArrayList-Klasse, denen man Lambdas übergeben kann
- replaceAll(UnaryOperator<ElementVonArrayList> operator)

```
List<String> aL = new ArrayList<>();
aL.add("Hallo");
aL.add("mein");
aL.add("Name");
aL.add("ist");
aL.replaceAll(str -> str.concat("Oh"));
System.out.println(aL);
```

[HalloOh, meinOh, NameOh, istOh]

```
for(int i = 0; i<aL.size(); i++){
   aL.set(i, aL.get(i).concat("Oh"));
}
System.out.println(aL);</pre>
```



Gegeben ist eine Liste von Zahlen. Entferne alle Zahlen, die kleiner als 10 sind, mit der Methode removelf.

- Schritt 1: Erstelle eine ArrayList<Integer> mit den Werten: [5, 12, 8, 20, 3, 15, 10].
- Schritt 2: Verwende removelf, um alle Zahlen kleiner als 10 zu entfernen.
- Schritt 3: Gib die bereinigte Liste aus.

Erwartete Ausgabe: [12, 20, 15, 10]





Eine Liste enthält Namen, die in Kleinbuchstaben geschrieben sind. Verwandle alle Namen in Großbuchstaben mithilfe der Methode replaceAll.

- Schritt 1: Erstelle eine ArrayList<String> mit den Werten: ["anna", "bob", "charlie", "david"].
- Schritt 2: Verwende replaceAll, um alle Namen in Großbuchstaben umzuwandeln.
- Schritt 3: Gib die transformierte Liste aus.

Erwartete Ausgabe: ["ANNA", "BOB", "CHARLIE", "DAVID"]





Gegeben ist eine Liste mit Städtenamen. Gib jeden Städtenamen aus, indem du forEach nutzt.

- Schritt 1: Erstelle eine List<String> mit den Werten: ["Berlin", "München", "Hamburg", "Stuttgart"].
- Schritt 2: Verwende forEach, um jeden Städtenamen auszugeben.

