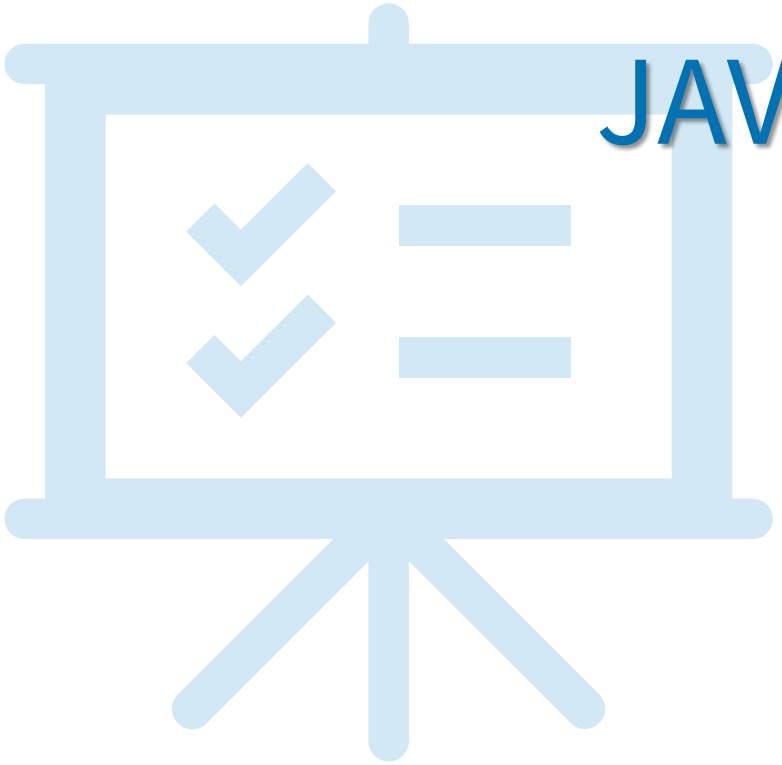
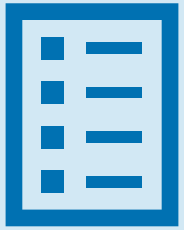


JAVA Programmierung

ECLIPSE & JAVA 8

Themenübersicht



01 List

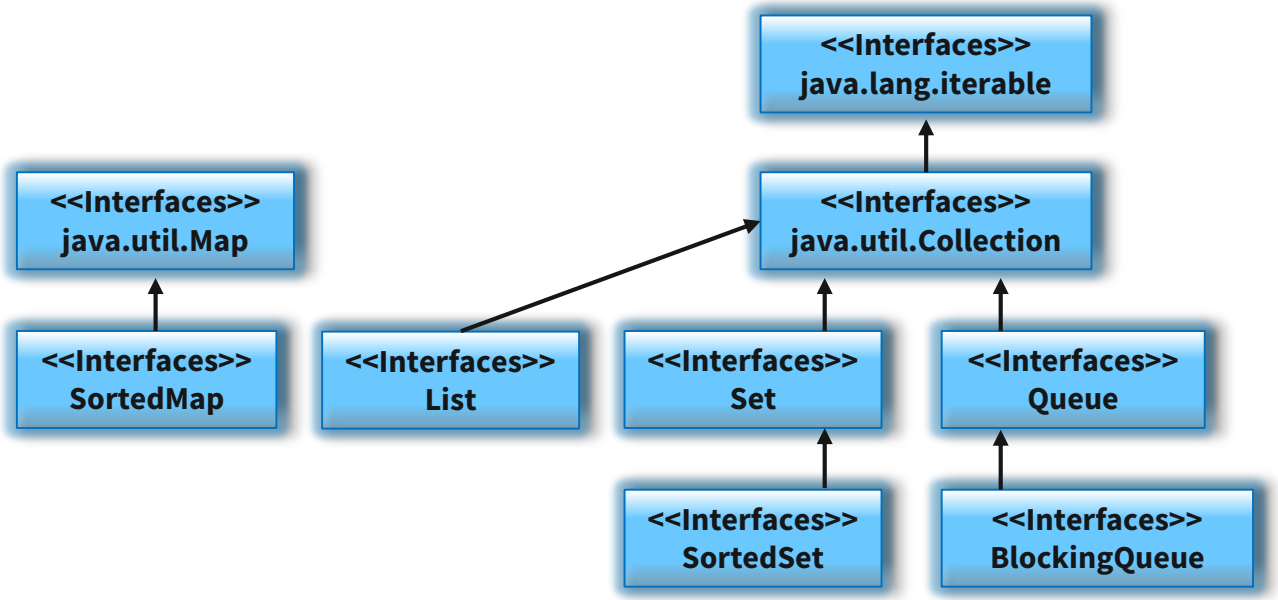
1. Generelles
2. ArrayList
   - Generelles
   - Aufgabe & Lösung
3. LinkedList
   - Generelles
   - Aufgabe & Lösung
4. Vector
   - Generelles
   - Aufgabe & Lösung

Überblick Collections

Überblick Collections

- **Generelles**

Für einen generellen Überblick über alle Mitglieder der Collection Familie schauen wir uns die folgende Hierarchie von Schnittstellen (Interfaces*) an, welche die unterschiedlichen Mitglieder implementieren.



*Schnittstelle (Interface): Definiert funktionale Eigenschaft für die implementierenden Klassen.

Überblick Collections

- **Generelles**

Überblick über Eigenschaften der reellen Klassen welche Familienmitglieder von Collections implementiert.

Den Stack werden wir zu einem späteren Zeitpunkt behandeln.

Klasse	Famlie (implem. Schnittst.)	Zugriff	Duplikate erlaubt	Ordnung
ArrayList	Liste(List)	wahlfrei über Index	Ja	geordnet
LinkedList	Liste(List)	wahlfrei über Index	Ja	geordnet
Vector	Liste(List)	wahlfrei über Index	Ja	geordnet
Stack	Liste(List)	sequentiell (letztes Element)	Ja	geordnet

⇒ https://docs.oracle.com/javase/7/docs/api/java/util/Collections.html

01 List

1. *ArrayList*
2. *LinkedList*
3. *Vector*

1.1 List

Generelles

Überblick Listen

- **Generelles**

Java stellt dem Anwender verschiedene Arten von Listen zur Verfügung. Sie können zwischen den folgenden Listen wählen. Welche Liste Sie benutzen hängt vom konkreten Anwendungsfall ab.

- java.util.ArrayList
- java.util.LinkedList
- java.util.Vector
- java.util.Stack (Sonderfall, den wir derzeit nur zur Kenntnis nehmen)

Die ArrayList ist die am häufigsten verwendete Liste. Alle Listen sind eine geordnete Menge von Objekten, auf die entweder sequentiell oder über ihren Index (ihre Position in der Liste) zugegriffen werden kann. Listen sind wie Arrays Index 0 basiert. Es ist möglich, an einer beliebigen Stelle der Liste ein Element einzufügen oder zu löschen. Die weiter hinten stehenden Elemente werden dann dementsprechend nach rechts bzw. links verschoben.

⇒ https://docs.oracle.com/javase/8/docs/api/java/util/List.html

1.2 List

ArrayList

1.2 ArrayList

- **Generelles**

Die ArrayList ist die am häufigsten verwendete Datenstruktur zum Erstellen eines Arrays mit dynamischer Größe.

Der Hauptunterschied zwischen Array und ArrayList besteht darin, dass das Array eine feste Größe hat, während die ArrayList dynamisch ist (wir können Elemente hinzufügen, entfernen oder ändern). Einige Methoden der ArrayList:

Modifikator und Typ	Methode	Beschreibung
boolean	add(E e)	Hängt das angegebene Element an das Ende der Liste an.
boolean	contains(Object o)	Gibt true zurück, wenn diese Liste das angegebene Element enthält.
Element	get(int index)	Gibt das Element an der angegebenen Position in dieser Liste zurück.
Element	remove(int index)	Entfernt das Element an der angegebenen Position

⇒ https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html

1.2 ArrayList

- **Beispiel**

```
public static void main(String[] args) {
    ArrayList<String> Farben = new ArrayList<String>();
    Farben.add("Rot");
    Farben.add("Blau");
    Farben.add("Grün");

    ArrayList<String> Farben2 = new ArrayList<String>(Arrays.asList("Rot", "Blau", "Grün"));
    System.out.println(Farben);
    System.out.println(Farben2);
}
```

Output:

```
<terminated> ErsteKlasse [Java Applicat
[Rot, Blau, Grün]
[Rot, Blau, Grün]
```

1.2 Aufgaben ArrayList

1. *Aufgabe*
2. *Lösung*

1.2 Aufgabe ArrayList

Aufgabe

Aufgabe ArrayList

Schreiben Sie ein Programm welches eine ArrayList implementiert und deren Daten ausgibt.

1.2 Aufgabe ArrayList

Lösung

Lösung ArrayList

```
8
9      public static void main(String[] args) {
10          ArrayList<String> cars = new ArrayList<String>();
11             cars.add("Volvo");
12             cars.add("BMW");
13             cars.add("Ford");
14             cars.add("Mazda");
15             System.out.println(cars);
16     }
17 }
```

```
<terminated> test [Java Application] C:\Program Fi
[Volvo, BMW, Ford, Mazda]
```

1.3 List

LinkedList

1.3 LinkedList

• Generelles

Die wichtigste Liste für den Anfang haben wir schon kennengelernt. Es gibt aber noch weiter Listen, welche wir uns auch anschauen wollen. Eine dieser Listen ist die Verkettete Liste (LinkedList). Bei diesem Typ steht das jeweilige Element in Verbindung zum Vor- bzw. Nachfolger.

Diese Liste unterscheidet sich in zwei wesentlichen Punkten von der Arraylist:

- Elemente können schneller hinzugefügt und gelöscht werden
- Das Auslesen von Elementen an bestimmten Positionen benötigt dafür mehr Zeit

⇒ https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html

1.3 LinkedList

• Generelles

Einige Methoden der ArrayList:

Modifikator und Typ	Methode	Beschreibung
boolean	add(E e)	Hängt das angegebene Element an das Ende der Liste an.
boolean	contains(Object o)	Gibt true zurück, wenn diese Liste das angegebene Element enthält.
Element	element()	Ruft den Kopf (erstes Element) dieser Liste ab, entfernt ihn aber nicht.
Element	pop()	Entnimmt ein Element aus dem Stapel, der durch diese Liste repräsentiert wird.

1.3 LinkedList

- **Beispiel**

```
LinkedList<String> list=new LinkedList<String>();

 list.add("Egon");
 list.add("Hans");
 list.add("Walter");

 list.addFirst("Flo");
 list.addLast("Rick");

 //Iterator für LinkedList => hierzu später mehr
 // wird benötigt um über die Liste zu iterieren
 Iterator<String> iterator=list.iterator();
 while(iterator.hasNext()){
   System.out.println(iterator.next());
 }
```

Output:

```
<terminated> First(
Flo
Egon
Hans
Walter
Rick
```

1.3 Aufgaben LinkedList

1. *Aufgabe*
2. *Lösung*

1.3 Aufgabe LinkedList

Aufgabe

Aufgabe LinkedList

Schreiben Sie ein Programm welches diese Art der Liste implementiert und die Daten ausgibt.

1.3 Aufgabe LinkedList

Lösung

Lösung LinkedList

```
LinkedList<Integer> list=new LinkedList<Integer>();

 list.add(1);
 list.add(3);
 list.add(4);

 list.addFirst(5);
 list.addLast(7);
 list.add(2,9);

 //Iterator für LinkedList => hierzu später mehr
 // wird benötigt um über die Liste zu iterieren
 Iterator<Integer> iterator=list.iterator();
 while(iterator.hasNext()){
   System.out.println(iterator.next());
 }
```

```
<terminated> F
5
1
9
3
4
7
```

1.4 List

Vector

1.4 Vector

• Generelles

Der Vector hat Ähnlichkeit mit dem Array, aber mit zwei Besonderheiten. Zum Einen ist die Größe dynamisch . Zum Anderen kann der Vector unterschiedliche Datentypen entgegen nehmen.

Einige Methoden des Vector:

Modifikator und Typ	Methode	Beschreibung
void	add (int index, Objekt e)	Fügt das Element dem angegebenen Index hinzu. Wenn der Index bereits ein Element enthält, wird es nach rechts verschoben
void	clear ()	Entfernt alle Elemente aus dem Vektor und wird leer
Element	elementAt (int index)	Gibt das Objekt am angegebenen Index zurück
Element	firstElement ()	Gibt die erste Komponente bei Index 0 zurück

⇒ https://docs.oracle.com/javase/8/docs/api/java/util/Vector.html

1.4 Vector

- **Beispiel**

```
    Vector vector = new Vector(4, 2);

    vector.add("Java");
    vector.add("C");
    vector.add("C++");
    vector.add(100);
    vector.add("Python");
    vector.add("Ruby");
    vector.add(1);
    System.out.println(vector);
```

Output:

```
<terminated> FirstClass [Java Application] C:\Program
[Java, C, C++, 100, Python, Ruby, 1]
```

1.4 Aufgabe Vector

1. *Aufgabe*
2. *Lösung*

1.4 Aufgabe Vector

Aufgabe

Aufgabe Vector

Schreiben Sie ein Programm, das einen Vector implementiert und die Daten ausgibt.

1.4 Aufgabe Vector

Lösung

Lösung Vector

```
12    public static void main(String[] args) {
13        Vector<String> mammals= new Vector<>();
14
15        mammals.add("Dog");
16        mammals.add("Horse");
17
18        mammals.add(2, "Cat");
19        System.out.println("Vector: " + mammals);
20
21        Vector<String> animals = new Vector<>();
22        animals.add("Crocodile");
23
24        animals.addAll(mammals);
25        System.out.println("New Vector: " + animals);
26    }
```

```
<terminated> test [Java Application] C:\Program Files\Java
Vector: [Dog, Horse, Cat]
New Vector: [Crocodile, Dog, Horse, Cat]
```

VIELEN DANK!