hhu.

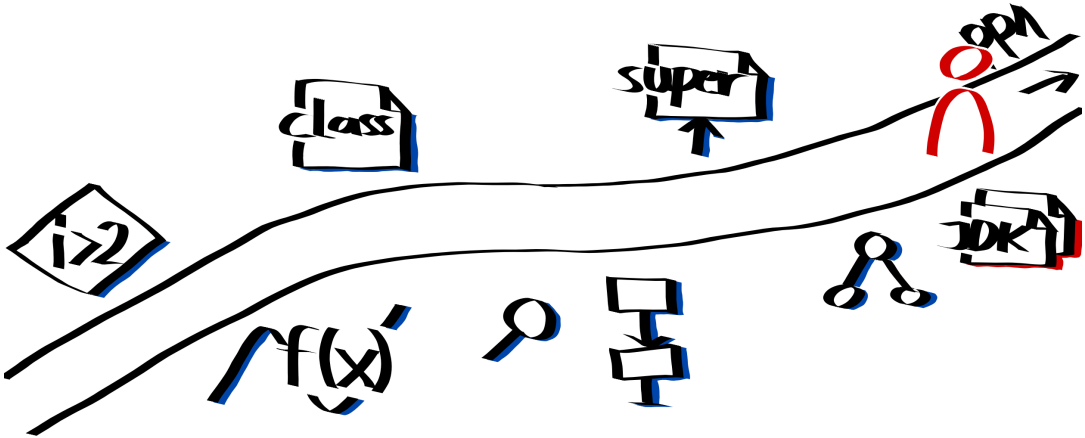
Kapitel 8: Packages, Frameworks & Co.

VL 27: Standard Collections Framework & IO

Wintersemester 2023/24

Stand 14. Dezember 2023, 10:09 Uhr

Wo stehen wir gerade?



Vorlesungsfeedback

Fertige Listen im Package `java.util`

Heinrich Heine Universität Düsseldorf

- `LinkedList<E>`[1]: (doppelt) verkettete Liste
  - ⇒ schnelles Einfügen
  - ⇒ langsamer Index-basierter Zugriff
- `ArrayList<E>`[2]: benutzt intern Array
  - ⇒ schneller Index-basierter Zugriff
  - ⇒ Einfügen in der Mitte langsam
- weitere, spezielle Listen, z. B. `Vector`, `CopyOnWriteArrayList`
- implementieren Interface `List<E>`[3]

[1] https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/LinkedList.html
[2] https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/ArrayList.html
[3] https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/List.html

Verwendungsbeispiel ArrayList

```java
import java.util.List;
import java.util.ArrayList;

public class Lists {
    public static void main(String[] args) {
        List<String> names = new ArrayList<>();
        names.add("Alex");
        names.add("Kim");
        names.add("Sascha");
        names.add("Max");
        names.remove("Kim");

        System.out.println(names.size());
        System.out.println(names.get(0));
        System.out.println(names);
        for(String name: names) {
            System.out.print(name);
        }
    }
}
```

Verwendungsbeispiel ArrayList

Heinrich Heine Universität Düsseldorf

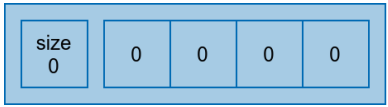
```java
import java.util.List;
import java.util.ArrayList;

public class Lists {
    public static void main(String[] args) {
        List<String> names = new ArrayList<>();
        names.add("Alex");
        names.add("Kim");
        names.add("Sascha");
        names.add("Max");
        names.remove("Kim");

        System.out.println(names.size());
        System.out.println(names.get(0));
        System.out.println(names);
        for(String name: names) {
            System.out.print(name);
        }
    }
}
```

```
3
Alex
[Alex, Sascha, Max]
Alex
Sascha
Max
```

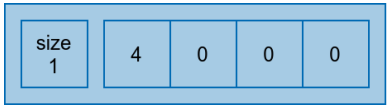
Interne Funktionsweise der ArrayList



(vom Prinzip her, Darstellung vereinfacht)

```
1 ArrayList<Integer> zahlen = new ArrayList<>();
```

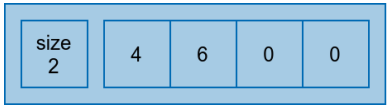
Interne Funktionsweise der ArrayList



(vom Prinzip her, Darstellung vereinfacht)

```
1 ArrayList<Integer> zahlen = new ArrayList<>();
2 zahlen.add(4);
```

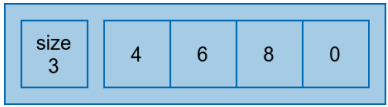
Interne Funktionsweise der ArrayList



(vom Prinzip her, Darstellung vereinfacht)

```
1 ArrayList<Integer> zahlen = new ArrayList<>();
2 zahlen.add(4);
3 zahlen.add(6);
```

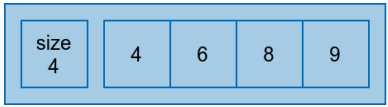
Interne Funktionsweise der ArrayList



(vom Prinzip her, Darstellung vereinfacht)

```
1 ArrayList<Integer> zahlen = new ArrayList<>();
2 zahlen.add(4);
3 zahlen.add(6);
4 zahlen.add(8);
```

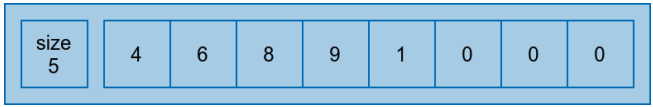
Interne Funktionsweise der ArrayList



(vom Prinzip her, Darstellung vereinfacht)

```java
ArrayList<Integer> zahlen = new ArrayList<>();
zahlen.add(4);
zahlen.add(6);
zahlen.add(8);
zahlen.add(9);
```

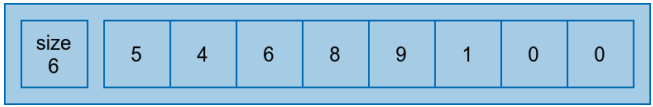
Interne Funktionsweise der ArrayList



(vom Prinzip her, Darstellung vereinfacht)

```
ArrayList<Integer> zahlen = new ArrayList<>();
zahlen.add(4);
zahlen.add(6);
zahlen.add(8);
zahlen.add(9);
zahlen.add(1);
```

Interne Funktionsweise der ArrayList



(vom Prinzip her, Darstellung vereinfacht)

```
1 ArrayList<Integer> zahlen = new ArrayList<>();
2 zahlen.add(4);
3 zahlen.add(6);
4 zahlen.add(8);
5 zahlen.add(9);
6 zahlen.add(1);
7 zahlen.add(0, 5); // an Index 0 einfügen
```

Sets/Maps in `java.util`

- `HashMap<K,V>`[4]: garantiert Einfügen/Suchen in konstanter Zeit
- `HashSet<E>`[5]: verwendet `HashMap` im Hintergrund
- `TreeSet<E>`[6]: verwendet Rot-Schwarz-Baum
- weitere, spezielle Interfaces/Klassen, z. B. `CopyOnWriteArraySet`
- Gemeinsame Oberklasse: `AbstractSet<E>` bzw. `AbstractMap<K,V>`
- implementieren Interface `Set<E>` bzw. `Map<K,V>`

[4] https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/HashMap.html
[5] https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/HashSet.html
[6] https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/TreeSet.html

Verwendungsbeispiel HashMap

Heinrich Heine Universität Düsseldorf

```java
import java.util.Map;
import java.util.HashMap;

public class Maps {
    public static void main(String[] args) {
        Map<String, Integer> people = new HashMap<>();
        people.put("Alex", 22);
        people.put("Kim", 19);
        people.put("Sascha", 21);
        people.put("Max", 22);
        people.remove("Kim");

        System.out.println(people.size());
        System.out.println(people.get("Alex"));
        System.out.println(people);
        for(String name: people.keySet()) {
            System.out.println(name);
        }
    }
}
```

Verwendungsbeispiel HashMap

Heinrich Heine Universität Düsseldorf

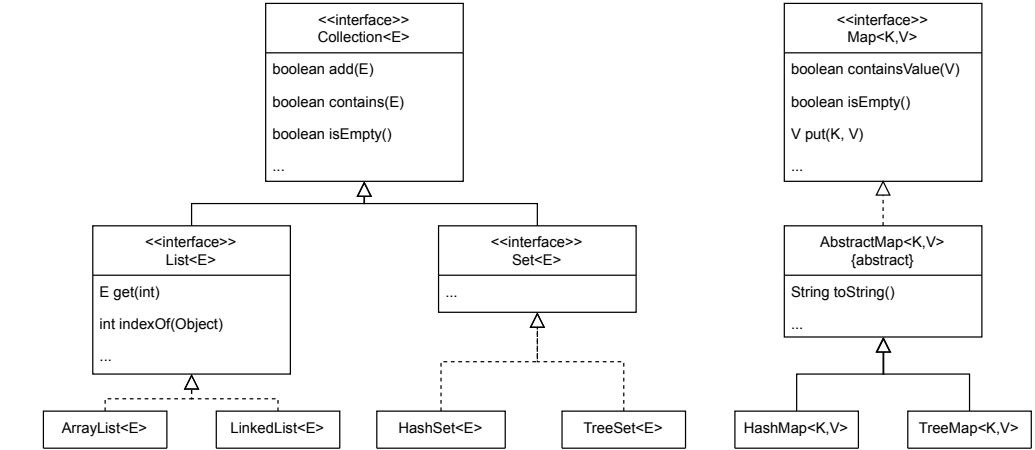
```java
import java.util.Map;
import java.util.HashMap;

public class Maps {
    public static void main(String[] args) {
        Map<String, Integer> people = new HashMap<>();
        people.put("Alex", 22);
        people.put("Kim", 19);
        people.put("Sascha", 21);
        people.put("Max", 22);
        people.remove("Kim");

        System.out.println(people.size());
        System.out.println(people.get("Alex"));
        System.out.println(people);
        for(String name: people.keySet()) {
            System.out.println(name);
        }
    }
}
```

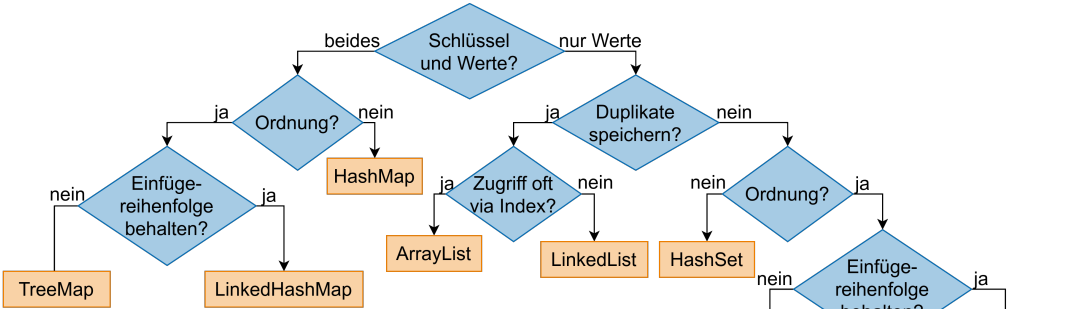
```
3
22
{Alex=22, Max=22, Sascha=21}
Alex
Max
Sascha
```

Datentypen im Java Collections Framework[7]



[7] Auswahl, vollständig unter
https://docs.oracle.com/javase/8/docs/technotes/guides/collections/reference.html

Wann welche Collection?



basierend auf https://lec.inf.ethz.ch/baug/informatik1/2019/slides/infLecture12.handout.2x2.pdf

Weitere Bestandteile des Collections Frameworks

Heinrich Heine
Universität Düsseldorf

- Algorithmen
  - sort, shuffle, min, . . .
- Interfaces
  - Comparable, Iterable, . . .
- Hilfsmethoden
  - asList, toArray, . . .
- Exceptions
  - NoSuchElementException, . . .
- . . .

Weitere Bestandteile des Collections Frameworks

- Algorithmen
  - sort, shuffle, min, . . .
- Interfaces
  - Comparable, Iterable, . . .
- Hilfsmethoden
  - asList, toArray, . . .
- Exceptions
  - NoSuchElementException, . . .
- . . .

```java
import java.util.Arrays;
import java.util.Collections;
import java.util.ArrayList;
import java.util.List;

public class Sort {
  public static void main(String[] args) {
    List<String> arguments = Arrays.asList(args);
    Collections.sort(arguments);
    System.out.println(arguments);
  }
}
```

```
% java Sort Rookie Champion Ultra
[Champion, Rookie, Ultra]
```

Muss man sich das jetzt alles merken?

Heinrich Heine
Universität Düsseldorf

Wo nachschauen, um zu sehen, was in JDK enthalten und was die Klassen können?

Muss man sich das jetzt alles merken?

Heinrich Heine Universität Düsseldorf

Wo nachschauen, um zu sehen, was in JDK enthalten und was die Klassen können?
Dokumentation: `https://docs.oracle.com/en/java/javase/11/docs/api/java.base/module-summary.html`

Packages

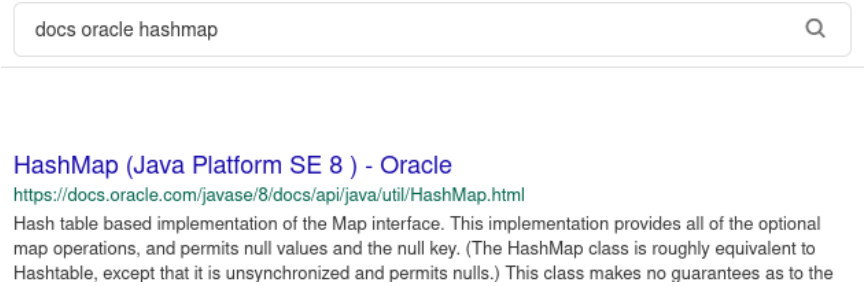
Exports

Package	Description
java.io	Provides for system input and output through data streams, serialization and the file system.
java.lang	Provides classes that are fundamental to the design of the Java programming language.
java.lang.annotation	Provides library support for the Java programming language annotation facility.
java.lang.invoke	The `java.lang.invoke` package provides low-level primitives for interacting with the Java Virtual Machine.
java.lang.module	Classes to support module descriptors and creating configurations of modules by means of resolution and service binding.
java.lang.ref	Provides reference-object classes, which support a limited degree of interaction with the garbage collector.
java.lang.reflect	Provides classes and interfaces for obtaining reflective information about classes and objects.
java.math	Provides classes for performing arbitrary-precision integer arithmetic (`BigInteger`) and arbitrary-precision decimal arithmetic (`BigDecimal`).
java.net	Provides the classes for implementing networking applications.
java.net.spi	Service-provider classes for the `java.net` package.
java.nio	Defines buffers, which are containers for data, and provides an overview of the other NIO packages.

Muss man sich das jetzt alles merken?

Heinrich Heine Universität Düsseldorf

Wo nachschauen, um zu sehen, was in JDK enthalten und was die Klassen können?



Collections Framework oder eigene Implementierung?

Collections Framework oder eigene Implementierung?

Heinrich Heine Universität Düsseldorf

Praxis: fertige Implementierungen vorziehen

- Zeitersparnis
- intensiv getestet $\Rightarrow$ weniger Fehler
- kompatibel mit existierenden Bibliotheken

Aber auch hier gilt:

- Vor-/Nachteile verschiedener Datenstrukturen kennen

Collections im Zusammenhang mit OOP

Heinrich Heine Universität Düsseldorf

- oft sinnvoll, Collections in eigener Klasse zu kapseln
  - ⇒ Operationen klar Datentyp zugeordnet
  - ⇒ Typfehler fallen zur Compilezeit auf

```
1  import java.util.List;
2  import java.util.ArrayList;
3
4  public abstract class Layout extends Widget {
5      private List<Widget> widgets = new ArrayList<>();
6
7      public void add(Widget widget) {
8          widgets.add(widget);
9          if(!isEnabled()) {
10             widget.setEnabled(false);
11         }
12     }
13
14     public int numberOfWidgets() {
15         return widgets.size();
16     }
```

Einlesen von Dateien

Heinrich Heine
Universität Düsseldorf

- bisher: Verwendung von Standardeingabe
- Einschränkungen:
  - Dateiname muss bei Programmaufruf bekannt sein
  - Datei muss bei Programmaufruf existieren
  - nur eine Datei verarbeitbar
- Lösung: Einlesen von Dateien zur Laufzeit
  → Klassen in Package java.nio.file

Beispiel: Datei einlesen und ausgeben

Heinrich Heine Universität Düsseldorf

```java
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.List;

public class Cat {
    public static void main(String[] args) {
        List<String> lines;
        Path filepath = Path.of(args[0]);

        try {
            lines = Files.readAllLines(filepath);
        } catch (IOException e) {
            System.out.println("Fehler beim Lesen der Datei " + filepath);
            System.out.println(e.getMessage());
            return;
        }

        for(String line: lines) {
            System.out.println(line);
        }
    }
}
```

Beispiel: Datei einlesen und ausgeben

Heinrich Heine
Universität Düsseldorf

```
% ls
Cat.class Cat.java isbns
% java Cat isbns
978-3-86425-065-1
978-3-86425-068-2
978-3-86425-152-8
```

Anwendungsbeispiel: Klassifikation von Textbeiträgen

Heinrich Heine Universität Düsseldorf

Eingabe

Die Ampelschaltung an der Kreuzung zur Universitätsstraße/Christopfstraße in Fahrtrichtung Süden ist ungünstig für Radfahrer. Die Rotphase ist viel zu lang.

Welche Kategorie passt am besten?

- Ampeln
- Beleuchtung
- Beschilderung
- Fahrradparken
- Hindernisse
- Radverkehrsführung
- Radwegqualität
- Sonstiges

Lösung: Machine Learning

Heinrich Heine Universität Düsseldorf

1. Lernprozess:
   - Eingabe: Datensatz mit Trainingsdaten (Zuordnung Text$\rightarrow$ Kategorie)
   - Pro Kategorie speichern, wie oft welche Wörter in entsprechenden Texten vorkommen
2. Klassifizierung:
   - Eingabe: ein Satz
   - Pro Kategorie:
     - schlage nach, wie häufig Wörter des Satzes in Kategorie vorkommen
     - multipliziere Häufigkeiten zusammen
   - Ausgabe: Kategorie mit dem höchsten Wert

Beispiel (verkürzt)

Heinrich Heine Universität Düsseldorf

Eingabe

Die Ampelschaltung . . . Kreuzung . . . Rotphase . . .

Gelernte Häufigkeiten:

- Ampeln: [Die: 10, Ampelschaltung: 5, Kreuzung: 3, Rotphase: 4]
- Fahrradparken: [Die: 11, Ampelschaltung: 1, Kreuzung: 2, Rotphase: 1]

Scores pro Kategorie:

- Ampeln: $10 \cdot 5 \cdot 3 \cdot 4 = 600$
- Fahrradparken: $11 \cdot 1 \cdot 2 \cdot 1 = 22$

Das war etwas vereinfacht . . .

Heinrich Heine
Universität Düsseldorf

Wer mehr wissen möchte:

- Vorlesungen:
  - Data Science
  - Machine Learning
  - Natural Language Processing
- Stichworte für eigene Recherche:
  - maximum likelihood estimator
  - training, validation, test set
  - accuracy, precision, recall
  - Laplace smoothing, stop words
  - tokenizer, stemmer, lemmatizer
  - imbalanced data sets

Lernziele

Heinrich Heine
Universität Düsseldorf

Sie können am Ende der Woche . . .

- Informationen über im JDK enthaltene Klassen **finden**.
- **begründen**, warum im JDK enthaltene Datenstrukturen i. d. R. vorzuziehen sind.

Wichtige Begriffe

Heinrich Heine
Universität Düsseldorf

ArrayList Collections Framework Dokumentation