Java OOP

Konrad Raue, Oliver Scholz

19. November 2019

Gliederung

1. Nachtrag
2. einfache Klassen
3. Vererbung
4. Generics
5. Listen, Sets, Maps
6. Nächste Woche

Nachtrag

einfache Klassen

Klassen

```java
public class Klasse{
    public static void main(String[] args){
        Zahl zahl = new Zahl(4);
        System.out.println(zahl.getValue());
        zahl.setValue(7);
        System.out.println(zahl.getValue());
    }
}
```

```java
public class Zahl{
    private int value;

    public Zahl(int value){
        this.value = value;
    }

    public int getValue(){
        return value;
    }

    public void setValue(int value){
        this.value = value;
    }
}
```

Vererbung

extends

```java
public class Person{
    private String name;

    public Person(String name){
        this.name = name;
    }

    public void work(){
        //do something
    }
}
```

```java
public class Student extends Person{
    private String studiengang;
    private int fachsemester;

    public Student(String name, String studiengang, int fachsemester
    ){
        super(name);
        this.studiengang = studiengang;
        this.fachsemester = fachsemester;
    }

    @Override
    public void work(){
        //do something else
    }
}
```

Generics

Generics

- Klasse kann auf verschiedene Klassen angewandt werden
- Generic<Klasse>
- in Generic wird Klasse als einzelner Großbuchstabe geschrieben

Generics

```java
public class Generics<T>{
    private T content;

    public <T> Generics(T content){
        this.content = content;
    }

    public T getContent(){
        return content;
    }
}
```

```java
public class Nutzung{
    private Generics<int> content;

    public Nutzung(Generics<int> content){
        this.content = content;
    }

    public Generics<int> getContent(){
        return content;
    }
}
```

Generics können auch Vererbung nutzen, über T extends Klasse

Listen, Sets, Maps

List

```java
public class Zahl{
    ...
}
```

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

public class Listen{
    private List<Zahl> zahlen;

    public Listen(){
        zahlen = new ArrayList<>();
    }

    public Zahl addZahl(Zahl newZahl){
        for(Zahl zahl : zahlen){
            if(zahl == newZahl){
                return newZahl;
            }
        }
        zahl.add(newZahl);
        return newZahl;
    }
}
```

wichtige Funktionen

- add(Klasse klasse)
- remove(Klasse klasse)
- get(int index)
- set(int index, Klasse klasse)
- contains(Klasse klasse)
- size()
- isEmpty()
- clear()
- weitere in JavaDoc:
  https://docs.oracle.com/javase/8/docs/api/java/util/List.html

Set

```
1 public class Zahl{
2     ...
3 }
```

```
1 import java.util.Set;
2 import java.util.HashSet;
3 import java.util.TreeSet;
4
5 public class Sets{
6     private Set<Zahl> zahlen;
7
8     ...
9 }
```

Sets sind Listen, nur dass jedes Element nur einmal vorkommen kann.

Man kann über sie iterieren.

wichtige Funktionen

- add(Klasse klasse)
- remove(Klasse klasse)
- contains(Klasse klasse)
- size()
- isEmpty()
- clear()
- weitere in JavaDoc:
  https://docs.oracle.com/javase/8/docs/api/java/util/Set.html

Map

```
1 public class Zahl{
2     ...
3 }
```

```
1 import java.util.Map;
2 import java.util.HashMap;
3 import java.util.TreeMap;
4
5 public class Maps{
6     private Map<String, Zahl> zahlen;
7
8     ...
9 }
```

Maps binden Values (hier Zahlen) an Keys (hier Sting). Jeder Key kommt nur einmal vor. Values können mehrfach vorkommen.

wichtige Funktionen

- containsKey(Key key)
- containsValue(Value value)
- (entrySet())
- keySet()
- values()
- put(Key key, Value value)
- remove(Key key)
- get(Key key)
- size()
- isEmpty()
- clear()
- weitere in JavaDoc:
  https://docs.oracle.com/javase/8/docs/api/java/util/Map.html

Nächste Woche

Nächste Woche

- abstract
- interface
- Sichtbarkeiten (private, protected, public, package-private)
- static, final
- Iterator
- Observer
- UML