

252-0027

Einführung in die Programmierung Übungen

Klassen

Henrik Pätzold

Departement Informatik


ETH Zürich

Heutiger Plan

- Änderungen an Bewertungen (GradeMe)
- Zusätzliches Material?
- Theorie
 - Felder
 - Methoden
 - Konstruktoren
 - **this**-Schlüsselwort
 - Sichtbarkeit
 - Getter- und Setter-Methoden
- Kurze Programmieraufgabe
- Kahoot

Wie definieren wir eine Klasse?


- Klassen sind „Baupläne“, die definieren, wie Daten gespeichert werden
- Wir benutzen das **class-Stichwort**, um eine Klasse zu definieren



```
1 public class Student {  
2     // Körper der Klasse  
3 }
```

Erstellung von Attributen (Fields)


- Attribute definieren den Zustand oder die Eigenschaften einer Klasse
- Ihnen kann normal ein Wert zugewiesen werden
- Ohne Zuweisung durch Konstruktor erhalten Sie bei Instanziierung sonst den Standardwert des Datentyps (**null** für Objekte)



```
1 public class Student {  
2     String name;  
3     int alter;  
4     String legi;  
5     boolean istEingeschrieben = true;  
6 }
```

Instanziierung eines Objekts auf einer Klasse

- Wenn Klassen keinen Konstruktor haben, wird ein Default-Konstruktor verwendet
- Wenn Default-Konstruktor verwendet wird, werden Instanzvariablen der Klasse auf Standardwerte gesetzt



```
1 public class Main {  
2     public static void main(String[] args) {  
3         Student student = new Student();  
4     }  
5 }
```

Zugriff auf Attribute eines Objektes


- wenn Sichtbarkeit es erlaubt, können wir auf ein Attribute zugreifen mit **nameObjekt.nameAttribut**
- wir können lesen und schreiben (Pass-By-Value & Pass-By-Reference) **(DEMO)**



```
1 public class Main {  
2     public static void main(String[] args){  
3         Student student = new Student();  
4         student.alter = 21;  
5     }  
6 }
```

Erstellung von Methoden

- Methoden sind Funktionen, die innerhalb einer Klasse definiert sind
- Methoden haben uneingeschränkten Zugriff auf die Attribute und andere Methoden der Klasse, in der sie definiert sind.




```
1 public class Student {  
2     String name;  
3     int alter;  
4     String legi;  
5     boolean istEingeschrieben = true;  
6     public void hasBirthday(){  
7         this.age += 1;  
8     }  
9 }
```

Parametrisierte Konstruktor-Methoden

- **default**-Konstruktor kann mit expliziter Beschreibung überschrieben werden
- Konstruktoren müssen **public** sein
- Methoden-Name ist gleich der Klasse
- Es gibt keinen deklarierten Rückgabetyp, ein Konstruktor gibt ein Objekt der Klasse zurück.


Parametrisierte Konstruktor-Methoden



```
1 public class Student {
2     String name;
3     int alter;
4     String legi;
5     boolean istEingeschrieben = true;
6
7     public Student(String StudentName, int StudentAlter, String StudentLegi) {
8         name = StudentName;
9         alter = StudentAlter;
10        legi = StudentLegi;
11    }
12 }
```

**Methoden (auch Konstruktoren) können
überladen werden**

Parametrisierte Konstruktor-Methoden




```
1 public class Student {
2     String name;
3     int alter;
4     String legi;
5     boolean istEingeschrieben = true;
6
7     public Student(String StudentName, int StudentAlter, String StudentLegi) {
8         name = StudentName;
9         alter = StudentAlter;
10        legi = StudentLegi;
11    }
12    public Student(String StudentName, int StudentAlter) {
13        name = StudentName;
14        alter = StudentAlter;
15    }
16 }
```

this-Schlüsselwort

- wird verwendet, um auf das aktuelle Objekt einer Klasse zu verweisen
- wir können es verwenden, um auf Attribute aus dem Objekt zuzugreifen, oder auf dem Objekt Methoden aufzurufen
(DEMO)

this-Schlüsselwort für Zuweisung von Feldern



```
1 public class Student {  
2     String name;  
3     int alter;  
4     String legi;  
5     boolean istEingeschrieben = true;  
6  
7     public Student(String name, int alter, String legi) {  
8         this.name = name;  
9         this.alter = alter;  
10        this.legi = legi;  
11    }  
12 }
```

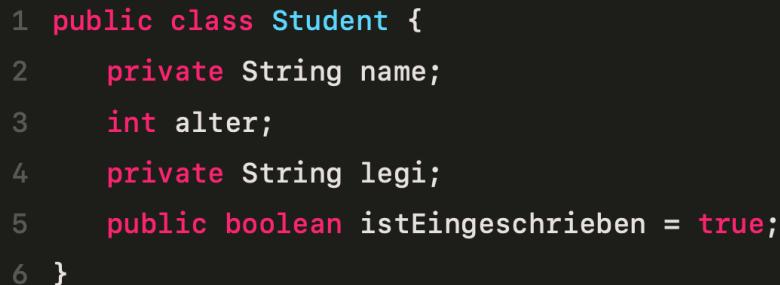
this-Schlüsselwort für Aufrufen von Methoden



```
1 public class Student {  
2     String name;  
3     int alter;  
4     String legi;  
5     boolean istEingeschrieben = true;  
6  
7     public Student(String name, int alter, String legi) {  
8         this.Student(String name, int alter);  
9         this.legi = legi;  
10    }  
11    public Student(String name, int alter){  
12        this.name = name;  
13        this.alter = alter;  
14    }  
15 }
```

Sichtbarkeit von Methoden und Feldern

- nützliches Konzept, wo bestimmte Informationen unzugänglich gemacht werden, um korrektes Verhalten zu garantieren
- **private** deklarierte Strukturen sind nur innerhalb der Klasse sichtbar
- auf **public** deklarierte Strukturen kann von jeder anderen Klasse aus zugegriffen werden
- Ausnahmefälle: **default** und **protected**



```
1 public class Student {  
2     private String name;  
3     int alter;  
4     private String legi;  
5     public boolean istEingeschrieben = true;  
6 }
```

Getter- und Setter-Methoden

- wir nutzen die Sichtbarkeit zum „Kapseln“ Feldern
- **getter**-Methoden geben Werte zurück => verhindern, dass wir sie überschreiben können
- **setter**-Methoden sind eine Sicherheitsbarriere, die korrektes überschreiben garantieren können (**DEMO**)

```
1 public class Student {  
2     private int alter;  
3     public int alter(){  
4         return this.alter;  
5     }  
6     public void setAlter(int alter) {  
7         if(alter >= 0){  
8             this.alter = alter;  
9         }  
10    }  
11 }
```


Vorbesprechung Übung 4 – Bonus & Zusätzliches Material

Kleine Programmierprüfung nächste Woche

Kahoot