

# Record-Klassen

---

Carsten Gips (FH Bielefeld)

Unless otherwise noted, this work is licensed under CC BY-SA 4.0.

## Motivation; Klasse Studi

```
public class Studi {  
    private final String name;  
    private final int credits;  
  
    public Studi(String name, int credits) {  
        this.name = name;  
        this.credits = credits;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public int getCredits() {  
        return credits;  
    }  
}
```

## Klasse Studi als Record

```
public record StudiR(String name, int credits) {}
```

# Klasse Studi als Record

```
public record StudiR(String name, int credits) {}
```

- Immutable Klasse mit Feldern `String name` und `int credits`  
=> “(String name, int credits)” werden “Komponenten” des Records genannt
- Standardkonstruktor setzt diese Felder (“Kanonischer Konstruktor”)
- Getter für beide Felder:

```
public String name() { return this.name; }  
public int credits() { return this.credits; }
```

# Eigenschaften und Einschränkungen von Record-Klassen

- Records erweitern implizit die Klasse `java.lang.Record`:  
Keine andere Klassen mehr erweiterbar! (Interfaces kein Problem)
- Record-Klassen sind implizit final
- Keine weiteren (Instanz-) Attribute definierbar (nur die Komponenten)
- Keine Setter definierbar für die Komponenten: Attribute sind final
- Statische Attribute mit Initialisierung erlaubt

# Records: Prüfungen im Konstruktor

```
public record StudiS(String name, int credits) {  
    public StudiS(String name, int credits) {  
        if (name == null) { throw new IllegalArgumentException("Name cannot be null!"); }  
        else { this.name = name; }  
  
        if (credits < 0) { this.credits = 0; }  
        else { this.credits = credits; }  
    }  
}
```

```
public record StudiT(String name, int credits) {  
    public StudiT {  
        if (name == null) { throw new IllegalArgumentException("Name cannot be null!"); }  
  
        if (credits < 0) { credits = 0; }  
    }  
}
```

# Getter und Methoden

```
public record StudiR(String name, int credits) {}

public static void main(String... args) {
    StudiR r = new StudiR("Sabine", 75);

    int x = r.credits();
    String y = r.name();
}
```

```
public record StudiT(String name, int credits) {
    public int credits() { return credits + 42; }
    public void wuppie() { System.out.println("WUPPIE"); }
}
```

- Records sind immutable Klassen:
  - `final` Attribute (entsprechend den Komponenten)
  - Kanonischer Konstruktor
  - Automatische Getter (Namen wie Komponenten)
- Konstruktoren und Methoden können ergänzt/überschrieben werden
- Keine Vererbung von Klassen möglich (kein `extends`)



# LICENSE



Unless otherwise noted, this work is licensed under CC BY-SA 4.0.