

# Programmierkurs

Einführung in Java

Tag 4

Michael Fischer

Wintersemester 2020/21

Ablauf

Methoden

Warum?

Wie?

mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

Klassenvariablen

Warum?

Wie?

Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

Quellen & Lizenz

# Ablauf

- ▶ 09:30 Lösungen des Vortages
- ▶ ab 10:00 Vorlesung
- ▶ 60 min Mittagspause
- ▶ gegen 12:30 / 13:00 Vorstellung Aufgabenstellung + Übungen
- ▶ ca 15 Uhr: Vorstellung Lösungen Tag 4

## Ablauf

### Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

### Klassenvariablen

Warum?  
Wie?

### Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

### Quellen & Lizenz

# Inhaltsübersicht Vorkurs

- ▶ Tag 1: Zustände, Variablen, Datentypen, Konvertierungen, Arithmetik, Eclipse Livedemo
- ▶ Tag 2: Kommentare, Boolesche Ausdrücke, If-Abfragen, Switch-Case
- ▶ Tag 3: Arrays, (Do-)While-Schleife, For-Schleifen, Weiterführung Debugging
- ▶ Tag 4: (statische) Methoden, Klassenvariablen

## Ablauf

### Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

### Klassenvariablen

Warum?  
Wie?

### Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

### Quellen & Lizenz

# Code-Beispiel

```
185 public boolean checkAircraftExists(String iataAirline, String registrationCode, boolean flight) {
186
187     boolean check = false;
188     boolean foundAirline = false;
189     Iterator<Aircraft> i = getAircraftList().iterator();
190     Iterator<Airline> ia = this.getAirlineList().iterator();
191     String tempAircraftRegistration = null;
192     String airlineNationality = null;
193
194     if (iataAirline != null && registrationCode != null) {
195         //Check for correct input of IATA-Code.
196         if ((iataAirline.length() == 1 || iataAirline.length() == 2) && iataAirline.matches("[A-Z0-9]+")) {
197             //Check for correct input of the registration code of the aircraft.
198             if ((registrationCode.length() >= 3 && registrationCode.length() <= 5)
199                 && registrationCode.matches("[A-Z0-9]+")) {
200
201                 //Check if airline exists.
202                 if (checkAirlineExists(iataAirline)) {
203
204                     //Get the nationality of the airline and create the temporary aircraft registration the method
205                     //is looking for. Needed to compare the aircraft registrations.
206                     while (ia.hasNext() && !foundAirline) {
207                         Airline tempAirline = ia.next();
208                         if (tempAirline.getIATACodeAirline().equals(iataAirline)) {
209                             airlineNationality = tempAirline.getNationalityAirline();
210                             tempAircraftRegistration = airlineNationality + "-" + registrationCode;
211                             foundAirline = true;
212                         }
213                     }
214                     //Check aircraft list if the given aircraft exists.
215                     if (!getAircraftList().isEmpty()) {
216                         //Walkthrough the airlineList.
217                         while (i.hasNext() && !check) {
218                             Aircraft tempAircraft = i.next();
219
220                             String aircraftReg = tempAircraft.getAircraftRegistration();
221
222                             //Check if the registration code is the same as the given one.
223                             if (tempAircraftRegistration.equals(aircraftReg)) {
224
225                                 //Check which commando is used.
226                                 if (flight) {
227                                     //Check if the aircraft belongs to the given airline.
228                                     if (tempAircraft.getAirline().getIATACodeAirline().equals(iataAirline)) {
229                                         check = true;
230                                     }
231                                 }
232                                 else {
```

## Ablauf

### Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

### Klassenvariablen

Warum?  
Wie?

### Aufgabe

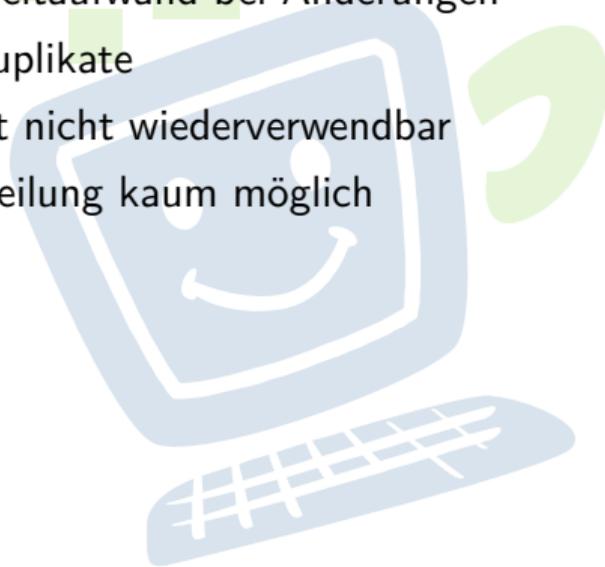
addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

### Quellen & Lizenz

# Warum ist dieser Code problematisch?

## Probleme:

- ▶ Unübersichtlich
- ▶ Hoher Zeitaufwand bei Änderungen
- ▶ Code-Duplikate
- ▶ Code oft nicht wiederverwendbar
- ▶ Arbeitsteilung kaum möglich



## Ablauf

## Methoden

### Warum?

Wie?

mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

## Klassenvariablen

Warum?

Wie?

## Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

## Quellen & Lizenz

# Warum ist dieser Code problematisch?

## Probleme:

- ▶ Unübersichtlich
- ▶ Hoher Zeitaufwand bei Änderungen
- ▶ Code-Duplikate
- ▶ Code oft nicht wiederverwendbar
- ▶ Arbeitsteilung kaum möglich

## Ziele:

- ▶ Gleichen Code auslagern
- ▶ Wiederverwendbaren Code schreiben
- ▶ **Wie? Methoden!**

## Ablauf

## Methoden

### Warum?

Wie?

mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

## Klassenvariablen

Warum?

Wie?

## Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

## Quellen & Lizenz

# Code-Beispiel

```
52  /**
53   * Sets the airline the booked flight belongs to.
54   *
55   * @param airline The given airline of the flight.
56   */
57  public void setAirline(Airline airline) {
58
59      if (airline != null) {
60
61          this.airline = airline;
62      }
63  }
64  /**
65   * Sets the flight number of the flight.
66   *
67   * @param flightNumber The given flight number.
68   */
69  public void setFlightNumber(int flightNumber) {
70
71      //Calculate the number of digits of the given flight number.
72      double numberDigits = Math.floor(Math.log10(flightNumber))+1;
73
74      //Check flight number for correct input.
75      if (flightNumber > 0 && numberDigits >= 2 && numberDigits <= 4) {
76          this.flightNumber = flightNumber;
77      }
78  }
```

## Ablauf

## Methoden

### Warum?

Wie?

mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

## Klassenvariablen

Warum?

Wie?

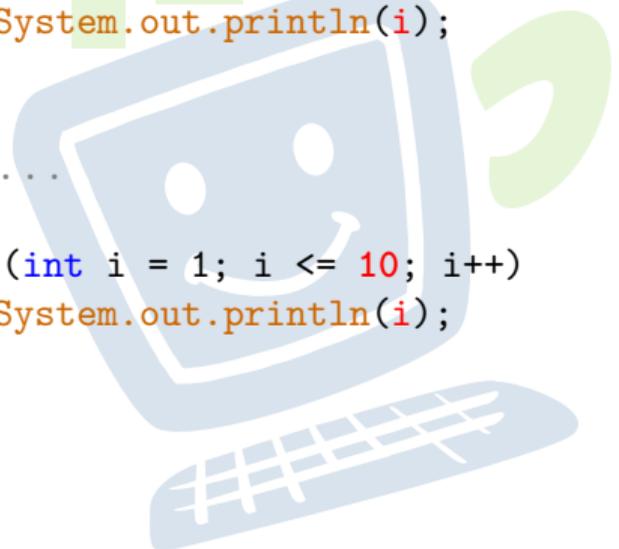
## Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

## Quellen & Lizenz

## Schlechter Code (Duplikate)

```
public class Main () {  
    public static void main() {  
        for (int i = 1; i <= 9; i++)  
            System.out.println(i);  
    }  
  
    // ...  
  
    for (int i = 1; i <= 10; i++)  
        System.out.println(i);  
    }  
}
```



### Ablauf

### Methoden

#### Warum?

Wie?

mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

### Klassenvariablen

Warum?

Wie?

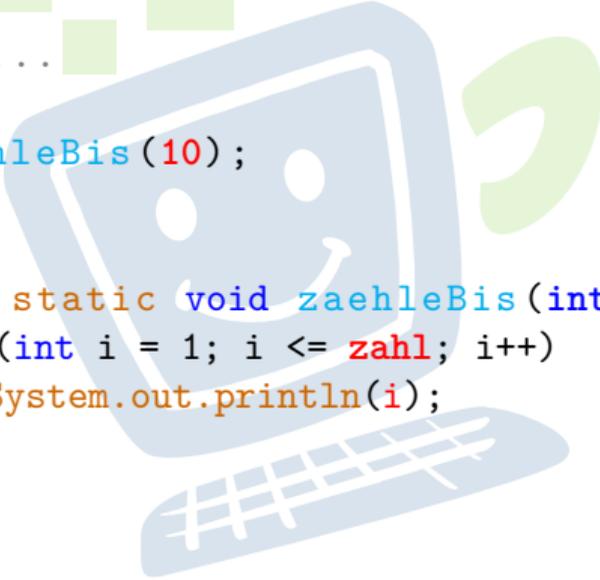
### Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

### Quellen & Lizenz

## Lösung mit Methoden

```
public class Main() {  
    public static void main() {  
        zaehleBis(9);  
  
        // ...  
  
        zaehleBis(10);  
    }  
  
    public static void zaehleBis(int zahl) {  
        for (int i = 1; i <= zahl; i++)  
            System.out.println(i);  
    }  
}
```



### Ablauf

### Methoden

#### Warum?

Wie?

mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

### Klassenvariablen

Warum?

Wie?

### Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

### Quellen & Lizenz

## Methodenaufbau

```
public static Datentyp Name(Parameter) {  
    // Methodenrumpf  
    return ... ;  
}
```

- ▶ **static**: Definiert eine Klassenmethode
  - ▶ Ermöglicht die Verwendung einer Methode direkt über die Klasse.

```
public class Main {  
    // Klassenrumpf  
}
```

- ▶ Hier: Wird nicht näher darauf eingegangen.

### Ablauf

#### Methoden

Warum?

Wie?

mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

#### Klassenvariablen

Warum?

Wie?

#### Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

#### Quellen & Lizenz

## Beispiel

```
public static void zaehlBisZehn () {  
    // ...  
}
```

**Anfang:**

public static

**Rückgabetyt:**

void

**Methodenname:**

zaehlBisZehn

**Parameter:**

keine

## Ablauf

## Methoden

Warum?

Wie?

mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

## Klassenvariablen

Warum?

Wie?

## Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

## Quellen &amp; Lizenz

## Verwenden einer Methode

```
// ...  
zaehlBisZehn();  
int x = gibMir42();  
int y = verdoppelWert(x);  
  
// ...
```

- ▶ `methodName(Parameter);`
  - ▶ Nach dem Methodennamen müssen **runde Klammern** folgen!
- ▶ Eine Methode **löst nur ein Problem!**

### Ablauf

#### Methoden

Warum?

Wie?

mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

#### Klassenvariablen

Warum?

Wie?

#### Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

#### Quellen & Lizenz

# Parameter

```
public static void zaehleBis(int zahl) {  
    // ...  
}
```

- ▶ In die runden Klammern kommen die Parameter
- ▶ Parameter werden mit Komma getrennt:  
`(int a, boolean b, double c)`
- ▶ Ein Parameter besteht aus **Datentyp** und **Bezeichner**

## Ablauf

## Methoden

Warum?

Wie?

### mit Parameter

mit Rückgabewert

Bessere Lesbarkeit

## Klassenvariablen

Warum?

Wie?

## Aufgabe

addNewTicket

getSum

calculateNewSum

resetSum

insertMoney

getAmountLeft

getChangeAmount

getChangeCoins

beginPayment

## Quellen & Lizenz

## Anwendung von Parametern

▶ `zaehleBis(9);`

```
public static void zaehleBis(int zahl) {  
    // "zahl" wird der Wert 9 zugewiesen  
}
```

▶ `sucheWortInText("Fachschaft");`

```
public static boolean sucheSchlagwort(String wort) {  
    // Sucht nach dem Wort innerhalb eines Textes  
    return gefunden;  
}
```

### Ablauf

### Methoden

Warum?

Wie?

### mit Parameter

mit Rückgabewert

Bessere Lesbarkeit

### Klassenvariablen

Warum?

Wie?

### Aufgabe

addNewTicket

getSum

calculateNewSum

resetSum

insertMoney

getAmountLeft

getChangeAmount

getChangeCoins

beginPayment

### Quellen & Lizenz

## Beispiel mit 2 Parametern

```
public class Main {  
    public static void main() {  
        zaehleVonBis(1, 9);  
        // ...  
        zaehleVonBis(5, 10);  
    }  
  
    public static void zaehleVonBis(int von, int bis) {  
        for (int i = von; i <= bis; i++) {  
            System.out.println(i);  
        }  
    }  
}
```

### Ablauf

### Methoden

Warum?

Wie?

mit Parameter

mit Rückgabewert

Bessere Lesbarkeit

### Klassenvariablen

Warum?

Wie?

### Aufgabe

addNewTicket

getSum

calculateNewSum

resetSum

insertMoney

getAmountLeft

getChangeAmount

getChangeCoins

beginPayment

### Quellen & Lizenz

# Rückgabewert

```
public static int gibMir42() {  
    ...  
    return 42;  
}
```

- ▶ Möchte man einen Wert zurück geben, so wird der entsprechende **Datentyp** angegeben (z.B. `int`, `double`, `String`)
- ▶ Wenn es keinen Rückgabewert gibt wird das Schlüsselwort `void` als Datentyp angegeben
- ▶ Wenn es einen Rückgabewert gibt, wird dieser in der Methode mit dem Befehl `return` zurückgegeben
- ▶ **Der Datentyp muss mit dem Rückgabewert übereinstimmen!**

## Ablauf

### Methoden

Warum?

Wie?

mit Parameter

mit Rückgabewert

Bessere Lesbarkeit

### Klassenvariablen

Warum?

Wie?

### Aufgabe

`addNewTicket`

`getSum`

`calculateNewSum`

`resetSum`

`insertMoney`

`getAmountLeft`

`getChangeAmount`

`getChangeCoins`

`beginPayment`

### Quellen & Lizenz

## Beispiel mit Rückgabewert

```
public class Main {  
    public static void main() {  
        int x = gibMir42();  
    }  
  
    public static int gibMir42() {  
        return 42;  
    }  
}
```

- ▶ `gibMir42()` wird ausgeführt und der Rückgabewert `42` in `x` gespeichert.

### Ablauf

#### Methoden

- Warum?
- Wie?
- mit Parameter
- mit Rückgabewert
- Bessere Lesbarkeit

#### Klassenvariablen

- Warum?
- Wie?

#### Aufgabe

- addNewTicket
- getSum
- calculateNewSum
- resetSum
- insertMoney
- getAmountLeft
- getChangeAmount
- getChangeCoins
- beginPayment

#### Quellen & Lizenz

## Lesbarkeit durch Methoden

- Berechnung der Fakultät:  $n! = 1 \times 2 \times \dots \times n$

```
public static long berechneFakultaet(int n) {  
    long fakultaet = 1L;  
  
    // Iterative Berechnung  
    for(int i = 1; i <= n; i++) {  
        fakultaet *= i;  
    }  
  
    return fakultaet;  
}
```



### Ablauf

#### Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

#### Klassenvariablen

Warum?  
Wie?

#### Aufgabe

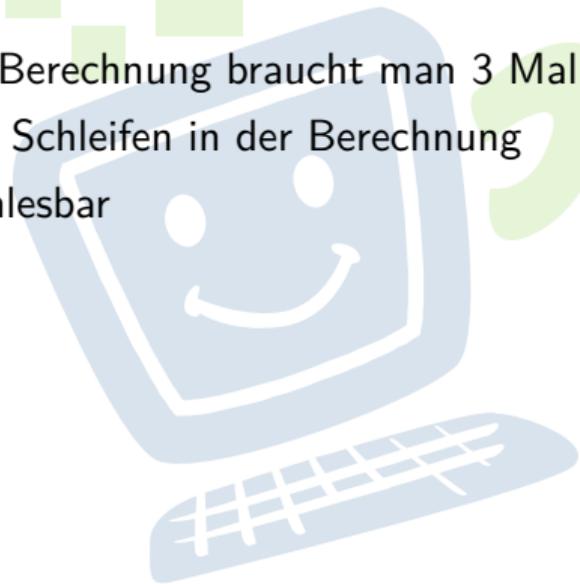
addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

#### Quellen & Lizenz

## Binomialkoeffizient berechnen

$$\binom{n}{k} = \frac{n!}{(n-k)! \times k!}$$

- ▶ Für die Berechnung braucht man 3 Mal eine Fakultät
- ▶ Somit 3 Schleifen in der Berechnung
- ▶ Code unlesbar



### Ablauf

### Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

### Klassenvariablen

Warum?  
Wie?

### Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

### Quellen & Lizenz

## Binomialkoeffizient berechnen

$$\binom{n}{k} = \frac{n!}{(n-k)! \times k!}$$

- ▶ Für die Berechnung braucht man 3 Mal eine Fakultät
- ▶ Somit 3 Schleifen in der Berechnung
- ▶ Code unlesbar

### Lösung

- ▶ Methode `long berechneFakultaet(int n)` benutzen

### Ablauf

### Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

### Klassenvariablen

Warum?  
Wie?

### Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

### Quellen & Lizenz

## Binomialkoeffizient berechnen

$$\binom{n}{k} = \frac{n!}{(n-k)! \times k!}$$

```
public static double binomialKoeffizient(int n, int k) {  
    double bin = berechneFakultaet(n) /  
        (berechneFakultaet(n - k) * berechneFakultaet(k));  
  
    return bin;  
}
```

## Ablauf

## Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

## Klassenvariablen

Warum?  
Wie?

## Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

## Quellen &amp; Lizenz

## Zusammenfassung Methoden

```
public static Datentyp methodenName(int Parameter) {  
    // Hier folgt Code...  
    return Rückgabewert;  
}
```

- ▶ Eine Methode mit Rückgabewert `void` hat kein `return`
- ▶ Methoden sollten möglichst klein sein. (Übersicht, leichter zu lesen & testen)
- ▶ Eine Methode **löst nur ein Problem!**

### Ablauf

#### Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

#### Klassenvariablen

Warum?  
Wie?

#### Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

#### Quellen & Lizenz



# Klassenvariablen

## Ablauf

## Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

## Klassenvariablen

Warum?  
Wie?

## Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

## Quellen & Lizenz

## Beispiel ohne Klassenvariablen

```
public class Vorkurs {
    public static void main() {
        int x = zaehleVonBis(1, 9);
        System.out.println(wurdeAusgegeben);
        // Das funktioniert so leider nicht
    }

    public static int zaehleVonBis(int von, int bis) {
        for (int i = von; i <= bis; i++) {
            System.out.println(i);
        }

        boolean wurdeAusgegeben = bis >= von;
        return bis - von + 1;
    }
}
```

### Ablauf

#### Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

#### Klassenvariablen

Warum?  
Wie?

#### Aufgabe

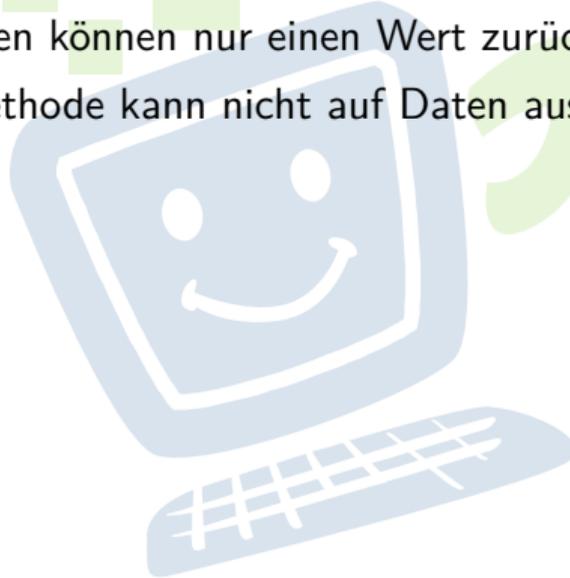
addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

#### Quellen & Lizenz

# Warum Klassenvariablen nutzen?

## Probleme

- ▶ Methoden können nur einen Wert zurückgeben
- ▶ Eine Methode kann nicht auf Daten aus anderen Methoden zugreifen



## Ablauf

### Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

### Klassenvariablen

#### Warum?

Wie?

### Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

### Quellen & Lizenz

# Warum Klassenvariablen nutzen?

## Probleme

- ▶ Methoden können nur einen Wert zurückgeben
- ▶ Eine Methode kann nicht auf Daten aus anderen Methoden zugreifen

## Lösung

- ▶ **Klassenvariablen**

**Hinweis:** Überlegt immer, ob ihr Klassenvariablen wirklich benötigt.

## Ablauf

### Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

### Klassenvariablen

#### Warum?

Wie?

### Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

### Quellen & Lizenz

## Beispiel mit Klassenvariablen

```
public class Vorkurs {
    public static boolean wurdeAusgegeben;

    public static void main() {
        int x = zaehleVonBis(1, 9);
        System.out.println(wurdeAusgegeben);
    }

    public static int zaehleVonBis(int von, int bis) {
        for (int i = von; i <= bis; i++) {
            System.out.println(i);
        }

        wurdeAusgegeben = bis >= von;
        return bis - von + 1;
    }
}
```

### Ablauf

#### Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

#### Klassenvariablen

Warum?  
Wie?

### Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

### Quellen & Lizenz

## Deklaration von Klassenvariablen

- ▶ Deklaration direkt nach Klassendeklaration
- ▶ `public static Datentyp Variablenname;`
- ▶ Sichtbar bzw. zugreifbar in der ganzen Klasse.
- ▶ Ohne manuelle Zuweisung wird der Defaultwert zugewiesen.
  - ▶ Defaultwerte werden vom Java-Compiler **automatisch** zugewiesen, wenn eine Variable **keinen Wert** zugewiesen bekommen hat.
  - ▶ `short/int/long = 0/0/0L` | `boolean = false` |  
`float/double = 0.0f/0.0` | `char = ""` | `String = null`
- ▶ Beispiel:  
`public static boolean wurdeAusgegeben;`

### Ablauf

### Methoden

Warum?

Wie?

mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

### Klassenvariablen

Warum?

Wie?

### Aufgabe

`addNewTicket`  
`getSum`  
`calculateNewSum`  
`resetSum`  
`insertMoney`  
`getAmountLeft`  
`getChangeAmount`  
`getChangeCoins`  
`beginPayment`

### Quellen & Lizenz



# Übungsaufgabe



Habt ihr noch Fragen?

## Ablauf

### Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

### Klassenvariablen

Warum?  
Wie?

## Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

## Quellen & Lizenz

# Vorstellung der Aufgabe

Abfahrtsbahnhof:  
Karlsruhe HBF

Zielbahnhof:  
< Bitte auswählen >

Bezahlen

Bitte wählen Sie einen Zielbahnhof aus:



München Frankfurt Weitere ...

Tickets  
München  
Frankfurt  
München (Bahncard)  
Frankfurt (Bahncard)

Gesamtpreis: 153,82 EUR Alle Tickets löschen

Abfahrtsbahnhof:  
Karlsruhe HBF

Zielbahnhof:  
< bitte auswählen >

Bezahlen

Bitte werfen Sie Geld ein.

Noch zu zahlen: 153,82 EUR



## Ablauf

## Methoden

Warum?  
Wie?  
mit Parameter  
Bessere Lesbarkeit

## Klassenvariablen

Warum?  
Wie?

## Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

## Quellen & Lizenz

# Vorstellung der Aufgabe

LIVE DEMO!!!

Abfahrtsbahnhof:  
Karlsruhe HBF

Zielbahnhof:  
< bitte auswählen >

**Bezahlen**

Bitte werfen Sie Geld ein.

Rückgeld: 6,18 EUR

Rückgeld:

3x		1x	
0x		1x	
0x		1x	
0x		1x	

Ablauf

Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

Klassenvariablen

Warum?  
Wie?

Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

Quellen & Lizenz

## addNewTicket

```
public static String[] addNewTicket (String[] oldTickets,  
    String newTicket, boolean bahncard)
```

### Beschreibung:

Soll den Parameter **oldTickets** um den Parameter **newTicket** erweitern und für den Fall dass der Parameter **bahncard true** ist um „(Bahncard)“ erweitern und anschließend zurück geben.

### Ablauf

### Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

### Klassenvariablen

Warum?  
Wie?

### Aufgabe

#### addNewTicket

getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

### Quellen & Lizenz

getSum

```
public static double getSum ()
```

**Beschreibung:**

Gibt den aktuellen Gesamtpreis als double in Euro zurück

## Ablauf

## Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

## Klassenvariablen

Warum?  
Wie?

## Aufgabe

addNewTicket  
**getSum**  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

## Quellen &amp; Lizenz

## calculateNewSum

```
public static void calculateNewSum(int distance, boolean  
    bahncard)
```

### Beschreibung:

Berechnet den Gesamtpreis aller bisher sowie dem aktuell ausgewählten Ticket.

Dabei gilt:

- ▶ Bis 200km: 10 € + 0.20 € pro km
- ▶ Ab 200km: 5 € + 0.15 € pro km
- ▶ Mit einer Bahncard erhält man immer 25% Rabatt

### Ablauf

### Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

### Klassenvariablen

Warum?  
Wie?

### Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

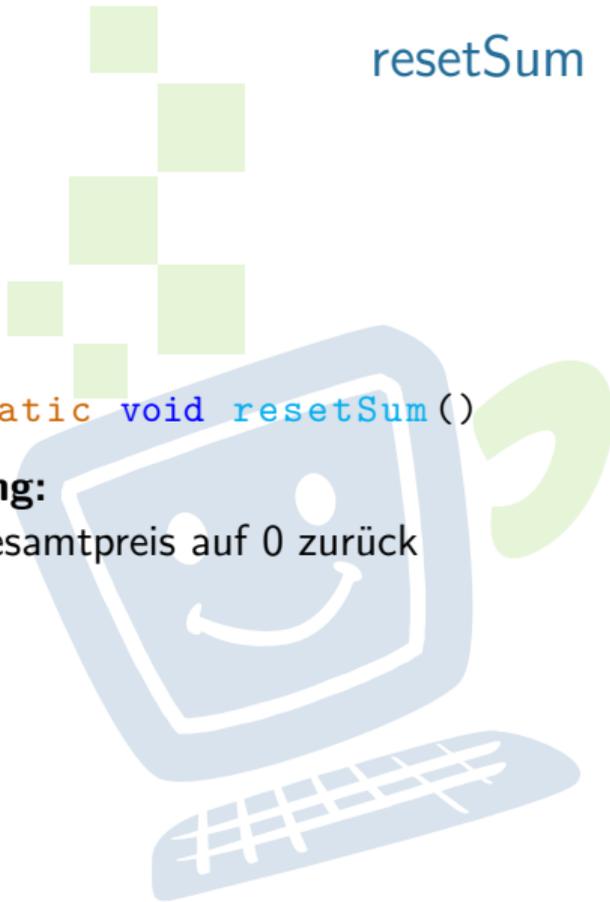
### Quellen & Lizenz

## resetSum

```
public static void resetSum()
```

**Beschreibung:**

Setzt den Gesamtpreis auf 0 zurück



## Ablauf

## Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

## Klassenvariablen

Warum?  
Wie?

## Aufgabe

addNewTicket  
getSum  
calculateNewSum  
**resetSum**

insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

## Quellen &amp; Lizenz

# insertMoney

```
public static void insertMoney(int amount)
```

## Beschreibung:

Wenn der Kunde einen Geldschein einwirft wird diese Methode aufgerufen. Der noch zu bezahlende Betrag muss dementsprechend angepasst werden.

## Ablauf

### Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

### Klassenvariablen

Warum?  
Wie?

### Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum

### insertMoney

getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

### Quellen & Lizenz

## getAmountLeft

```
public static double getAmountLeft()
```

### Beschreibung:

Als Rückgabewert soll hier der noch zu zahlende Betrag als double in Euro zurück gegeben werden.

### Ablauf

### Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

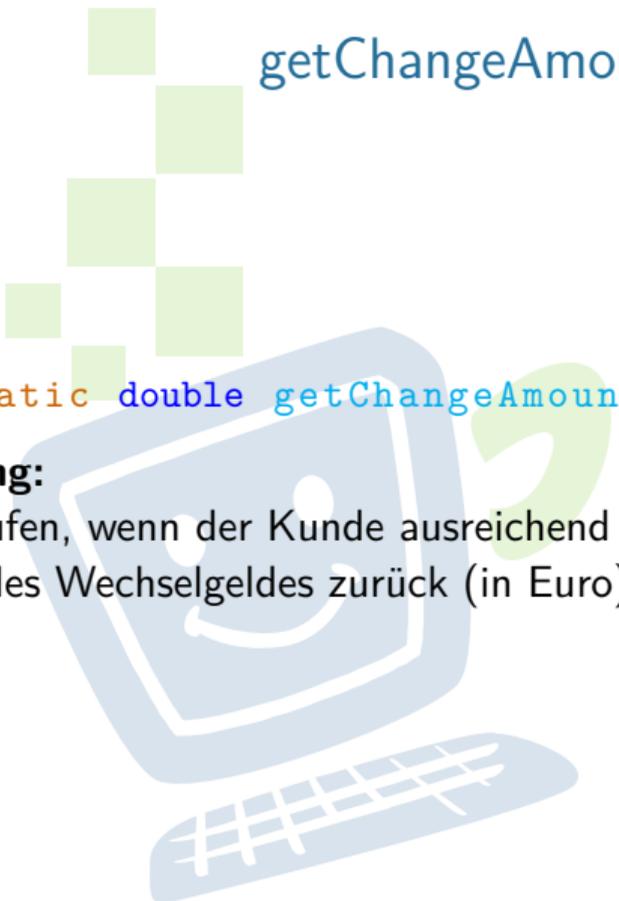
### Klassenvariablen

Warum?  
Wie?

### Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

### Quellen & Lizenz



## getChangeAmount

```
public static double getChangeAmount ()
```

### Beschreibung:

Wird aufgerufen, wenn der Kunde ausreichend Geld eingeworfen hat und gibt den Betrag des Wechselgeldes zurück (in Euro). Dieser muss positiv sein.

### Ablauf

#### Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

#### Klassenvariablen

Warum?  
Wie?

#### Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
**getChangeAmount**  
getChangeCoins  
beginPayment

#### Quellen & Lizenz

## getChangeCoins

```
public static int[] getChangeCoins ()
```

### Beschreibung:

Berechnet, wie viele Münzen von jeder Sorte der Kunde zurück bekommt.  
Gibt ein Array zurück, das die Anzahl der entsprechenden Münzen enthält

- ▶ Rückgabe[0]: enthält die Anzahl der 2 €-Münzen
- ▶ Rückgabe[1]: enthält die Anzahl der 1 €-Münzen
- ▶ ...
- ▶ Rückgabe[7]: enthält die Anzahl der 1 ct-Münzen

### Ablauf

#### Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

#### Klassenvariablen

Warum?  
Wie?

#### Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment

#### Quellen & Lizenz

## beginPayment (OPTIONAL)

```
public static void beginPayment ()
```

- ▶ Diese Methode ist optional
- ▶ Wird nicht immer benötigt
- ▶ Kann für spezielle Aktionen zu Beginn des Bezahlvorgangs verwendet werden

### Ablauf

### Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

### Klassenvariablen

Warum?  
Wie?

### Aufgabe

addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
**beginPayment**

### Quellen & Lizenz

## Quellen und Lizenz

FACHSCHAFT INFORMATIK

HS Karlsruhe



- ▶ Original von Samuel Zeitvogel
- ▶ Überarbeitet 2012 von Daniel Hoff
- ▶ Überarbeitet 2013 von Tristan Wagner
- ▶ Überarbeitet 2015 von Tobias Kerst
- ▶ Überarbeitet 2016 von Christian Wernet
- ▶ Überarbeitet 2018 von Jan Oliver Zerulla
- ▶ Überarbeitet 2018 von Johannes Beierle
- ▶ Überarbeitet 2020 von Jonas Westenhoff (SS) und Michael Fischer (WS)

## Ablauf

## Methoden

Warum?  
Wie?  
mit Parameter  
mit Rückgabewert  
Bessere Lesbarkeit

## Klassenvariablen

Warum?  
Wie?

## Aufgabe

```
addNewTicket  
getSum  
calculateNewSum  
resetSum  
insertMoney  
getAmountLeft  
getChangeAmount  
getChangeCoins  
beginPayment
```

## Quellen &amp; Lizenz