



# Programmierungsvorkurs

## Tag 4

Michael Martel

# Ablauf

- 09:30** Tag 3 Übungsbesprechung
- 10:00** Vorlesung
- 12:00** 60 Minuten Mittagspause
- 13:00** Übungen im LI 136 und LI 137

**Danach**



# Inhaltsübersicht Vorkurs

---



Tag 1	Zustände, Variablen, Datentypen, Konvertierungen, Arithmetik, Eclipse Live-Demo
Tag 2	Kommentare, Boolesche Ausdrücke, If-Abfragen, Switch-Case
Tag 3	Arrays, (Do-)While-Schleife, For-Schleifen, Weiterführung Debugging
<b>Tag 4</b>	<b>Methoden, Klassen, JavaDoc</b>



# Prozesse digitalisieren

---





# Methoden / Funktionen

Die Begriffe Methode und Funktion werden oft synonym verwendet

Ziel einer Methode / Funktion ist es eine bestimmte zugeteilte Aufgabe zu erledigen

```
public class Beispiel {  
  
    public static void main(String[] args) {  
        System.out.println("Hello World");  
        double sqrt = Math.sqrt(2.0);  
        doMath();  
    }  
  
    public static void doMath() {  
        int a = 42;  
        int b = a * 1337;  
        System.out.println(b);  
    }  
}
```

# Methoden / Funktionen

**Rückgabetyp**      **Name**      **Parameter**

```
public static String getErdbeerkaese(int anzahl) {  
    return anzahl + " Portionen Erdbeerkäse";  
}
```

**Rumpf**

Eine Methode ohne Rückgabewert wird mit dem Schlüsselwort **void** gekennzeichnet!

## Aufgabe:

Implementiere eine Hilfsfunktion um `System.out.println(String)` zu verkürzen auf `println(String)`

```
public static void println(String text) {  
    System.out.println(text);  
}
```

## Aufgabe:

Addiere zwei (ganze | reelle) Zahlen

```
public static int addiere(int a, int b) {  
    return a + b;  
}
```

```
public static double addiere(double a, double b) {  
    return a + b;  
}
```



## Aufgabe:

Betrag eines 2D-Vektors

$$|\vec{a}| = \sqrt{a_x^2 + a_y^2} \quad \text{Math.sqrt()}$$

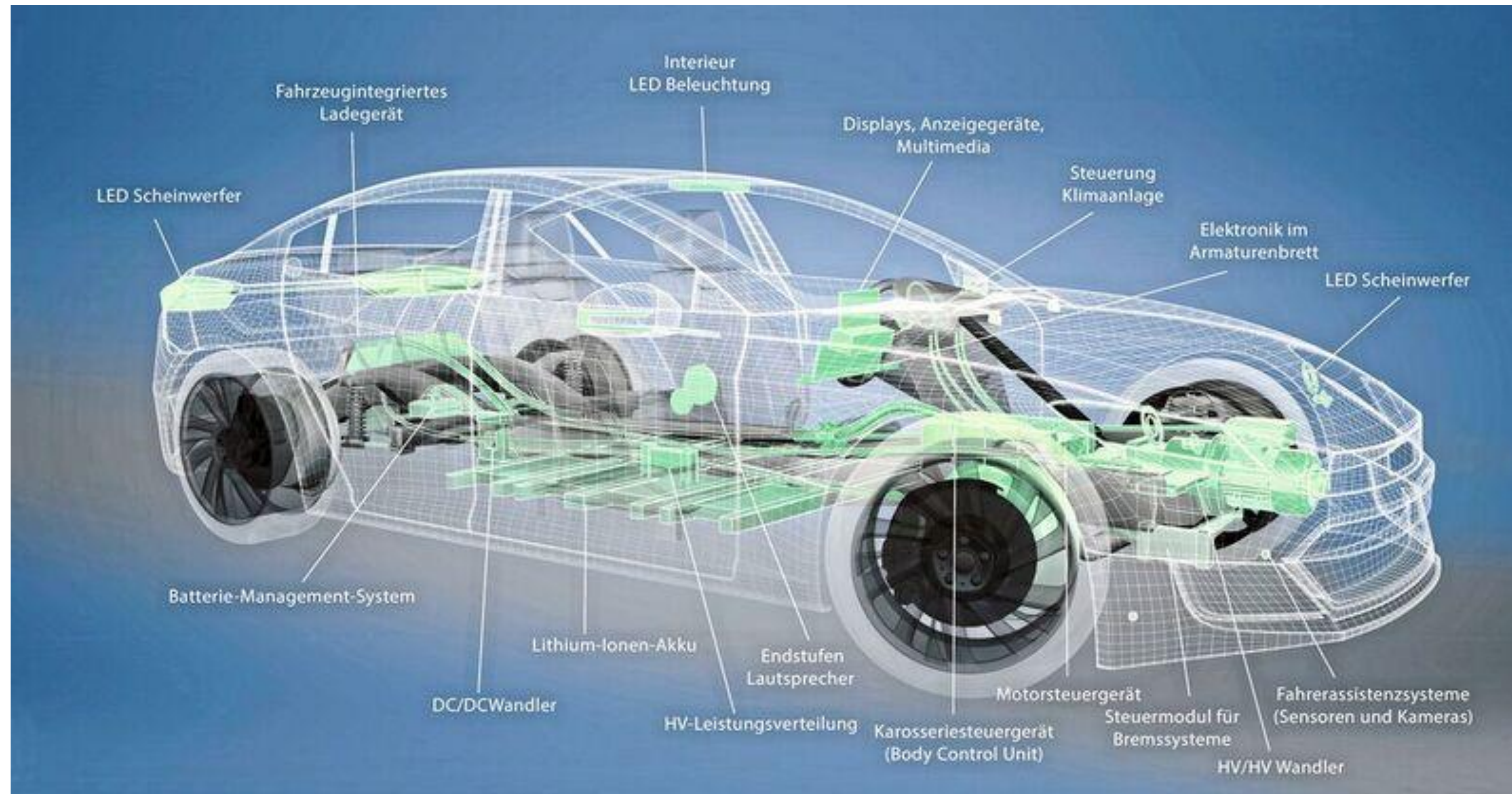
```
public static double vektorBetrag(double[] vec) {  
    return Math.sqrt(vec[0] * vec[0] + vec[1] * vec[1]);  
}
```

# Methoden / Funktionen



## Fragen?







# Klassen

Kaffeefach

Wassertank

Heizelement

Aufschäumer



```
public class Kaffeemaschine {  
    private Wassertank wassertank;  
    private Heizelement heizelement;  
  
    public Kaffeemaschine(Wassertank wassertank, ...) {  
        this.wassertank = wassertank;  
        // ...  
    }  
  
    public Kaffee kocheKaffee(double menge) {  
        // ...  
        double wasserMenge = wassertank.getWasser(menge);  
        // ...  
        return kaffee;  
    }  
}
```

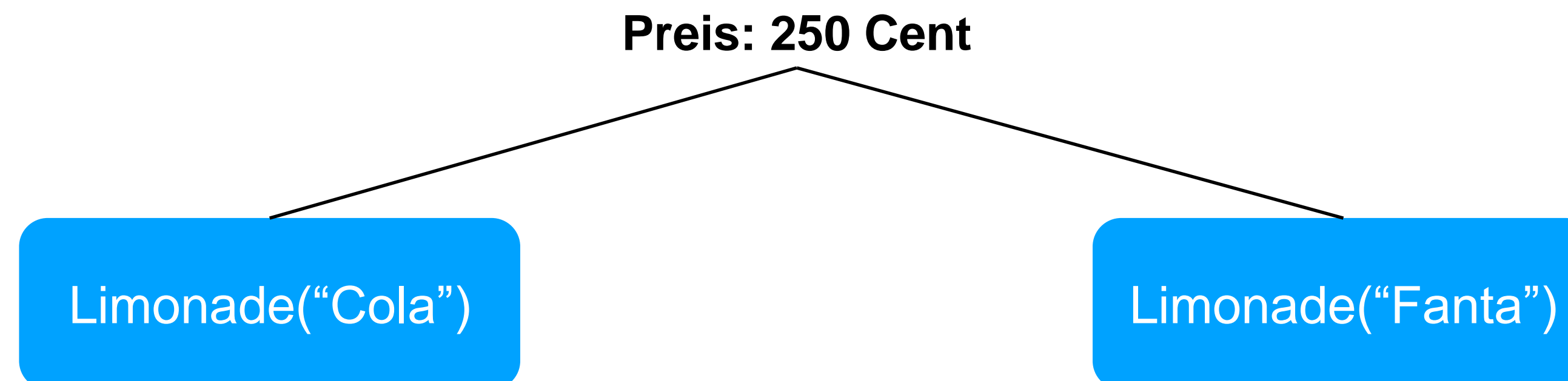
# Klassenvariablen

```
public class Kaffeemaschine {  
    private Wassertank wassertank;  
    private Heizelement heizelement;  
  
    public Kaffeemaschine(Wassertank wassertank, ...) {  
        this.wassertank = wassertank;  
        // ...  
    }  
  
    public Kaffee kocheKaffee(double menge) {  
        // ...  
        double wasserMenge = wassertank.getWasser(menge);  
        // ...  
        return kaffee;  
    }  
}
```



# Statisch und Instanz

```
public class Limonade {  
    public static int preis = 250;  
    public String name;  
  
    public Limonade(String name) {  
        this.name = name;  
    }  
}
```



# Statisch und Instanz



```
public static void main(String[] args) {  
    Limonade cola = new Limonade("Cola");  
    Limonade fanta = new Limonade("Fanta");  
  
    System.out.println(cola.name); // Cola  
    System.out.println(fanta.name); // Fanta  
  
    System.out.println(Limonade.preis); // 250  
}
```

# Beispiel Applikation





# Klassen und Klassenvariablen



## Fragen?

# JavaDoc



```
1 package kaffee;
2
3 /**
4  * Das ist eine super tolle kaffee.Kaffeemaschine.
5  */
6 public class Kaffeemaschine {
7
8     2 usages
9     private Wassertank wassertank;
10    1 usage
11    private Heizelement heizelement;
12
13    1 usage
14    public Kaffeemaschine(Wassertank wassertank, Heizelement heizelement) {
15        this.wassertank = wassertank;
16        this.heizelement = heizelement;
17    }
18
19    /**
20     * Kocht digitalen Kaffee. Vorsicht heiß
21     *
22     * @param menge Kaffeemenge in ml
23     * @return Ein schmackhafter digitaler Kaffee
24     */
25    1 usage
26    public Kaffee kocheKaffee(double menge) {
27        Kaffee kaffee = new Kaffee();
28
29        kaffee.addWasser(wassertank.getWasser(menge));
30        kaffee.addKaffePulver(menge * 9.425);
31
32        return kaffee;
33    }
34 }
```

```
/**
 * Das ist eine Kaffeemaschine.
 */
public class Kaffeemaschine {

    /**
     * Kocht digitalen Kaffee. Vorsicht heiß!
     *
     * @param menge Kaffeemenge in ml
     * @return Ein schmackhafter digitaler Kaffee
     */
    public Kaffee kocheKaffee(double menge) {
        // ...
        return kaffee;
    }
}
```



# JavaDoc reale Beispiele



```
and the material regarding "Systems of Time" at:  
  
https://www.usno.navy.mil/USNO/time/master-clock/systems-of-time  
  
which has descriptions of various different time systems including UT, UT1, and UTC.  
  
In all methods of class Date that accept or return year, month, date, hours, minutes, and seconds values, the following representations are used:  


- A year y is represented by the integer y - 1900.
- A month is represented by an integer from 0 to 11; 0 is January, 1 is February, and so forth; thus 11 is December.
- A date (day of month) is represented by an integer from 1 to 31 in the usual manner.
- An hour is represented by an integer from 0 to 23. Thus, the hour from midnight to 1 a.m. is hour 0, and the hour from noon to 1 p.m. is hour 12.
- A minute is represented by an integer from 0 to 59 in the usual manner.
- A second is represented by an integer from 0 to 61; the values 60 and 61 occur only for leap seconds and even then only in Java implementations that actually track leap seconds correctly. Because of the manner in which leap seconds are currently introduced, it is extremely unlikely that two leap seconds will occur in the same minute, but this specification follows the date and time conventions for ISO C.

  
In all cases, arguments given to methods for these purposes need not fall within the indicated ranges; for example, a date may be specified as January 32 and is interpreted as meaning February 1.  
  
Since: 1.0  
See Also: DateFormat,  
Calendar,  
TimeZone  
  
Author: James Gosling, Arthur van Hoff, Alan Liu  
  
3 inheritors  
133 public class Date  
134 implements java.io.Serializable, Cloneable, Comparable<Date>  
135 {  
136 private static final BaseCalendar gcal =  
137 CalendarSystem.getGregorianCalendar();  
138 private static BaseCalendar jcal;  
139  
140 private transient long fastTime;  
141  
142 /*
```



Ende! Fragen?

