

Script generated by TTT

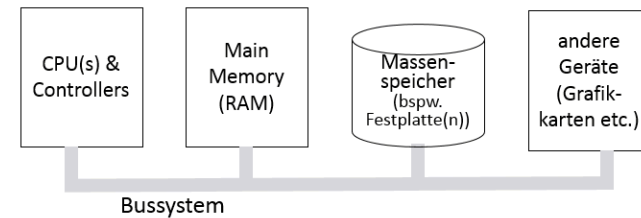
Title: groh: profile1 (13.05.2016)

Date: Fri May 13 13:29:48 CEST 2016

Duration: 83:22 min

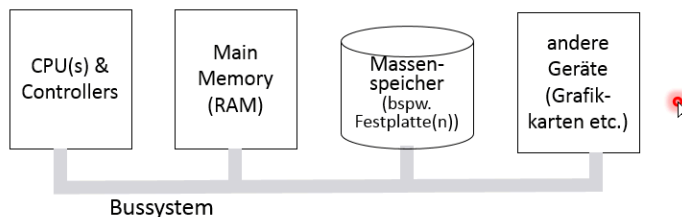
Pages: 31

- **Programme**¹ bestehen aus **Instruktionen** (Deklarationen und Statements) in einer formalen Programmiersprache (--> Syntax und Semantik)
- Statements¹ werden **schrittweise** auf einem Computersystem **ausgeführt**. Schritte ändern den **Zustand** (bspw. Speicherzustand) des Computersystems.
- Typisches Computersystem: **Von-Neumann-Architektur** (stark vereinfacht):



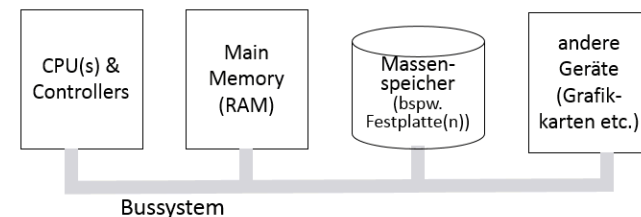
⁽¹⁾: bezieht sich hauptsächlich auf imperative Sprachen

- **Programme**¹ bestehen aus **Instruktionen** (Deklarationen und Statements) in einer formalen Programmiersprache (--> Syntax und Semantik)
- Statements¹ werden **schrittweise** auf einem Computersystem **ausgeführt**. Schritte ändern den **Zustand** (bspw. Speicherzustand) des Computersystems.
- Typisches Computersystem: **Von-Neumann-Architektur** (stark vereinfacht):



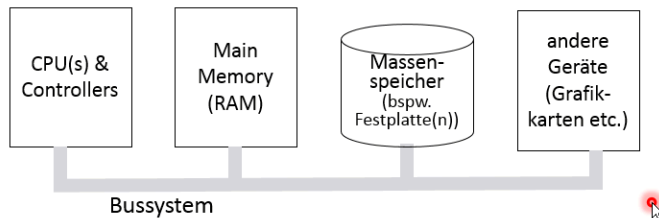
⁽¹⁾: bezieht sich hauptsächlich auf imperative Sprachen

- **Programme**¹ bestehen aus **Instruktionen** (Deklarationen und Statements) in einer formalen Programmiersprache (--> Syntax und Semantik)
- Statements¹ werden **schrittweise** auf einem Computersystem **ausgeführt**. Schritte ändern den **Zustand** (bspw. Speicherzustand) des Computersystems.
- Typisches Computersystem: **Von-Neumann-Architektur** (stark vereinfacht):



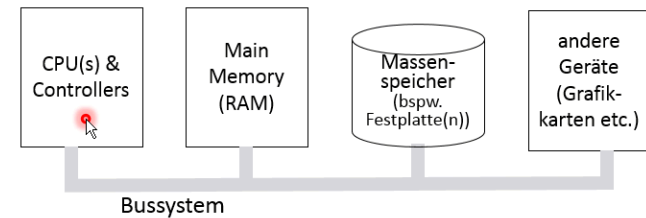
⁽¹⁾: bezieht sich hauptsächlich auf imperative Sprachen

- Programme¹ bestehen aus **Instruktionen** (Deklarationen und Statements) in einer formalen Programmiersprache (--> Syntax und Semantik)
- Statements¹ werden **schrittweise** auf einem Computersystem **ausgeführt**. Schritte ändern den **Zustand** (bspw. Speicherzustand) des Computersystems.
- Typisches Computersystem: **Von-Neumann-Architektur** (stark vereinfacht):

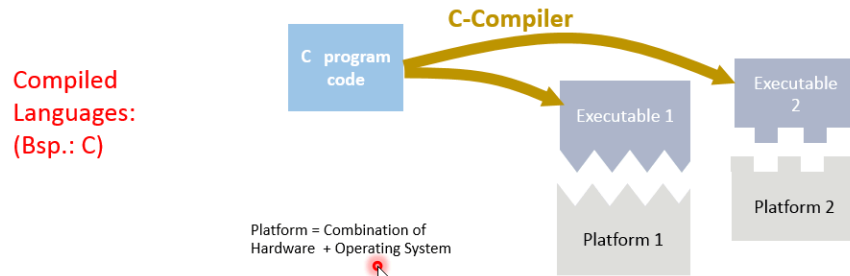


(¹): bezieht sich hauptsächlich auf imperative Sprachen

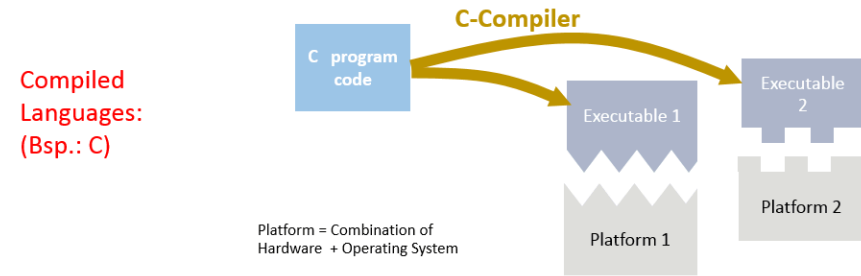
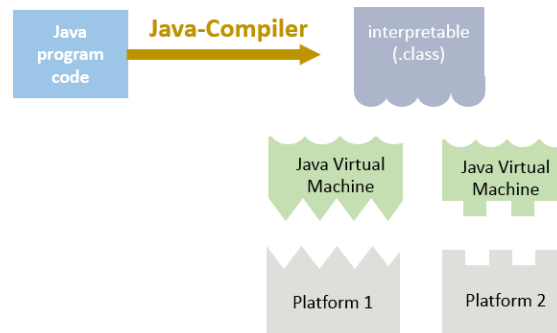
- Programme¹ bestehen aus **Instruktionen** (Deklarationen und Statements) in einer formalen Programmiersprache (--> Syntax und Semantik)
- Statements¹ werden **schrittweise** auf einem Computersystem **ausgeführt**. Schritte ändern den **Zustand** (bspw. Speicherzustand) des Computersystems.
- Typisches Computersystem: **Von-Neumann-Architektur** (stark vereinfacht):



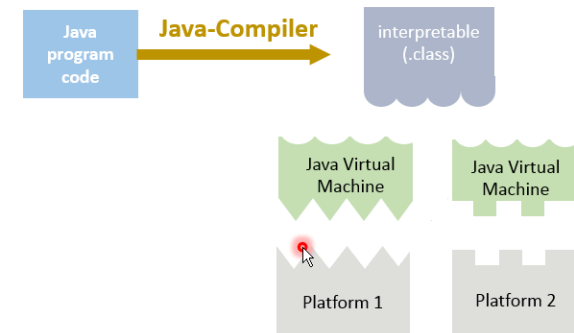
(¹): bezieht sich hauptsächlich auf imperative Sprachen



„Virtual Machine Languages“: (Bsp.: Java)



„Virtual Machine Languages“: (Bsp.: Java)



Programm und vereinfachtes Speichermodell

```

...
int horst;
int heiner;
int fritz;
horst = 101;
heiner = 2;
fritz = horst + heiner;
horst = 2000;
...
    
```

Vereinfachtes Speicher-Modell		
Zell-Nr (Adresse)	Zell-Name (Variablenname)	Zell-Inhalt
		⋮
1124		
1125	horst	
1126	heiner	
1127		
1128	fritz	
		⋮
4027		
4028		int horst;
4029		int heiner;
4030		int fritz;
4029		horst = 101;
4030		heiner = 2;
		⋮

Daten

Programm

8

Programm und vereinfachtes Speichermodell

```

...
int horst;
int heiner;
int fritz;
horst = 101;
heiner = 2;
fritz = horst + heiner;
horst = 2000;
...
    
```

Variablenname

Deklarationen

Statements

Vereinfachtes Speicher-Modell		
Zell-Nr (Adresse)	Zell-Name (Variablenname)	Zell-Inhalt
		⋮
1124		
1125	horst	
1126	heiner	
1127		
1128	fritz	
		⋮
4027		
4028		int horst;
4029		int heiner;
4030		int fritz;
4029		horst = 101;
4030		heiner = 2;
		⋮

Daten

Programm

9

Programm und vereinfachtes Speichermodell

```

...
int horst;
int heiner;
int fritz;
horst = 101;
heiner = 2;
fritz = horst + heiner;
horst = 2000;
...
    
```

Variablenname

Deklarationen

Statements

Vereinfachtes Speicher-Modell		
Zell-Nr (Adresse)	Zell-Name (Variablenname)	Zell-Inhalt
		⋮
1124		
1125	horst	
1126	heiner	
1127		
1128	fritz	
		⋮
4027		
4028		int horst;
4029		int heiner;
4030		int fritz;
4029		horst = 101;
4030		heiner = 2;
		⋮

Daten

Programm

9

Programm und vereinfachtes Speichermodell

```

...
int horst;
int heiner;
int fritz;
horst = 101;
heiner = 2;
fritz = horst + heiner;
horst = 2000;
...
    
```

Variablenname

Deklarationen

Statements

Vereinfachtes Speicher-Modell		
Zell-Nr (Adresse)	Zell-Name (Variablenname)	Zell-Inhalt
		⋮
1124		
1125	horst	
1126	heiner	
1127		
1128	fritz	
		⋮
4027		
4028		int horst;
4029		int heiner;
4030		int fritz;
4029		horst = 101;
4030		heiner = 2;
		⋮

Daten

Programm

9

```
...
int horst;
int heiner;
int fritz;
horst = 101;
heiner = 2;
fritz = horst + heiner;
horst = 2000;
...
```



Was ist der Wert der Speicherstelle 1128 nachdem das rote Statement ausgeführt wurde?

A	103
B	0
C	101
D	101 oder 2

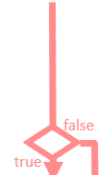
Vereinfachtes Speicher-Modell

Zell-Nr (Adresse)	Zell-Name (Variablenname)	Zell-Inhalt
1124		⋮
1125	horst	101
1126	heiner	2
1127		⋮
1128	fritz	???
⋮		⋮
4027		⋮
4028		int horst;
4029		int heiner;
4030		int fritz;
4029		horst = 101;
4030		heiner = 2;
⋮		⋮

Daten

Programm

```
...
int horst;
int heiner;
int fritz;
horst = 101;
heiner = 2;
fritz = horst + heiner;
horst = 2000;
if (heiner == 2)
    horst = 10;
else
    horst = 11;
fritz = 17;
...
```



Vereinfachtes Speicher-Modell

Zell-Nr (Adresse)	Zell-Name (Variablenname)	Zell-Inhalt
1124		⋮
1125	horst	2000
1126	heiner	2
1127		⋮
1128	fritz	103
⋮		⋮
4027		⋮
4028		int horst;
4029		int heiner;
4030		int fritz;
4029		horst = 101;
4030		heiner = 2;
⋮		⋮

Daten

Programm

```
...
int horst;
int heiner;
int fritz;
horst = 101;
heiner = 2;
fritz = horst + heiner;
horst = 2000;
if (heiner == 2)
    horst = 10;
else
    horst = 11;
fritz = 17;
...
```



Was passiert nachdem das gesamte if-Statement ausgeführt wurde?

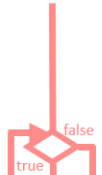
Vereinfachtes Speicher-Modell

Zell-Nr (Adresse)	Zell-Name (Variablenname)	Zell-Inhalt
1124		⋮
1125	horst	2000
1126	heiner	2
1127		⋮
1128	fritz	103
⋮		⋮
4027		⋮
4028	A	nichts
4029	B	Speicherzelle 1125 wird auf 10 gesetzt
4029	C	horst wird auf 11 gesetzt
4030	D	horst wird auf 10 gesetzt

Daten

Programm

```
...
int horst;
int heiner;
int fritz;
horst = 101;
heiner = 2;
fritz = horst + heiner;
horst = 2000;
while (heiner > 0)
    heiner = heiner - 1;
fritz = 17;
...
```



Vereinfachtes Speicher-Modell

Zell-Nr (Adresse)	Zell-Name (Variablenname)	Zell-Inhalt
1124		⋮
1125	horst	2000
1126	heiner	2
1127		⋮
1128	fritz	103
⋮		⋮
4027		⋮
4028		int horst;
4029		int heiner;
4030		int fritz;
4029		horst = 101;
4030		heiner = 2;
⋮		⋮

Daten

Programm

Kontrollfluss: Schleife

```

...
int horst;
int heiner;
int fritz;
horst = 101;
heiner = 2;
fritz = horst + heiner;
horst = 2000;
while (heiner > 0)
    heiner = heiner - 1;
    fritz = 17;
...
    
```

Vereinfachtes Speicher-Modell

Zell-Nr (Adresse)	Zell-Name (Variablenname)	Zell-Inhalt
1124		⋮
1125	horst	2000
1126	heiner	1
1127		
1128	fritz	103
		⋮
4027		
4028		int horst;
4029		int heiner;
4030		int fritz;
4029		horst = 101;
4030		heiner = 2;
		⋮

Daten

Programm

Kontrollfluss: Schleife

```

...
int horst;
int heiner;
int fritz;
horst = 101;
heiner = 2;
fritz = horst + heiner;
horst = 2000;
while (heiner > 0)
    heiner = heiner - 1;
    fritz = 17;
...
    
```

Vereinfachtes Speicher-Modell

Zell-Nr (Adresse)	Zell-Name (Variablenname)	Zell-Inhalt
1124		⋮
1125	horst	2000
1126	heiner	1
1127		
1128	fritz	103
		⋮
4027		
4028		int horst;
4029		int heiner;
4030		int fritz;
4029		horst = 101;
4030		heiner = 2;
		⋮

Daten

Programm

Kontrollfluss: Schleife

```

...
int horst;
int heiner;
int fritz;
horst = 101;
heiner = 2;
fritz = horst + heiner;
horst = 2000;
while (heiner > 0)
    heiner = heiner - 1;
    fritz = 17;
...
    
```

Vereinfachtes Speicher-Modell

Zell-Nr (Adresse)	Zell-Name (Variablenname)	Zell-Inhalt
1124		⋮
1125	horst	2000
1126	heiner	0
1127		
1128	fritz	17
		⋮
4027		
4028		int horst;
4029		int heiner;
4030		int fritz;
4029		horst = 101;
4030		heiner = 2;
		⋮

Daten

Programm

Antwort A ist richtig!

Prozedurale Programmierung

Gruppieren Sie Sequenzen von Instruktionen in benannte „Prozeduren“ („Funktionen“, „Methoden“, „Subroutinen“ etc.)

```

int doSelfSumSquare(int someNumber) {
    int a;
    a = someNumber + someNumber;
    a = a * a;
    return a;
}
    
```

$$f(x) = (x + x)^2$$

Gruppieren Sie Sequenzen von Instruktionen in benannte „Prozeduren“ („Funktionen“, „Methoden“, „Subroutinen“ etc.)

```
int doSelfSumSquare(int someNumber){
    int a;
    a = someNumber + someNumber;
    a = a * a;
    return a;
}
```

$$f(x) = (x + x)^2$$

```
...
int horst;
int heiner;
int fritz;
horst = 101;
heiner = 2;
fritz = horst + heiner;
horst = 2000;
fritz = doSelfSumSquare(5);
fritz = doSelfSumSquare(heiner);
...
```



```
int doSelfSumSquare(int someNumber){
    int a;
    a = someNumber + someNumber;
    a = a * a;
    return a;
}
```



Zell-Nr (Adresse)	Zell-Name (Variablenname)	Zell-Inhalt
		:
1124		
1125	horst	2000
1126	heiner	2
1127		
1128	fritz	103
		:
2024		
2025		
		:
4027		
4028		int horst;
4029		int heiner;
4030		int fritz;
4029		horst = 101;
4030		heiner = 2;
		:
8756		int a;
8757		a=someNumber+someNumber;
8758		a = a * a;
		:

Daten
Programm

```
...
int horst;
int heiner;
int fritz;
horst = 101;
heiner = 2;
fritz = horst + heiner;
horst = 2000;
fritz = doSelfSumSquare(5);
fritz = doSelfSumSquare(heiner);
...
```



```
int doSelfSumSquare(int someNumber){
    int a;
    a = someNumber + someNumber;
    a = a * a;
    return a;
}
```

Zell-Nr (Adresse)	Zell-Name (Variablenname)	Zell-Inhalt
		:
1124		
1125	horst	2000
1126	heiner	2
1127		
1128	fritz	103
		:
2024		
2025		
		:
4027		
4028		int horst;
4029		int heiner;
4030		int fritz;
4029		horst = 101;
4030		heiner = 2;
		:
8756		int a;
8757		a=someNumber+someNumber;
8758		a = a * a;
		:

```
...
int horst;
int heiner;
int fritz;
horst = 101;
heiner = 2;
fritz = horst + heiner;
horst = 2000;
fritz = doSelfSumSquare(5);
fritz = doSelfSumSquare(heiner);
...
```



```
int doSelfSumSquare(int someNumber){
    int a;
    a = someNumber + someNumber;
    a = a * a;
    return a;
}
```

Zell-Nr (Adresse)	Zell-Name (Variablenname)	Zell-Inhalt
		:
1124		
1125	horst	2000
1126	heiner	2
1127		
1128	fritz	103
		:
2024	someNumber	5
2025	a	
		:
4027		
4028		int horst;
4029		int heiner;
4030		int fritz;
4029		horst = 101;
4030		heiner = 2;
		:
8756		int a;
8757		a=someNumber+someNumber;
8758		a = a * a;
		:

Daten
Programm

```

...
int horst;
int heiner;
int fritz;
horst = 101;
heiner = 2;
fritz = horst + heiner;
horst = 2000;
fritz = doSelfSumSquare(5);
fritz = doSelfSumSquare(heiner);
...
    
```

```

int doSelfSumSquare(int someNumber){
    int a;
    a = someNumber + someNumber;
    a = a * a;
    return a;
}
    
```

• Was ist der Wert der Speicherstelle 2025 nachdem das Statement `a = a * a` ausgeführt wurde?

Zell-Nr (Adresse)	Zell-Name (Variablenname)	Zell-Inhalt
1124		⋮
1125	horst	2000
1126	heiner	2
1127		⋮
1128	fritz	103
		⋮
2024	someNumber	5
2025	a	???
		⋮
4027		
4028		int horst;
4029		int heiner;
4030	A	10
4029	B	undefiniert
4030	C	100
8756		int a;
8757		a=someNumber+someNumber;
8758		a = a * a;

Daten

Programm

```

...
int horst;
int heiner;
int fritz;
horst = 101;
heiner = 2;
fritz = horst + heiner;
horst = 2000;
fritz = doSelfSumSquare(5);
fritz = doSelfSumSquare(heiner);
...
    
```

```

int doSelfSumSquare(int someNumber){
    int a;
    a = someNumber + someNumber;
    a = a * a;
    return a;
}
    
```

• Was ist der Wert der Speicherstelle 2025 nachdem das Statement `a = a * a` ausgeführt wurde?

Zell-Nr (Adresse)	Zell-Name (Variablenname)	Zell-Inhalt
1124		⋮
1125	horst	2000
1126	heiner	2
1127		⋮
1128	fritz	103
		⋮
2024	someNumber	5
2025	a	???
		⋮
4027		
4028		int horst;
4029		int heiner;
4030	A	10
4029	B	undefiniert
4030	C	100
8756		int a;
8757		a=someNumber+someNumber;
8758		a = a * a;

Daten

Programm

```

...
int horst;
int heiner;
int fritz;
horst = 101;
heiner = 2;
fritz = horst + heiner;
horst = 2000;
fritz = doSelfSumSquare(5);
fritz = doSelfSumSquare(heiner);
...
    
```

```

int doSelfSumSquare(int someNumber){
    int a;
    a = someNumber + someNumber;
    a = a * a;
    return a;
}
    
```

Vereinfachtes Speicher-Modell

Zell-Nr (Adresse)	Zell-Name (Variablenname)	Zell-Inhalt
1124		⋮
1125	horst	2000
1126	heiner	2
1127		⋮
1128	fritz	100
		⋮
2024		
2025		
		⋮
4027		
4028		int horst;
4029		int heiner;
4030		int fritz;
4029		horst = 101;
4030		heiner = 2;
		⋮
8756		int a;
8757		a=someNumber+someNumber;
8758		a = a * a;

Daten

Programm

Java.pptx - PowerPoint

DATEI START EINFÜGEN ENTWURF ÜBERGÄNGE ANIMATIONEN BILDSCHIRMPRÄSENTATION ÜBERPRÜFEN ANSICHT Anmelden

Von Ab aktueller Folie Online Benutzerdefinierte Bildschirmpräsentation + Folie ausblenden Neue Anzeigedauern testen Kommentare wiedergeben Anzeigedauern verwenden Bildschirmre

Beginn an Bildschirmpräsentation starten Bildschirmpräsentation einrichten Bildschirmpräsentation aufzeichnen + Mediensteuererelemente anzeigen Einrichten

33 34 35 36 37

Pingo Mini-Aufgabe – <http://pingo.upb.de>

```

...
int horst;
int heiner;
int fritz;
horst = 101;
heiner = 2;
fritz = horst + heiner;
horst = 2000;
fritz = doSelfSumSquare(5);
fritz = doSelfSumSquare(heiner);
...
    
```

```

int doSelfSumSquare(int someNumber){
    int a;
    a = someNumber + someNumber;
    a = a * a;
    return a;
}
    
```

• was passiert nachdem das Statement ausgeführt worden ist?

Zell-Nr (Adresse)	Zell-Name (Variablenname)	Zell-Inhalt
1124		⋮
1125	horst	2000
1126	heiner	2
1127		⋮
1128	fritz	100
		⋮
2024		
2025		
		⋮
4027		
4028		int horst;
4029		int heiner;
4030	A	heiner wird auf 4 gesetzt
4029	B	fritz wird auf 10000 gesetzt
4030	C	heiner wird auf 10000 gesetzt
	D	fritz wird auf 4 gesetzt
8757		a=someNumber+someNumber;
8758		a = a * a;

Klicken Sie, um Notizen hinzuzufügen

FOLE 36 VON 181 ENGLISH (USA) NOTIZEN KOMMENTARE 66% 14:44 13.05.2016

- **Lernziele** der Veranstaltung: **Praktische** Fähigkeit, kleinere Probleme mit Datenbanken und Java lösen zu können + **Grundlagen** für vertiefte Einarbeitung legen.
- **Begrenzte Zeit** in Veranstaltung --> Viele grundlegende Informatikkonzepte können nur **vereinfacht** behandelt werden.
- Die meisten **Bücher** ≈ „Einführung in die Informatik“ haben Informatiker als Zielgruppe --> hier nur eingeschränkt zu empfehlen
- --> **Wikipedia**-Artikel als Hintergrundliteratur. Ergänzend / alternativ: [1] und [2]



The screenshot shows a PowerPoint slide titled "Pingo Mini-Aufgabe" with a URL <http://pingo.upb.de>. The slide contains Java code for a class with a method `doSelfSumSquare`. A red arrow points from a line of code to a table. The table has columns for "Zell-Nr (Adresse)", "Zell-Name (Variablenname)", and "Zell-Inhalt".

Zell-Nr (Adresse)	Zell-Name (Variablenname)	Zell-Inhalt
1124		...
1125	horst	2000
1126	heiner	2
1127		
1128	fritz	100
2024		...
2025		...

Below the table, there are four options (A, B, C, D) for what happens after a statement is executed. Option D is selected with a red circle and arrow.

● was passiert nachdem das Statement ausgeführt worden ist?

A: heiner wird auf 4 gesetzt
 B: fritz wird auf 10000 gesetzt
 C: heiner wird auf 10000 gesetzt
 D: fritz wird auf 4 gesetzt

Empfehlungen zum Lernen

- **Minimaler Anspruch:**
Machen Sie sich die Inhalte der Folien klar!
- **Standard-Vorgehen:**
Minimaler Anspruch + Lesen Sie [1]
- **Interessierte Studierende**
Standard-Vorgehen + Lesen Sie die entsprechenden Abschnitte aus [3a,b,c] oder Kapitel 0, 2 und 6 aus [2].
- **Superinteressierte Studierende:**
Vorgehen für Interessierte + Lesen Sie die restlichen angegebenen Artikel aus [3] und wählen Sie zwei weitere Artikel aus, auf die Sie dabei über Links in [3] stoßen. Alternativ: weitere Abschnitte aus Kapitel 1 - 6 aus [2], die Sie interessieren..

Empfehlungen zum Lernen

- **Minimaler Anspruch:**
Machen Sie sich die Inhalte der Folien klar!
- **Standard-Vorgehen:**
Minimaler Anspruch + Lesen Sie [1]
- **Interessierte Studierende**
Standard-Vorgehen + Lesen Sie die entsprechenden Abschnitte aus [3a,b,c] oder Kapitel 0, 2 und 6 aus [2].
- **Superinteressierte Studierende:**
Vorgehen für Interessierte + Lesen Sie die restlichen angegebenen Artikel aus [3] und wählen Sie zwei weitere Artikel aus, auf die Sie dabei über Links in [3] stoßen. Alternativ: weitere Abschnitte aus Kapitel 1 - 6 aus [2], die Sie interessieren..