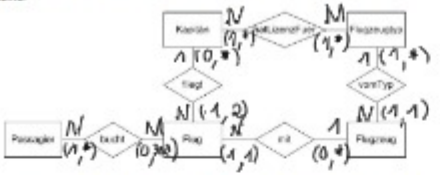
(1) (4 P)

Gegeben sei folgendes unvollständiges **ER-Diagramm** eines **Flughafen-Management-systems.**



Treffen Sie für die Beziehungen **sinnvolle Funktionalitäten** (also 1:1, 1:N, N:M, ...) **UND Angaben** in **(min, max)** Notation ein!

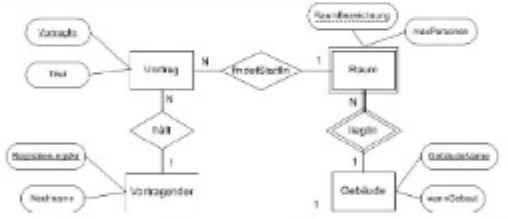
Gehen Sie dabei insbesondere **davon aus**, dass:

- Flugzeuge und / oder Kapitäne geben kann, die für keinen Flug eingesetzt sind.
- für einen Flug noch keine Buchungen vorliegen können.
- die Maximalzahl der Passagiere pro Flug generell auf 300 festgesetzt wurde.
- im System keine lizenzlosen Kapitäne und keine nicht-buchenden Passagiere erfasst werden.

(2) (4 P)

(a) Überführen Sie das folgende ER Diagramm **initial (d.h. OHNE Verfeinerungen)** in ein **relationales Schema** (Typen nicht vergessen!)

(b) Geben Sie im Hinblick auf eine **Verfeinerung** lediglich **an, welche** Relationen **wie** mit welchen anderen Relationen **zusammengefasst** werden können!



1 Vortrag : { VortragNr : Int, Titel : String }

2 Vortragender : { RegistrierungsNr : Int, Nachname : String }

3 hält : { VortragsNr : Int, RegistrierungsNr : Int }

4 Raum : { RaumBezeichng : String, maxPersonen : Int }

5 Gebäude : { Gebäude Name : String, wann Gebaut : Date }

6 liegt In : { RaumBezeichng : String, Gebäude Name : String }

7 findet Statt In : { VortragsNr : Int, RaumBezeichng : String }

1 Vortrag : { VortragNr : Int, Titel : String, RegistrierungsNr : Int, RaumBezeichng : String }

2 Vortragender : { RegistrierungsNr : Int, Nachname : String }

3 ~~hält : { VortragsNr : Int, RegistrierungsNr : Int }~~

4 Raum : { RaumBezeichng : String, maxPersonen : Int, Gebäude Name : String }

5 Gebäude : { Gebäude Name : String, wann Gebaut : Date }

6 ~~liegt In : { RaumBezeichng : String, Gebäude Name : String }~~

7 ~~findet Statt In : { VortragsNr : Int, RaumBezeichng : String }~~

$\{1,2,3\} \times \{4,5\} = \{(1,4), (1,5), (2,4), (2,5), (3,4), (3,5)\}$

SQL

(6 Punkte)

Gegeben sei ein Ausschnitt der aus der Zentralübung bekannten Universitätsdatenbank.

Professoren

PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Studenten

MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

Vorlesungen

VorlNr	Titel	SWS	gelesenVon
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	3	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

voraussetzen

Vorgänger	Nachfolger
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

hören

MatrNr	VorlNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022
29555	5001

Assistenten

PersNr	Name	Fachgebiet	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachanalyse	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Welt	2134

prüfen

MatrNr	VorlNr	PersNr	Note
27550	5001	2137	1
25403	5041	2125	2
27550	4630	2137	2

(1) (3 P)

Geben Sie ein `select` statement an, das **äquivalent** zu folgendem (sehr einfachen !) **relationale Algebra Ausruck** ist!

$$\Pi_{sLVorgaenger}\Big(\sigma_{v.VorlNr=v2.Nachfolger \wedge v2.Vorgaenger=v1.Nachfolger \wedge v.Titel='Bioethik'}\big((\rho_v(Vorlesungen)' \times \rho_{v2}(voraussetzen) \times \rho_{v1}(voraussetzen))\big)\Big)$$

(2) (3 P)

Bestimmen Sie mit einem `select` statement die **Namen** von **Professoren, die in Summe mehr als 5 SWS Lehrveranstaltungen halten!**

1)
```
select v1.Vorgaenger from Vorlesungen v, Voraussetzen v2, Voraussetzen v1
distinct where v.VorlNr = v2.Nachfolger and v2.Vorgaenger = v1.Nachfolger
                                        and v.Titel = 'Bioethik'
```

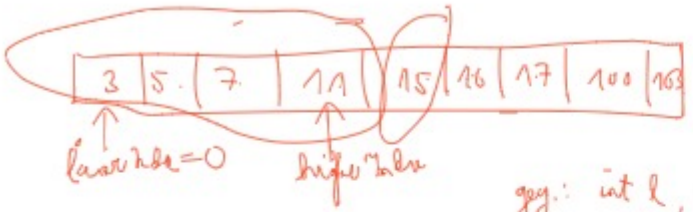
2)
```
select Name from Professoren, Vorlesungen
    where gelesenVon = PersNr
    groupBy gelesenVon, PersNr
    having sum(SWS) > 5
```

```
select p.Name from Professoren p
    where p.PersNr in (select gelesenVon from
                       Vorlesungen groupBy gelesenVon
                       having sum(SWS) > 5)
```

having count(*) > 5

select p.Name from Professoren p
where p.PersNr in (select gelesenVon from Vorlesungen group by gelesenVon having count(*) > 5)

Suche 5



gegeben: int l, int r

int m = (l-r)/2

(l+r)/2

Bsp.:

l = 10 r = 20

m = (10+20)/2 = 30/2 = 15

Datenstrukturen und Algorithmen

(6 Punkte)

(1) (3 P)

Binäre Suche in Java:

Gegeben sei folgende Java Methode, die die **binäre Suche auf einem sortierten Array** implementiert. **Ergänzen** Sie die Methode an den **drei** angegebenen Stellen sinnvoll!

```
public static int locate(int key, int[] a) { //a must be sorted ascendingly
        int lowerIndex = 0;
        int higherIndex = a.length - 1;
        if(key > a[higherIndex])
            return -1; //key is larger than the largest key in a --> treat this case extra
        int middleIndex = 0;
        while(lowerIndex <= higherIndex) { //while we are not finished with searching
            middleIndex = (lowerIndex + higherIndex) / 2; //choose new mid
            if(key < a[middleIndex])
                higherIndex = middleIndex - 1; //continue search in lower half
            else if(key > a[middleIndex])
                lowerIndex = middleIndex + 1; //continue search in upper half
            else
                return middleIndex; //we found it!
        }
        //reaching this part of the code means that a does not contain key
        //--> return index of smallest key k' with
        if(key <= a[middleIndex])
            return middleIndex;
        else
            return middleIndex + 1;
    }
```

(1) (3P)

```
public static int locate(int key, int[] a) { //a must be sorted ascendingly
        int lowerIndex = 0;
        int higherIndex = a.length - 1;
        if(key > a[higherIndex])
            return -1; //key is larger than the largest key in a --> treat this case extra
        int middleIndex = 0;
        while(lowerIndex <= higherIndex) { //while we are not finished with searching
            middleIndex = lowerIndex + (higherIndex - lowerIndex) / 2; //choose new mid
```

l = 10
r = 20

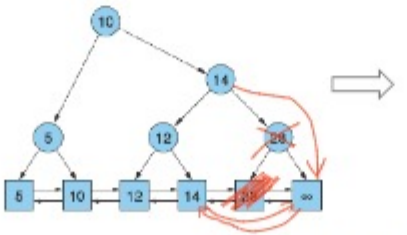
m = l + (r - l)/2 =

= 10 + 10/2 = 10 + 5 = 15

(2) (1.5 P)

Binäre Suchbäume:

Gegeben sei ein **binärer Suchbaum** mit zugehöriger verketteter Liste. **Zeichnen** Sie den Baum **nach** der Operation `remove(28)`!



(Aufgabe 3 geht auf nächster Seite noch weiter)

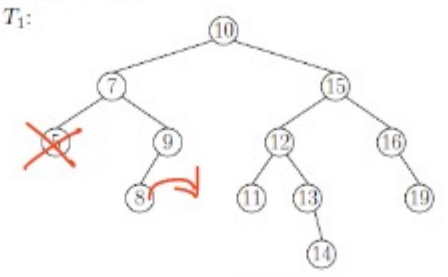
Ein AVL ist ein binärer Suchbaum (d.h. für jeden Knoten ist das linke Kind kleiner und das rechte Kind größer als der Knoten) bei dem für jeden Knoten die Höhen des linken und rechten Teilbaums sich um höchstens 1 unterscheiden.

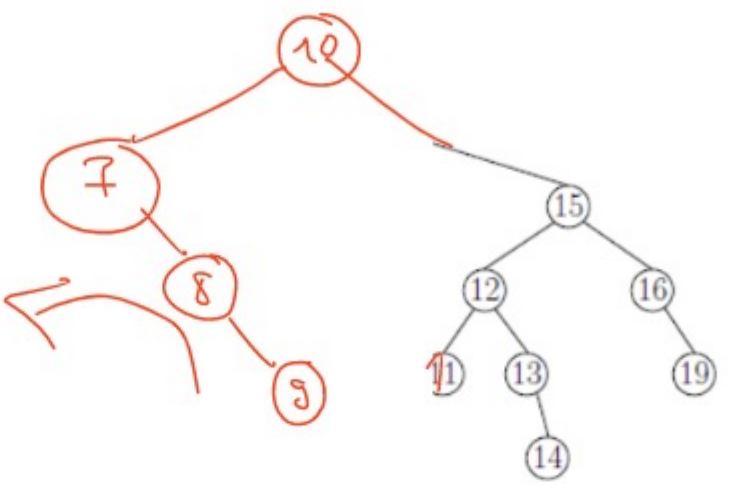
Hausaufgabe 3 (AVL-Bäume): **(3+2+2 Punkte)**

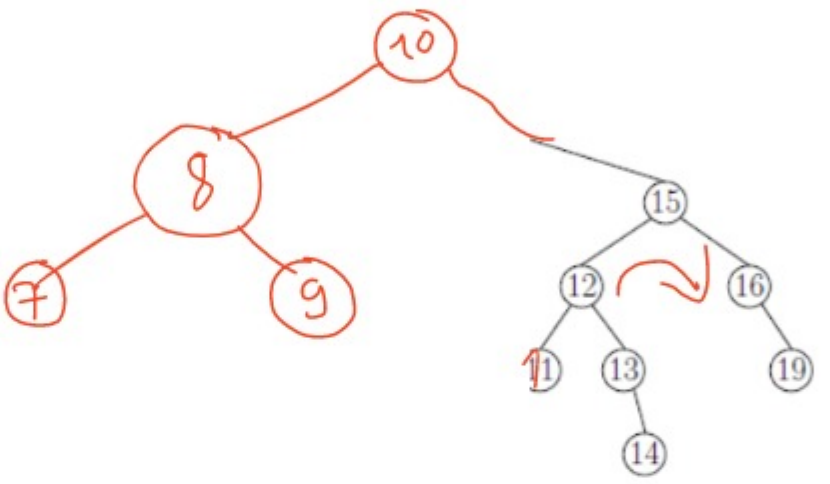
Betrachte in den Aufgabenteilen a) bis d) den Baum, der in der jeweiligen Abbildung dargestellt ist. Führe die Operation des jeweiligen Aufgabenteils und die damit verbundenen Restrukturierungsmaßnahmen zum Erhalt der AVL-Eigenschaft auf dem entsprechenden Baum aus. Zeichne dabei das Resultat nach jeder einzelnen ausgeführten Operation INSERT, DELETE und RESTRUCTURE in einen separaten Baum:

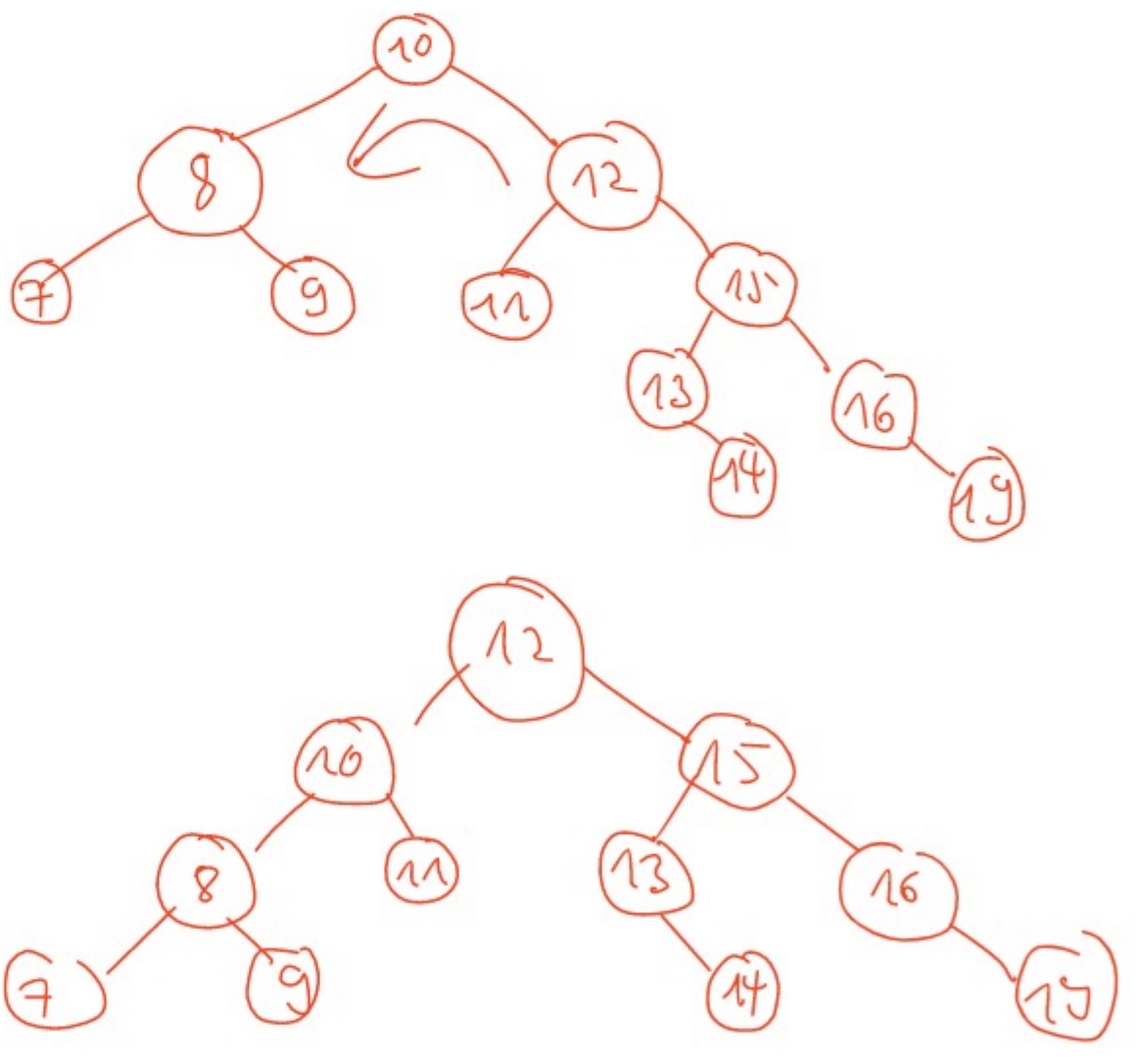
a) DELETE($T_1, 5$)

T_1:



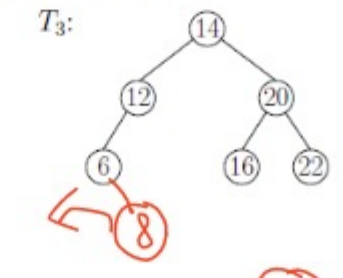


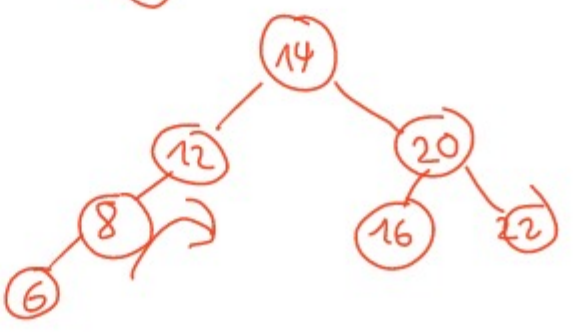


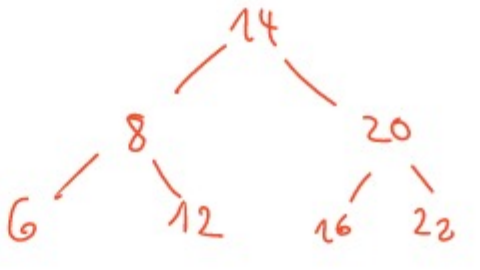


c) INSERT($T_3, 8$)

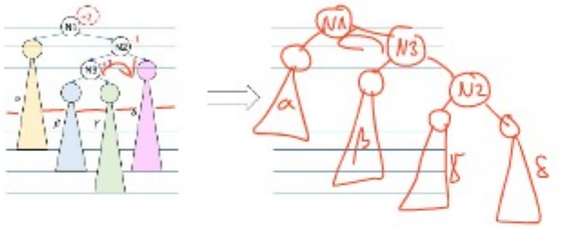
T_3:

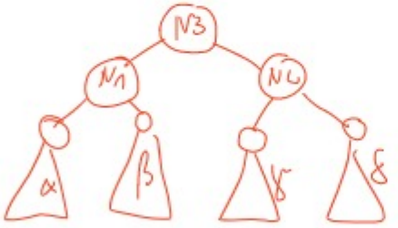




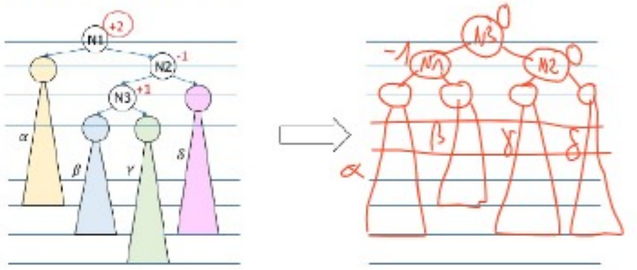


(3) (1.5 P)
AVL Bäume:
In einem AVL Baum ist durch Einfügen in den Teilbaum γ folgender **ungünstiger Zustand entstanden**. **Korrigieren** Sie ihn durch eine **Doppelrotation nach links**! **Geben** Sie auch die neuen **Δh Angaben** für A und B **an**!

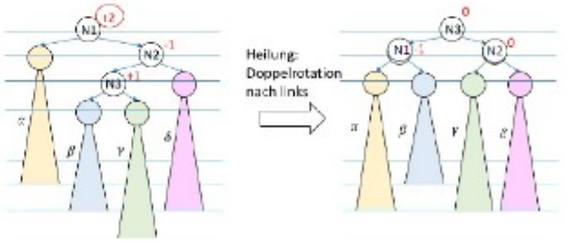




(3) (1.5 P)
AVL Bäume:
In einem AVL Baum ist durch Einfügen in den Teilbaum γ folgender **ungünstiger Zustand entstanden**. **Korrigieren** Sie ihn durch eine **Doppelrotation nach links**! **Geben** Sie auch die neuen **Δh Angaben** für A und B **an**!



(3) (1.5 P)



Aufgabe 6: Java (I)

(7 Punkte)

(1) (5 P)
Gegeben sei folgender **Java Code**:

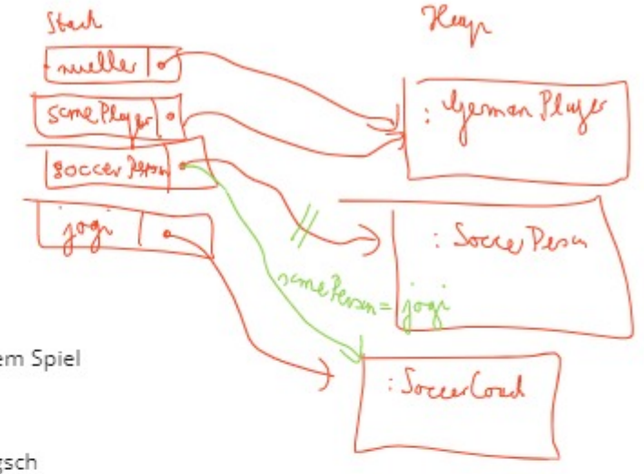
```
public class SoccerDemo {
    public static void main(String[] args) {
        SoccerPlayer somePlayer;
        GermanPlayer mueller = new GermanPlayer();
        PortugesePlayer ronaldo = new PortugesePlayer();
        somePlayer = mueller;
        SoccerPerson somePerson = new SoccerPerson();
        SoccerCoach jogi = new SoccerCoach();
        ronaldo.saySomething();
        mueller.saySomething();
        somePlayer.saySomething();
        somePerson.saySomething();
        somePerson = jogi;
        somePerson.saySomething();
        jogi.saySomething("Hodanoi heut schwitz i abr heffdiggsch!");
    }
}
```

(b) (2 P)
Welche Ausgabe produziert SoccerDemo?

(b) (3 P)
Welche der drei Konzepte **Overloading, Overriding, Polymorphie** werden in obigen Code benutzt, welche nicht?
BEGRÜNDEN Sie jeweils kurz!

```
public interface SoccerPlayer {
        public void saySomething();
}

public class GermanPlayer implements SoccerPlayer{
        public void saySomething(){
                System.out.println("Tooooooooor!");
        }
}

public class PortugesePlayer {
        public void saySomething(){
                System.out.println("Ich I would kiss me if I could.");
        }
}

public class SoccerPerson {
        public void saySomething(){
                System.out.println("Jagutäck... nach dem Spiel ist vor dem Spiel!");
        }
}

public class SoccerCoach extends SoccerPerson{
        public void saySomething(){
                System.out.println("Des isch dess höggschde!");
        }
        public void saySomething(String what){
                System.out.println(what);
        }
}
```

- I would kiss me if I could
- Toooooo...or
- Too... or
- jagutäck... Nach dem Spiel ist vor dem Spiel
- Des isch dess höggschde
- Hlodanoi heut schwitz i abr heffdiggsch



Bis 17:50 Uhr!

Overloading: Wenn eine Methode mit demselben Namen in der gleichen Klasse zweimal vorkommt , aber sich in Anzahl und/oder Typ der Parameter unterscheiden

z.B. kommt Overloading in SoccerCoach vor

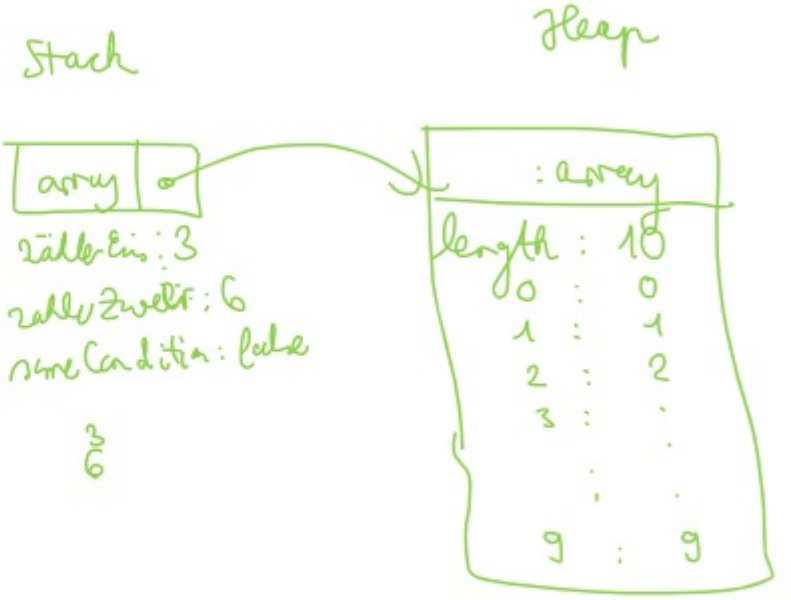
Overriding: Eine Methode von einer Superklasse wird in einer Subklasse überschrieben

z.B. passiert das bei der Methode saySomething() in SoccerPerson und SoccerCoach

(2) (2 P)

Gegeben sei folgender **Java Code**:

```
1  public static void main(String[] args) {
2      int[] array = new int[10];
3      for(int i=0; i<array.length; i++){
4          array[i] = i;
5      }
6      int zaehlerEins = 0;
7      int zaehlerZwei = 0;
8      for(int j=0; j<3; j++){
9          zaehlerEins = zaehlerEins + array[j];
10     }
11     boolean someCondition = (1 == 1);
12     while(someCondition && zaehlerZwei<10){
13         zaehlerZwei++;
14         if(zaehlerZwei > 5)
15             someCondition = false;
16     }
17     System.out.println(zaehlerEins);
18     System.out.println(zaehlerZwei);
   }
```



Welche Ausgabe produziert `main`?
BEGRÜNDEN Sie jeweils ganz kurz!

Aufgabe 7: Java (II)

(7 Punkte)

(1) (2 P)

Gegeben ist folgende unvollständige Java Methode `fibonacciRecursive` zur Berechnung der n-ten Zahl der **Fibonacci-Folge**:

$$f(n) = \begin{cases} 1 & \text{if } n = 1, \\ 1 & \text{if } n = 2, \\ f(n-1) + f(n-2) & \text{if } n > 2. \end{cases}$$

```
public static int fibonacciRecursive(int n) {
    if (n == 1) {
        return 1;
    } else if (n == 2) {
        return 1;
    } else {
        return fib(n-1) + fib(n-2);
    }
}
```

Ergänzen Sie die fehlenden Elemente **sinnvoll**!

$f_1 = 1$
$f_2 = 1$
$f_3 = f_1 + f_2 = 1+1 = 2$
$f_4 = 1+2 = 3$
$f_5 = f_2 + f_4 = 5$
$f_6 = 8$
$f_7 = 13$
$f_8 = 21$
$f_9 =$

(2) (2 P)

Gegeben seien zwei Java Methoden, die die Funktion $f(x) = x^n$ berechnen:

```
public static double potenzMitWhile(double x, int n){
    double result = 1.0;
    int i = 1;
    while (i <= n) {
        result = result * x;
        i++;
    }
    return result;
}

public static double potenzMitFor(double x, int n){
    double result = 1.0;
    for (int i=0; i<n; i++) {
        result = result * x;
    }
    return result;
}
```

Ergänzen Sie die fehlenden Elemente **sinnvoll**!

$2^3 = 2 \cdot 2 \cdot 2 = 8$

$2^5 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 32$