



## Aufgabe 6.1 (Ü) Objekte verstehen

Diskutieren Sie, was in dem folgenden Programm passiert.

```
1 class A {
2     private int i;
3     private int j;
4     private B b;
5     public A(int i, int j, B b){
6         setIandJandB(i, j, b);
7     }
8     public void setIandJandB(int i, int j, B neuesB){
9         i=i;
10        this.j=j;
11        b=neuesB;
12    }
13    public String toString(){
14        return "Toll, ich bin A und habe "+i+" / "+j+" / "+b.toString()
15        ;
16    }
17 class B {
18     private String name;
19     private int i;
20     public B(String name, int i){
21         this.i=i;
22         this.name=name;
23     }
24     public String toString(){
25         return "Wow, ich bin B "+name+" mit i="+i;
26     }
27     public void increase(){
28         i++;
29     }
30     public boolean isSameName(String s){
31         return (s==name);
32     }
33     public boolean isReallyNotSameName(String s){
34         return !(s.equals(name));
35     }
36 }
37 public class OO{
38     public static void main(String [] args){
39         int i=1, j=2;
40         B b = new B("TestB", 0);
```

```

41     A a = new A(i, j, b);
42     System.out.println(a.toString());
43     i++;
44     b.increase();
45     System.out.println(a.toString());
46     b = new B("hollaDieWaldfee", 0);
47     System.out.println(a.toString());
48     String [] s = {"holla", "Die", "Waldfee"};
49     System.out.println(b.isSameName(s[0]+s[1]+s[2]));
50     System.out.println(b.isReallyNotSameName(s[0]+s[1]+s[2]));
51 }
52 }

```

### Aufgabe 6.2 (Ü) Vorlesung

Gegeben sei die Klasse **Student** mit dem privaten Attribut **int id**.

```

class Student{
    private int id;
    public Student(int id){
        this.id = id;
    }
    public int getId(){
        return id;
    }
}

```

- Implementieren Sie die Klasse **Vorlesung**, die maximal  $n$  **Student**-Objekte aufnehmen kann. Die maximale Studentenzahl soll dabei im Konstruktor übergeben werden.
- Erweitern Sie die Klasse **Vorlesung** um eine Methode

```
boolean anmelden(Student s)
```

die einen Studenten **s** zu dieser Vorlesung anmeldet. Dabei soll die Methode **anmelden** den Wert **true** zurückliefern, wenn ein Student für die Vorlesung erfolgreich angemeldet werden konnte. Die Anmeldung desselben Studenten soll dabei keinen Effekt haben. Ansonsten soll die Methode **false** zurückgeben.

- Definieren Sie eine private Methode:

```
private void bereitsAngemeldet(int id)
```

die anzeigt, daß die ID des bereits angemeldeten Studenten registriert ist.

- Modifizieren Sie die Methode **anmelden** so, daß diese die funktion **bereitsAngemeldet** aufruft, falls sich ein bereits für die Vorlesung angemeldeter Student nochmals anmeldet.

- Implementieren Sie zudem in der Klasse **Vorlesung** eine Methode

```
public boolean abmelden(Student s)
```

die einen Studenten **s** von der Vorlesung wieder abmeldet, d.h. der Eintrag des Studenten wird aus der Vorlesung entfernt. Die Methode **abmelden** soll den Wert **true** zurückliefern, wenn ein Student für die Vorlesung erfolgreich abgemeldet werden konnte. Ansonsten soll die Methode **false** zurückgeben.

- Implementieren Sie eine Methode

```
public void printTeilnehmer()
```

die sowohl die Anzahl als auch die IDs aller an der Vorlesung angemeldeten Teilnehmer ausgibt.

**Aufgabe 6.3 (H) Quersumme**

Schreiben sie ein Programm `Quersumme.java`, das die Quersumme von einer ganzen Zahl

- a) *iterativ* in der Funktion `static int quersummeIt(int zahl)`
- b) *rekursiv* in der Funktion `static int quersummeRec(int zahl)`

berechnet und anschließend ausgibt.

**Hinweis:** Der Modulo-Operator ist bei dieser Aufgabe sehr hilfreich!