static - Komponenten

Das Schlüsselwort static steht vor einer Klassenkomponente (Attribut oder Methode) und kennzeichnet diese als zur Klasse gehörig und nicht zu einem bestimmten Objekt.

Was bedeutet das?

1. Die damit bezeichnete Komponente (Attribut oder Methode) kann über den *Klassennamen* angesprochen und genutzt werden, auch ohne existierendes Objekt.
2. Wenn es jedoch Objekte dieser Klasse gibt, dann kann die Komponente auch über den *Objektnamen* angesprochen werden.
3. Ein static-Attribut hat für ALLE Objekte ein und denselben Wert / Inhalt.
4. Wenn ein static-Attribut verändert wird, dann hat es für alle Objekte den veränderten Wert.
5. Static-Komponenten können auch von einer fremden Klasse (ohne Objektreferenz) genutzt werden, weil sie ja über den Klassennamen angesprochen werden.

Syntax:
Klasse . Komponente

```
anzahl++;          // Zugriff in der Klasse
Student.anzahl++;  // Ausserhalb der Klasse
```

Static-Attribute eignen sich zum Beispiel dazu, einen Objektzähler zu implementieren (Beispiel 1).

BEISPIEL 1

```
////////////////////////////
// Zugriff auf ein Klassenattribut
// in einer fremden Klasse.

class Student {

      static  int  anzahl;        // Klassenattribut

      private int  matrikelnr;    // Objektattribute

      public Student() {  // Konstruktor
            matrikelnr = ++ anzahl;
      }
} // end class Student

////////////////////////////////////////////

class Test {

  public static void main(String args[]) {

      Student eins = new Student(); // neues Objekt

      System.out.println(Student.anzahl);  // -> 1
      System.out.println(eins.matrikelnr); // -> 1

  } //end main
} //end class Test
```

static-Methoden

Für static-**Methoden** (Klassenmethoden) gilt das gleiche Prinzip wie für static-Attribute. Static-Methoden können über den Klassennamen aufgerufen werden, ohne dass ein Objekt existieren muss. Static-Methoden werden bspw. verwendet um auf static-Attribute zuzugreifen. **`Main()`** ist eine solche static-Methode, da beim Aufruf durch die Laufzeitumgebung noch kein Objekt dieser Klasse existieren kann.

Klassenmethoden werden nicht mit dem Objektnamen, sondern mit dem Klassennamen angesprochen. Es ist aber auch möglich, static-Methoden über einen Objektnamen anzusprechen, wenn es ein Objekt gibt.

Syntax:
Klasse.Methode();

```
gibAnzahl();             // Aufruf in der Klasse
Student04.gibAnzahl();   // Ausserhalb der Klasse
```

BEISPIEL2

```
////////////////////////////
// Zugriff auf ein statisches Attribut
// in einer fremden Klasse
// mittels einer Klassenmethode

class Student05 {

     ////////////////////////////
     // Klassenattribut

     static  int  anzahl ;

     ////////////////////////////
     // Klassenmethode

     static void gibAnzahl() {

        System.out.println(anzahl);
     }

     ////////////////////////////
     // Konstruktor

     Student05() {

        anzahl++;       // Objektzaehler inkrementieren
     }

} // end class Student05

//////////////////////////////////////////

class Test {

  public static void main(String args[]) {

   Student05 temp1 = new Student05();
```

```
        Student05 temp2 = new Student05();
        Student05 temp3 = new Student05();

        Student05.gibAnzahl(); // Anzahl Objekte ausgeben: 3

      } //end main

    } //end class Test
```

BEISPIEL 3

```
////////////////////////////
// Objektzaehler mit
// static-Attribut

class Student06 {

     int    matrikelnr;   // Attribute
     double einkommen;
     String vorname, nachname;

     static int anzahl;  // Klassenattribut

    public Student06() {
        anzahl++;                    // Objektzähler erhöhen
        this.matrikelnr = anzahl;
        this.einkommen  = 2000;
        this.vorname    = "default";
        this.nachname   = "default";
        System.out.println("Konstruktor aufgerufen...");
     }

     public void gibAus() {  // Objekt-Attribut ausgeben

        System.out.println("Student:    " + vorname +
                           " " + nachname);
        System.out.println("Einkommen:  " + einkommen);
        System.out.println("MatrikelNr: " + matrikelnr);
     }

     public static void gibAnzahl () {  // Klassenmethode
        System.out.println("Es gibt " + anzahl + " Objekte!");
     }

   public static void main(String args[]) {

          Student06[] vektor = new Student06 [10];
          for(int i=0; i <10; i++)
                    vektor[i] = new Student06();

            for(int i=0; i <10; i++)
             vektor[i].gibAus();

             Student06.gibAnzahl(); // Klassenmethode aufrufen
        }
} // end class Student06
```

Nutzung einer static-Methode:

```
////////////////////////////////////////////////////

class Finanzamt {

    static double berechneSteuer(double e) {
           return e * 0.1;
    }
}
////////////////////////////////////////////////////

class Student {

    static int zaehler = 0;  // Objektzaehler

    private String vorname  ="leer";
    private String nachname ="leer";

    private int    matrikel;
    private double einkommen;

    ////////////////////////////////////////////
    // Standard-Konstruktor:

    Student() {

        zaehler++;
        matrikel = zaehler;
        System.out.println("Standardkonstruktor aufgerufen...");
    }

    ////////////////////////////////////////////
    // überladener Konstruktor:

    Student(String vn, String nn) {

        vorname  = vn;
        nachname = nn;
        System.out.println("Konstruktor mit zwei Args...");
    }

    ////////////////////////////////////////////
    // noch ein überladener Konstruktor:

    Student(String vorname, String nn, double e) {

        zaehler++;
        matrikel = zaehler;

        this.vorname  = vorname;
        nachname = nn;
        einkommen= e;
        System.out.println("Konstruktor mit drei Args...");
    }

    ////////////////////////////////////////////////////////////
```

```
    void zeige() {

        System.out.println("Student: "+      nachname + ", " +
                                            vorname  + "\n" +
                           "Matrikel:" +    matrikel + "\n");
    }

    ////////////////////////////////////////////////////////////////
/

    String setze(String vn, String nn, double e) {

        this.zeige();

        if (nn.compareTo(" ")==0)
            return "ungueltig!";
        vorname  = vn;
        nachname = nn;
        einkommen= e;
        return "gueltig!";

     }
     ////////////////////////////////////////////////////////////////

     void steuerErklaerung() {

         double st = Finanzamt.berechneSteuer(einkommen);

        System.out.println("Meine Steuer ist: "+ st);

     }
} // end class

////////////////////////////////////////////////////////////////////

class StudentenSteuer {

    public static void main(String[] xyz ) {

        Student eins = null;
        Student zwei = null;

        Student drei = new Student();  // Standard-Konstruktor
        Student vier = new Student();

        drei.zeige();
        vier.zeige();

        eins = new Student("Hans",   "Mueller"); // Konstruktor 2
        zwei = new Student("Susanne","Meier"  );

        Student fuenf = new Student("Hans", "Meier", 3000.0);

        eins.setze("A","B", 3000);
        zwei.setze("C","D", 4000);

        eins.zeige();
        zwei.zeige();
        drei.zeige();
        vier.zeige();
```

```
            eins.steuerErklaerung();
            zwei.steuerErklaerung();
            drei.steuerErklaerung();
            vier.steuerErklaerung();
        }
    }
```