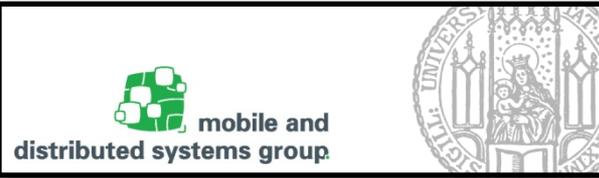




LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN



Teil 2: Weitere Aspekte der Objektorientierung



So wie es Instanzvariablen gibt, die zu einer gewissen Instanz (Objekt) gehören und deren Attribute speichern, so gibt es auch **Klassenvariablen**:

- Speichern Werte für eine ganze Klasse
 - Sprich: Eine Änderung wirkt sich auf alle Objekte der Klasse aus!
- Beschreiben keine objektspezifischen Eigenschaften
- Werden auch als statische Attribute bezeichnet
 - Eingeleitet mit dem Schlüsselwort `static`
 - Bsp.: `static int instances;`
- Statische Attribute können, wie Instanzvariablen auch `public`, `private`, oder `protected` sein
- Auf Klassenvariablen kann (je nach Sichtbarkeit) auch zugegriffen werden, selbst wenn kein Objekt existiert
 - Über: `Klassenname.klassenvariable;`
 - Bsp.: `Auto.instances;`
- Zugriff über ein direktes Objekt ist auch möglich
 - Über: `objektname.klassenvariable;`
 - Bsp.: `fiat.instances;`
 - Sollte allerdings vermieden werden

Ein bekanntes Beispiel für die Anwendung von Klassenvariablen stellt ein Instanz-Zähler dar:

```
// Beispiel für einen Instanz-Zähler über Klassenvariablen

public class Auto {

    //Klassenvariable
    public static int instances;

    //Instanzvariablen
    private String name;
    private int preis;

    //Konstruktor
    public Auto(String name, int preis){

        this.name = name;
        this.preis = preis;

        // Klassenvariable instances wird im Konstruktor erhöht
        Auto.instances++;

    }

    //Weitere Methoden...
}
```

Zugriff auf die Klassenvariable:

```
public class Autohaus {  
    public static void main(String[] args) {  
        System.out.println("Willkommen im Autohaus!");  
        System.out.println(" Es sind bereits "+Auto.instances+" Autos erzeugt worden!");  
        Auto fiatPunto = new Auto("fiatPunto", 9000);  
        Auto mercedes = new Auto("Mercedes",30000);  
        System.out.println("Es sind bereits "+Auto.instances+" Autos erzeugt worden!");  
    }  
}
```

Ausgabe:

Willkommen im Autohaus!

Es sind bereits 0 Autos erzeugt worden!

Es sind bereits 2 Autos erzeugt worden!

Neben den Klassenvariablen gibt es natürlich auch **Klassenmethoden**:

- Gehören zur Klasse und nicht zu den Objekten
 - Können daher **NUR** auf Klassenvariablen arbeiten und **NICHT** auf Instanzvariablen operieren!
 - Beschreiben also kein Objektverhalten und manipulieren daher auch keine Instanzvariablen
 - Können nur Statische Methoden aufrufen
 - Die Verwendung von `this` ist nicht möglich, da kein Objekt zugeordnet ist!
- Können direkt auf Klassen aufgerufen werden
 - auch ohne, dass ein Objekt der Klasse existiert
 - Beispiele:
 - Main-Methode: `public static void main(String[] args)`
 - Wird beim Programmstart aufgerufen, an dem noch keine Objekte existieren
 - `public static int getInstances() {return Auto.instances};`
- Das Aufrufen über Objekte ist auch möglich, aber unerwünscht!

Leichte Veränderung des Beispiels von vorher:

```
// Beispiel für einen Instanzzähler über private Klassenvariablen
```

```
public class Auto {  
    //Klassenvariable  
    private static int instances;  
  
    //Instanzvariablen  
    private String name;  
    private int preis;  
  
    //Konstruktor  
    public Auto(String name, int preis){  
        this.name = name;  
        this.preis = preis;  
  
        // Klassenvariable instances wird im Konstruktor erhöht  
        Auto.instances++;  
    }  
    //Weitere Methoden..  
  
    // Statischer Getter für statische Variable  
    public static int getInstances(){  
        return Auto.instances;  
    }  
}
```

Zugriff auf die Klassenvariable nun über den statischen Getter:

```
public class Autohaus {  
    public static void main(String[] args) {  
        System.out.println("Willkommen im Autohaus!");  
        System.out.println(" Es sind bereits "+Auto.getInstances()+" Autos erzeugt  
worden!");  
        Auto fiatPunto = new Auto("fiatPunto", 9000);  
        Auto mercedes = new Auto("Mercedes",30000);  
        System.out.println("Es sind bereits "+Auto.getInstances()+" Autos erzeugt  
worden!");  
    }  
}
```

Ausgabe:

Willkommen im Autohaus!

Es sind bereits 0 Autos erzeugt worden!

Es sind bereits 2 Autos erzeugt worden!

Ein Anwendungsfall für statische Attribute sind Konstanten, die ihren Initialwert nicht mehr verändern können.

- Beispiele für bekannte Konstanten:
 - `Math.PI`, `Math.E`
- Wir können Konstanten, die eine Klasse anbieten soll selbst definieren:
 - Mit dem Schlüsselwort `final`
 - Sollten nach Konvention in GROßBUCHSTABEN deklariert werden
 - Sollten `public` sein, um direkt von Außen darauf zugreifen zu können

```
// Beispiel für eigene Konstanten

public class MeineKlasse{
    public static final int MAX_VALUE = 10;

    public static final double MEIN_WERT = 1.2;

    public static final double PI = 3.14;

    //Weitere Attribute und Methoden
}
```

```
// Konstanten benutzen

int max_wert = MeineKlasse.MAX_VALUE;

for(int i=0;i<MeineKlasse.MAX_VALUE;i++)
{
    // Do something...
}
```

Mit `final` lassen sich auch Instanzvariablen deklarieren, die nur einmal mit einem Wert belegt werden dürfen, der sich danach nicht mehr verändern lässt

- Das kann sehr nützlich sein, wenn sich Initialbelegungen von Objekten nicht mehr ändern dürfen
- Beispiel: Der Vorname eines Menschen ändert sich nicht mehr.

```
public class Mensch{  
  
    private final String vorname;  
    private String name;  
  
    // Einmalige Belegung durch Konstruktor erlaubt  
    public Mensch(String vorname, String name){  
        this.vorname = vorname;  
        this.name = name;  
    }  
  
    //Aber kein Setter für vorname erlaubt!  
}
```

Nutzen Sie nun Ihre Klasse Student vom Anfang der Stunde und verändern Sie die Klasse wie folgt:

- Fügen Sie einen Instanz-Zähler ein, welcher es mittels einer Klassenvariablen und einer entsprechenden Getter-Methode erlaubt, die Anzahl der erzeugten Instanzen zu zählen und widerzugeben.
- Ändern Sie nun die Instanzvariable „matrikelnr“, so dass diese nur einmal mit einem Wert belegt werden darf.

Prüfen Sie nun in Ihrem Programm (Main-Methode) den Instanz-Zähler und lassen Sie sich die Anzahl der erzeugten Studenten ganz am Anfang und am Ende Ihres Programms ausgeben.

Legen Sie eine neue Klasse MyPoint2D an

- Die Klasse hat zwei Attribute x und y beide vom Typ double
- Ein Punkt soll gleich über den Konstruktor mit x und y initialisiert werden können
- Legen Sie eine Klassenvariable points vom Typ ArrayList<Point2D> an in der alle erzeugten Punkte gespeichert werden
- Schreiben Sie eine Klassenmethode, die für alle Punkte in points, deren x- und y-Werte auf die Konsole schreibt (eine Zeile pro Punkt)
- Erweitern Sie die Klasse um eine Klassen-Methode, die zwei Punkte als Parameter nimmt und den euklidischen Abstand berechnet

Prüfen Sie nun in Ihrem Programm (Main-Methode) indem Sie sich ein paar Punkte anlegen und sich den Abstand ausgeben lassen. Lassen Sie sich außerdem nach jedem neu erzeugten Punkt alle erzeugten Punkte über die Klassenmethode ausgeben.