

### Was bedeutet "this"?

---

-um das Schlüsselwort this zu verstehen, muss man zunächst verstehen, was eine Referenz ist  
-Referenz ist eine Adresse, die zum reservierten Speicherplatz eines Objekts führt  
=> das Prinzip von Referenzen wird im zugehörigen Video sehr detailliert und anschaulich erklärt (falls noch nicht geschehen, unbedingt ansehen)

-das Schlüsselwort this liefert immer die Referenz des Objekts zurück  
-Java weiß durch das Schlüsselwort this in Kombination mit einem Attribut der Klasse sofort, dass es sich um das Attribut der Klasse handelt und nicht um einen Übergabeparameter (siehe Beispiel Set-Methode)  
=> wenn man also innerhalb einer Klasse ein Attribut ansprechen möchte, dann sollte man mit dem Schlüsselwort this arbeiten

```
public class Car {  
  
    String carBrand;  
    int horsepower;  
    int yearOfConstruction;  
    String color;  
    int xPosition;  
    int yPosition;  
  
    public void setHorsePower(int horsepower) {  
        this.horsePower = horsepower;  
    }  
}
```

### Konstruktoren/Methoden überladen

---

-durch das Überladen von Methoden/Konstruktoren kann der Methodenname in einer Klasse mehrfach verwendet werden  
-wenn man den gleichen Methodennamen allerdings zweimal oder mehrmals verwendet, dann muss sichergestellt werden, dass sich die Parameterlisten voneinander unterscheiden

**Hinweis:** In Java kann man eine Methode nicht überladen, indem man unterschiedliche Rückgabewerte verwendet

-im Folgenden siehst du ein Beispiel für das überladen von Konstruktoren (für normale Methoden funktioniert dies analog)

```

public class Car {

    String carBrand;
    int horsepower;
    int yearOfConstruction;
    String color;
    int xPosition;
    int yPosition;

    Car() {
        this.xPosition = 10;
        this.yPosition = 10;
    }

    Car(int x, int y) {
        this.xPosition = x;
        this.yPosition = y;
    }

    Car(int horsepower) {
        this.horsePower = horsepower;
    }
}

```

## Klassenvariablen/Statische Variablen (static)

---

- zwei verschiedene Objekte besitzen keinen gemeinsamen Datenpool
- beide Objekte sind separat abgespeichert und besitzen somit ihre eigenen Attribute
- statische Variablen bzw. Klassenvariablen hingegen sind nur einmal pro Klasse verfügbar  
=> jedes Objekt kann auf diese zugreifen, da diese in einem gemeinsamen Datenpool liegen
- das Schlüsselwort static muss hierfür noch zusätzlich verwendet werden (nach Angabe des Sichtbarkeitsmodifizierers)
- Zugriff auf Klassenvariablen erfolgt wieder über den Punkt Operator  
=> man gibt diesmal aber kein spezielles Objekt an, sondern den Klassennamen, dann den "." und dann den Bezeichner der Klassenvariable
- man kann beispielsweise einen Zähler erstellen, der mitzählt, wie viele Objekte einer Klasse bereits erstellt wurden:

```
public class Program {
```

```
    public static void main(String[] args) {
```

```
        Car car1 = new Car(30, 40);
```

```
        Car car2 = new Car(50, 50);
```

```
        Car car3 = new Car(30, 30);
```

```
        System.out.println(Car.counter);
```

```
    }
```

```
}
```

```
public class Car {
```

```
    public static int counter = 0;
```

```
    String carBrand;
```

```
    int horsepower;
```

```
    int yearOfConstruction;
```

```
    String color;
```

```
    int xPosition;
```

```
    int yPosition;
```

```
    Car(int x, int y) {
```

```
        this.xPosition = x;
```

```
        this.yPosition = y;
```

```
        counter++;
```

```
    }
```

Console

<terminated> Program [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0\_121.jdk/Content

3

## Klassenmethoden/Statische Methoden (static)

---

-das Prinzip von statischen Variablen kann man 1 zu 1 auf statische Methoden ummünzen

-auch statische Methoden werden über den Klassennamen aufgerufen


-statische Methoden sind also auch wieder, genauso wie statische Variablen, keiner bestimmten Instanz zugeordnet, sondern der ganzen Klasse

**Hinweis:** Auch die main-Methode ist eine static-Methode. Muss auch so sein, denn es gibt ja noch überhaupt kein Objekt von der Klasse, in der sich die main-Methode befindet. Diese Methode kann also nur aufgerufen werden, falls es sich um eine statische Methode handelt.

**Wichtig:** Innerhalb von statischen Methoden kann man nur auf statische Attribute zugreifen und **nicht** auf Instanzvariablen.

```
public class Program {  
  
    public static void main(String[] args) {  
  
        Car car1 = new Car(30, 40);  
        Car car2 = new Car(50, 50);  
        Car car3 = new Car(30, 30);  
  
        System.out.println(Car.getCounter());  
  
    }  
}
```

```
public class Car {  
  
    private static int counter = 0;  
  
    String carBrand;  
    int horsepower;  
    int yearOfConstruction;  
    String color;  
    int xPosition;  
    int yPosition;  
  
    Car(int x, int y) {  
        this.xPosition = x;  
        this.yPosition = y;  
        counter++;  
    }  
  
    public static int getCounter() {  
        return counter;  
    }  
}
```

 Console 

<terminated> Program [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0\_121.j

3