

# Objektorientiertes Programmieren 1

## Einstieg in Klassen und Objekte

Stephan Kessler, BSc in Systems Engineering

Kontakt: [stephan.kessler@edu.teko.ch](mailto:stephan.kessler@edu.teko.ch)

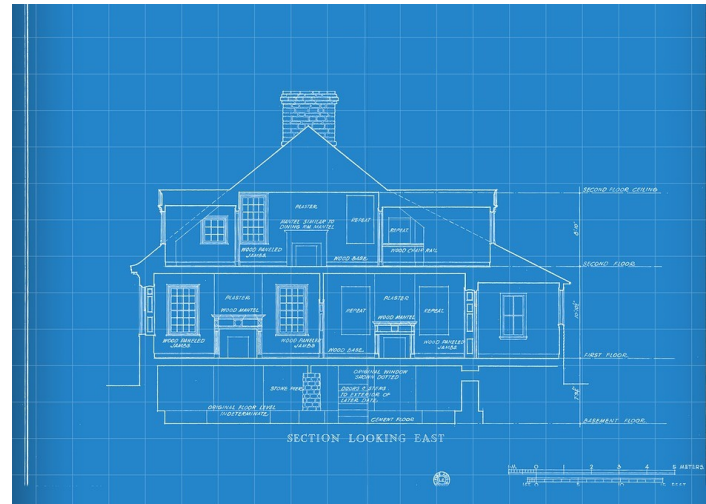


# Vorbereitung

Lesen Sie folgende Abschnitte im Buch ***Sprechen Sie Java?*** als Vorbereitung für dieses Thema. Beachten Sie, dass bei Angaben von Unterkapiteln nur diese auch gelesen werden müssen:

- Klassen (S. 129-142)
- Objektorientierung (S. 145-149)
  - Methoden in Klassen
  - Konstruktoren

# Klassen



- Klassen können analog zum Gebäudebau als **Baupläne** angeschaut werden. Das daraus erbaute Haus wäre ein Objekt.
- Klassen beschreiben den Aufbau eines Objektes.
- Aus einer Klasse können mehrere Objekte erstellt (**instanziert**) werden.
- Eine Klasse kann Variablen und Methoden beschreiben, die ein Objekt nach dem instanziierten besitzt.
- Klassen können auch „eigene“ Variablen und Methoden besitzen. (**static**)

# Klassen

```
public class Hauptklasse
{
    public static void main(String[] args){
        System.out.println("Hallo Welt");
    }
}
```

Betrachten wir wieder unser erstes Programm: „*HelloWorld*“

# Klassen

```
public class Hauptklasse
{
    public static void main(String[] args){
        System.out.println("Hallo Welt");
    }
}
```

Nun können wir alles betrachten. Wir befinden uns nämlich in einer Klasse (hier *Hauptklasse*).

# Klassen

```
public class Hauptklasse
{
    public static void main(String[] args){
        System.out.println("Hallo Welt");
    }
}
```

Genau wie bei den bisherigen Methoden (und teils auch Variablen) fängt bei diesem Beispiel die Klasse mit dem Schlüsselwort *public* an. Dieses Schlüsselwort ist ein **Zugriffsmodifikator**, welcher erst durch die Betrachtung von Klassen und Objekten Sinn ergibt. Damit kann die Sichtbarkeit geregelt werden.

Wir nutzen vorerst weiterhin *public* da wir damit auf Klassen, Methoden und Variablen von überall zugreifen können.

Auf die Zugriffsmodifikatoren wird im Fach OOP2 noch weiter eingegangen.

# Klassen

```
public class Hauptklasse
{
    public static void main(String[] args){
        System.out.println("Hallo Welt");
    }
}
```

Mit dem Schlüsselwort *class* wird angegeben, dass es sich bei diesem Aufbau um eine Klasse handelt.

# Klassen

```
public class Hauptklasse
{
    public static void main(String[] args){
        System.out.println("Hallo Welt");
    }
}
```

Als dritte Komponente kann der Name der Klasse angegeben werden. (In diesem Beispiel *Hauptklasse*).



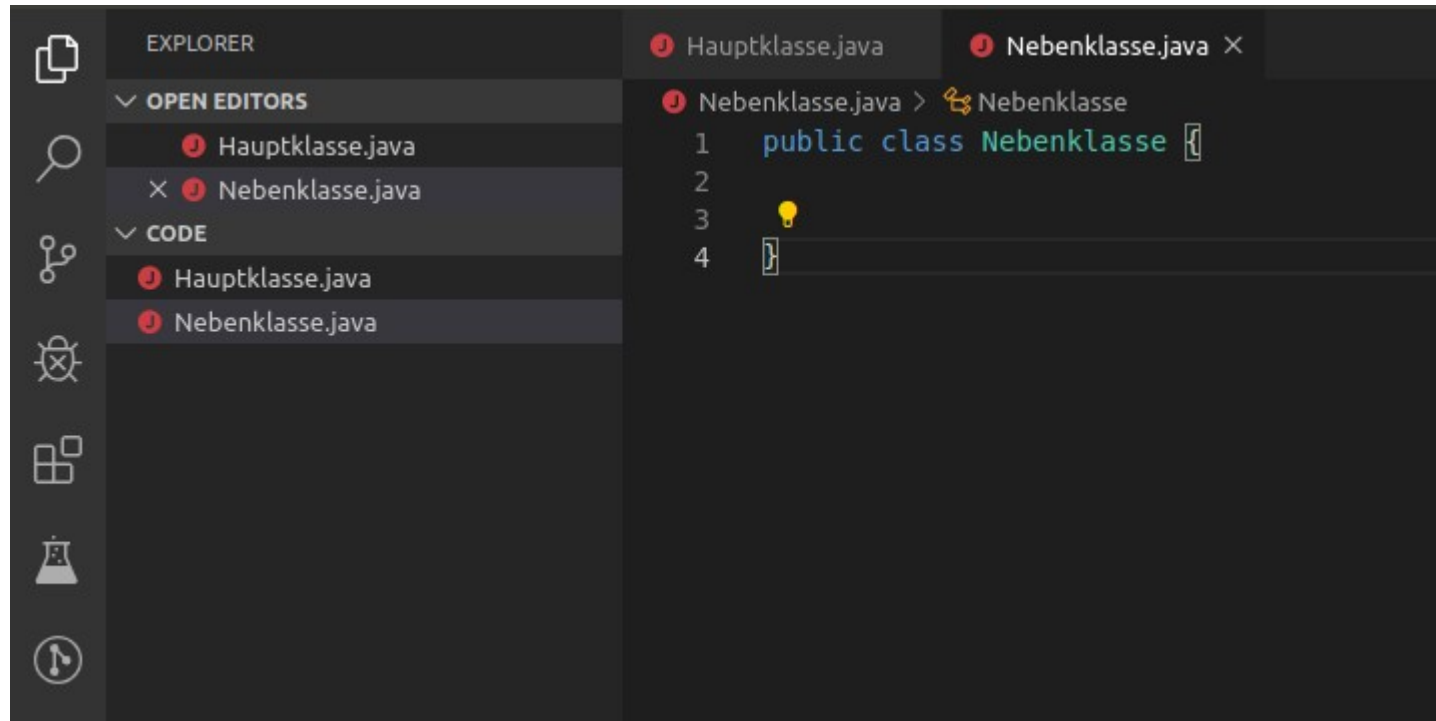
# Klassen

```
public class Hauptklasse
{
    public static void main(String[] args){
        System.out.println("Hallo Welt");
    }
}
```

Da wir nun auch Klassen betrachten, können wir jetzt auch das Schlüsselwort *static* bei den Methoden anschauen. *static* bedeutet hier, dass die Methode *main* eine „Klassenmethode“ ist und sie sich somit auf die Klasse und nicht auf ein aus dieser Klasse erstelltes Objekt bezieht.

Daher kann diese Methode über die Klasse und nicht über ein instanziiertes Objekt zugegriffen werden.

# Klassen



Um das zu veranschaulichen, können wir eine weitere Klasse in unserem Projekt erstellen (im Beispiel: Nebenklasse). In VS Code wird dafür einfach eine neue Java-Datei im Projektordner angelegt und entsprechend beschrieben.

# Klassen

```
public class Nebenklasse  
{  
    public static String text;  
}
```

In dieser neuen Klasse können wir, wie wir das schon bei den Methoden gesehen haben, eine Variable auf *static* setzten.

# Klassen

```
public class Hauptklasse
{
    public static void main(String[] args){
        System.out.println("Hallo Welt");
        Nebenklasse.text = "Text";
    }
}
```

In der Hauptklasse können wir dann auf diese Variable zugreifen.

# Klassen

```
public class Hauptklasse
{
    public static void main(String[] args){
        System.out.println("Hallo Welt");
        Nebenklasse.text = "Text";
    }
}
```

Dazu wird der Name der Klasse und der Name der Variable verwendet und durch einen Punkt abgetrennt.

# Klassen

```
public class Nebenklasse
{
    public static String text;

    public static void schreibeText(){
        System.out.println(text);
    }
}
```

Wir können dies auch mit *static*-Methoden durchführen. (hier *schreibeText*)

# Klassen

```
public class Hauptklasse
{
    public static void main(String[] args){
        System.out.println("Hallo Welt");
        Nebenklasse.text = "Text";
        Nebenklasse.schreibeText();
    }
}
```

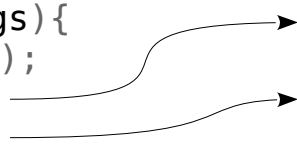
Das Ausführen in der anderen Klasse wird auf die selbe Art wie das Zugreifen auf Variablen durchgeführt.

# Klassen

```
public class Hauptklasse
{
    public static void main(String[] args){
        System.out.println("Hallo Welt");
        Nebenklasse.text = "Text";
        Nebenklasse.schreibeText();
    }
}

public class Nebenklasse
{
    public static String text;

    public static void schreibeText(){
        System.out.println(text);
    }
}
```



Die einzelnen Aufrufe aus der *Hauptklasse* sind dabei wie oben dargestellt mit den Elementen aus der *Nebenklasse* verbunden.

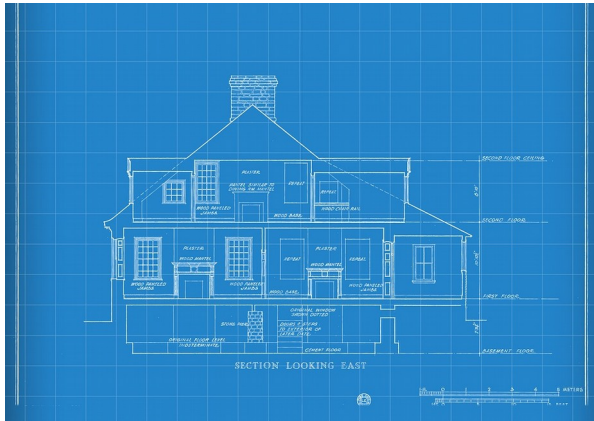




# Aufgabe

- Erstellen Sie ein neues Projekt.
- In diesem Projekt erstellen Sie zwei Klassen.
- In der ersten Klasse programmieren Sie wie gewohnt die *main*-Methode.
- In der zweiten Klasse programmieren Sie zwei Float-Variablen
- Programmieren Sie nun eine Methode (*static*) in der zweiten Klasse. Diese Methode soll die beiden Float-Variablen zusammen addieren und an die Konsole ausgeben. Da die Variablen in der selben Klasse sind, müssen Sie nicht durch den Punkt auf diese zugreifen (z.B. anstelle *Klasse2.float\_variable1* direkt *float\_variable1*)
- Füllen Sie in der ersten Klasse die beiden Float-Variablen aus der zweiten Klasse mit einem Wert und führen Sie die Methode aus der zweiten Klasse aus.

# Objekte



- Wenn Klassen analog als Bauplan angeschaut werden können, sind Objekte die errichteten **Gebäude**.
- Objekte werden aus Klassen erzeugt (**instanziiert**)
- Die nicht statischen Eigenschaften (ohne *static*) einer Klasse werden vom instanziierten Objekt aufgenommen.
- Der Typ eines Objektes wird durch dessen Klasse bestimmt (in Java wird der Name der Klasse verwendet)
- Es können mehrere Objekte aus einer Klasse instanziiert werden.
- Variablen die zu einem Objekt gehören, werden **Attribute** oder **Eigenschaften** genannt.

# Objekte

```
public class Nebenklasse  
{  
    public String text;  
}
```

Möchten wir ein Objekt instanziiieren, welches eine Variable (**Attribut**) besitzt, können wir das gleich wie beim Klassen-Beispiel machen. Beachten Sie jedoch, dass das Schlüsselwort *static* fehlt, da wir nun nicht mehr Klassenbezogen die Variable nutzen wollen.

Der Zugriff aus der Hauptklasse mit *Nebenklasse.text* ist damit nicht mehr möglich.

# Objekte

```
public class Hauptklasse
{
    public static void main(String[] args){
        Nebenklasse objekt1;
        objekt1 = new Nebenklasse();
    }
}
```

In der Hauptklasse können wir nun die Nebenklasse als Objekt instanziiieren.

# Objekte

```
public class Hauptklasse
{
    public static void main(String[] args){
        Nebenklasse objekt1;
        objekt1 = new Nebenklasse();
    }
}
```

Dazu deklarieren wir eine Variable mit dem Typ der Klasse. Diese Variable wird benötigt um nach der Instanziierung weiterhin auf das Objekt zugreifen zu können.

# Objekte

```
public class Hauptklasse
{
    public static void main(String[] args){
        Nebenklasse objekt1;
        objekt1 = new Nebenklasse();
    }
}
```

Mit *new Nebenklasse();* wird nun ein Objekt aus der Klasse instanziiert. Beachten Sie, dass Sie nach dem Namen der Klasse Klammern schreiben. Warum diese gebraucht werden, wird im Abschnitt *Konstruktor* erklärt.

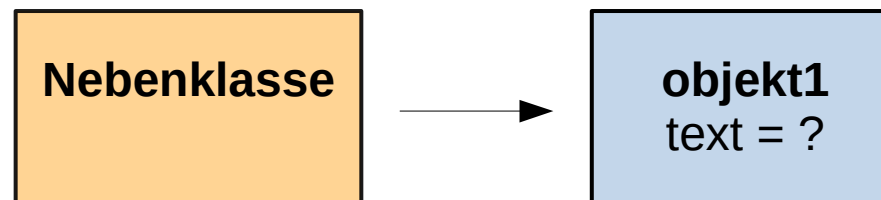
# Objekte

```
public class Hauptklasse
{
    public static void main(String[] args){
        Nebenklasse objekt1;
        objekt1 = new Nebenklasse();
    }
}
```

Die Referenz des Objektes wird dann direkt in die Variable *objekt1* des Typs *Nebenklasse* abgespeichert. Etwas anders ausgedrückt (aber nicht ganz korrekt) kann man sagen, dass das Objekt in *objekt1* abgespeichert ist.

# Objekte

```
public class Hauptklasse
{
    public static void main(String[] args){
        Nebenklasse objekt1;
        objekt1 = new Nebenklasse();
    }
}
```

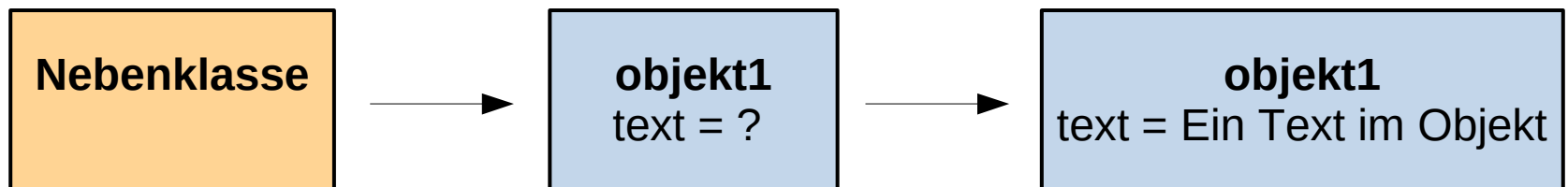


Somit wurde aus der Klasse *Nebenklasse* ein Objekt (*klasse1*) instanziiert.



# Objekte

```
public class Hauptklasse
{
    public static void main(String[] args){
        Nebenklasse objekt1;
        objekt1 = new Nebenklasse();
        objekt1.text = "Ein Text im Objekt";
    }
}
```



Möchte man auf die Variable zugreifen, kann dies ähnlich wie bei den Klassen über den Punkt ausgeführt werden. Jedoch wird hier anstelle der Name der Klasse, der Name der Variable verwendet (hier *objekt1*).

# Objekte

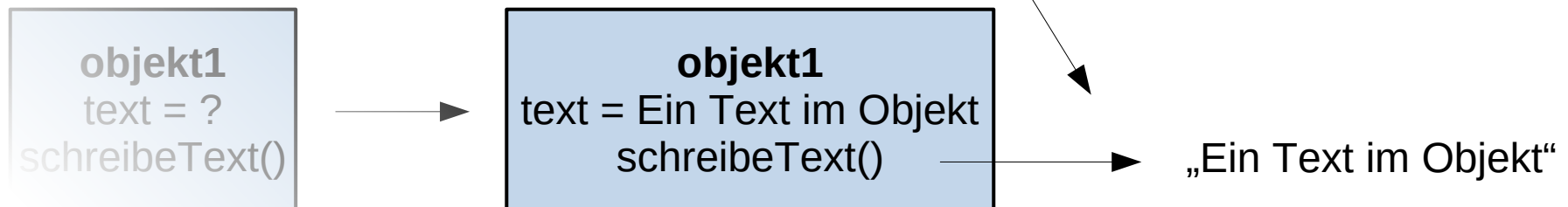
```
public class Nebenklasse
{
    public String text;

    public void schreibeText(){
        System.out.println(text);
    }
}
```

Methoden können auch Objekt-bezogen verwendet werden. Dazu schreiben Sie die Methode ohne *static* in der Klasse hin. In diesem Beispiel wird nun mit der Methode *schreibeText* der Text aus der Variable *text* ausgegeben.

# Objekte

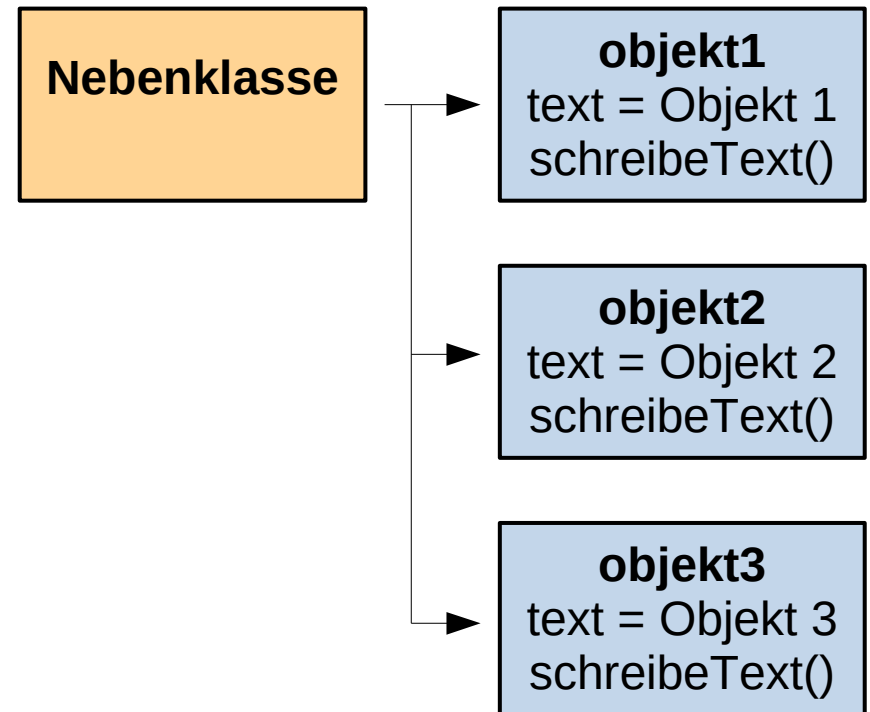
```
public class Hauptklasse
{
    public static void main(String[] args){
        Nebenklasse objekt1;
        objekt1 = new Nebenklasse();
        objekt1.text = "Ein Text im Objekt";
        objekt1.schreibeText();
    }
}
```



Und wieder können wir ähnlich wie bei den Klassen die Methode über einen Punkt nach dem Namen der Variable aufrufen.

# Objekte

```
public class Hauptklasse
{
    public static void main(String[] args){
        Nebenklasse objekt1;
        Nebenklasse objekt2;
        Nebenklasse objekt3;
        objekt1 = new Nebenklasse();
        objekt2 = new Nebenklasse();
        objekt3 = new Nebenklasse();
        objekt1.text = "Objekt 1";
        objekt2.text = "Objekt 2";
        objekt3.text = "Objekt 3";
        objekt1.schreibeText();
        objekt2.schreibeText();
        objekt3.schreibeText();
    }
}
```



Der Vorteil von Objekten gegenüber dem Klassen-bezogenen Zugriff ist, dass man beliebig viele Objekte instanziiieren kann und diese die gleichen Eigenschaften besitzen, **die jedoch auf die Objekte bezogen sind**.

Daher wird bei diesem Beispiel „Objekt 1“, „Objekt 2“, „Objekt 3“ jeweils ausgegeben.



# Aufgabe

- Erstellen Sie ein neues Projekt.
- Erstellen Sie eine Klasse *Person* welche folgende Variable besitzt: Name (String), Alter (Integer), Grösse (Float). Schreiben Sie zudem eine Methode in die Klasse, die, die Variablen formatiert ausgeben.
- Instanziiieren Sie 5 Personen-Objekte in einer Hauptklasse mit einer *main*-Methode.
- Suchen Sie sich 5 Personen aus der Vorlesungs-Klasse aus und nehmen Sie die Werte für die Variablen auf. (Grösse in Meter)
- Geben sie nun alle 5 Daten der Personen mit der Objekt-Methode aus.

# Array mit Objekte

```
public class Hauptklasse
{
    public static void main(String[] args){
        int size = 10;
        Nebenklasse[] objekte = new Nebenklasse[size];

        for(int i = 0; i < size; i++){
            objekte[i] = new Nebenklasse();
            objekte[i].text = "Objekt " + i;
        }

        objekte[5].schreibeText();
        objekte[8].schreibeText();
    }
}
```

Sie können auch ein Array wie gewohnt erzeugen, welches nun die Objekte als Elementen besitzen kann.



# Aufgabe

- Nehmen Sie Ihr letztes Programm (Personen) zur Hand.
- Verändern Sie dieses, so, dass die Objekte nun in einem Array gehalten werden.
- Erzeugen Sie nun anstelle von 5 Person-Objekte 10 und nehmen Sie zu den 5 weiteren Objekten noch die Fehlenden Daten von weiteren 5 Personen auf.
- Rechnen Sie nun das Durchschnittsalter und die Durchschnittsgrösse der 10 Personen mit Hilfe des Arrays aus.
- Geben Sie alle Personen aus, die über dem Durchschnittsalter und unter der Durchschnittsgrösse liegen. Dies sollen Sie wieder mit Hilfe des Arrays erreichen.

# Objekte als Parameter

```
public class Hauptklasse
{
    public static void main(String[] args){
        Nebenklasse objekt1 = new Nebenklasse();
        erweitereText(objekt1);
        objekt1.schreibeText();
    }

    public static void erweitereText(Nebenklasse objekt){
        objekt.text = objekt.text + "noch mehr";
    }
}
```

Sie können auch Objekte als Parameter bei einer Methode übergeben. Dazu wird als Typ der Name der Klasse verwendet.

Beachten Sie, dass im Gegensatz zu den primitiven Datentypen (*int*, *float*, *char* usw.) die Objekte **nicht kopiert** werden. Dadurch bleiben Änderungen bezogen auf das Objekt innerhalb der Methode auch nach dem Verlassen der Methode erhalten. (**call by reference**).



# Konstruktor

```
public class Nebenklasse
{
    public String text;

    public Nebenklasse(){
        text = "Anfangstext";
    }
}
```

Ein Konstruktor kann in einer Klasse beschrieben werden und ist ähnlich wie eine Methode aufgebaut. Speziell am Konstruktor ist, dass dieser **am Anfang beim Instanzieren** eines Objektes ausgeführt wird.

Dadurch hat man die Möglichkeit während der Instanziierung das Objekt „vorzubereiten“.

# Konstruktor

```
public class Nebenklasse
{
    public String text;

    public Nebenklasse(){
        text = "Anfangstext";
    }
}
```

**Kriterium 1:** Um in Java einen Konstruktor zu beschreiben, muss der Name der Klasse verwendet werden.

# Konstruktor

```
public class Nebenklasse
{
    public String text;

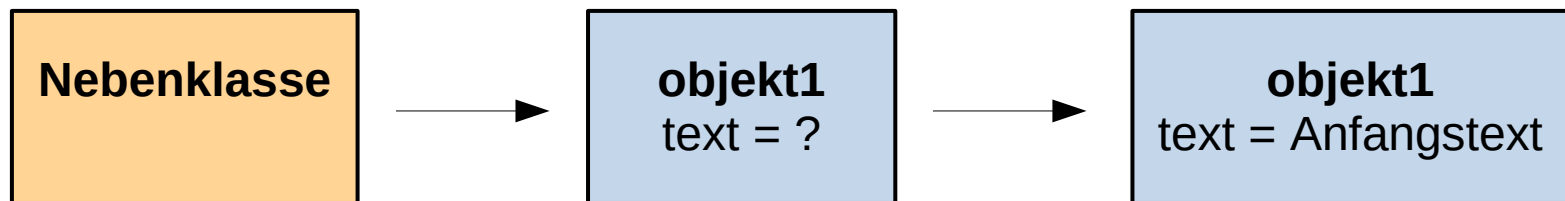
    public Nebenklasse(){
        text = "Anfangstext";
    }
}
```

**Kriterium 2:** Ausserdem wird das Schlüsselwort für den Rückgabewert ausgelassen (kein *void*, *int*, *float* usw.).

Sind diese **beiden Kriterien erfüllt**, handelt es sich um einen Konstruktor.

# Konstruktor

```
public class Hauptklasse
{
    public static void main(String[] args){
        Nebenklasse objekt1;
        objekt1 = new Nebenklasse();
        System.out.println(objekt1.text)
    }
}
```



Wird nun in diesem Beispiel ein Objekt Nebenklasse instanziiert, wird der Konstruktor ausgeführt.

Dadurch besitzt der String in diesem Objekt automatisch den Anfangswert „Anfangstext“.

Im oberen Beispiel wird "Anfangstext" an die Konsole ausgegeben.

# Konstruktor

```
public class Nebenklasse
{
    public String text;

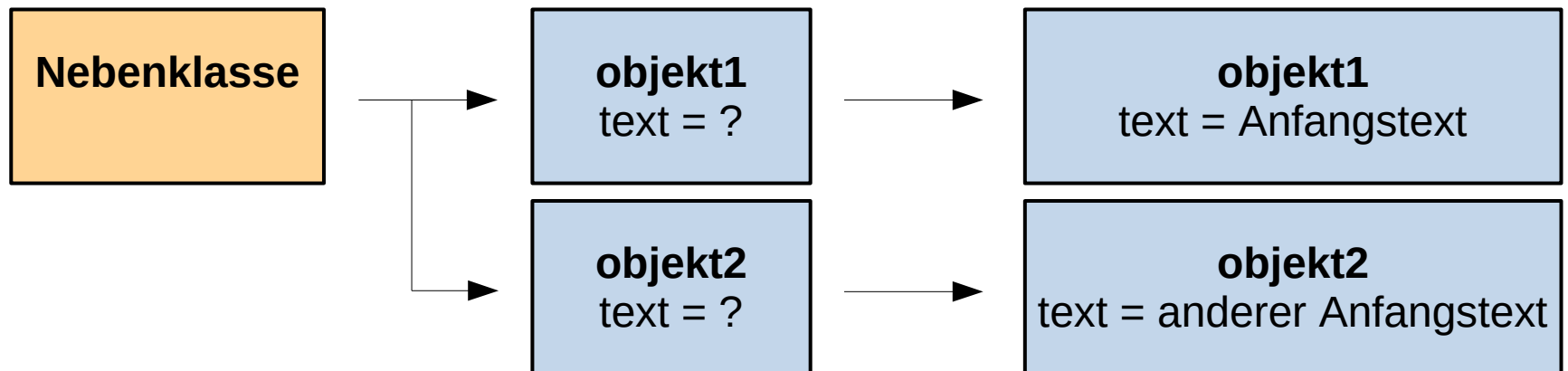
    public Nebenklasse(){
        text = "Anfangstext";
    }

    public Nebenklasse(String anfangstext){
        text = anfangstext;
    }
}
```

Sie können auch Konstruktoren mit Parametern schreiben. Dadurch können Sie auch, wie in diesem Beispiel, mehrere Konstruktoren in der selben Klasse besitzen.

# Konstruktor

```
public class Hauptklasse
{
    public static void main(String[] args){
        Nebenklasse objekt1 = new Nebenklasse();
        Nebenklasse objekt2 = new Nebenklasse("anderer Anfangstext");
        System.out.println(objekt1.text);
        System.out.println(objekt2.text);
    }
}
```



Beim Instanzieren können Sie die Parameter in den Klammern setzen. In diesem Beispiel wird beim Objekt *objekt1* der erste Konstruktor und beim Objekt *objekt2* der zweite Konstruktor verwendet.

Daher brauchen Sie beim Instanzieren die Klammern.



# Aufgabe

- Nehmen Sie wieder Ihr letztes Programm (Personen) zur Hand.
- Erweitern Sie nun Ihr Programm, so dass Sie die drei Werte der Personen über einen Konstruktor des Objektes setzen.