Objektorientierte Modellierung und Programmierung

Klasse und Objekt

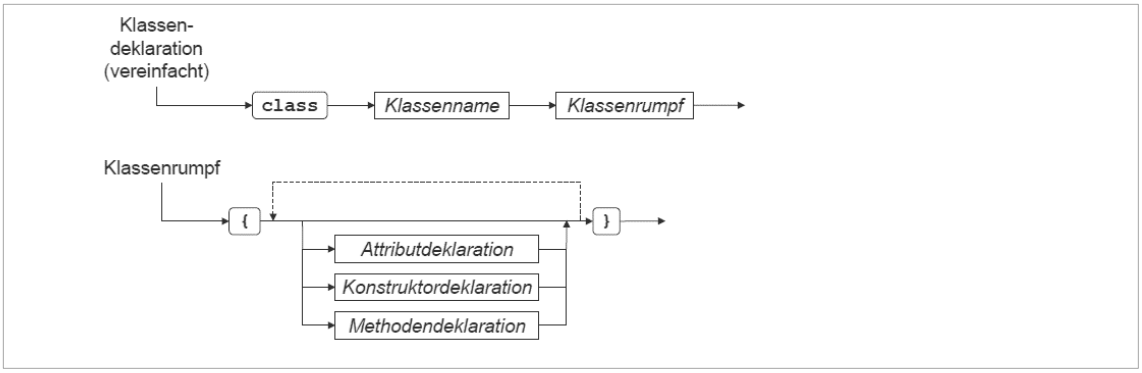
- Klasse (auch: Objekttyp, engl.: class, object type): Konzept der objektorientierten Modellierung und Programmierung, mit dem neue Datentypen definiert werden können.
- Die Deklaration einer Klasse stellt damit eine Art Bauanleitung oder Schablone für sog. Objekte (auch: Exemplare, Instanzen) dieser Klasse dar.
- Konkret wird in der Klassendeklaration festgelegt:
  - aus welchen Datenkomponenten (sog. Attributen), ein Objekt besteht,
  - welche Operationen (sog. Methoden), z. B. zur Manipulation der zugehörigen Attributwerte, zur Verfügung stehen und
  - welche sog. Konstruktoren zur Erzeugung von Objekten (man sagt auch: Instanziierung) verwendet werden können

Beispiel: Mitglieder einer Bibliothek

```
class BibMitglied {
        // --- Attribute ---
        String name;
        String privAnschrift;
        int mitgliedsNr;
        // --- Konstruktor ---
        BibMitglied(String n, String pA, int mNr) {
                name = n;
                privAnschrift = pA;
                mitgliedsNr = mNr;
        }
        // --- Methoden ---
        // --- Getter und Setter ---
        String getName() {
                return name;
        }
        void setName(String n) {
                name = n;
        }
        String getPrivAnschrift() {
                return privAnschrift;
        }
        void setPrivAnschrift(String pA) {
                privAnschrift = pA;
        }
        int getMitgliedsNr() {
                return mitgliedsNr;
        }
        void setMitgliedsNr(int mNr) {
                mitgliedsNr = mNr;
        }
        // andere Methoden
        void print() {
                System.out.println("Mitgliedsnr.: " + mitgliedsNr);
                System.out.println("Name: " + name);
                System.out.println("Anschrift: " + privAnschrift);
        }
}
```

Deklaration von Klassen

- Klassendeklaration legt den Namen einer Klasse sowie ihre Attribute, Konstruktoren und Methoden fest und definiert damit einen neuen Datentyp.
- Sie hat folgenden Aufbau:



Deklaration von Attributen und Methoden

- Attribute sind die Datenkomponenten, aus denen die Objekte einer Klasse bestehen.
  - Deklaration und Benennung erfolgen analog zu Variablen.
  - Erlaubt sind Attribute primitiven und zusammengesetzten Typs
  - Beispiele: int mitgliedsNr; Datum geburtsDatum; Buch[] entlieheneBuecher;
- Methoden (Deklaration und Benennung) sind Ihnen schon bekannt.
  - Im Methodenrumpf kann auf die Attribute eines Objekts zugegriffen werden.
  - Später: Unterschied zwischen Methoden mit/ohne static

Benennung von Klassen

- Klassennamen
  - sollten selbsterklärend sein und die Lesbarkeit eines Programms fördern, z. B. Spielfeld
  - sollten Substantive sein und mit einem Großbuchstaben beginnen

Deklaration von Konstruktoren

- sind Operationen, die es ermöglichen, Objekte einer Klasse zu instanziieren, d. h. anzulegen o. zu erzeugen.
- Alle Konstruktoren einer Klasse haben denselben Namen, nämlich den der Klasse; zusätzlich können sie Parameter haben.
- Ein Rückgabetyp wird nicht angegeben und sie enthalten damit nur optional eine return-Anweisung.
- Beispiel:

```
BibMitglied(String n, String pA, int mNr) {
        name = n;
        anschrift = pA;
        mitgliedsNummer = mNr;
}
```

Man unterscheidet zwischen zwei Arten von Konstruktoren

- explizite Konstruktoren
  - Sie werden bei der Klassendeklaration angegeben und i. d. R. dazu genutzt, die Attributwerte des zu erzeugenden Objekts auf für den Anwendungszweck sinnvolle Startwerte zu setzen.
  - Explizite Konstruktoren können Parameter haben.
  - Eine Klasse kann mehrere explizite Konstruktoren haben, die aber unterschiedliche Parameterlisten haben müssen (entscheidend sind dabei unterschiedliche Typen und nicht nur unterschiedliche Parameterbezeichner!)
- impliziter Konstruktor (auch: Standard-Konstruktor, Default-Konstruktor)
  - Wird kein expliziter Konstruktor bei der Klassendeklaration angegeben, so legt der Übersetzer automatisch einen impliziten Konstruktor an.
  - Ein impliziter Konstruktor hat immer die Form <Klassenname>() { } und damit keine Parameter

Objektorientiertes Programm in Java (vom Typ Applikation)

- ...besteht aus Folge von Klassendeklarationen, die jeweils Attribut-, Konstruktor- und Methodendeklarationen enthalten.
- In einer der Klassen muss die Methode main deklariert sein, mit der die Ausführung des Gesamtprogramms startet.
- Klasse, die die Methode main deklariert, nimmt i. d. R. Sonderrolle ein, nämlich die der Spezifikation des Hauptprogramms und nicht die der Deklaration des Datentyps einer Klasse von Objekten.
- In der Regel beginnt dort die Erzeugung der für die Problemlösung relevanten Objekte anhand der Klassendeklarationen.
- Diese Objekte verwalten eigene Daten (Werte der Attribute) und sie reagieren auf Nachrichten (Aufrufe von Methoden, die in der Klasse des Objekts deklariert wurden).
- Problemlösung durch schrittweise Objekterzeugung, -verknüpfung und -kommunikation.

Objekt

- Objekt ist Datenstruktur, deren Datentyp (Struktur, Verhalten) durch eine Klasse festgelegt ist.
- Objekt ist konkrete Ausprägung einer Klasse, deshalb auch Instanz (engl. instance) genannt.
- Klassendeklaration legt fest, welche Attribute und Methoden für ein Objekt der Klasse zur Verfügung stehen und in welcher Form eine Instanziierung stattfinden kann.
- Während eine Klasse die Attribute i. d. R. nur deklariert, werden bei einem Objekt die Attribute durch den verwendeten Konstruktor oder durch Methodenaufrufe mit Attributwerten belegt.
- Die aktuelle Belegung der Attribute eines Objekts mit Attributwerten definiert den aktuellen Zustand des Objekts.

Objektvariablen

- Attribute von primitivem Typ sind „Behälter im Objekt“. Sie verhalten sich wie die Ihnen bekannten Variablen.
- Attribute vom Typ einer Klasse (sog. Objektvariablen) können einen Verweis/eine Referenz auf ein Objekt enthalten.
    - Bei der Deklaration einer Objektvariable wird eine Variable angelegt, die einen Verweis (auch: Referenz, Zeiger) auf ein Objekt der zugehörigen Klasse aufnehmen kann (deshalb wird solche Variable auch als Referenzvariable bezeichnet; vgl. „Referenzsemantik“)
    - Es wird bei der Deklaration noch keine Objektinstanz erzeugt.
    - Wert der neu angelegten Variable ist eine sog. Null-Referenz
        - zeigt an, dass die Objektvariable (noch) nicht auf ein Objekt verweist.
        - Schlüsselwort null (z. B. für Vergleiche und Wertzuweisungen).

Instanziierung von Objekten

- Instanziierung:
    - bezeichnet die Erstellung eines neuen Objekts einer Klasse.
    - Objekt wird erzeugt, d. h. der entsprechende Platz im Speicher reserviert.
    - Verweis (Referenz, Zeiger) auf dieses Objekt bzw. auf den Speicherbereich wird zurückgegeben.
- Form: Schlüsselwort new gefolgt von Konstruktoraufruf.
    - Beispiel: new BibMitglied("Petra Buch-Wurm", "Alte Gasse 1, 86807 Buchloe",12345);
- erfolgt durch den new-Operator, gefolgt von Angabe eines Konstruktors aus der Klassendeklaration des Objekts mit zugeh. Argumenten
- Objektvariablen können bereits bei ihrer Deklaration mit einem Verweis auf ein Objekt initialisiert werden

Wertzuweisung bei Objektvariablen

- Dieselbe Form wie für Variablen primitiven Datentyps.
- Auf der linken Seite des Zuweisungsoperators „=“ steht Objektvariable, auf der rechten Seite steht ein Ausdruck, der einen mit der Objektvariablen kompatiblen Klassentyp hat
- Deklaration und erste Wertzuweisung kann wie auch bei primitiven Datentypen kombiniert werden
- Initialisierung kann aber auch erst später erfolgen

Annahme: es wird eine weitere Objektvariable deklariert : BibMitglied bm2;

- Nun: Wertzuweisung bm2 = bm1;
- Wirkung:
    - Der in bm1 gespeicherte Wert (Referenz) wird bm2 zugewiesen.
    - bm1 und bm2 verweisen damit auf dasselbe Objekt.
- Wichtig: bei Zuweisung an Referenzvariablen wird der Verweis kopiert, nicht aber das Objekt, auf das verwiesen wird.

Vergleich Wertsemantik versus Referenzsemantik

- Variablen primitiven Typs
    - Mit Variablen einfachen Typs wird bei deren Vereinbarung zugleich ein Behälter für den Wert bereitgestellt.
    - Mit Angabe des Namens wird unmittelbar der in der zugeordneten Speicherzelle abgelegte Wert ausgelesen (sog. Wertsemantik).
- Variablen zusammengesetzten Typs/Objektvariablen
    - Bei solchen Variablen wird nur ein Behälter für einen Zeiger (auch: Referenz, Verweis) auf ein Objekt des entsprechenden Typs angelegt (sog. Referenzsemantik).
    - Der eigentliche Speicherplatz für das eigentliche Objekt muss dann noch angelegt werden und ist somit nicht fest angebunden.
- Wirkungen:
    - Zuweisungen bei Wertsemantik

```
int i1 = 1;
int i2;
i2 = i1; // i2: 1
 i1 = 2; // i1: 2; i2: 1
```

    - Zuweisungen bei Referenzsemantik

```
int[] a1 = {1, 2, 3};
int[] a2;
a2 = a1;   // a2 zeigt jetzt auch auf das durch a1
           // referenzierte Objekt
a1[2] = 0; // a1: 1, 2, 0 und a2: 1, 2, 0
```

    - Faustregel: bei Objekten, die i. d. R. mit new erzeugt werden, gilt die Referenzsemantik

Vorbesetzte Attribute

Beispiel: Uhrenklasse, deren Objekte eine übergebene Minutenzahl in Stunden und Minuten umrechnen

```
class Uhr {
        // Attribute
        long minuten;
        long stunden;
        TimeZone zeitzone = new TimeZone("UTC");

        // expliziter Konstruktor
        Uhr(long min) {
                stunden = min / 60; // Division ohne Rest (Ganzzahl)
                minuten = min % 60; // Modulo-Operation
        }
}
```

- Attribute werden bei Objekterzeugung mit dem angegebenen Wert initialisiert (analog für primitive Typen)
- TimeZone sei eine an anderer Stelle deklarierte Klasse.
- UTC: Coordinated Universal Time, koordinierte Weltzeit, wird zur Berechnung der Mitteleuropäischen Zeit verwendet

Referenzkonstanten

Beispiel: Uhrenklasse, deren Objekte eine übergebene Minutenzahl in Stunden und Minuten umrechnen

```
class Uhr {
        // Attribute
        long minuten;
        long stunden;
        TimeZone zeitzone = new TimeZone("UTC");
        final Date herstellungsdatum = new Date();

        // expliziter Konstruktor
        Uhr(long min) {
                stunden = min / 60; // Division ohne Rest (Ganzzahl)
                minuten = min % 60; // Modulo-Operation
        }
}
```

- Referenzkonstanten fixieren den Verweis.
- das Objekt, auf das sie verweisen, kann geändert werden (bei bleibender Identität).
- Semantik: es gibt genau ein unveränderliches Herstellungsdatum für eine Uhr

Methodenaufruf

Hier können zwei Fälle unterschieden werden:

- Fall 1 (Zugriff von Außen, nicht immer erlaubt, vgl. Sichtbarkeits-modifikatoren, folgen noch): Deklaration einer Objektvariablen, Instanziierung eines Objekts und dann Aufruf einer Methode (d. Objekts).
  - Aufruf einer Methode geschieht für ein Objekt der Klasse, in der auch die Methode deklariert ist, durch Aufruf der Methode mit vorangestellter Angabe des Objekts getrennt durch einen Punkt (sog. Punktnotation)
  - Beispiel: p.length();
- Fall 2 (Zugriff von Innen): Aufruf einer Methode innerhalb derselben oder einer anderen Methode der Klasse, in der sie deklariert ist.
- Aufruf erfolgt wie Funktionsaufruf (also ohne Angabe eines Objekts und Punkt „.“) oder mittels sog. this-Referenz (folgt auch noch).
- Beispiel: weitere Methode der Klasse Vec3 void printLength() { System.out.println("Länge: " + length()); }

Zugriff auf Attribute eines Objekts

Zwei Fälle (analog zu Methodenaufrufen):

- Fall 1 (Zugriff von Außen, nicht immer erlaubt, vgl. Sichtbarkeitsmodifikatoren, folgen noch):
  - über <objektname>.<attributname>
- Fall 2 (Zugriff von Innen): über den Attributnamen

Selbstreferenz: this-Referenz

- Jedes Objekt führt eine Referenz auf sich selbst mit sich.
- Auf diese kann man mit dem Schlüsselwort this im Konstruktor und in Methoden zugreifen.
- Damit ist ein alternativer Zugriff eines Objekts auf die eigenen Attribute und Methoden möglich.

Selbstreferenz: this-Referenz

Das Schlüsselwort this ermöglicht den expliziten Bezug auf Attribute oder Methoden des Objekts, in dessen Klasse es verwendet wird.

```
// --- Konstruktor ---
BibMitglied(String name, String privAnschrift, int mitgliedsNr) {
        this.name = name;
        this.privAnschrift = privAnschrift;
        this.mitgliedsNr = mitgliedsNr;
}
```

Vergleiche bei Referenzvariablen

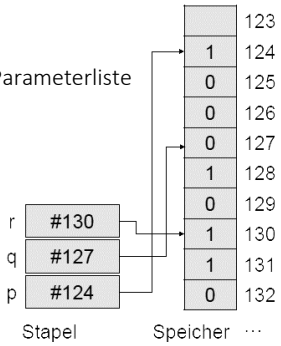
Vergleich der Referenzvariablen prüft die Identität und nicht die Bestandteile (nur die Referenzen)

Objektreferenz als Rückgabewert einer Methode

Verschiedene Methoden derselben Klasse dürfen denselben Namen verwenden, sofern sie sich in der Parameterliste unterscheiden).

Programmstapel und Objektvariablen

- Objekt wird im Speicher angelegt.
- Referenz auf das Objekt liegt auf dem Stapel, nicht aber das Objekt selbst
- Am Ende eines Blocks werden Objekte im Speicher nicht automatisch gelöscht!



Einschub: Speicherbereinigung,

- Frage: was geschieht mit Objekten, die nicht mehr benötigt werden?
- andere Sprachen: Destruktor
  - Spezielle Methode, die ein Objekt entfernt, d. h. den von ihm belegten Speicherplatz wieder frei gibt.
  - Fehleranfällig, wenn dieser von Hand im Programm aufgerufen werden muss.
- Java: automatische Speicherbereinigung („garbage collection“)
  - Speicherbereiniger ist ein spezieller Teil der Laufzeitumgebung (virtuellen Maschine), der die Objektgraphen im Speicher durchsucht.
  - Wird ein Objekt gefunden, das nicht mehr referenziert wird, wird dieses gelöscht und der belegte Speicher frei gegeben.
  - Vorteil: Programmierer/-in muss sich nicht darum kümmern.

Zählen der instanziierten Objekte einer Klasse

- Beobachtung: die Vergabe von Mitgliedsnummern von Außen ist möglich, erfordert aber eine Instanz, die sich um die Zuordnung kümmert.
- Frage: Kann man das auch innerhalb der Klasse BibMitglied selbst automatisieren?
- Strategie: es müsste gezählt werden, wie viele Objekte der Klasse BibMitglied bereits erzeugt wurden.

Klassenattribut

```
class BibMitglied {
        // Klassenattribut
        static int mitgliederZaehler = 0;

        // --- Attribute ---
        String name;
        String privAnschrift;
        int mitgliedsNr;
        // --- Konstruktor ---
        BibMitglied(String n, String pA) {
                name = n;
                privAnschrift = pA;
                mitgliedsNr = BibMitglied.mitgliederZaehler++;
        }

        // --- Methoden ---
        // unveraendert
}
```

Klassenattribut

- Für jedes „normale“ Attribut gibt es Speicherplatz pro Objekt.
- Klassenattribut (auch: statisches Attribut) ist Sonderfall, bei dem sich alle Objekte der Klasse denselben Attributwert teilen.
- Schreiboperationen in einem Objekt wirken sich damit auf alle anderen Objekte aus.
- Deklaration hat die Form:
  - static <Datentyp> <Klassenattributname> (= <Anfangswert>);
- Verwendung:
  - Der Zugriff auf „normale“ Attribute von außen erfolgte in der Form: <objektname>.<attributname>
  - Der Zugriff auf Klassenattribute von außen hat die Form: <klassenname>.<attributname> oder <objektname>.<attnme>
  - Von innen jeweils in der Form: <attributname>

Klassenmethode

```
// --- Klassenmethoden ---
static void setMitgliederZaehler(int mz) {
        BibMitglied.mitgliederZaehler = mz;
}
static int getMitgliederZaehler() {
        return BibMitglied.mitgliederZaehler;
}
// ...
```

- Klassenmethoden: auch ohne instanziiertes Objekt verwendbar
- Getter/Setter wären auch als nicht-statische Methoden erlaubt, dann aber nur verwendbar, wenn Objekt instanziiert wurde

```
BibMitglied.setMitgliederZaehler(1000);
BibMitglied bm1 = new BibMitglied( "Petra Buch-Wurm", "Alte Gasse 1, 86807 Buchloe");
BibMitglied bm2 = new BibMitglied( "Paul-Les Eratte", "Schillerstr. 2, 89290 Buch");
BibMitglied bm3 = new BibMitglied( "David B.O. Okworm", "Buchenring 3, 76297 Büchig");
System.out.println(BibMitglied.getMitgliederZaehler());
...
```

Klassenmethoden (engl. class method)

- Methoden werden im Allg. von den Objekten einer Klasse ausgeführt.
- Klassenmethoden (auch: Klassenoperationen, statische Operationen/Methoden) sind einer Klasse und nicht einem Objekt zugeordnet und bieten damit die Möglichkeit, Methoden direkt auf einer Klasse selbst auszuführen.
- In diesem Fall ist keine Objekterzeugung vor dem Methodenaufruf erforderlich.
- Deklaration: wie normale Methode nur mit vorangestelltem Schlüsselwort static
- Beispiele:
  - die statische Methode main (Programmstart),
  - alle Methoden der vergangenen Lehreinheiten.
- Frage: Wann und wozu verwendet man Klassenmethoden?
  - als Getter und Setter für Klassenattribute
  - wenn für die Methode kein Objekt der Klasse benötigt wird oder generell zu einer Klasse keine Objekte instanziiert werden sollen, z. B. Java-Bibliotheksklasse Math, die mathematische Konstanten und Funktionen bereitstellt
    - Bibliotheksklassen sind bereits programmiert und können direkt verwendet werden
  - so sparsam wie möglich, da eigentlich nicht der objektorientierten Philosophie entsprechend
- Hinweis: Klassenmethoden können nicht auf Instanz- variablen (z. B. Attribute, Konstanten), sondern nur auf Klassen-variablen (z. B. Klassenattribute, Klassenkonstanten) operieren.
- Erlaubt ist aber:

```
static Vec3 addAndIncX(Vec3 v1, Vec3 v2) {
Vec3 r = v1.add2(v2);
r.x++;
return r;
}
```

- Es werden Objektreferenzen v1, v2 als Parameter übergeben.
- Für die übergebenen Objekte dürfen (soweit erlaubt) deren Instanzmethoden (z. B. add2) aufgerufen und auf deren Attribute (z. B. x) werden.
- Es ist nicht erforderlich, ein Objekt der Klasse Vec3 vor dem Aufruf von addAndIncX zu erzeugen

Sichtbarkeit von Information

- Vorteil der objektorientierten Sichtweise:
  - Es reicht, die Signaturen/Schnittstellen der Methoden eines Objekts zu kennen, um mit diesem „umgehen“ zu können.
  - Implementierungsdetails sind für die Anwendung einer Methode nicht relevant.
  - Man sagt, Implementierungsdetails sind verdeckt bzw. geheim.
- Oft (insb. in größeren Software-Projekten):
  - nur wichtig, was ein Objekt leistet,
  - nicht aber, wie es realisiert ist.
  - Konsequenz: es werden Programmiertechniken benötigt, mit denen man den Zugriff auf Attribute und Methoden von Objekten einschränken kann.
  - Die Anwendung dieser Techniken fördert die arbeitsteilige Entwicklung in Software-Projekten, da nicht jede/r Entwickler/-in den gesamten Programmtext verstehen muss.

Datenkapselung

- bezeichnet das Verbergen von Programmierdetails vor dem direkten Zugriff von Außen.
- Direkter Zugriff auf interne Datenstruktur wird unterbunden und erfolgt stattdessen über wohldefinierte Schnittstellen („Black-Box-Modell“).
- Objektorientierte Programmierung:
  - Kontrollierter Zugriff auf Attribute und Methoden von Objekten.
  - Objekte können den internen Zustand anderer Objekte nicht in unerwarteter Weise lesen oder verändern.
  - Klasse hat definierte Schnittstelle, die festlegt, auf welche Weise mit ihren Objekten umgegangen werden kann.
  - Vom Innenleben der Klasse (z. B. verwendete Algorithmen) soll Anwender möglichst wenig wissen (sog. Geheimnisprinzip).
  - Implementierung einzelner Methoden oder ganzer Klassen kann so ausgetauscht und verbessert werden, ohne dass das Programm an anderen Stellen geändert werden muss.

Sichtbarkeitsmodifikatoren private und public

- ... beeinflussen die Sichtbarkeit, d. h. den Zugriff z. B. auf Attribute und Methoden von Objekten.
- ... werden einer Deklaration vorangestellt, der damit ein Sichtbarkeitsgrad zugeordnet wird.
- private:
  - bewirkt bei einer Attributdeklaration, dass auf das Attribut nur innerhalb der Klassendeklaration lesend und schreibend zugegriffen werden kann, aber nicht von außerhalb.
  - bewirkt bei einer Methodendeklaration, dass die Methode nur innerhalb der Klassendeklaration aufgerufen werden kann, aber nicht von außerhalb.
  - Es können auch Konstruktoren oder ganze Klassen (sinnvoll z. B. in sog. Paketen) als private deklariert werden.
- public:
  - erlaubt vollen Zugriff von innerhalb und außerhalb der Klasse (wenn die Klasse nicht selbst als private deklariert wurde).

	Eigenschaft ist sichtbar aus/von			
Modifikator	eigener Klasse	selben Paket	Unterklasse	überall
`public`	ja	ja	ja	ja
`protected`	ja	ja	ja	nein
ohne/paketsichtbar	ja	ja	nein	nein
`private`	ja	nein	nein	nein

Bemerkungen

- Instanzvariablen sollten nur mit wirklich gutem Grund öffentlich zugänglich sein.
- Besser: öffentliche Getter- und Setter-Methoden verwenden, die den Wert der Instanzvariablen setzen bzw. zurückliefern.
- Vorteile: ein Setter kann gegenüber der reinen Zuweisung bspw. überprüfen, ob ein übergebener Wert innerhalb eines gewünschten Wertebereichs liegt.