# Getter- u setter-Methoden

Bisher haben wir uneingeschränkt von überall lesend u. schreibend direkt auf die Attribute zugegriffen.

Dies kann zu schwer findbaren Fehlern führen, wenn in den Attributen plötzlich inkonsistente Werte auftreten,  
da kaum nachvollziehbar ist, von wo aus sie geschrieben wurden.

Deshalb wollen wir den Zugriff einschränken, keinen direkten Zugriff von überall mehr zulassen,  
sondern einen "Umweg erzwingen", bei dem wir Regeln aufstellen können.

Z.B. wollen wir verhindern, dass Seitenlängen von Rechteck oder Dreieck oder der Radius eines Kreises negativ wird.

class … {

direkter schreibender Zugriff

class Rechteck {

int laenge … ;

}

… main … {

Rechteck r1 = new Rechteck();

direkter lesender Zugriff

r1.laenge = …;

Sys…ln(r1.laenge);

}

}

Wir unterbinden den direkten lesenden u. schreibenden Zugriff,

indem wir vor das entsprechende Attribut das Schlüsselwort "private" schreiben.

Der Versuch, trotzdem (von außerhalb der Klasse) direkt zuzugreifen, wird mit einer Fehlermeldung verhindert.

class … {

class Rechteck {

**private** int laenge … ;

}

direkter schreibender Zugriff

… main … {

Rechteck r1 = new Rechteck();

direkter lesender Zugriff

r1.laenge = …;

Sys…ln(r1.laenge);

}

}

Wie greifen wir stattdessen indirekt zu?

Lesend über eine getter-Methode, schreibend über eine setter-Methode:

class … {

direkter schreibender Zugriff

class Rechteck {

**private** int laenge … ;

}

… main … {

Rechteck r1 = new Rechteck();

direkter lesender Zugriff

r1.laenge = …;

**in**direkter schreibender Zugriff

setter-Methode

Sys…ln(r1.laenge);

**in**direkter lesender Zugriff

getter-Methode

}

}

Herleitung getter-Methode:

Die getter-Methode liefert den Wert des entsprechenden gekapselten Attributs zur aufrufenden Stelle,   
auch außerhalb der Klasse.

In den Namen der Methode muss eingehen:

1. Name der Variablen
2. Unterscheidung: getter- oder setter-Methode.

Der Name der Variablen geht direkt in den Methodennamen ein.

Die Unterscheidung erfolgt durch Voranstellen der Vorsilbe "get\_" oder "set\_":

Hier heißt die getter-Methode für das Attribut "laenge": get\_laenge()

Sie liefert den Wert des entsprechenden Attriutes zurück.

Sie hat keinen Parameter.

getter-Methode für das int-Attribut "laenge":

public int get\_laenge(){

return laenge;

}

Statt bisher mit

System.out.println (r1.laenge);

greifen wir indirekt zu über

System.out.println (r1.get\_laenge());

Herleitung setter-Methode:

Die setter-Methode setzt das Attribut, auf das sie sich bezieht, auf den übergebene Wert.

Für das int-Attribut "laenge" heißt sie: "set\_laenge()".

Sie hat einen Parameter, aber keinen Rückgabewert.

Somit sind getter- u. setter-Methoden komplementär,

wie wir es schon bei und- u. oder-Gattern kennengelernt haben

u. bei den Operationen "push()" u. "pop()" auf einen Stack wiederfinden werden:

Eine getter-Methode hat einen Rückgabewert, aber keinen Parameter.

Eine setter-Methode hat keinen Rückgabewert, aber einen Parameter

1. Version: Wir geben dem Parameter einen Namen,

der sich vom Attribut unterscheidet, aber eine Bezug dazu herstellt, z.B. "uebergabe\_laenge":

vorläufige Setter-Methode für das int-Attribut "laenge":

public void set\_laenge (int uebergabe\_laenge){

laenge = uebergabe\_laenge;

}

Aus Bequemlichkeit überlegen wir, wie wir es vermeiden können,

uns für den Paramter einen neuen Namen ausdenken zu müssen.

Wir fragen uns, ob wir dem Parameter nicht den gleichen Namen geben können wie den des Attributes:

class Rechteck {

int **laenge** = 0;

public void set\_laenge (int **laenge**){

laenge = laenge;

}

}

Nun steht da: "laenge=laenge".

Da sieht merkwürdig aus.

Ist das eine Selbstzuweisung? Dann hätte sie keinen Effekt.

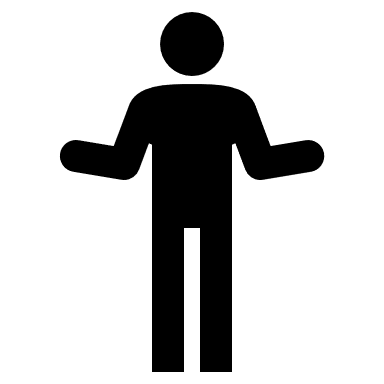
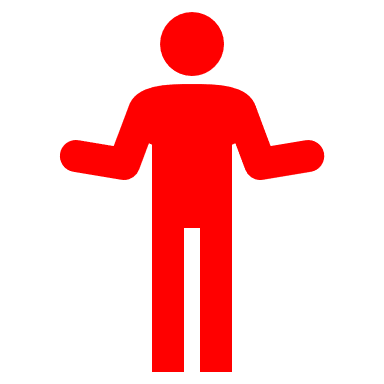
Ist die Laenge in der Methode die rote oder die grüne Länge?

Oder einmal die eine und einmal die andere?

Welche ist dann welche?

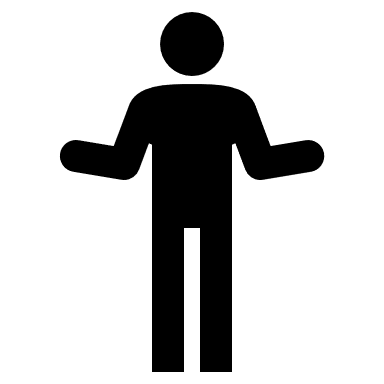
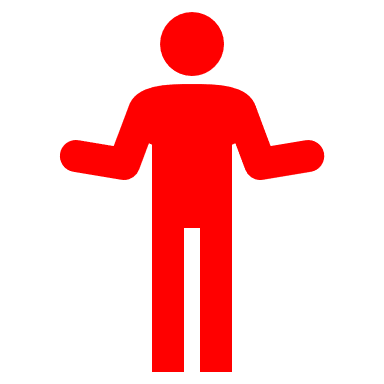
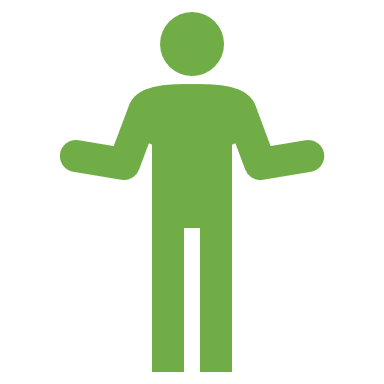
Um die Fragen zu beantworten, müssen wir den Effekt der "Verschattung" oder "Verdeckung" kennen:

Stellen Sie sich vor, Sie stehen an einer Stelle u. sehen eine rote Person:

Sie blicken rote Person

Jetzt stellt sich eine grüne Person davor:

Jetzt sehen Sie nur noch die grüne Person, die sich vor die rote gestellt hat.

Die grüne Person verdeckt den Blick auf die rote.

Übertragen auf die Situation oben jeweils mit entsprechenden Farben:

Global ist das rote Attribut "laenge" deklariert.

Wenn Sie lokal eine Variable oder einen Parameter gleichen namens deklarieren,

hier der grüne Parameter "laenge",

verdeckt oder verschattet der lokale grüne Parameter die Sicht auf das globale rote Atribut.

Damit sprechen Sie in der Methode in beiden Fällen den Parameter an.

class Rechteck {

int **laenge** = 0;

public void set\_laenge (int **laenge**){

**laenge** = **laenge**;

} lokaler Parameter in beiden Fällen

}

Das hatten wir uns anders vorgestellt!

Wir haben uns gleich drei (!) Eigentore geschossen:

1. Eine Selbstzuweisung hat keinen Effekt!
2. Wir nehmen die Zuweisung vor an den Parameter,  
   der nach Verlassen der Methode wieder freigegeben wird.  
   Deshalb hat zum zweiten die Zuweisung keinen Effekt nach außen,  
   insbesondere nicht die Weiterreichung des Parameters an das Attribut.
3. Da das rote Attribut verschattet ist,  
   können wir in der Methode nicht mehr darauf zugreifen,  
   insbesondere nicht der Aufgabe gerecht werden,  
   dem Attribut den Wert des übergebenen Parameters zuzuweisen.

Ergebnis: unbefriedigend!

Was würden wir uns wünschen?

1. Wir wollen unserer Aufgabe nachkommen können wie in der Vorversion,  
   den Parameter an das Attribut weiterzureichen
2. Wir wollen an der Bequemlichkeit festhalten,  
   dem Parameter den gleichen Namen zu geben wie das Attribut.

Lösung: Schlüsselwort "this" mit Punkt vor den Bezeichner,

der sich auf das Attribut beziehen soll.

Damit wir der lokale Kontext überwunden u. der Blick wieder frei für das Attribut.

Man kann es sich bildlich vostellen, dass der grüne Kasten ist wie eine Mauer,

in der nur der grüne lokale Parameter zu sehen ist.

Durch Voranstellen von "this." "klettert man auf die Mauer",

überwindet somit die lokale Einschränkung u. kann wieder das rote Attribut sehen.

Im Beispiel mit den Personen könnte man sagen,

man stellt sich auf einen Hocker u. sieht damit über die grünen Person hinweg

wieder die rote.

class Rechteck {

int **laenge** = 0;

public void set\_laenge (int **laenge**){

**this.**laenge = **laenge**;

}

}

Damit sind beide Wünsche erfüllt:

1. Wir können unserer Aufgabe nachkommen,   
   den Wert des Parameters in das Attribut zu schreiben,
2. trotzdem an der Bequemlichkeit festhalten,   
   den Parameter so zu nennen wie das Attribut selbst.

endgültige setter-Methode für das Attribut "laenge":

public void set\_laenge (int laenge){

this.laenge = laenge;

}

Technisch gesehen können wir uns das "this" vorstellen wie ein Zeiger auf das Objekt.

Wenn wir ihm folgen,

landen wir auf der globalen, äußersten Ebene

u. haben damit den lokalen Kontext der setter-Methode überwunden,

die uns den Blick auf das Attribut verschattet hat.

Die Technik werden wir auch später noch einmal aufgreifen

beim Thema "Verkettung von Konstruktoren".