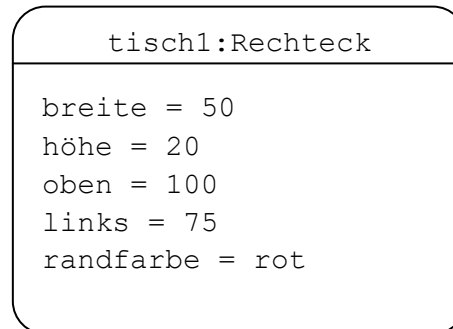


Darstellung im Klassendiagramm bzw. Objektdiagramm

Klassenkarte



Objektkarte



Objektorientierung

Objektorientierung beschreibt das Lösen von Problemstellungen durch Analyse der beteiligten Objekte und deren Zusammenspiel.

Punktnotation Methodenaufrufs ohne oder mit Parameter

```
Objektname.methode(Parameter);
Objektname.attribut = Attributswert;
```

Beispiel:

```
tisch1.breite = 50;
tisch1.verschieben(10,10);
tisch1.löschen();
karol.schritt();
karol.hinlegen();
karol.linksdrehen();
```

Datentypen

Datentyp	Beschreibung	Werte	Speicherbedarf
int	ganze Zahl	-2.147.483.648 bis 2.147.483.647	4 Bytes
long	ganze Zahl	-2^{63} bis $2^{63}-1$	8 Bytes
float	Dezimalzahl	$\pm 1,4E-45$ bis $\pm 3,4E+38$	4 Bytes
double	Dezimalzahl	$\pm 4,9E-324$ bis $\pm 1,7E+308$	8 Bytes
char	ein Zeichen	0 bis 65.535	2 Byte
boolean	Wahrheitswert	true / false	1 Bit
String	Zeichenfolge	“Zeichenfolge variabler Länge ...“	2 Byte*Zeichen+Overhead

Klassen in JAVA

Hund
name:String alter:int ...
Hund(_name,_alter) belle():void setzeName (neuername):void gibAlter():int

Hund
String name int alter ...
Hund(String _name,int _alter) void belle() void setzeName(String neuername) int gibAlter()

Umsetzung der Klassenkarte in JAVA-Code

```
① public class Hund
{ // Attribut werden deklariert
  String name;
  int alter;

  //Konstruktor
② Hund(String _name; int _alter)
  { //Initialisierung der Attribute
    name = _name;
    alter = _alter;
  }

  //Methoden
③ void belle()
  {
④   println(name+": Wuff, Wuff!");
  }

  void setzeName(String neuername) name = neuername;
⑤ int gibAlter() return alter;
}
```

Erstellen des Objekts und ausführen von Methoden in JAVA-Code

```
⑥ Hund hund1 = new Hund("Nero",7);
hund1.belle();
hund1.setzeName("Pluto");
hund1.belle();
```

Informatik 10 – Grundwissen

Fachbegriff im JAVA-Code

① Klassenkopf

Der **Klassenkopf** steht am Anfang der Klassendeklaration und beinhaltet den Klassennamen (immer in Großschreibung), manchmal zusätzliche Befehle wie `extends` (siehe Vererbung).

② Konstruktor

Der **Konstruktor** ist eine spezielle Methode die aufgerufen wird, um ein Objekt einer Klasse zu erzeugen (siehe ⑥). Sein Methodename lautet genauso wie der Klassenname und hat keinen Rückgabetyt (**kein** `void`, `int`, ...).

Der Konstruktor wird verwendet um einen „Startzustand“ der erzeugten Objekte festzulegen, z.B. durch **Initialisieren** der Attributwerte.

③ Methodenkopf

Der **Methodenkopf** steht am Anfang der Methodendeklaration. Er beinhaltet den Methodennamen (Kleinschreibung).

Vor dem Methodennamen steht der Datentyp des **Rückgabewerts** (z.B. `int`, `float`, `String`, ...) oder `void`, wenn die Methode keine Werte zurückliefert.

Hinter dem Methodennamen stehen die Übergabeparameter und ihre Datentypen, diese können dann im Methodenrumpf als **lokale Variablen** genutzt werden.

④ Methodenrumpf

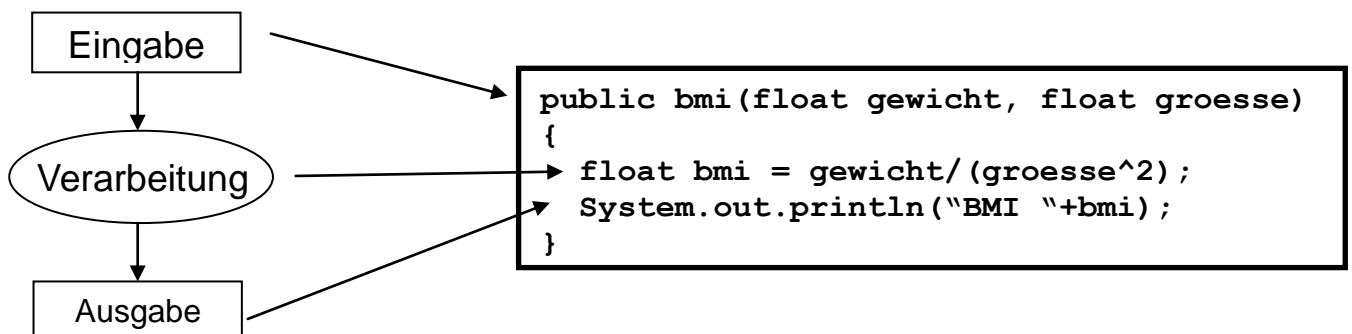
Im **Methodenrumpf** wird die Programmsequenz angegeben, welche die Funktion der Methode in JAVA-Code beschreibt.

Hat die Methode einen Rückgabewert (siehe ③) dann muss die Sequenz mit dem Befehl `return <Rückgabewert>` beendet werden.

⑤ Getter- /Settermethoden

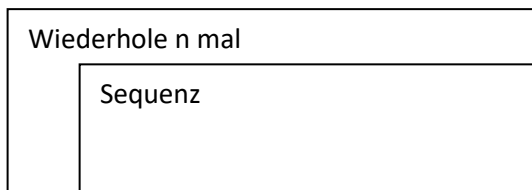
Getter- und **Settermethoden** sind sehr einfache Methoden, welche zum Abrufen (meist `gibAttributwert()`) von Attributwerten und zum Festlegen (meist `setzeAttributwert(neuer Wert)`) von Attributwerten verwendet werden.

E-V-A Prinzip



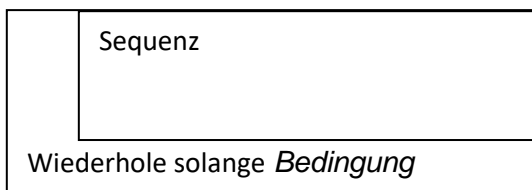
Kontrollstrukturen (Struktogramm)

Wiederholung mit fester Anzahl



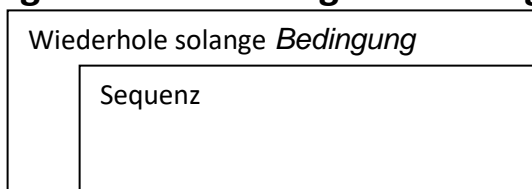
```
for(int i=1; i<=n; i++)  
{  
    Sequenz;  
}
```

bedingte Wiederholung mit Endbedingung



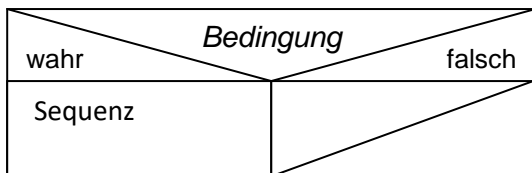
```
do  
{  
    Sequenz;  
} while (Bedingung);
```

bedingte Wiederholung mit Anfangsbedingung



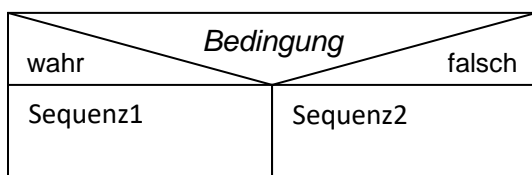
```
while (Bedingung)  
{  
    Sequenz;  
}
```

Bedingte Anweisung (1-seitig)



```
if (Bedingung)  
{  
    Sequenz;  
}
```

Bedingte Anweisung (2-seitig)

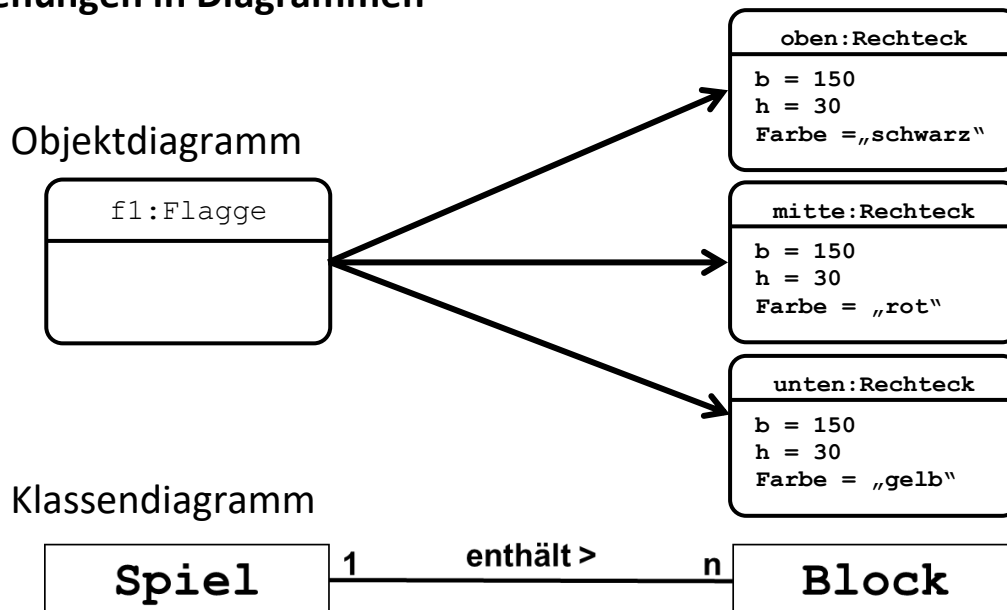


```
if (Bedingung)  
{  
    Sequenz1;  
}  
else  
{  
    Sequenz2;  
}
```

Bedingungen (Beispiele)

```
x > 0  
b == 5  
c != 20  
(a > 5 && a < 10)  
(a > 6 || a < -6)  
karol.istWand()  
!karol.istZiegel()
```

Beziehungen in Diagrammen



Die **Aggregation** („Enthält-Beziehung“) wird im enthaltenen Objekt als Attribut implementiert. Dessen Wert ist eine **Referenz** (Verweis auf den Speicherplatz) auf das enthaltene Objekt.

Arten von Beziehungen

Kardinalität	Multiplizität
1 : 1	oder 1 .. 1
1 : n	1 .. *
n : m	* .. *

Feld (Array)

- Zusammenfassung mehrerer Variablen oder Objekte der gleichen Klasse in einer **zusammenhängenden Folge fester Länge**.
- Die einzelnen Feldelemente sind durchnummeriert. Diese Nummern nennt man **Index**.
- Zugriff auf einzelne Elemente über Indizes (Nummerierung beginnt bei 0).

```
int zahl[];
zahl = new int[1000];
zahl[5] = 10;
```

Deklaration (Festlegen des Datentyps)
Instanziierung (Erzeugen des Objekts)
Initialisierung (Zuweisung eines Werts)

Kommunikation zwischen Objekten

Objekte können mit einander kommunizieren, durch...

...lesen/verändern der öffentliche Attributwerte eines anderen Objektes lesen

...Aufruf der öffentliche Methoden eines anderen Objektes. (zu bevorzugen!!)

Hinweis: Attribute sollten nicht öffentlich sein (**Datenkapselung**), **Getter**- & **Setter**-Methoden verwenden.