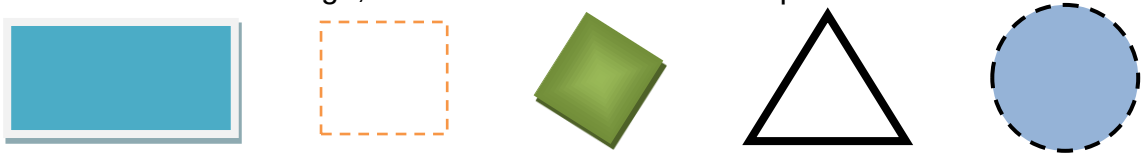
Unterscheidung Klasse und Objekt

Objekte sind konkrete Dinge, z.B. in der Miniwelt des Computers:



Objekte mit gemeinsamen Merkmalen und Fähigkeiten fasst man unter einer **Klasse** zusammen. Die Klasse beschreibt die Gemeinsamkeiten all dieser Objekte und dient somit als eine Art Bauplan für die Objekte dieser Klasse.
Beispiel: Die ersten drei Zeichnungsobjekte gehören zur Klasse Rechteck.

Klassenkarte

Rechteck
posX posY füllfarbe...
verschieben(dx,dy) setzeFüllfarbe(farbe) gibFüllfarbe() ...

Objektkarte

kiste:Rechteck
posX = 100 posY = 75 füllfarbe = "blau"

Umsetzung der Klassenkarte in JAVA-Code

```
①  public class Rechteck
   { // Attribut werden deklariert
     float posX;
     float posY;
     String füllfarbe;

     //Konstruktor
②    Rechteck(float para1; float para2)
     { //Initalisierung der Attribute
       posX = para1;
       posY = para2;
       farbe = "grau";
     }

     //Methoden
③    void verschieben(float dx, float dy)
     {
④       posX = posX + dx;
        posY = posY + dy;
     }

⑤    String gibFüllfarbe() return füllfarbe;

     void setzeFüllfarbe(String farbe) füllfarbe = farbe;
   }
```

Erstellen des Objekts passend zur Objektkarte in JAVA-Code

⑥
```
Rechteck kiste = new Rechteck(100,75);
kiste.setzeFüllfabe("blau");
```

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 Methodenname lautet genauso wie der Klassenname und hat keinen Rückgabetyp (**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 Methodenname steht der Datentyp des **Rückgabewerts** (z.B. `int, float, String`, …) oder `void`, wenn die Methode keine Werte zurückliefert.
Hinter dem Methodenname 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.

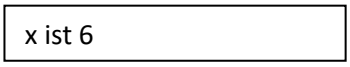
⑤ Getter- /Settermethoden

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.

Natur und Technik - Informatik 9 – Grundwissen

Bausteine für Algorithmen/Kontrollstrukturen

Anweisung

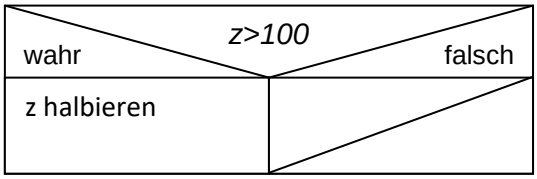



```
x = 6;
```

Sequenz

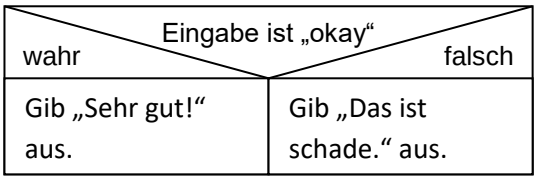

```
x = 6;
x = x*2;
println(x);
```

einseitig bedingte Anweisung



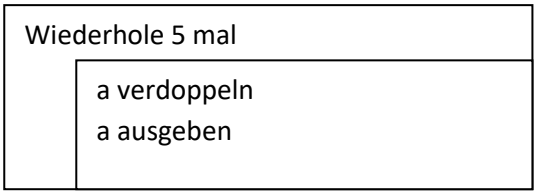

```
if(z > 100)
{
  z = z / 2;
}
```

zweiseitig bedingte Anweisung



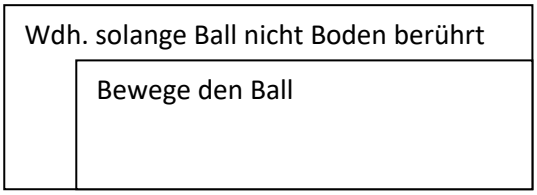

```
if(eingabe == "okay")
{
  println("Sehr gut!");
}
else
{
  println("Das ist schade.");
}
```

Wiederholung mit fester Anzahl



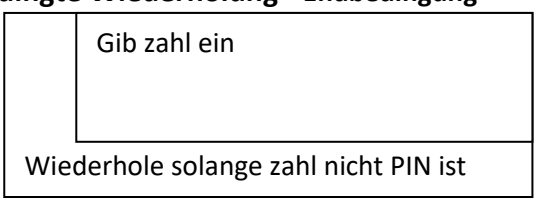

```
for (int n = 0;n < 5; n++)
{
  a = a*2;
  println(a);
}
```

bedingte Wiederholung - Anfangsbedingung




```
while (!ball.collidesWith(boden))
{
  bewegeBall();
}
```

bedingte Wiederholung - Endbedingung




```
do
{
  zahl = Input.readInt("Pin?");
}
(zahl != PIN)
```

*Die Befehle der Online-IDE sind an manchen Stellen vereinfacht, so wird z.B. aus JAVA `system.out.println(…)` bei der Online-IDE nur `println(…)` um die Programmierung zu vereinfachen. (siehe https://www.learnj.de/doku.php?id=unterschiede_zu_java:start)

Ablaufbeschreibung von Algorithmen

Ein **Algorithmus** ist eine eindeutige Handlungsvorschrift zur Lösung eines Problems oder einer Klasse von Problemen. Algorithmen bestehen aus endlich vielen, wohldefinierten Einzelschritte.

Beispiele: Packen einer Schultasche, Kochrezept, Bauanleitung, Programm

Darstellung im Klassendiagramm bzw. Objektdiagramm

Klassenkarte*

Rechteck
oben links breite höhe füllfarbe randfarbe randart randstärke ...
drehen(winkel) verschieben(dx,dy) löschen() ...

Objektkarte

kiste:Rechteck
breite = 50 höhe = 20 oben = 100 links = 75 randfarbe = blau

Punktnotation bei Darstellung eines Attributwertes bzw. Methodenaufrufs ohne oder mit Parameter

```
allgemein:
Objektname.Methode (Wert);

Beispiele:
kiste.breite = 50;
kiste.verschieben(10,10);
kiste.löschen();
```

Vererbung

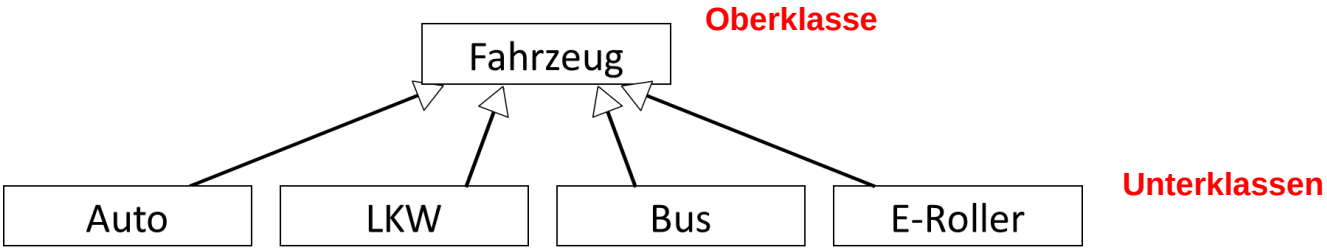
Bei der Vererbung besteht zwischen zwei Klassen eine **Ist-Ein-Beziehung**.
Die **Unterklasse** erbt die Attribute und Methoden der **Oberklasse**.
Diese müssen in der Unterklasse nicht erneut implementiert werden.
Die Unterklasse kann um weitere Attribute und Methoden ergänzt werden.
Zudem können geerbte Methoden überschrieben werden, sodass sie sich anderes verhalten als die gleichnamigen der Oberklasse (**Spezialisierung**).

JAVA-Code zur Erstellung einer Unterklasse

```
public class Unterklasse extends Oberklasse {

  //zusätzliche Attribute der Unterklasse werden deklariert
  <Datentyp> neuesAttribut1;
  <Datentyp> neuesAttribut2;
  …

  //Konstruktor der Unterklasse
  Unterklasse (<Datentyp> parameter1;<Datentyp> parameter2; …)
  {
    //Mit super wird der Konstruktor der Oberklasse aufgerufen
    super(parameter1; parameter2; …)

    //Neue Attribute werden initalisiert
    neuesAttribut1 = paramter4;
    neuesAttribut2 = paramter5;
  }

  //Methode der Oberklasse wird überschrieben
  @Override
  public void methoderDerOberklasse ()
  {
     <neue Sequenz>
  }

  //neue Methode der Unterklasse wird deklariert
  public void neueMethoderDerUnterklasse ()
  {
     <neue Sequenz>
  }
}
```

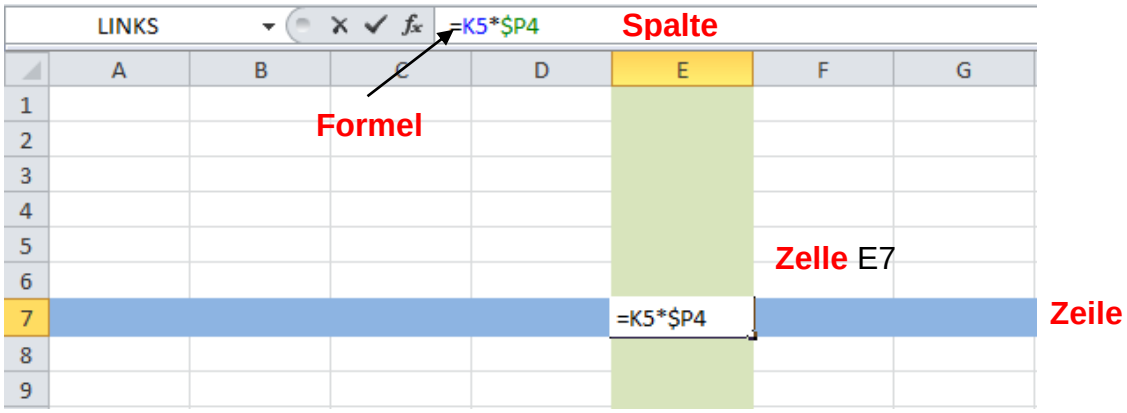
Klassendiagramm der Vererbung

Im Klassendiagramm werden geerbte Attribute und nicht überschriebene Methoden nicht angegeben.

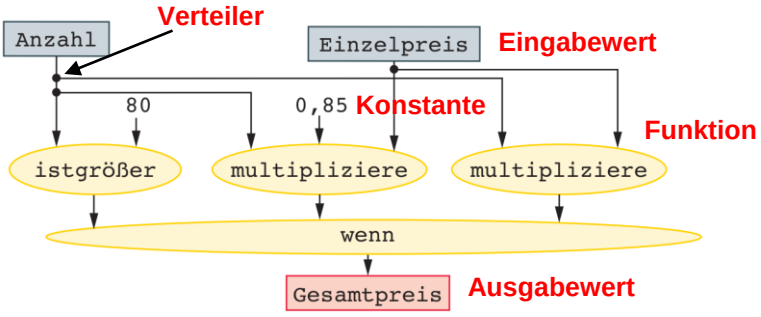


Tabellenkalkulation

Ein Rechenblatt besteht aus **Zellen**, welche durch **Spalten** und **Zellen** nummeriert werden. In den Zellen sind **Daten** oder **Funktionen** gespeichert.
Mit Hilfe von **relativen** (z.B. **D2**) und **absoluten** (z.B. **A3**) Bezügen kann auf die Zellinhalte zugegriffen werden.



Flussdiagramm



Ein **Flussdiagramm** stellt **zusammengesetzte Funktionen** mit Hilfe von **Eingabewerten**, **Ausgabewert**, **Verteiler**, **Konstanten** und **Funktionen** dar.

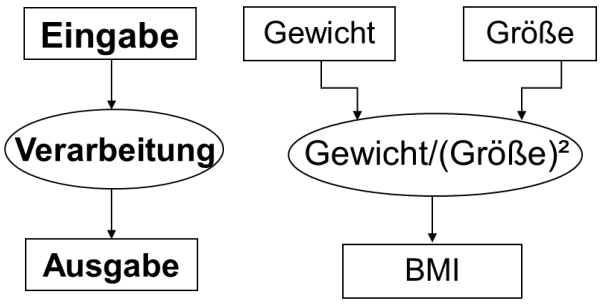
Funktion zum Diagramm:

```
Gesamtpreis = wenn(Anzahl>80;Anzahl*0,85*Einzelpreis;Anzahl*Einzelpreis)
=wenn(Anzahl>80; Anzahl*Einzelpreis*0,85; Anzahl*Einzelpreis)
```

Natur und Technik - Informatik 9 – Grundwissen

E-V-A-Prinzip

Ein Algorithmus kann auch als informations-verarbeitender Prozess gesehen werden, welcher nach dem **E-V-A** Prinzip funktioniert.
E-V-A = **Eingabe – Verarbeitung - Ausgabe**



Datenbanken

Eine **Datenbank** enthält **Tabellen** diese bestehen aus **Datensätzen** mit passenden **Attributen** und entsprechenden **Datentypen** (Text, Zahl, Wahrheitswert, Datum, Zeit). Jeder Datensatz ist durch einen eindeutigen **Primärschlüssel** (unterstrichenes Attribut) identifizierbar.

Primärschlüssel

		Freund	**Tabellenname**		
Name	Vorname	Geburtsdatum	Nickname	…	**Attribute**
Altmann	Achmed	2007-07-09	achmed2091		
Groß	Luisa	2008-04-01	luisagrgr		
Marlov	Aaron	2006-09-13			**Datensatz**
Marlov	Luisa	2009-02-14	lusi		

Informationen werden durch SQL-Abfragen gewonnen:

```
SELECT <Attribute>
FROM <Tabelle>
WHERE <Bedingungen>
ORDER BY <Attribute>
```

Für Beispieltabelle:

```
SELECT Name, Vorname
FROM Freund
WHERE Name LIKE 'A%'
ORDER BY Vorname
```

Ergebnistabelle

Name	Vorname
Altmann	Achmed