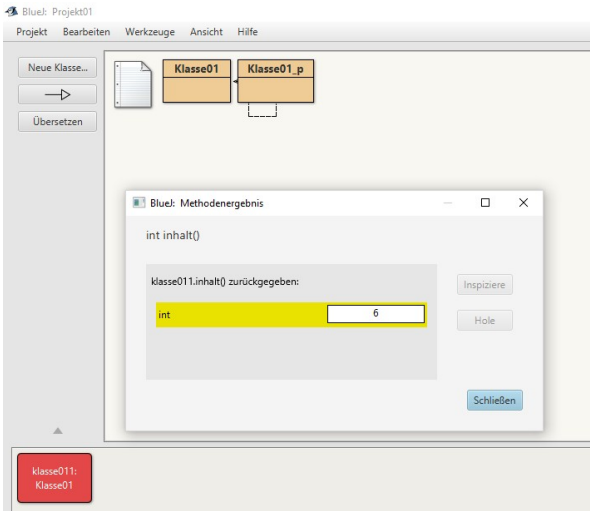
Zunächst einmal ist java die weltweit verbreitetste Programmiersprache, insbesondere deshalb weil die Programme auf jedem Computer laufen.
Um mit java arbeiten zu können, benötigt man auf jeden Fall das JDK. Dann kann man mit jedem Texteditor spezielle Dateien schreiben und diese mit der Endung .java abspeichern.
Wie es dann weitergeht wird später erläutert.
Im folgenden arbeiten wir mit dem Entwicklungssystem BlueJ, das extra für die Schulung entwickelt wurde.
Unter www.bluej.org können wir BlueJ kostenlos downloaden.

Zu Projekt 01:

BlueJ starten, Projekt Neues Projekt anklicken, Projekt01 eingeben, der Ordner Projekt01 wird erzeugt und es erscheint das Hauptfenster mit einem Textdatei-icon,
das man als memo benutzen kann.
Im linken Nebenfenster klicken wir auf Neue Klasse... und geben Klasse01 ein,
im Ordner Projekt01 wird die Datei Klasse01.java erzeugt im Hauptfenster erscheint
ein entsprechendes icon. Nach Rechtsklick darauf und bearbeiten erscheint ein Beispieltext.
Diesen löschen wir und ersetzen ihn durch

```
public class Klasse01
{
   // Instanzvariablen
   private int laenge;
   private int breite;
   //Konstruktor für Objekte der Klasse Klasse01
   public Klasse01(int la, int br)
   {
      // Instanzvariable initialisieren
      laenge=la;
      breite=br;
   }
   //Methode 1
   public int inhalt()
   {
      return laenge*breite;
   }
   //Methode2
   public int umfang()
   {
       return 2*(laenge+breite);
   }
}
```

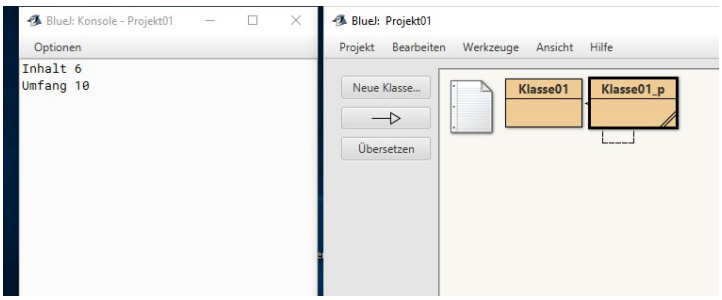
//… sind Kommentare
Nach Übersetzen wird im Ordner die ausführbare Datei Klasse01.class erzeugt.
Nach Schließen, Rechtsklick auf icon Klasse01 und New ... wählen.
Gibt man in das Fenster die gewünschten Werte für laenge und breite ein erscheint
im unteren Fenster ein rotes icon, das das neue Objekt repräsentiert.



Mit Rechtsklick auf dieses rote icon kann man sich Inhalt und Umfang eines Rechtecks mit den eingegebenen Maßen ansehen indem man die entsprechenden Methoden wählt..
Für ein anderes Rechteck muss man den Vorgang ab New ... wiederholen.
Weitere Erklärungen:
Objektorientiertes Programmieren bedeutet für eine Klasse realer Objekte eine geeignete Klasse zu programmieren mit deren Hilfe man reale Objekte berechnen kann.
Wenn man mit einem Computer ein Rechteck berechnen will, so frage man sich zunächst nach den Eingabedaten, diese werden in den Instanzvariablen gespeichert.
Dann frage man was berechnet werden soll, dies wird in den Methoden ausgeführt.
Entsprechend wird die Klasse aufgebaut.
Mit den geschweiften Klammern werden zusammengehörige Programmteile zusammengefasst.
Sie treten stets geschachtelt auf, {{...}{…}} wären 2 kleine Schachteln in einer großen.
Sowohl Variable als auch Methoden müssen nach Typ deklariert werden.
int bedeutet Ganzzahl (double bedeutet Kommazahl, String Text in Anführungszeichen).
Instanzvariable und Methoden können außerdem public (extern zugänglich) oder private (nur intern zugänglich) spezifiziert werden.
Der Konstruktor heißt wie die Klasse und hat 2 rund Klammern hinter der Bezeichnung.
Zunächst denkt man, dass dies kein Mensch kapiert, jedoch ist es immer gleich strukturiert.
Nimmt man diese Klasse als Vorlage, so kann man (fast) problemlos jedes reale Objekt programmieren.
Standalone mit Doppelklick auf icon funktionert aber noch nicht.
Dazu benötigt man zunächstneine spezielle Klasse mit der main-Methode.
Dazu gehe man vor wie bei Klasse01 und ersetzen den vorgegebenen Text durch

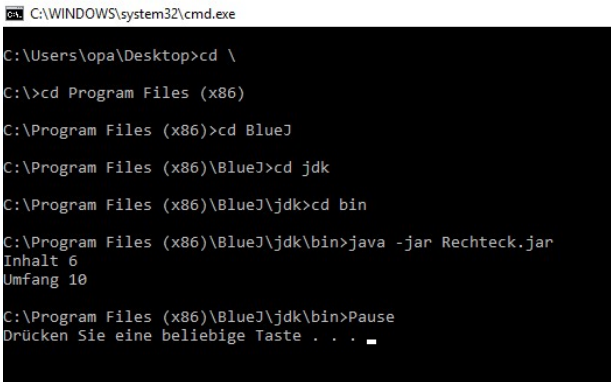
```
public class Klasse01_p
{
   public static void main(String[] args)
   {
   Klasse01 OKlasse01=new Klasse01(3,2);
   System.out.println("Inhalt "+OKlasse01.inhalt());
   System.out.println("Umfang "+OKlasse01.umfang());
   }
}
```

Nach Übersetzen Rechtsklick bei Klasse01_p und void main(String[] args) anklicken ergibt in der Konsole.
Inhalt 6
Umfang 10
(das zunächst eingeblendete Eingabefenster kann ohne Eingabe geschlossen werden)

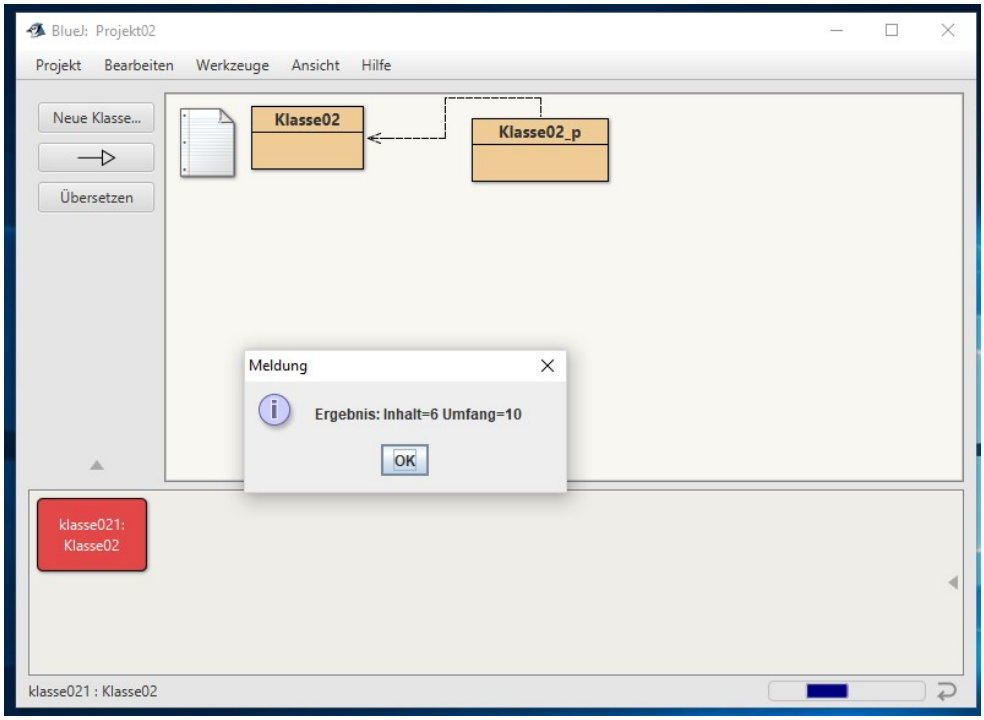


Erläuterungen:
Der Interpreter kann auf die Klasse direkt zugreifen (public), die Methode kann unmittelbar angewandt werden (static) und gibt nichts zurück (void).
String[] args bedeutet, dass man das Programm mit einer Zeichenketten-Parameterliste args[0], args[1], … starten kann.
OKlasse01.inhalt() startet die Methode inhalt() mit den entprechenden Parameterwerten.
System.out.println() ist eine analoge bereits vordefinierte Methode zur Ausgabe.
Als Argument kann man Werte von Variablen additiv verknüpft mit Zeichenketten (in Anführungszeichen) wählen. System.out.print() druckt ohne Zeilenvorschub.

Jetzt geht es auch standalone: (kann übersprungen werden)
Projekt Als jar-Archiv speichern...
Wird die .jar Datei mit Rechteck bezeichnet, wird Rechteck.jar im ProjektOrdner erzeugt.
Diese muss man in C:\Program Files(x86)\BlueJ\jdk\bin kopieren
Dann die Textdatei mit Inhalt
cd \
cd Program Files (x86)
cd BlueJ
cd jdk
cd bin
java -jar Rechteck.jar
Pause
unter javare.bat auf Desktop speichern und öffnen.
Dann erscheint erfolgt die Ausgabe im Terminal.



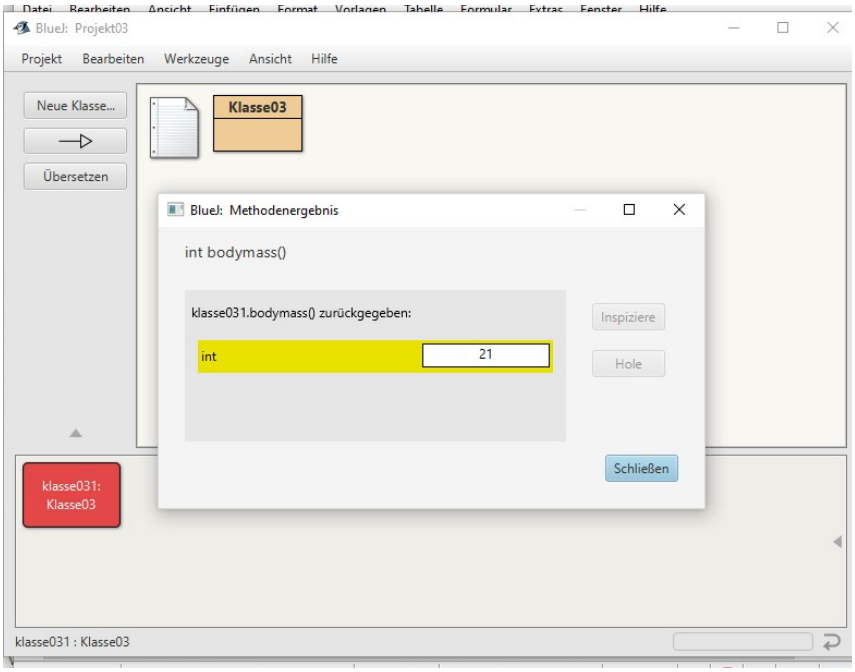
Ein professionelles Programm benötigt natürlich eine grafische Oberfläche.
Hier nur ein einfaches Beispiel, da man vorhandene Klassen benötigt. BlueJ ermöglicht dies zwar, unterstützt es aber nicht besonders. Man kann Projekt02 überspringen.
Ansonsten wie bei Projekt01…
Zunächst geht je Fenster für die Eingabe der Länge und die Breite auf.
So sieht dann die Ausgabe aus, es wurde ein Eingabefenster „missbraucht“.



```
import javax.swing.JOptionPane;//normalerweise nicht vorhandenes muss importiert werden
public class Klasse02 {
   String la;
   String br;
   String in="";
   String um="";
   String f="";
   public Klasse02()
   {
      la = JOptionPane.showInputDialog("Laenge");//so geht's
      br = JOptionPane.showInputDialog("Breite");
     }
   void inhalt_umfang()
     {
      try//bei eigenen Eingaben können Fehler auftreten
      {
      int lai=Integer.parseInt(la);//Umwandlung String in Ganzzahl
      int bri=Integer.parseInt(br);
      int ini, umi;
         ini=lai*bri;
         in=String.valueOf(ini);//Umwandlung Zahl in String
         umi=2*(lai+bri);
         um=String.valueOf(umi);
      }
      catch(Exception e)
      {
       in="Fehler";
       um="Fehler";
      }
     JOptionPane.showMessageDialog(null, "Ergebnis: " + "Inhalt="+in+" Umfang="+um);
      System.exit(0);

      }
}


public class Klasse02_p
{
 public static void main(String[] args)
 {
    Klasse02 OKlasse02=new Klasse02();
    OKlasse02.inhalt_umfang();
   }
}
```

Bis Projekt05 verzichten wir auf eine main-Klasse, da man innerhalb BlueJ diese nicht braucht.
Wir ändern jetzt das Rechteckprogramm so ab, dass es zum Bodymassprogramm wird.
Dabei ergibt sich das Bodymass, indem man das Gewicht in Kilogramm durch die Größe in Meter teilt. Da man hier bereits mit Eingabefehlern zu rechnen hat,wird in Zentimeter gerechnet, so erklärt sich der Faktor 10000=100*100.
Ansonsten alles wie bei Projekt01.

```
public class Klasse03
{
   // Instanzvariablen
   private int laenge;
   private int gewicht;
   /**
    * Konstruktor für Objekte der Klasse Klasse03
    */
   public Klasse03(int la, int gw)
   {
      // Instanzvariable initialisieren
      laenge=la;
      gewicht=gw;
   }

   //Methode 1
   public int bodymass()
   {
      return 10000*gewicht/(laenge*laenge);
   }

}
```



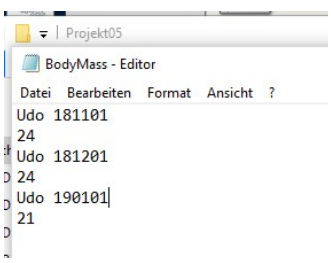
Nach Rechtsklick auf Klasse03 und New … wurde 180 für die Länge und 70 für das Gewicht eingegeben, dann Rechtsklick auf das rote Objekt und int bodymass() wählen.
Im folgenden geht dies dann abgesehen von Bezeichnungen immer so.

Als kleine Ergänzung kommt in Projekt04 eine Prüfung auf Division durch 0 hinzu. Dies wird von BlueJ bei der Eingabe nicht erkannt, aber bei der Ausführung in der Konsole gemeldet.
Ansonsten Vorgang wie bei Projekt01.

```
public class Klasse04
{
    // Instanzvariablen
    private int laenge;
    private int gewicht;
    //Konstruktor
   public Klasse04(int la_cm, int gw_kg)
    {
        // Instanzvariable initialisieren
        laenge=la_cm;
        gewicht=gw_kg;
    }
    //Methode 1
    public int bodymass()
    {
        int bm=1;
          if(laenge*gewicht>0)//wenn wahr, dann Anweisungen in {}
        bm= 10000*gewicht/(laenge*laenge);
        else
        bm=0;
   return bm;
}
}
//das Konstrukt if(Bedingung)[Anweisungen} else {Anweisungen} ist nötig, da /0 möglich
//bei einer Anweisung sind keine Klammern erforderlich
```

Das Projekt05 kann man auch überspringen, es zeigt, wie man Daten dauerhaft auf die Festplatte schreibt. Die Daten werden in der Datei BodyMass.txt gespeichert, die im Ordner Projekt05 angelegt wird. Sie kann mit jedem Texteditor gelesen werden.
Das Lesen aus einer Textdatei funktioniert ganz ähnlich.
(im browser suchen "java textdatei lesen", dann kommen gleich Vorschläge)

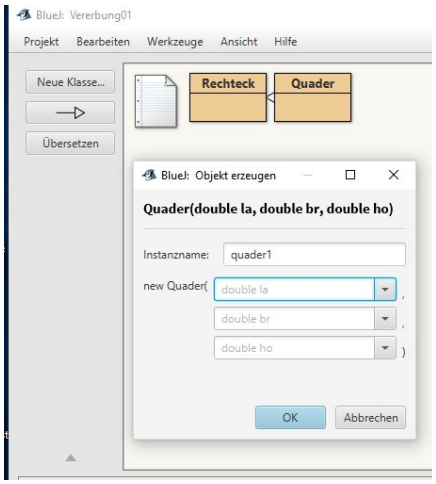
```
import java.io.File;//nötige Anweisungen einbinden
import java.io.FileWriter;//nötige Anweisungen einbinden
import java.io.IOException;//nötige Anweisungen einbinden
public class Klasse05
{
  private int laenge;
   private int gewicht;
   private String info;
  public Klasse05(int la_cm, int gw_kg, String inf_nn_jjmmtt)
   {
      laenge=la_cm;
      gewicht=gw_kg;
      info=inf_nn_jjmmtt;
   }
   public void bodymass()
   {
      int bmi=1;
      String bm="";
        if(laenge*gewicht>0)//wenn wahr, dann Anweisungen in {}
      bmi= 10000*gewicht/(laenge*laenge);
      else
      bmi=0;
      bm=String.valueOf(bmi);
  System.out.println("Bodymass="+bm);
  schreiben(bm);
}
private void schreiben(String bm){//nur intern aufrufbar
   FileWriter writer;
 File file;
   file = new File("BodyMass.txt");// File anlegen
    try {
     writer = new FileWriter(file ,true);//Datei anlegen überschreibt ggf
     writer.write(info);
     writer.write(System.getProperty("line.separator"));//Zeilenumbruch
     writer.write(bm);
     writer.write(System.getProperty("line.separator"));
     writer.flush();
     writer.close();
   } catch (IOException e) {
     e.printStackTrace();
   }
 }
}
```

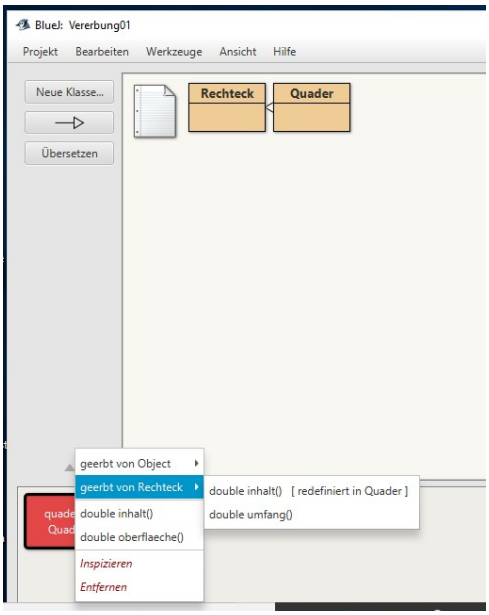


In Projekt Vererbung01 kommt zunächst wieder die Rechteckklasse aus Projekt01 zu Ehren, wobei man jetzt auch Kommazahlen eingeben kann. Bei der Eingabe ist aber ein Punkt
zu verwenden.
Es kommt jetzt aber eine zweite Klasse Quader hinzu, die von der ersten Klasse "erbt".
Dadurch bleibt sie kleiner und weniger fehleranfällig. Angeblich ist eine Marssonde am Mars vorbeigeflogen, da ein Strichpunkt gefehlt hat.
Bei umfangreicheren Projekten kann man diese auf mehrere Programmierer aufteilen.
Ansonsten Vorgangsweise wie bei Projekt01.
Die Erklärungen sind als Kommentare angefügt.

```
public class Rechteck {
double laenge;//double bedeutet Kommazahl
double breite;
//Konstruktor!
Rechteck(double la, double br)
{
laenge=la; breite=br;
 }
double umfang(){//Methode1
double umf;//Hilfsvariable, existiert extern nicht!
umf=2*(laenge+breite);
return umf;
}
double inhalt(){//Methode2
double inh;//Hilfsvariable
inh=laenge*breite;
return inh;
}
}
```

```
public class Quader extends Rechteck {
double hoehe;//Attribut, extern festlegbar und benutzbar
//Konstruktor!
Quader(double la, double br, double ho)
{
 super(la,br);
 hoehe=ho;

}
double inhalt(){//Ueberschreibt Methode der Superklasse
double inh;//Hilfsvariable
inh=super.inhalt()*hoehe;
return inh;
}
double oberflaeche(){//Methode2
double oberfl;//Hilfsvariable
oberfl=2*(laenge*breite+laenge*hoehe+breite*hoehe);
return oberfl;
}
}
```


BlueJ: Vererbung01
Projekt Bearbeiten Werkzeuge Ansicht Hilfe
Neue Klasse...
Übersetzen
Rechteck
Quader
BlueJ: Objekt erzeugen
Quader(double la, double br, double ho)
Instanzname:
quader1
new Quader(
double la
double br
double ho
OK
Abbrechen


BlueJ: Vererbung01
Projekt Bearbeiten Werkzeuge Ansicht Hilfe
Neue Klasse...
Übersetzen
Rechteck
Quader
geerbt von Object
geerbt von Rechteck
double inhalt() [redefiniert in Quader]
double umfang()
double inhalt()
double oberflaeche()
Inspizieren
Entfernen

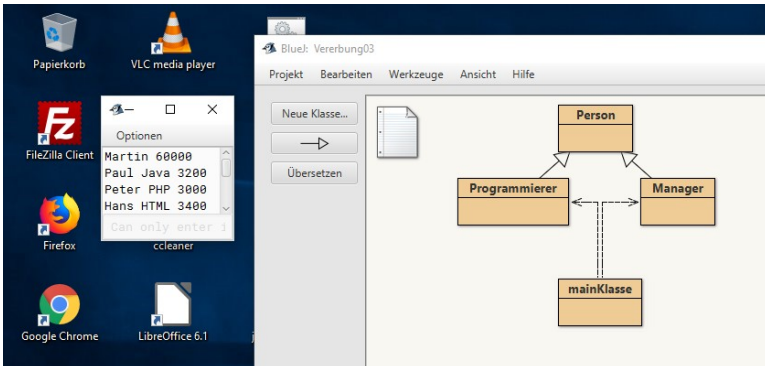
Beim Projekt Vererbung02 wurde einfach das Projekt Vererbung01 übertragen.
Die Eingabe einer Variablen vom Typ String muss mit Anführungszeichen erfolgen.

```
public class Auto
{
  String modell;
  int preis;

  public Auto(String mo,  int pr) {
    modell = mo;
    preis = pr;
  }

  public String getname() {
    return "Modell "+modell;
  }

  public int getpreis() {
    return preis;
  }
}

public class Auto_gebraucht extends Auto
{
  int kilometerstand;
  int baujahr;
  public Auto_gebraucht(String mo,int pr,int bj, int km) {
    super(mo,pr);//Aufruf des übergeordneten Konstruktors
    baujahr=bj;
    kilometerstand = km;
  }
  public String getname() {
    return super.getname()+" (km="+kilometerstand+")";
  }
  public int getpreis() {
    return preis/(kilometerstand/15000);
  }
}
```

Bei dem Projekt Vererbung03 erben die Klassen Programmierer und Manager von der Klasse Person.
Außerdem gibt es eine main-Klasse, da noch die for-Anweisung erklärt werden soll. Diese wird für Variablenlisten benötigt. (siehe Projekt Arrays)

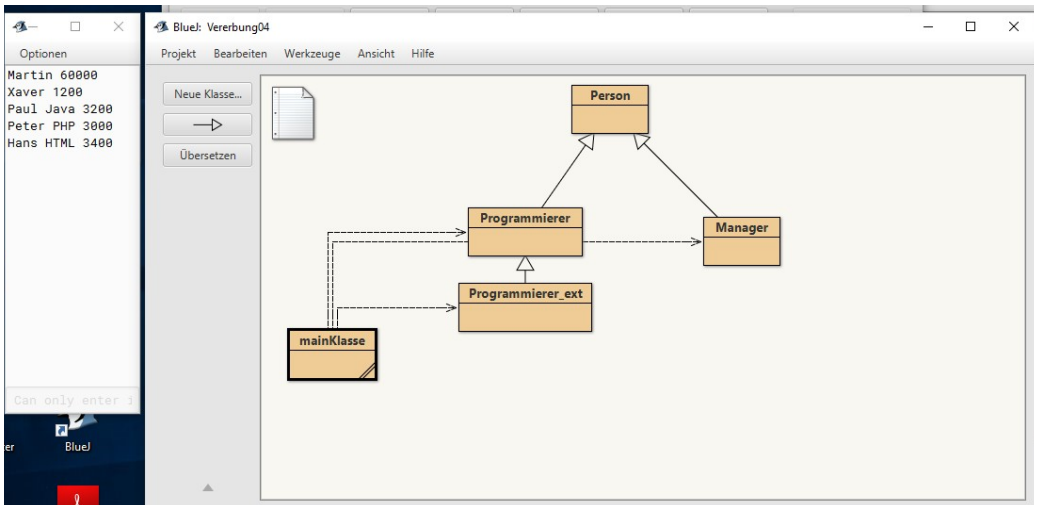
```
public class Person {
private String name;
public Person(String na) {
name = na;
}
public String getName() {
return name;
}
}

public class Programmierer extends Person {
private String lieblingsSprache;
private int stunden;
public Programmierer(String na, String sp, int st) {
super(na);
lieblingsSprache = sp;
stunden=st;
}
public String getlieblingsSprache() {
return lieblingsSprache;
}
public int getGehalt() {
return 20*stunden;
}
}

public class Manager extends Person {
private int gehalt;
public Manager(String na, int gh) {
super(na);
gehalt = gh;
}
public int getGehalt() {
return gehalt;
}
}
```

```
public class mainKlasse
{
public static void main(String[] args)
{
Manager martin = new Manager("Martin", 60000);
Programmierer[] programmierer=new Programmierer[3];//Liste programmierer mit 3 Elementen
programmierer[0]=new Programmierer("Paul", "Java",160);
programmierer[1]=new Programmierer("Peter", "PHP",150);
programmierer[2]=new Programmierer("Hans", "HTML",170);
System.out.println(martin.getName() + " " + martin.getGehalt());
for(int i=0;i<3;i++)
{
System.out.println(programmierer[i].getName() + " " + programmierer[i].getlieblingsSprache()+"
"+programmierer[i].getGehalt());
}
}
}
//Bemerkungen:
//1. Innerhalb BlueJ kann man dieses Programm als ganz einfaches Datenbankprogramm ansehen
//mit
//dem Manko, dass die Daten im Programm stehen
//2. Normalerweise werden Anweisungen nacheinander ausgeführt
//Dies kann man mit Kontrollstrukturen ändern
//if-Kontollstruktur für Programmverzweigungen
//if-Bedingung wahr?
//ja-Anweisung1
//...
//ja-Anweisungx
//Programmzähler auf Anweisungz stellen
//nein Anweisung1
//...
//nein Anweisungy
//Anweisungz
//...
//for-Kontrollstruktur für Programmschleifen
//for-Zählerstart setzen
//Anweisung1
//...
//Anweisungx
//Zählerwert erhöhen/erniedrigen
//for Bedingung wahr ?
//ja: Programmzähler auf Anweisung1 stellen
//...
```



Beim Projekt Vererbung04 erbt Programmierer_ext von Programmierer, die main-Klasse wurde angepasst, die Klassen Person, Programmierer und Manager sind gleich.

```
public class Programmierer_ext extends Programmierer
{
    public Programmierer_ext(String na, String sp, int st)
    {
     super(na, sp, st);   // Instanzvariable initialisieren
    }
public int getGehalt() {
return 15*stunden;//stunden muss in Programmierer public sein!
}
}

public class mainKlasse
{
  public static void main(String[] args) {
Manager martin = new Manager("Martin", 60000);
Programmierer_ext xaver = new Programmierer_ext("Xaver","Java",80);
Programmierer[] programmierer=new Programmierer[3];//Liste mit 3 Elementen
programmierer[0]=new Programmierer("Paul", "Java",160);
programmierer[1]=new Programmierer("Peter", "PHP",150);
programmierer[2]=new Programmierer("Hans", "HTML",170);
System.out.println(martin.getName() + " " + martin.getGehalt());
System.out.println(xaver.getName() + " " + xaver.getGehalt());
for(int i=0;i<3;i++)
{
System.out.println(programmierer[i].getName() + " " + programmierer[i].getlieblingsSprache()+"
"+programmierer[i].getGehalt());
}
}
}
```



Ergänzung Arrays:

Für die Verwendung von Listen und Tabellen gibt es den Datentyp array,
int [] liste = new int[6];//erzeugt liste {[0],liste[1],...,liste[5]}

oder

int [] liste1 ={a, b, c, d, e. f};

int [][] tabelle1 = new int[2][3]//erzeugt tabelle mit 2 Zeilen und 3 Spalten {{a, b, c},{d, e, f}}

andere Datentypen gehen auch, höherdimensional durch Anfügen weiterer [].

Projekt Projekt_Arrays, Klassen Array01, Array02 anlegen,folgendes hineinkopieren und ausprobieren.

```
public class Array01
{
int [] arr=new int[5];   // Erzeugung eines Arrays mit der Feldgröße 5
public Array01()
{
//int[] arr1 = {0:1,2,3,4}; geht auch
for(int i=0;i<5;i++)
{
arr[i] =i;
}
}
public void ausgeben()
{
for(int i=0;i<5;i++)
{
System.out.println(arr[i]);
}
}
}
//Konsole:
//0
//..
//4
```

```
public class Array02
{
int[][]  array2d=new int[7][7];
public Array02()
{
for(int i=0; i<7;i++)
{
   for(int j=0; j<7; j++)
   {
      array2d[i][j]=(i+1)*(j+1);
   }
}
}
public void ausgeben()
{
for(int i=0; i<7;i++)
{
   for(int j=0; j<7; j++)
   {
      System.out.print(array2d[i][j]+" ");
      if(array2d[i][j]<10)
      System.out.print(" ");
   }
   System.out.println();
}
}
}
/*Konsole:
1  2  3  4  5  6  7
2  4  6  8  10 12 14
3  6  9  12 15 18 21
4  8  12 16 20 24 28
5  10 15 20 25 30 35
6  12 18 24 30 36 42
7  14 21 28 35 42 49
```

array2d[3][4] wäre 20

*/

Ergänzung String:

String ist kein Datentyp sondern eine Klasse auf die man in der üblichen Weise zugreifen kann,

Projekt_String anlegen, Klassen String01, ...,String05 anlegen und entsprechende Klasse hineinkopieren, Objekte anlegen und void anzeigen() aufrufen.

```
public class String01
{
String s;//Instanzvarable
public String01()//Konstruktor
{
s="Hallo!";//Zeichenkette in Anführungszeichen, eigentlich s=new String("Hallo!");
}
public void anzeigen()
{
System.out.println(s);
}
}
//Konsole: Hallo!

public class String02
{
String s1, s2;//Instanzvarable
public String02()//Konstruktor
{
s1="Hallo";//Zeichenkette in Anführungszeichen, eigentlich s=new String("Hallo!");
s2="Welt!";
}
public void anzeigen()
{
System.out.println(s1+" "+s2);//Verkettung von Strings
System.out.println(1+1+"="+2);//1+1 wird vor Verkettung berechnet
}
}
```

```
//Konsole
//Hallo Welt!
//2=2

public class String03
{
   String s="Hallo",t="hallo";
   public void anzeigen()
   {
if (s.equals(t))//Methode der Klasse String, daher String auch groß
   System.out.println("gleich");
else
   System.out.println("ungleich");
}
}
//Konsole:
//ungleich

public class String04
{
String s;//Instanzvarable
public String04()//Konstruktor
{
s="Hallo!";//Zeichenkette in Anführungszeichen, eigentlich s=new String("Hallo!");
}
public void anzeigen()
{
System.out.println(s.length());
}
}
//Konsole:
//6
```

```
public class String05
{
String s;//Instanzvarable
public String05()//Konstruktor
{
s="1234567";
}
public void anzeigen()
{
String s1=s.substring(2, 5);
System.out.println(s1);
String s2=s.substring(0,1);//die Zählung beginnt bei 0
System.out.println(s2);
String s3=s.substring(2);
System.out.println(s3);
}
}
//Konsole:
//345
//1
//34567
```

Abschließende Bemerkung:

Das schwierigste beim Programmieren ist jedenfalls nicht der Formalismus, sondern der Algorithmus, z.B. Umwandlung von Dezimalzahlen in Dualzahlen.

$1001 \rightarrow 9$

$12 \rightarrow 1100$

Ein einfaches gui-Programm

```
import javax.swing.*;//Attribute und Methoden importieren
import java.awt.event.*;
public class BodyMassGui implements ActionListener{//implements ActionListener ähnlich extends class
   //Instanzvariable
   JTextField tf1,tf2,tf3;
   JButton b1;
   JLabel l0,l1,l2;
   //Konstruktor
   BodyMassGui(){
      JFrame f= new JFrame();//Bildschirm
      l0=new JLabel("Bodymass(Ende durch schließen des Fensters)");//Beschriftung allgemein
      l0.setBounds(10,0, 350, 50);//wo und Ausdehnung (x,y,Pixelbreit, Pixelhoch)
      tf1=new JTextField();//Textfeld 1
      tf1.setBounds(120,50,150,20);
      l1=new JLabel("Größe in cm");//Beschriftung Textfeld
      l1.setBounds(10,35, 110, 50);
      tf2=new JTextField();
      tf2.setBounds(120,100,150,20);
      l2=new JLabel("Gewicht in kg");
      l2.setBounds(10,85,110,50);
      tf3=new JTextField();
      tf3.setBounds(120,150,150,20);
      tf3.setEditable(false);//Ausgabefeld
      b1=new JButton("rechne");//Druckknopf
      b1.setBounds(120,200,100,20);
      b1.addActionListener(this);//Verknüpfung Button mit ActionListener
      f.add(l0);f.add(l1);f.add(l2);f.add(tf1);f.add(tf2);f.add(tf3);f.add(b1);//Elemente einfügen
      f.setSize(400,300);//Größe Bildschirm
      f.setLayout(null);//Standardlayout
      f.setVisible(true);//sichtbar
      f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);//beim schließen des Fensters auch Programm beenden
   }
```

```
public void actionPerformed(ActionEvent e) {//implements
java.awt.event.ActionListener.actionPerformed
        String s1=tf1.getText();//Eingabe holen
        String s2=tf2.getText();
        int a;int b;
        try{//Fehler abfangen
        a=Integer.parseInt(s1);//In int wandeln
        b=Integer.parseInt(s2);
        int bm=0;
        bm=10000*b/(a*a);
        String result=String.valueOf(bm);//in String wandeln
        tf3.setText("Bodymass="+result); }//Ausgabe
        catch(Exception f){
        tf3.setText("Fehler");
        }
    }
public static void main(String[] args) {//Hauptprogramm
new BodyMassGui();//erzeugt Objekt aus Klasse
}
 }
```

//Quellcode https://www.ckaden.de bei downloads java mit bluej ganz am Ende der pdf-Datei