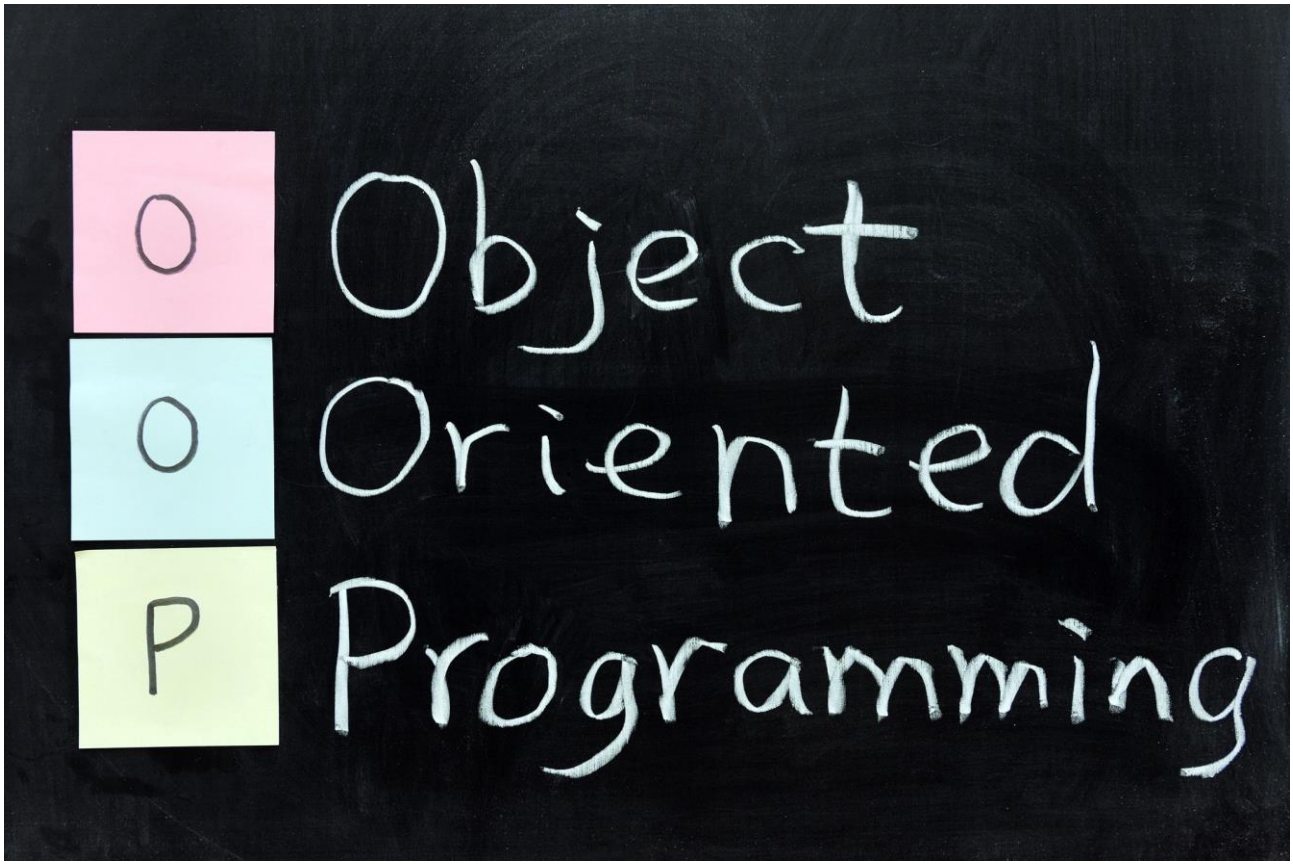
BS Info München

Objektorientierte Programmierung

Ein Einstieg in die Objektorientierte Programmierung



Stand: **08.05.2019**
Autor: **M. Völkl**

1. Einstieg in die objektorientierte Programmierung

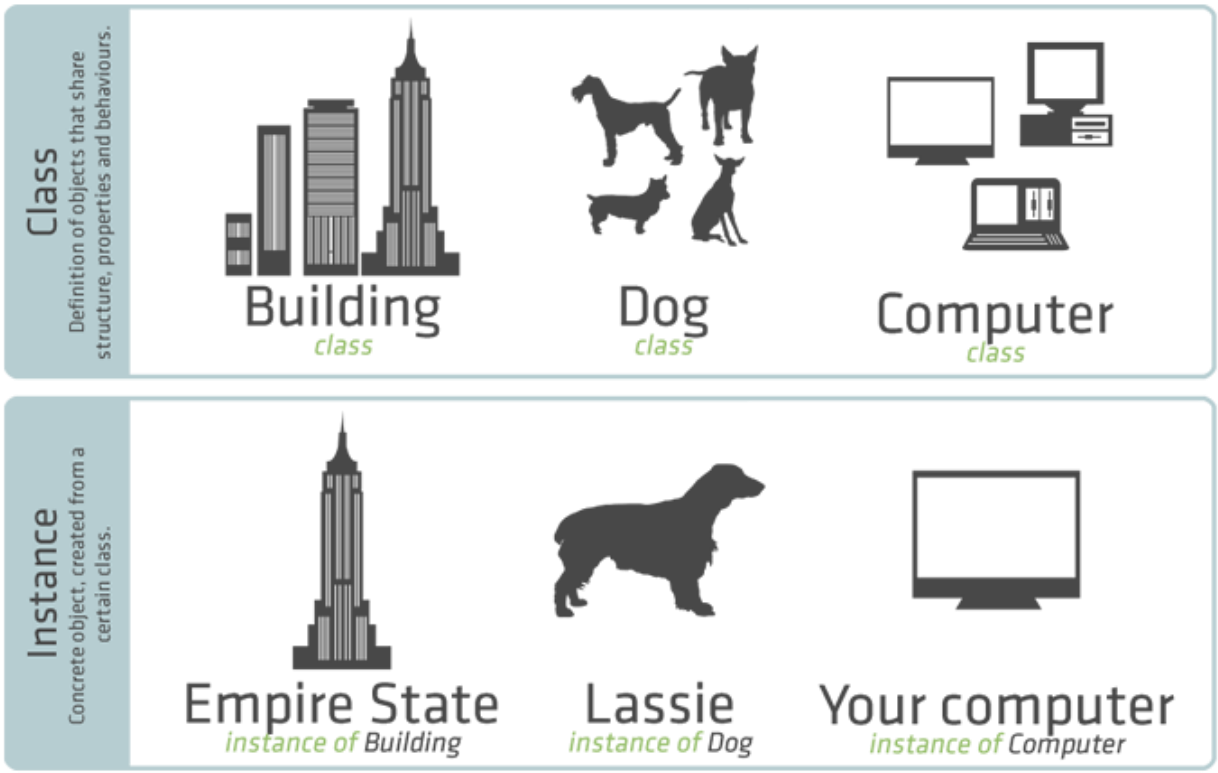
Objekte

Ein Objekt (auch Instanz genannt) bezeichnet in der objektorientierten Programmierung (OOP) ein Exemplar eines bestimmten Datentyps oder einer bestimmten Klasse. Objekte sind konkrete Ausprägungen („Instanzen“) eines Objekttyps und werden während der Laufzeit erzeugt (Instanziierung).

Instanziierung

Als Instanziierung bezeichnet man in der objektorientierten Programmierung das Erzeugen eines konkreten Objekts einer bestimmten Klasse. Während der Instanziierung eines Objekts wird in vielen Programmiersprachen ein sogenannter Konstruktor ausgeführt.

Die Instanz einer Klasse ist ein konkretes Exemplar mit konkreten Ausprägungen. Zum Beispiel könnte es in einem Programm, das mit Autos arbeitet, eine Klasse „Auto“ geben, in der die Eigenschaften (Farbe, PS, etc.) definiert sind. Wenn in einem solchen Programm ein konkretes Auto, zum Beispiel „Porsche, schwarz, Fahrzeugnummer X“, erstellt wird, das dann dem Programm zur weiteren Verwendung zur Verfügung steht, so nennt man dieses nun eine Instanz der Klasse „Auto“, und der Vorgang der Objekterstellung heißt Instanziierung.



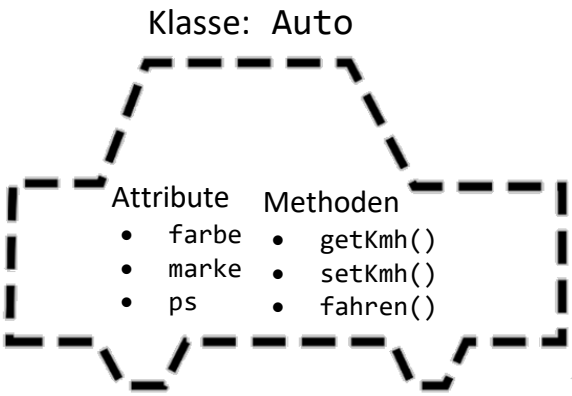
https://docs.sencha.com/extjs/6.0.2/guides/other_resources/images/classes_instances.png

Klassen

Klassen sind das wichtigste Merkmal objektorientierter Programmiersprachen. Eine Klasse definiert einen neuen Typ, beschreibt die Eigenschaften der Objekte und gibt Operatoren und Methoden den Bauplan an. Jedes Objekt ist eine Instanz (Exemplar, Ausprägung) einer Klasse.

Eine Klasse deklariert im Wesentlichen zwei Dinge:

- Attribute (was das Objekt hat)
- Methoden (was das Objekt kann)

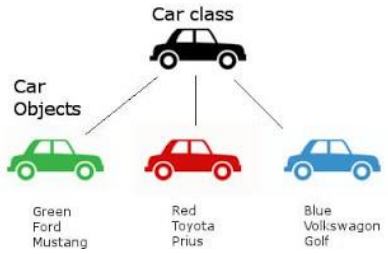


Beschreiben sie mit eigenen Worten den Unterschied zwischen einer Klasse und einem Objekt!

Beispiele zur Programmierung

```
// Bauplanklasse
public class Auto {

  public String farbe = "";
  public String marke = "";
  public String typ = "";

  public void ausgabe(){
    System.out.println("Die Marke ist: " + marke);
    System.out.println("Der Typ ist: " + typ);
    System.out.println("Die Farbe ist: " + farbe);
  }
```

```
// Ausführbare Klasse
public class AutoTest {

  public static void main(String[] args) {
    Auto meinBmw = new Auto();
    meinBmw.marke = "Bmw";
    meinBmw.typ = "320d";
    meinBmw.farbe = "schwarz";
    meinBmw.ausgabe();
  }
```



Die Instanziierung eines neuen Objektes erfolgt in der ausführbaren Klasse (Klasse mit `public static void main (String[]args)`) mit folgendem Programmcode:

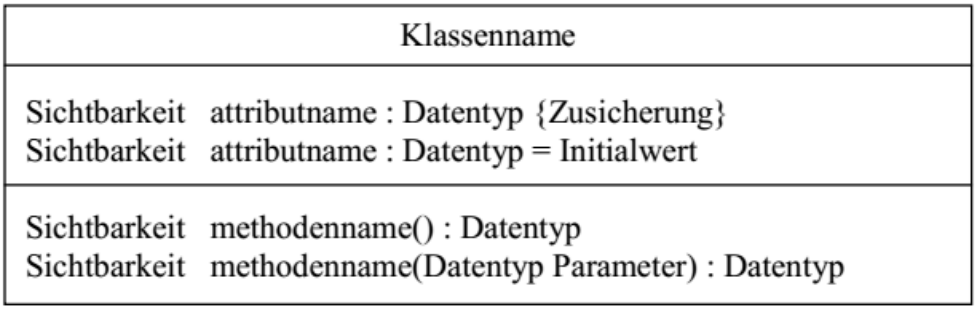
```
Auto meinBmw = new Auto();
```

Klassenname — Objektname — Konstruktor (spezielle Methode) erzeugt das Objekt

Beschreiben Sie kurz, was bei dem oben dargestellten Programmcode im Beispiel passiert!

Klassendiagramm

Ein Klassendiagramm ist ein Strukturdiagramm der Unified Modeling Language (UML) zur grafischen Darstellung (Modellierung) von Klassen sowie deren Beziehungen.



Regeln für die Benennung

- Klassenname großschreiben (erste Stelle)
- Attribute, Methoden klein schreiben (erste Stelle)
- zusammengesetzte Begriffe werden zusammengeschrieben, wobei jedes neue Wort mit einem großen Buchstaben beginnt → Kamelnotation

Zugriffsmodifizierer (Sichtbarkeit)

In der Objektorientierung kennt man so genannte Zugriffsmodifizierer, die die Rechte anderer Objekte einschränken (Kapselung) oder die ein bestimmtes Verhalten von einem Unterobjekt verlangen.

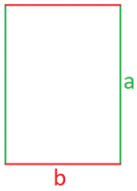
Modifizierer	Symbol im Klassendiagramm	Sichtbarkeit in der Klasse selbst und in inneren Klassen	Sichtbarkeit im Paket	Sichtbarkeit in vererbten Klassen	Sichtbarkeit in sonstigen Klassen
private	-	ja	nein	nein	nein
public	+	ja	ja	ja	ja
protected	#	ja	ja	ja	nein
default		ja	ja	nein ja (im selben Paket)	nein ja (im selben Paket)

Erstellen Sie zum Programmcode der Klasse Auto das entsprechende Klassendiagramm!

```
public class Auto {

  public String marke = "";
  public String typ = "";
  public String farbe = "";

  public void ausgabe(){
    System.out.println("Die Marke ist: " + marke);
    System.out.println("Der Typ ist: " + typ);
    System.out.println("Die Farbe ist: " + farbe);
  }
```


Programmierauftrag

Gegeben ist folgendes Klassendiagramm zur Klasse `Rechteck` und das Struktogramm zur ausführbaren Klasse `RechteckTest`. Setzen Sie die Vorgaben in der Programmiersprache Java um!



+Rechteck
+laenge:double +breite:double
+flaecheBerechnenAusgeben():void

Eingabe Länge und Breite durch den Benutzer
Aufruf der Methode zur Berechnung und Ausgabe der Fläche

2. Methoden und Rückgabewerte

Eine Methode setzt sich aus mehreren Bestandteilen zusammen. Dazu gehören der Methodenkopf und der Methodenrumpf. Der Kopf besteht aus einem Rückgabetyp, dem Methodennamen und einer optionalen Parameterliste.

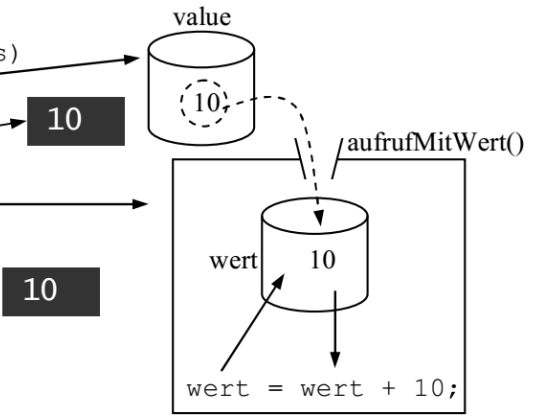
Methoden besitzen einen bestimmten Wert, der zurückgegeben wird. Die Methode `random()` der Klasse `Math` liefert eine Zahl zwischen 0 und kleiner 1 zurück. Die Methode wird über `Math.random()` aufgerufen.

Das Schlüsselwort `void` (*ungültig*, *leer*) wird anstelle eines Datentyps benutzt, um anzugeben, dass keine Daten übergeben werden. Syntaktisch wird `void` wie ein Datentyp behandelt, aber es kann nur an bestimmten Stellen stehen. So ist es zum Beispiel nicht möglich, eine Variable vom Typ `void` zu deklarieren.

call-by-value

Bei CBV werden die an eine Funktion übergebenen Daten kopiert. Es besteht dann keine Verbindung mehr zwischen den Daten beim Aufrufer und den Daten in der Funktion. Werden große Objekte auf diese Weise übergeben, so ist das sehr kostspielig in Bezug auf Rechenzeit (der Kopiervorgang) und Speicherplatz (Daten sind mehrmals vorhanden).

```
class TestCBV
{
  public static void main(String[] args)
  {
     int value = 10;
     System.out.println(value);
     KlasseA objekt = new KlasseA();
     objekt.aufrufMitWert(value);
     System.out.println(value);

  }
}

class KlasseA
{
  void aufrufMitWert(int wert)
  {
    wert = wert + 10;
  }
}
```



Wert der Variablen „value“ wird in die Variable „wert“ kopiert → Wert 10 ist doppelt vorhanden.

Methode arbeitet mit Kopie (Variable „wert“). Der Wert der Variablen value wird nicht verändert.

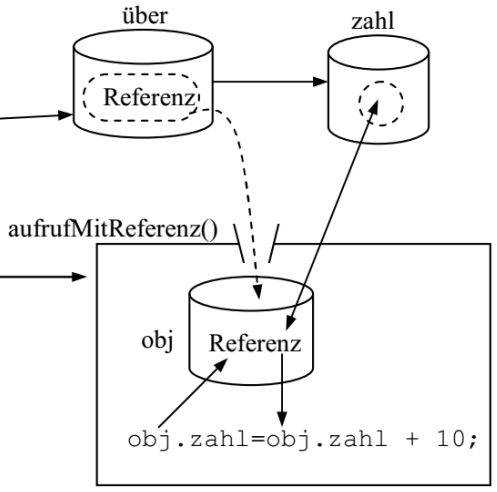
call-by-reference

Anstatt die Daten zu kopieren werden Referenzen auf die Daten übergeben. Man kann sich das als Zeiger vorstellen: Beim Aufrufer zeigt eine Variable auf ein bestimmtes Datum, in der Funktion zeigt eine andere Variable auf dasselbe Datum. Methodenaufrufe an einem so übergebenen Objekt arbeiten also auf demselben Objekt, das auch außerhalb sichtbar ist.

```
class TestCBR
{
  public static void main(String[] args)
  {
     KlasseB über = new KlasseB();
     über.zahl = 10;
     über.zahlAnzeigen();

     KlasseC objekt = new KlasseC();
     objekt.aufrufMitReferenz(über);

     System.out.println(über.zahl);
  }
}

class KlasseB
{
  int zahl;

  void zahlAnzeigen()
  {
    System.out.println(zahl);
  }
}

class KlasseC
{
  void aufrufMitReferenz(KlasseB obj)
  {
    obj.zahl=obj.zahl + 10;
  }
}
```

10

20



Wert der Referenz „über“ (Verweist auf Speicherplatz „zahl“) wird in „obj“ kopiert → Referenz (Verweis) ist doppelt vorhanden.

Methode arbeitet über die Referenz mit dem originalen Speicheplatz. Der Wert der Variablen „zahl“ wird dadurch unwiderruflich verändert (auch für den Zugriff mittel „über“).

Programmierauftrag

Ergänzen Sie das bereits entwickelte Programm, in dem Sie die Elemente des folgenden Klassendiagramms erweitern. Ergänzen Sie den Code in der ausführbaren Klasse so, dass das Ergebnis ausgegeben wird!

+Rechteck
+laenge:double +breite:double
+flaecheBerechnenAusgeben():void **+umfangBerechnen():double**

Erstellen sie im Anschluss ein neues Projekt mit dem Namen `Quader`. Berücksichtigen Sie das nachfolgende Klassendiagramm!

+Quader
+laenge:double +breite:double +hoehe:double
+flaecheBerechnenAusgeben():void +umfangBerechnen():double +volumenBerechnen():double

3. Parameter einer Methode übergeben

In vielen Fällen kann es erforderlich sein, einen bestimmten Wert einer Methode direkt zu übergeben. Man spricht von einem sogenannten **Übergabeparameter**. Der Übergabeparameter wird im **Stack** gespeichert. Diese Parameter sind nur in der Methode vorhanden, in der sie deklariert wurden.

Überblick

```
public datentypDesRueckgabewertes methodenName(datentyp Uebergabeparameter) {

      …

      return Rueckgabewert

}
```

Beispiel

Die Klassen `QuadratTest` und `QuadratTest2` greifen auf die Methode `flaecheBerechnen(double laenge)` der Bauplanklasse `Quadrat` zu. In diesem Fall muss der Methode ein Wert (die Länge) übergeben werden. Im ersten Fall der Wert `120` und im zweiten Fall der Inhalt der Variable `eingegebenenLaenge`, die ein Benutzer über die Tastatur eingibt.

```
//Bauplanklasse
public class Quadrat {
      double flaecheBerechnen(double laenge){
            return Math.pow(laenge,2);
      }
}

//eine ausführende Klasse mit Methodenaufruf
class QuadratTest {
      public static void main(String[] args) {
            Quadrat einQuadrat = new Quadrat();
            double flaeche = einQuadrat.flaecheBerechnen(120);
            System.out.println(flaeche);
      }
}
```

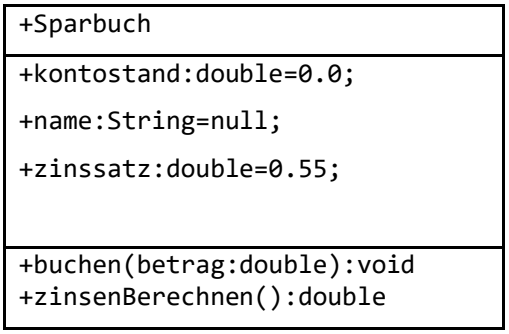
//eine weitere ausführende Klasse mit Methodenaufruf

```
import java.util.Scanner;
public class QuadratTest2 {
   public static void main(String[] args) {
      Quadrat einQuadrat = new Quadrat();
      Scanner berta = new Scanner(System.in);
      double eingegebeneLaenge = berta.nextDouble();
      double flaeche = einQuadrat.flaecheBerechnen(eingegebeneLaenge);
      System.out.println(flaeche);
      berta.close();
   }
}
```

Aufgabe (Sparbuch.java - SparbuchTest.java)

In einer Bank werden die Konten (Sparbuch) der Kunden mit Hilfe eines objektorientierten Computerprogramms verwaltet. Vereinfacht dargestellt besitzt jeder Kunde ein Sparbuch. Daher müssen von jedem Kunden der Name und der anfängliche Kontostand bekannt sein.

1. Wie lautet der entsprechende Java-Programmcode für das folgende Klassendiagramm? Den Inhalt des Methodenkörpers (in den geschweiften Klammern {}) lassen Sie frei.



_____

_____

_____

_____

_____

_____

2. Wie nennt man folgende Variablen?

`kontostand` _____

`name` _____

`zinssatz` _____

`betrag` _____

3. Wie lautet der Programmcode in einer ausführbaren Klasse (SparbuchTest.java) eine Instanz aus der Klasse Sparbuch zu erstellen?

_____

4. Sie wollen den zu buchenden Betrag in die Variable `betrag` speichern. Wie lautet der Programmcode nachdem Sie einen Scanner mithilfe der Programmzeile
   `Scanner berta = new Scanner(System.in)` in der ausführbaren Klasse erstellt haben?

_____

5. Wie lautet der Programmcode in einer ausführbaren Klasse (SparbuchTest.java) um die Methode `buchen(betrag:double)` aufzurufen?

_____

Programmierauftrag

Erstellen Sie ein Programm, das den Benutzer auffordert seinen Namen und den gewünschten Betrag zur Buchung einzugeben. Wird der Betrag mit einem Minus eingegeben, wird der Betrag vom Konto abgebucht, ansonsten addiert.
Der Kontostand soll für unser Beispiel den Wert 1000 besitzen.

- Erstellen Sie eine "Bauplanklasse" (Sparbuch.java), die die beschriebenen Funktionalitäten besitzt. Berücksichtigen Sie auch das gegebene Klassendiagramm!
- Erstellen Sie eine ausführbare Klasse (SparbuchTest.java), in der ein Kundenobjekt erzeugt wird.
- In der Klasse Sparbuchtest sollen zudem Bildschirmausgaben (für: Namen, Kontostand, Zinsen) stattfinden.
  Bsp.: Einzahlung / gedeckte Auszahlung
  Bsp.: nicht gedeckte Auszahlung

Methoden

`buchen(betrag:double):void`

Über die Methode buchen() kann der Kunde Ein- bzw. Auszahlungen vornehmen (Auszahlungen werden durch ein vorangestelltes Minuszeichen „–“ beim Buchungsbetrag dargestellt).

`zinsenBerechnen():double`

Das Konto wird mit dem Zinssatz (0.55%) verzinst. Durch Ausführen der Methode `zinsenBerechnen()` werden die Zinsen für den aktuellen Kontostand ermittelt und an die aufrufende Stelle zurückgegeben.

Zusatzaufgabe 1

Die Bank zahlt den gewünschten Betrag vom Konto nur aus, wenn dieser auch in voller Höhe gedeckt ist (Kontostand >= Auszahlungsbetrag).
Eine Ausgabemeldung soll den jeweiligen Vorgang dokumentieren ("Buchung erfolgreich" / "Buchung nicht möglich").

Zusatzaufgabe 2 - für Experten

Das Programm soll nach erfolgreicher Buchung nicht gleich beendet sein. Der Benutzer kann entscheiden ob er noch eine weitere Buchung vornehmen möchte oder ob das Programm beendet werden soll.

Programmierauftrag – BMI Berechnen

Ein Programm soll das Körpergewicht eines Patienten überprüfen. Dazu müssen vom Patienten neben dem Namen und dem Geschlecht, das aktuelle Gewicht (in kg), sowie die Körpergröße (in m) bekannt sein.
Um das aktuelle Körpergewicht richtig bewerten zu können, kann man auf den Body-Mass-Index (BMI) zurückgreifen. Dieser berechnet sich nach der Formel:
BMI = Körpergewicht in kg / (Körpergröße in m • Körpergröße in m)

BMI		
m	**w**	**Bewertung**
< 20	< 19	Untergewicht
20 <= bmi < 25	19 <= bmi < 24	Normalgewicht
25 <= bmi < 30	24 <= bmi < 30	Übergewicht
30 <= bmi < 40	30 <= bmi < 40	Starkes Übergewicht (Adipositas)
>= 40	>= 40	Extremes Übergewicht

Das Programm soll nun sowohl den BMI (Methode `bmiBerechnen()`) wie auch das Idealgewicht (Methode `idealBerechnen()`) berechnen (vereinfacht: Idealgewicht (Mann): BMI von 22, Idealgewicht (Frau): BMI von 21). Zudem soll eine Diagnose nach dem obigen Bewertungsschema (Methode `zustandBestimmen`) ermittelt werden.

Patient
+name:String +geschlecht:char +gewicht:double +groesse:double
+bmiBerechnen():double +idealBerechnen():double +zustandBestimmen(bmi:double):String

`bmiBerechnen() : double`
Methode zur Berechnung des BMI-Wertes nach oben beschriebener Formel
Rückgabewert: BMI-Wert als double

`idealBerechnen() : double`
Methode zur Berechnung des Idealgewichts nach oben beschriebenen Informationen
Rückgabewert: Idealgewicht als double

`zustandBestimmen(bmi:double) : String`
Methode zur Auswertung des BMI-Wertes nach oben dargestellter Tabelle.
Übergabewert: bmi-Wert als double
Rückgabewert: Bewertungstext als String

Erstellen Sie die Klassen (Patient.java und PatientenTest.java) entsprechend der gegebenen Beschreibungen!

Erweiterung (optional)
Neben der Gewichtsbewertung über den BMI kann ein Idealgewicht auch nach Paul Broca (franz. Arzt 1824–1880) berechnet werden: Idealgewicht (in kg) = Körpergröße (in cm) – 100.
Erweitern Sie die Klasse Patient um eine Methode `idealBorca()` für diese Variante der Idealgewichtsbestimmung. Verändern Sie die Klasse `PatientenTest`, so dass der „Borca-Idealgewichtswert“ unter dem bestehenden Idealgewichtswert angezeigt wird.

Programmmierauftrag – Goldrausch (Optional)

Ein Programm soll zur Wertbestimmung von Gegenständen aus Gold dienen. Hierzu sind folgende Klassen geplant:

`class Gold`

Eigenschaft

`unze` als `double`

Gewicht des zu bestimmenden Gegenstandes (Maßangabe in Feinunze)

Methoden

`grammInUnze`

Methode ohne Rückgabewert.

Der Methode wird ein Wert in Gramm übergeben. Dieser wird in einen entsprechenden Feinunzenwert umgerechnet und der Eigenschaft unze zugewiesen.

(1 Feinunze (oz. tr., *troy ounce*) = 31.1034768 Gramm (g) Quelle: wikipedia.de)

Parameter: Grammwert als `double`

`bekommeGoldwert`

Methode mit Rückgabewert.

Der Methode wird der tagesaktuelle Goldpreis (in € je Feinunze) übergeben. Mit diesem und dem Gewichtswert (Konstante unze) wird der Goldwert ermittelt und an die aufrufende Stelle zurückgegeben.

Parameter: `Goldtagespreis` als `double`

Rückgabe: `Goldwert` als `double`

`class Goldrausch`

Methode

main

Methode ohne Rückgabewert.

In der Methode wird ein Objekt der Klasse Gold erzeugt.

Mit Hilfe von Testwerten wird die Funktionalität der Klasse Gold überprüft.

Parameter: String-Array

Erstellen Sie das **Klassendiagramm** für die Klasse Gold

Erstellen Sie die Klassen entsprechend der gegebenen Beschreibungen!

4. Konstruktor

Auszug aus: https://javabeginners.de/Grundlagen/Konstruktor.php

Konstruktoren sind spezielle methodenähnliche Klassenstrukturen, die den Namen ihrer Klasse tragen und beim Erzeugen von Objekten der Klasse über das Schlüsselwort new aufgerufen werden.
Ein Konstruktor dient dazu, ein neu gebildetes Objekt einer Klasse in einen definierten Anfangszustand zu versetzen. Welcher dies ist, hängt davon ab, welcher Konstruktor bei der Objektbildung aufgerufen wird. Wird lediglich ein leerer (Standard-) Konstruktor ohne Parameterübergabe benötigt, so muss dieser nicht ausdrücklich angegeben werden. Er wird dann automatisch erzeugt. Allerdings ist dies nur der Fall, solange kein weiterer Konstruktor deklariert wurde. Werden neben weiteren Konstruktoren auch ein leerer Konstruktor benötigt, so muss dieser explizit angegeben werden.
Kontruktoren tragen immer den Namen der Klasse und besitzen keinen Rückgabewert.

Bauplanklasse:

```
public class KonstruktorClass {

    private int x=0;

    // Standardkonstruktor muss nicht angegeben werden, macht aber Sinn
    public KonstruktorClass() {

    }

    // zweiter Konstruktor
    public KonstruktorClass(final int i){
        this.x=i;
    }

}
```

Das Beispiel zeigt eine Klasse mit zwei Konstruktoren. Der erste ist ein Standardkonstruktor, der zweite dient zur Instanzierung einer int Instanzvariablen. Beim Aufruf dieses Konstruktors wird somit ein int-Attribut bereits beim Erzeugen einer Instanz initialisiert. Soll also ein Objekt der Klasse KonstruktorClass erzeugt werden, kann dies ohne oder mit Angabe eines Anfangswertes für die Instanzvariable x geschehen:

Auszug aus der ausführbaren Klasse:

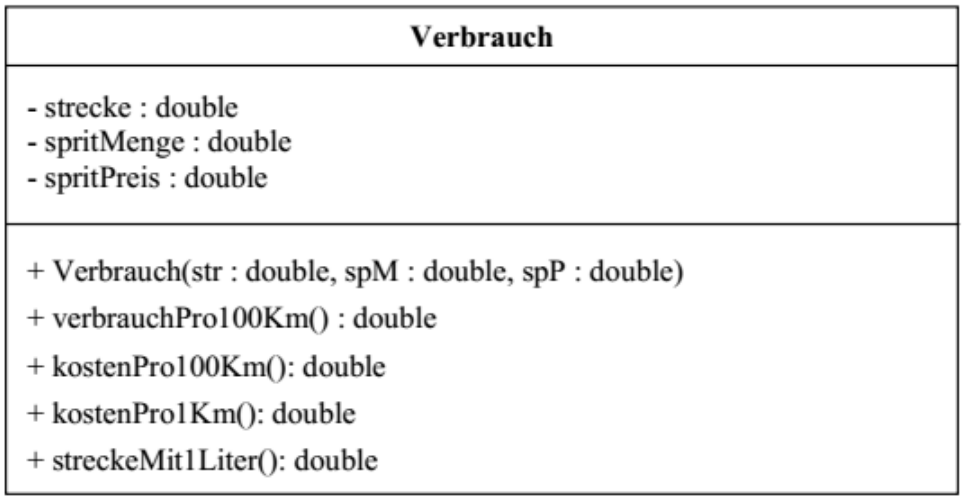
```
KonstruktorClass konst = new KonstruktorClass();
```

oder:

```
KonstruktorClass konst = new KonstruktorClass(2);
```

Programmierauftrag – Spritpreis berechnen

Ein Programm soll Autofahrer/-innen helfen den Kraftstoffverbrauch ihres Autos zu bestimmen.



Klasse „Verbrauch“

`strecke, spritMenge, spritPreis : double`

Eigenschaften der Verbrauchsberechnung

`+ Verbrauch(str:double,spM:double,spP:double)`

Konstruktor. Erzeugt ein Verbrauchsobjekt, wobei die Eigenschaften angelegt und die übergebenen Werte den entsprechenden Variablen zugewiesen werden.

`str → strecke  spM → spritMenge  spP → spritPreis`

`+ verbrauchPro100Km() : double`

Die Methode berechnet den Verbrauch pro 100 km aufgrund der Eigenschaftswerte.
Die Methode gibt den Verbrauchswert zurück.

`+ kostenPro100Km() : double`

Die Methode berechnet die entstehenden Kosten bei einer Strecke von 100 km aufgrund der Eigenschaftswerte. Die Methode gibt den Kostenwert zurück.

`+ kostenPro1Km() : double`

Die Methode berechnet die Kosten für 1 km aufgrund der Eigenschaftswerte.
Die Methode gibt den Kostenwert zurück.

`+ streckeMit1Liter(): double`

Die Methode berechnet den Strecke, die mit einem Liter Sprit zurückgelegt werden kann bei entsprechendem Verbrauch aufgrund der Eigenschaftswerte.
Die Methode gibt den Streckenwert zurück

Klasse „Verbrauchsrechner“

In der Klasse „Verbrauchsrechner“ soll ein Objekt aus der Klasse „Verbrauch“ erzeugt werden. Beachten Sie, dass hierzu der Konstruktor mit Übergabewerten verwendet werden muss. Anschließend sollen alle Methoden der Klasse aufgerufen und die ermittelten Werte angezeigt werden.

Verbrauchsrechner
+main(args:String)[]):void {static}

Erweiterung

Verändern Sie im Programm die Variablenbezeichner für die Übergabeparameter des Konstruktors nach folgender Vorgabe:

Variablenname ALT	Variablenname NEU
str	strecke
spM	spritMenge
spP	spritPreis

Verbrauch
- strecke : double - spritMenge : double - spritPreis : double
+ Verbrauch(strecke : double, spritMenge : double, spritPreis : double) …

Compilieren Sie die Klasse Verbrauch und führen Sie mit der Klasse Verbrauchsrechner einen Testlauf durch. Welche Auswirkung hat die Veränderung? Begründen Sie ihre Aussage!

_____

_____

_____

Fügen Sie folgenden Quellcode in die Klasse Verbrauchsrechner ein:

```
Verbrauch test = new Verbrauch();
```

Lässt sich die Klasse Verbrauchsrechner nun compilieren? Begründen Sie ihre Aussage!

_____

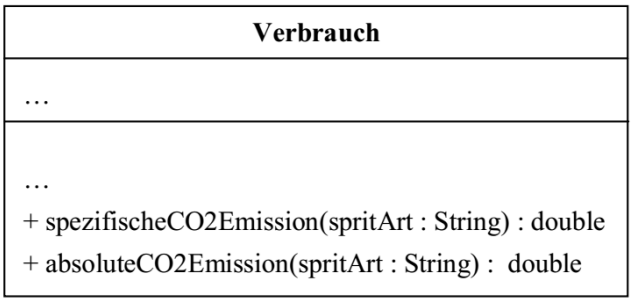
_____

Welche Veränderungen müsste man ggf. in der Klasse Verbrauch durchführen, damit die Klasse Verbrauchsrechner wieder compilierbar wird?

_____

Erweiterung 2

„Die Europäische Union hat sich verpflichtet, die Treibhausgasemissionen bis 2020 um mindestens 20 % (gegenüber dem Stand von 1990) zu reduzieren. Mit einem Anteil von ca. 26 % trägt der Verkehr erheblich zu den CO2 -Gesamtemissionen in der EU bei. Der Pkw-Verkehr ist dabei mit ca. 12 % für in etwa die Hälfte der Emissionen verantwortlich.“[1] Daher hat die EU für PKWs Zielwerte für den CO2-Ausstoß festgelegt. Zur Zeit ist für Neuwagen ein Wert von 130 Gramm CO2 je Kilometer festgelegt. Bis zum Jahr 2020 ist ein Wert von 95g/km angestrebt. Erweitern Sie ihre Klasse Verbrauch um zwei Methoden entsprechend der Beschreibung und testen Sie diese mit ihrer Klasse Verbrauchsrechner aus.



`spezifischeCO2Emission(spritArt : String) : double`

Die Methode berechnet den CO2-Ausstoß in Gramm pro Kilometer für die übergebene Kraftstoffart (mögliche Werte: Benzin, Diesel).
Berechnungsformel: Ausstoß = Kraftstoffverbrauch [l/100 km] • Faktor [kg/10 l]
Faktor für Benzin: **23,2** kg/10 l *
Faktor für Diesel: **26,2** kg/10 l * (*Werte aus Wikipedia.de)
Die Methode gibt den CO2-Emissionswert in g/km zurück.
(Bei falscher Kraftstoffart wird der Wert 0.0 zurückgegeben)

`absoluteCO2Emission(spritArt : String) : double`

Die Methode berechnet den CO2-Ausstoß in Kilogramm für die gefahrene Strecke bei der übergebenen Kraftstoffart (mögliche Werte: Benzin, Diesel).
Die Methode gibt die CO2-Menge in kg zurück.
(Bei falscher Kraftstoffart wird der Wert 0.0 zurückgegeben)

[1] www.bmu.de

5. Klassengrößen static

Eine Klassenvariable ist eine besondere Art von Variable, die innerhalb einer Klasse und außerhalb jeder Methode mit vorangestelltem Schlüsselwort **`static`** deklariert wird.

Besonderheiten

- Einer Klassenvariablen wird nur einmal pro Klasse Speicherplatz zugewiesen, da sie nur ein einziges Mal existiert (unabhängig von der Anzahl der Instanzen).
- Auf eine Klassenvariable kann zugegriffen werden, ohne zuvor ein Objekt erzeugt zu haben.

Der Zugriff erfolgt dann über den Klassennamen.

```
Klassenname.variablenname
```

Beispiel

```
public class Variablen {
	static int klassenvar = 0;
}

public class VariablenTest {
	public static void main (String[] args) {
		System.out.println(Variablen.klassenvar);
	}
}
```

Klassenmethoden

Das Schlüsselwort static kann auch in einem Methodenkopf erscheinen. Dann spricht man von einer Klassenmethode. Auf Klassenmethoden kann **zugegriffen werden, ohne** dass eine einzige **Instanz** der Klasse, in der sie deklariert wurde, existiert.

Aufruf einer Klassenmethode

```
public static typ methodenName(Übergabeparameter)
```

Hinweis
Die main-Methode muss immer als static deklariert werden, da zum Programmstart noch keine Instanzen bestehen können. Ein Zugriff ist aber über die vorhandene Klasse möglich (Klassenname muss bei der Programmausführung angegeben werden).

Beispiel

```
public class Textaus {
	public static void ausgabe(){
		String var ="Anwendungsentwicklung";
		System.out.println(var);
	}
}
public class Text {
	public static void main(String args[]){
		Textaus.ausgabe(); //Aufruf der Klassenmethode
	}
}
```

Programmierauftrag Helfer.java – RundenTest.java

Erstellen Sie eine öffentliche Klasse Helfer, die eine öffentliche Klassenmethode runden() besitzt.

`runden()`

Die Methode rundet eine beliebige Zahl auf eine vorgegebene Anzahl von Nachkommastellen. Die zu rundende Zahl so wie auch die Anzahl der Stellen sollen der Methode übergeben werden. Die Methode liefert den gerundeten Wert an die aufrufende Stelle zurück.

Parameter:	zu rundende Zahl als double
	Anzahl der Stellen als int
Rückgabe:	gerundete Zahl als double

Erstellen Sie eine Klasse `RundenTest`, mit der Sie die Methode runden() ohne Objekterzeugung austesten. Die notwendigen Werte sollen dabei unter Verwendung der Klasse Scanner eingelesen werden.

Hinweis: Bei der Lösung der Methode `runden()` soll die Potenzfunktion (`pow()`) sowie die Rundungsfunktion (`round()`) der Klasse Math eingesetzt werden

Erweiterung (optional):
Überladen des Konstruktors: `runden(double d, int stellen), runden (double d)` -> auf zwei Stellen

Programmierauftrag Helfer.java – ZufallTest.java

Erweitern Sie die Klasse Helfer um eine öffentliche Klassenmethode **mit** Rückgabewert mit dem Namen `zufallsInt()`

`zufallsInt()`

Die Methode `zufallsInt()` ermittelt einen ganzzahligen Zufallswert in einem Bereich, der durch die Übergabeparameter bestimmt wird. Der Bereich kann dabei im Minus- und Plusbereich des Integer-Datentyps liegen. Die Zufallszahl muss mit der Methode `random()` der Klasse Math bestimmt werden.

Übergabeparameter:	Anfangswert des Bereichs (als int)
	Endwert des Bereichs (als int)
	(Die beiden Werte sind Teil des Zufallsbereichs)
Rückgabewert:	ganzzahliger Zufallswert

Erstellen Sie eine Klasse `ZufallTest`, mit der Sie die Methode `zufallsInt()` austesten.

Nur für Experten: Sortierer

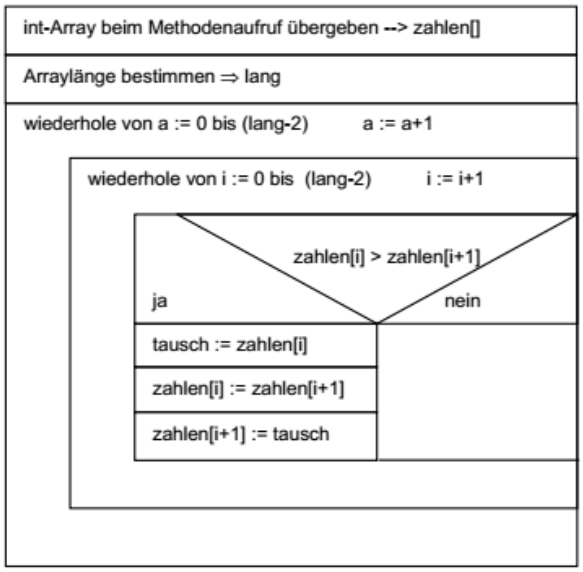
- Erweitern Sie die öffentliche Klasse Helfer um eine öffentliche Klassenmethode **ohne** Rückgabewert mit dem Namen sortieren().

  `sortieren(...)`

  Die Methode sortiert eine übergebene Menge von Integer-Werten in aufsteigende Reihenfolge (von klein nach groß) nach unten stehendem Algorithmus.

  Parameter: zu sortierende int-Werte mittels int-Array

  Sortieralgorithmus:



Sortierverfahren
Bubble-Sort
Der Algorithmus vergleicht der Reihe nach zwei benachbarte Elemente und vertauscht sie, falls sie in der falschen Reihenfolge vorliegen. Dieser Vorgang wird solange wiederholt, bis keine Vertauschungen mehr nötig sind. Hierzu sind in der Regel mehrere Durchläufe erforderlich.
Je nachdem, ob auf- oder absteigend sortiert wird, steigen die größeren oder kleineren Elemente wie Blasen im Wasser (daher der Name) immer weiter nach oben, das heißt, an das Ende der Reihe. Auch werden immer zwei Zahlen miteinander in „Bubbles" vertauscht.
(Quelle: Wikipedia)

- Erstellen Sie eine Klasse SortiererTest, mit der Sie die Methode sortieren() austesten.

  Die notwendigen Werte sollen dabei mit Hilfe der Klasse Scanner eingelesen werden.

  Hinweise
  - Bevor die zu sortierenden Zahlen durch den Benutzer eingegeben werden, soll ein Wert für die Menge der Zahlen eingelesen werden, um so die Größe des notwendigen Arrays zu erhalten.
  - Beim Aufruf der Methode muss nur der Name des Arrays (ohne „[]") übergeben werden!
  - Nach dem Methodenaufruf kann der Inhalt des Arrays am Bildschirm angezeigt werden.

Weitere Sortierverfahren (Beispiele) (Quelle: Wikipedia)

- **Insertion-Sort**
  Vorgehen ähnlich der Sortierung eines Spielkartenblatts. Am Anfang liegen die Karten des Blatts verdeckt auf dem Tisch. Die Karten werden nacheinander aufgedeckt und an der korrekten Position in das Blatt, das in der Hand gehalten wird, eingefügt. Um die Einfügestelle für eine neue Karte zu finden wird diese sukzessive (von links nach rechts) mit den bereits einsortierten Karten des Blattes verglichen. Zu jedem Zeitpunkt sind die Karten in der Hand sortiert und bestehen aus den zuerst vom Tisch entnommenen Karten.
- **Selection-Sort**
  Sei S der sortierte Teil des Arrays und U der unsortierte Teil. Am Anfang ist S noch leer, U entspricht dem ganzen Array. Das Sortieren durch Auswählen funktioniert so:
  Suche das kleinste Element in U und vertausche es mit dem ersten Element.
  Danach ist das Array bis zu dieser Position sortiert. S ist um ein Element gewachsen, U um ein Element kürzer geworden. Anschließend wird das Verfahren solange wiederholt, bis das gesamte Array abgearbeitet worden ist.