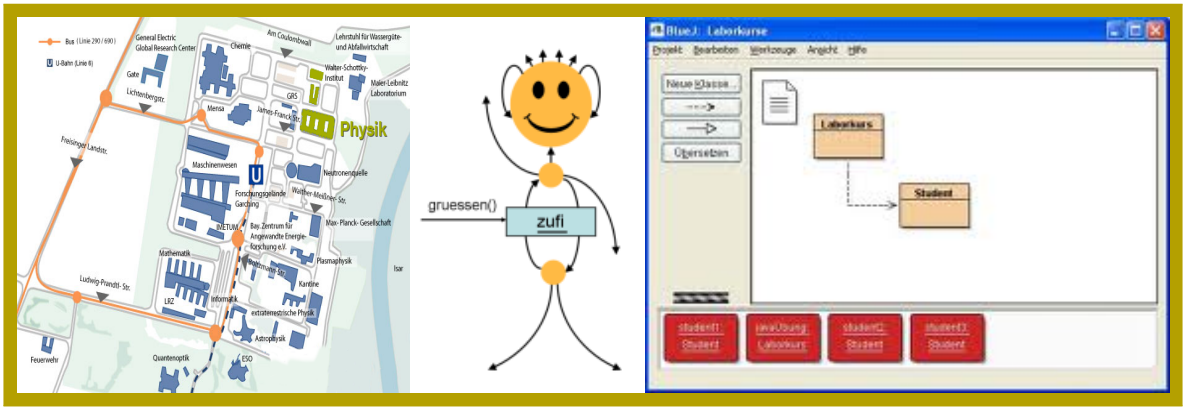
Objektorientierte Konzepte

M. Jakob

Gymnasium Pegnitz

24. September 2015



Inhaltsverzeichnis

In diesem Abschnitt

Fachkontext

Moderne Programmiersprachen wie zum Beispiel *Java, C++, html* basieren auf dem Prinzip der Objektorientierung. Dadurch werden die Programme in der Regel übersichtlicher, kürzer und sehr viel leistungsfähiger.

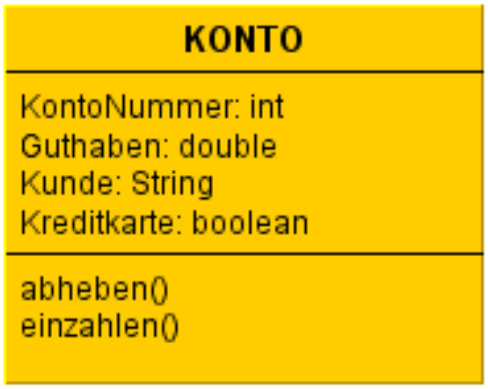
Klassen. . .

. . . kann man sich als Fabriken vorstellen, die bestimmte Objekte mit bestimmten Eigenschaften herstellt.
Beispiele . . .

- Vordefinierte Klassen — Kaffeeautomat

Klassenkarten

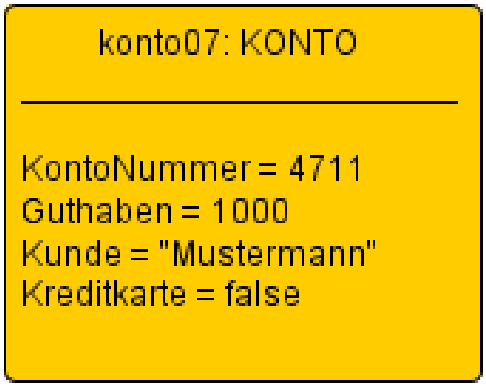
Um eine Klasse möglichst einfach und übersichtlich zu beschreiben, benutzt man eine Klassenkarte (mit scharfen Ecken).



- oben steht der Name der Klasse, in Großbuchstaben.
- dann folgen die Attribute mit ihrem Datentyp (genaue Erklärung unten).
- unten stehen die Methoden, stets mit Klammern.

Objektkarten

Eine Objektkarte haben immer abgerundete Ecken und geben die Attributwerte eines Objekts übersichtlich an.



- Oben steht der Name der Objekts, gefolgt von dem Namen der Klasse.
- Im zweiten Teil stehen die Attribute mit den zugeordneten Werten.
- Es werden keine Methoden aufgeführt.

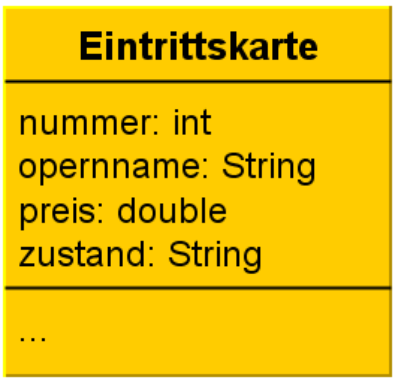
Grundbegriffe

Allgemein	Beispiel
Eine Klasse ist ein Oberbegriff für gleichartige Objekte mit gleichen Attributen.	Klasse `Konto` mit den Attributen `KontoNummer`, `Guthaben`, ...
Objekte haben einen eindeutigen Namen und haben ggf. verschiedene Attributswerte.	Das Konto `konto07` mit `KontoNummer=4711` und `Guthaben=1000`.
Attributswerte werden mit Methoden (oder Diensten) verändert.	Die Methode `abheben()` verändert den Wert des Attributs `Guthaben`.

Datentypen in Klassenkarten

Datentypen in Klassenkarten

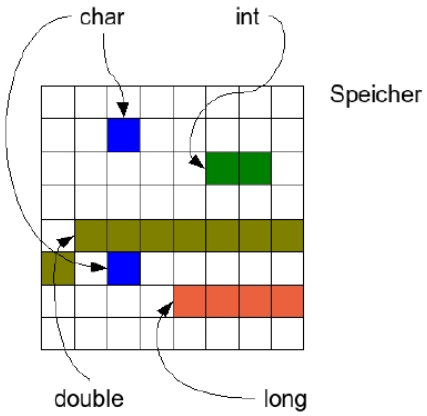
Der Datentyp wird in Klassenkarten hinter einem Doppelpunkt nach dem Attributnamen angegeben.



Warum Datentypen?

Warum Datentypen?

Attribute allgemeiner Variablen sind Container (Speicherplätze) für veränderliche Werte. Abhängig vom ihrer Art, dem sog. Datentyp, muss genug Platz (gemessen in Byte) im Datenspeicher reserviert werden.



Datentypen in Java — Zusammenfassung

Datentyp	Beschreibung	Literale	Bit
int	ganze Zahl	$[-2^{31}; 2^{31} - 1]$	32
boolean	boolescher Wert	true, false	1
double	Gleitkommazahl	$[-1{,}7 \cdot 10^{308}; 1{,}7 \cdot 10^{308}]$	64
char	Zeichen	'a','b','c',')'	16
String	Zeichenkette	"Hallo Welt"	Objekttyp
...			

Beachte

- primitive Datentypen werden klein geschrieben, Objekttypen groß (Erklärung später).
- Zeichen werden in einfache, Zeichenketten in doppelte Hochkommata eingeschlossen..

Übungen

- Ü 1.1: Die Klasse `Rechteck`

1. Zeichne eine Klassenkarte der Klasse `RECHTECK` mit mindestens 4 Attribute und 3 Methoden.
2. Zeichne die Objektkarten der Objekte `tuere`, `fussballfeld`, `buch`, der Klasse `RECHTECK` zeichnen

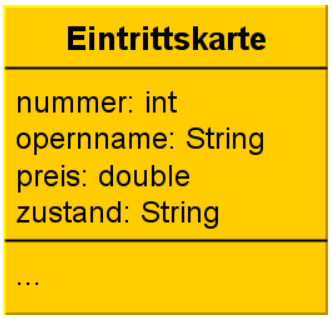
- Ü 1.2: Die Klasse `PLANET`

1. Zeichne eine Klassenkarte der Klasse `PLANET` mit den Attributen `name`, `mond`, `bewohnerzahl`, `bahnradius` und den Methoden `bewohnerzahlGeben()`, `umlaufzeitGeben()`, `bewohnerzahlAendern()`.
2. Zeichne die Objektkarten der Objekte `erde`, `jupiter` und `uranus`.

- Ü 1.3: Erstelle die Klassenkarte einer Klasse deiner Wahl und drei dazugehörige Objektkarten

In diesem Abschnitt

Von der Klassenkarte zur Klassendefinition




```
1 public class
      Eintrittskarte {
2     private int nummer;
3     private String
          opernname;
4     private double preis;
5     private String
          zustand;
6 }
```

- Ü 1.4: Java und BlueJ installieren
  `run:Hinweise/InstallationUndKonfigurationPUmgebung.pdf`
- Ü 1.5: Eintrittskarte deklarieren
- Ü 1.6: Rechteck und Planet deklarieren

Die Zugriffmodifizierer `private` und `public`. . .

. . . können Daten, nach außen hin versteckt und so vor unerlaubten Zugriffen geschützt.

Java bestizt 4 Zugriffsebenen (Zugriffsmodifizierer)

Modifier	Schutz	Zugriff auf Daten von außen
`public`	minimal	uneingeschränkt möglich
`private`	maximal	nicht möglich
`package`		für uns derzeit unbedeutend
`protected`		für uns derzeit unbedeutend

Beispiel

```
1 public class Konto{
2     private int kontoNummer;
3     private double guthaben;
4     private String kunde;
5     private boolean kreditkarte;
```

- Auf die Klasse `Konto` muss von außen zugegriffen werden (logisch); deshalb **`public`**
- Auf die Attribute soll der Zugriff von außen verweigert werden(**`public`**) weil
  - Fehleingaben abgesicherte werden müssen: z.B. kontoNummer:=3.14 oder `guthaben:=“zuWenig"`,
  - Daten konsistent bleiben müssen, z.B. Kreditkarteneinzug bei `guthaben<0`.

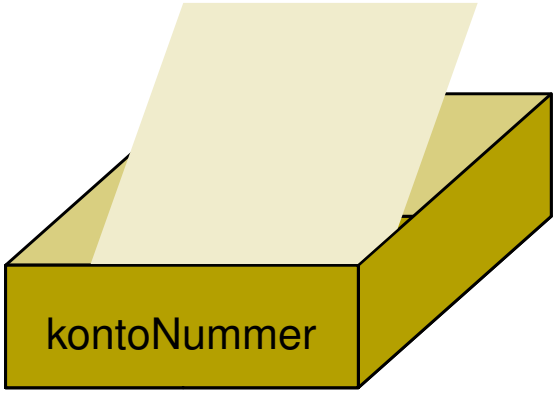
Datenkapselung, die heilige Regel der Attributdeklaration

Attribute werden stets private deklariert und der Zugriff ausschließlich über Methoden ermöglicht.

Attributimplementierung im Detail

Beispiel: `private int kontoNummer;`

- `int` stellt den Speicherplatz im Rechner bereit
- `kontoNummer` gibt dem Speicherplatz einen Zugriffs-Namen



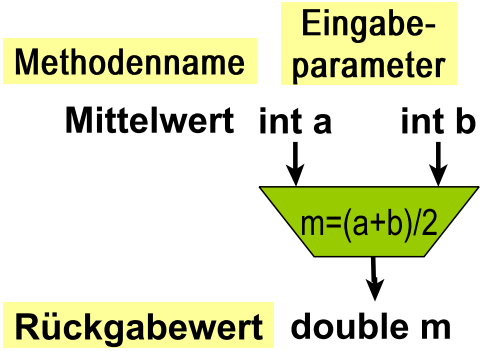
In diesem Abschnitt

Aufbau von Methoden

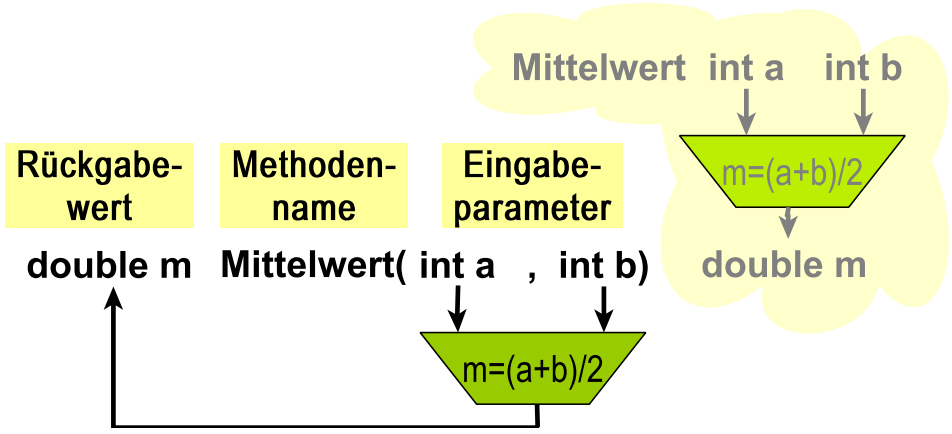
- Mittelwert(), f(x), HerdEinschalten(); HerdAusschalten(); KontostandAendern();

Methoden müssen folgende Informationen erhalten:

- Die Eingabewerte samt Datentypen
- Was sie tun sollen
- Was sie als Ergebnis zürckliefern sollen samt Datentyp



Hinführung: Syntax in Java



formal mit Modifyer:

```
public double Mittelwert(int a, int b)
```

- obige Beispiele implementieren

Aufbau von Methoden — Methodensignatur

Aufbau von Methoden — Methodensignatur

Die Methodensignatur besteht aus

- Einen Zugriffsmodifier,
- einen Bezeichner,
- eine Liste der formalen Parameter mit deren Datentypen,
- einen Ergebnistyp.

Formal:

```
<Modifier> <E-Typ> <Bezeichner> (<P-Liste>)
```

Beachte: Wenn die Methode kein Ergebnis liefert, muss als Ergebnistyp **`void`** (= leer) angegeben werden.

Übung

- Ü 1.7: Die Klasse `Planet`

Implementiere in der Klasse `Planet` alle Methoden zum Verändern und auslesen der Attributwerte `art`, `name`, `mond`, `bewohnerzahl`. Nenne sie `artSetzen(String neueArt)`, `artGeben()`, `bewohnerzahlAendern(int neuerWert)` ...
Erzeuge alle Planeten als Objekte und teste die Methoden ausführlich.

- Ü 1.8: Die Klasse `Rechteck`

Implementiere in der Klasse `Reckteck` alle Methoden zum Verändern und auslesen der Attributwerte. Gehe genauso vor, bei bei der vorherigen Aufgabe mit der Klasse `Planet`.

Übung

- Ü 1.9: Fehlersuche in der Klasse `Konto`

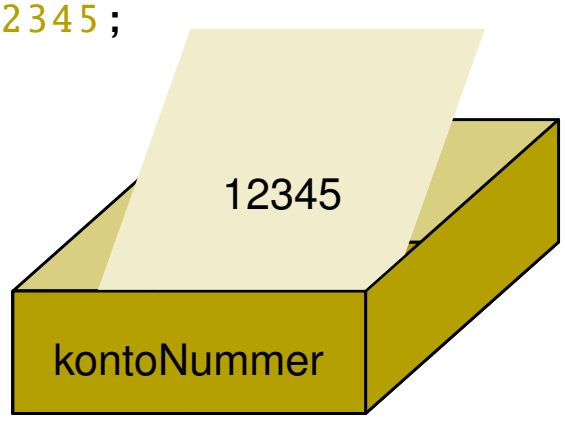
Tippe den Quellcode (siehe AB) ab, kommentiere die fehlerhaften Programmteile aus und ersetze sie durch den richtigen Code. Erkläre in einem Kommentar auch worin der Fehler besteht.

In diesem Abschnitt

Wertzuweisung — Bespiel 1

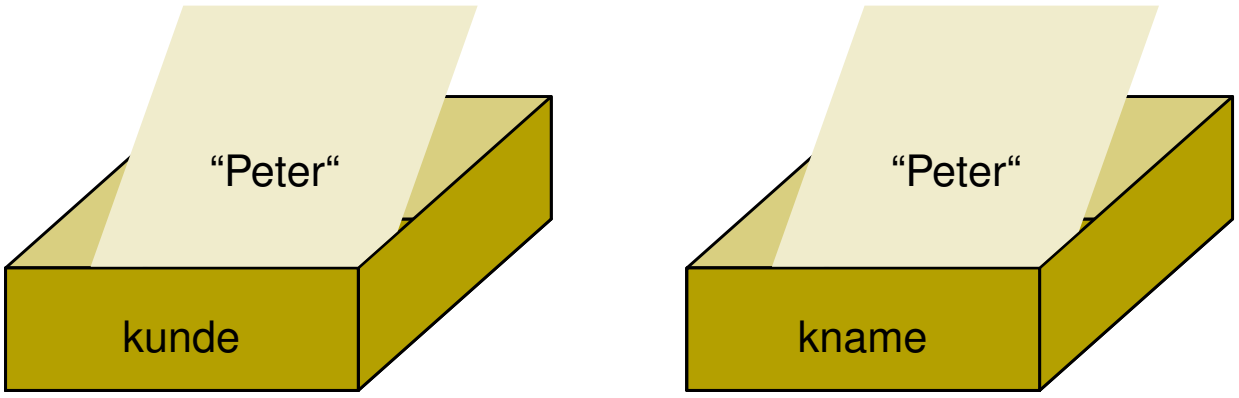
```
1  public class Konto{
2      private int kontoNummer;
3      ...
4      // Konstruktor
5      public Konto(String kname){
6          kontoNummer=12345;
7       }
8  }
```

`kontoNummer=12345` belegt die Speicherzelle mit dem Name `kontoNummer` mit dem Wert `12345`



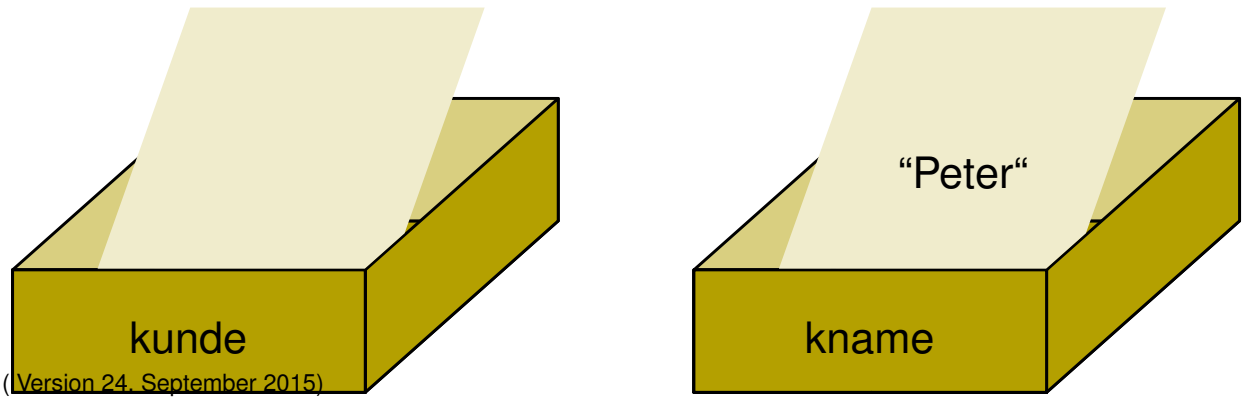
Wertzuweisung — Bespiel 2

```
1 private String kunde;
2 private String kname;
3 kname="Peter"
4 ...
5 kunde=kname;
6 ...
```



Wertzuweisung — Bespiel Holzweg

```
1 private String kunde;
2 private String kname;
3 kname="Peter"
4 ...
5 kname=kunde;
6 ...
7 }
```



Wertzuweisung

Definition (Wertzuweisung)

Bei einer Wertzuweisung der Form A=B wir der Variable mit der Bezeichnung A der Inhalt der Variable mit der Bezeichnung B zugeordnet.
Statt dem „=“-Zeichen wäre ein „⇐“ besser geeignet.

Beachte:

- A=B hat eine andere Bedeutung wie B=A.
- i=i+1 heißt: Zelle i wird ein Wert zugewiesen, der um ein größer ist, als das, was derzeit in i steht. Der Inhalt der Zelle i erhöht sich also um 1.

Übung

- Ü 1.10: Buchaufgaben
  - S.44, 2.25
  - S.45, 2.26
  - S.45, 2.27 (vorher gem. Beispiel)
  - S. 45, 2.28

In diesem Abschnitt

Objekte erzeugen — Der Konstruktor

- Objekt von Planet erzeugen und inspizieren

Definition (Konstruktor)

Der Konstruktor ist eine Methode zur Erzeugung eines Objektes. Dabei werden alle Attribute initialisiert, d.h. es wir ihnen ein fester Wert zugewiesen. Er wird mit dem Befehl `new()` aufgerufen.

Implementation von eigenen Konstruktoren

Java-Konstruktoren

Jede Klasse kann beliebig viele Konstruktoren haben, sie tragen alle den Namen der Klasse müssen sich aber in der Parameterliste unterscheiden.

Ist kein selbst geschriebener Konstruktor vorhanden, verwendet Java einen Standardkonstruktor, der alle Attribute auf `null` setzt.

Implementation von Konstruktoren — Beispiel

```
1
2
3  public class Konto{
4     private int kontoNummer;
5     private double guthaben;
6     private String kunde;
7     private boolean kreditkarte;
8
9     // Konstruktoren
10    public konto(String neuerKunde){
11        kunde=neuerKunde;
12        guthaben=0;
13    }
14
15    public Konto(String neuerKunde,int neueKNr)
          {
16        kunde=neuerKunde;
17        kontoNummer=neueKNr;
18        guthaben=0;
19        kreditkarte=false;
20    }
```

Übungen

- Ü 1.11: Die Klasse `Planet`

Implementiere in der Klasse `Planet` zwei unterschiedliche Konstruktoren. Dem ersten Konstruktor sollen als Parmeter der Name und die Bewohnerzahl übergeben werden. Der zweite Konstruktor soll alle Attributwerte als Parameter erhalten.
Erzeuge alle Planeten als Objekte und teste die Konstruktoren ausführlich.

- Ü 1.12: Die Klasse `Rechteck`

Implementiere in der Klasse `Rechteck` zwei unterschiedliche Konstruktoren. Gehe genauso vor, bei bei der vorherigen Aufgabe mit der Klasse `Planet`, wähle jedoch geeignete Parameterlisten selbst.

In diesem Abschnitt

Lokale Variablen — Beispiel

```
1  public class Kugel{
2
3      private double radius;
4
5      public double VolumenBerechnen(){
6          double volumen;
7          volumen = 4/3*3.1414*radius*radius*
               radius;
8          return volumen;
9      }
10 }
```


Lokale Variablen — allgemein

Lokale Variablen. . .

. . . werden nur in einzelnen Methoden benötigt und auch nur dort deklariert und zwar ohne Modifizierer.
Im Gegensatz zum Attribut belegen sie nur während der Ausführungszeit einer der Methode Speicherplatz.
Lokale Variablen dürfen in verschiedenen Methoden den gleichen Namen tragen, haben geringeren Speicherbedarf und erzeugen übersichtlicheren Programmcode.

Übungen

- Ü 2.1: Lokale Variablen der Klassen `Kreis`, `Kugel` und `Planet`

Vervollständige die Klassen nach den Vorgaben im Programmcode

In diesem Abschnitt

Methoden zur Benutzerinformation

Es soll eine Methode `kontodatenAnzeigen()` erstellt werden, die zu einem Konto die Kontodaten übersichtlich darstellt.

Bildschirmausgaben in Java. . .

werden durch die Befehle

- `System.out.print(daten)` und
- `System.out.println(daten)`

erzeugt. Der Unterschied liegt darin, dass nach dem zweiten Befehl der Cursor in die nächste Zeile springt, beim ersten aber nicht. Ein Zeilenumbruch kann auch mit dem Befehl `\n` erreicht werden.

Beispiel Kontodaten

```
1  public void kontodatenAnzeigen(){
2      String kopfzeile;
3      String inhaberText;
4      String kNrText;
5      String guthabenText;
6      String kartenText;
7
8      kopfzeile    = "Kontodaten\n==========
           ";
9      inhaberText  = "Inhaber: "+ kunde;
```

Bespiel Kontodaten

```
1  public void kontodatenAnzeigen(){
2     String kopfzeile;
3     String inhaberText;
4     String kNrText;
5     String guthabenText;
6     String kartenText;
7
8     kopfzeile    = "Kontodaten\n==========";
9     inhaberText  = "Inhaber: "+ kunde;
10    kNrText      = "Kontonummer: "+
         kontoNummer;
11    guthabenText = "Guthaben: "+ guthaben+ "
         Euro";
12    kartenText   = "besitzt Kreditkarte: "+
         kreditkarte;
13
14    System.out.println(kopfzeile);
15    System.out.println(inhaberText);
16    System.out.println(kNrText);
17    System.out.println(guthabenText);
18    System.out.println(kartenText);
19 }
```


Übungen

- Ü 2.2: Benutzerinfo in den Klassen `KONTO`, `PLANET` und `ELEMENT`

Ergänze in den Klassen `KONTO`, `PLANET` und `ELEMENT` jeweils die Methode `datenAnzeigen`, die die Daten eines Objektes übersichtlich auf dem Bildschirm ausgibt. Ergänze bei Bedarf zusätzlich fehlende Getter- und Settermethoden.

- Ü 2.3: Die Klassen `SEEWOLF`

Führe die Methode `Ausgabe()` der Klasse `SEEWOLF` aus und erkläre das Verhalten.

- Ü 2.4: S.51, 2.36

- Ü 2.5: S.51, 2.37

Übungen

- Ü 2.6: Die Klassen `Land` und `Stadt`

Implementiere die Klassen `Land` und `Stadt` in Java. Gehe dabei nach folgendem Schema vor:

- Sachsituation analysieren
- Klassenkarte erstellen
- Klasse deklarieren und testen
- Konstruktor festlegen und testen
- Setter- Gettermethoden implementieren und testen
- Weitere Methoden implementieren und testen

Beim letzten Punkt „Weitere Methoden implementieren und testen“ soll in beiden Klassen die Methode `datenAnzeigen()` implementiert werden, die dem Benutzer die Attribute übersichtlich darstellt.