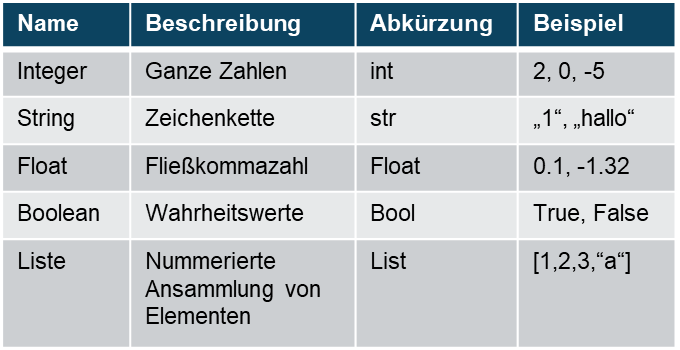
# Objektorientierte Programmierung

# und Benutzeroberfläche

**Vorkenntnisse:**

* Datentypen:
* ****Variable: Name mit verknüpften Datenobjekt
* Prozedur: Eigenständiges Unterprogramm

(unabhängig vom Hauptprogramm)

* Funktion: Prozeduren, die am Ende ein Ergebnis

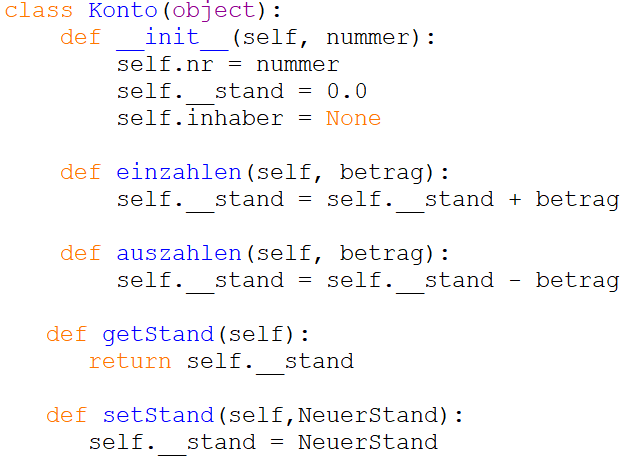
ausgeben (z.B. return)

**Objekt:**

* Eine Einheit, die Daten mit Hilfe von Attributen

****verwaltet und Operationen mit Methoden durchführt.

* + - Attribute: Variablen, die zu einem Objekt gehören.
    - ****Methoden: Prozeduren oder Funktionen, die an ein Objekt gebunden sind.



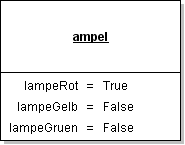
Konstruktor-methode

**Klasse:**

* Bauplan für Objekte
* Festlegung von Eigenschaften

(Attribute) und Fähigkeiten (Methoden)

**Objektdiagramm: Klassendiagramm:**

****

Objektname

Aktueller

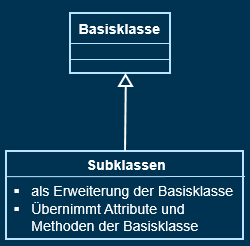
Objektzustand

Klassenname

Attribute (Datentyp)

Methoden

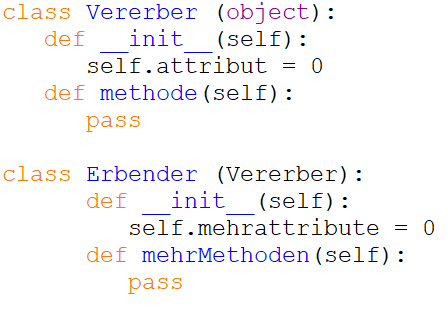
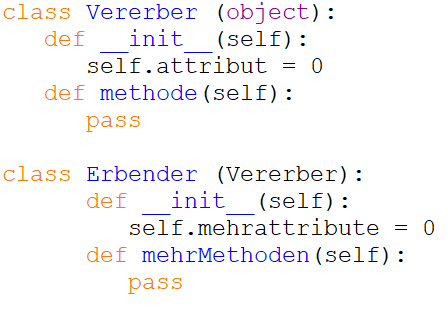
**Datenkapselung:** public (+) und private (-), um ungewünschtes Bearbeite zu verhindern

**Vererbung:**

* Subklasse übernimmt Attribute und Methoden der

Basisklasse

* Generalisierung oder Spezialisierung

****

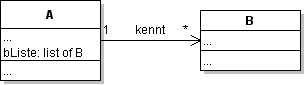
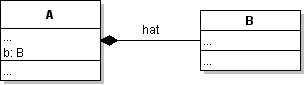
**Beziehungen zwischen Objekten**

Kennt-Beziehung:

* Objekt A kennt Objekt B, wenn B an die Konstruktormethode an A übergeben wird
* Mehrere A Objekte können ein B Objekt kennen

Hat-Beziehung:

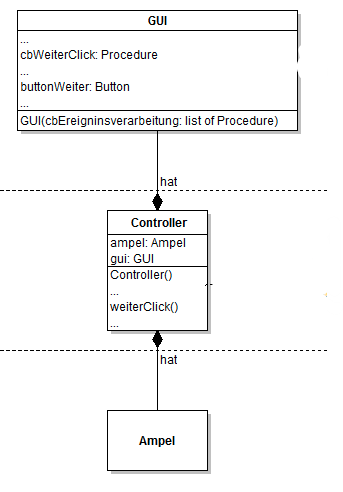
* Objekt A besitzt B, wenn B in der Konstruktor Methode A erzeugt wird (nicht unbedingt nur bei Erzeugung)
* Jedes neue Objekt A besitzt ein neu erzeugtes Objekt B

****

Grafische Benutzeroberfläche

**Model-View-Control-Prinzip**

Aufteilung der Zuständigkeitsbereiche:



* Model: verwaltet später angezeigte Daten
* View: Grafische Benutzeroberfläche

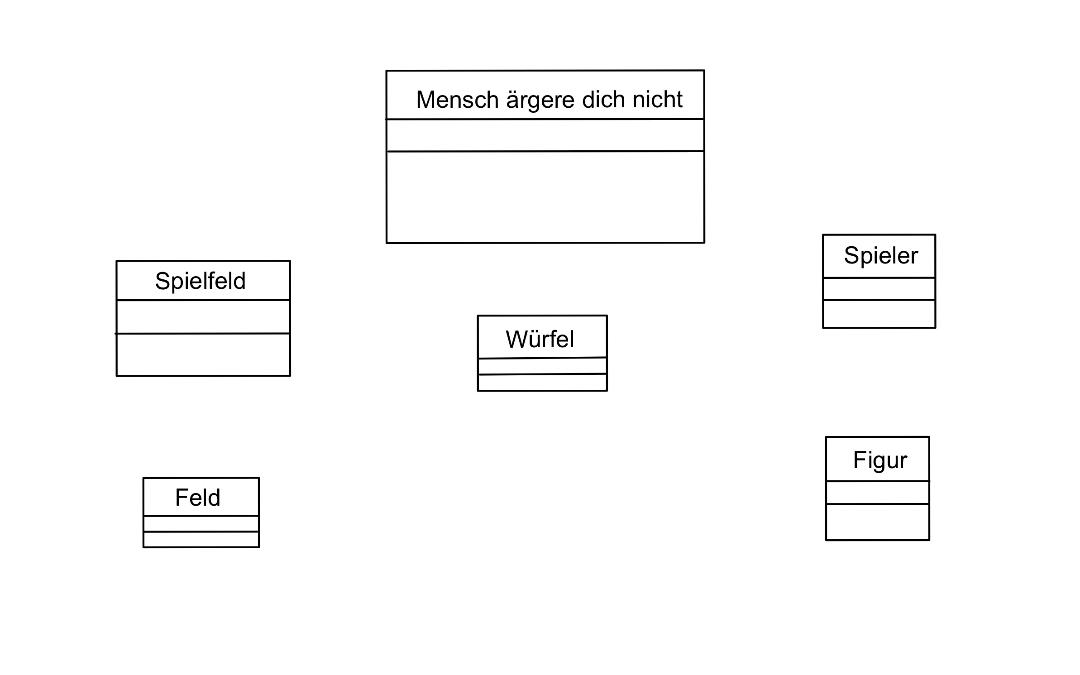
Bekommt notwendigen Daten durch

„Callback-Funktion“

* Controller: hat Model und View, strikte Trennung

Vorteile:

* Änderung, Austausch oder Erweiterung erleichtert
* Vereinfachtes und geordnetes Arbeiten möglich

****