Objektorientierte Programmierung OOP

Ronja Düffel
WS2012/13

08. Oktober 2013

Objektorientierte Programmierung

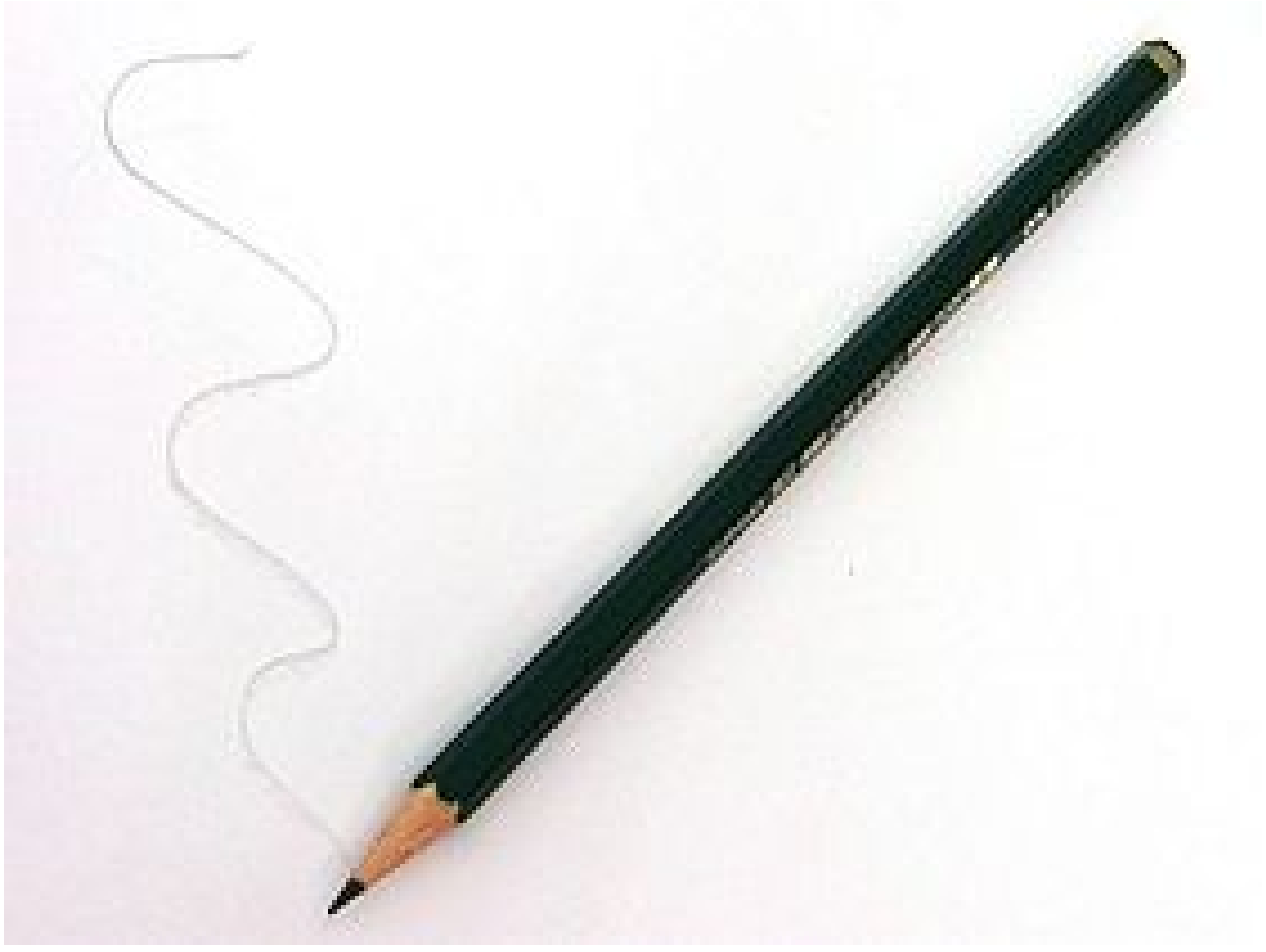
Übersicht

1. Was ist das?
2. Wie geht das?
3. Warum gibt es das?
4. Wie geht das in Python?

Übersicht

Was ist das?

- ein Programmierparadigma (Programmierstil)
- Art und Weise an ein Problem heranzugehen, es zu modellieren und somit auch zu programmieren.
- bisher: **Prozedurale Programmierung**
  - Zerlegung in Variablen, Datenstrukturen und Funktionen
  - Funktionen operieren direkt auf Datenstrukturen
- Objektorientierung: Beschreibung eines Systems anhand des Zusammenspiels kooperierender Objekte



Übersicht

Objekte

- sind überall
- werden von uns als solche wahrgenommen

reale Welt	OO-Programmierung
• Zustand • Verhalten	• Attributwerte • Methoden

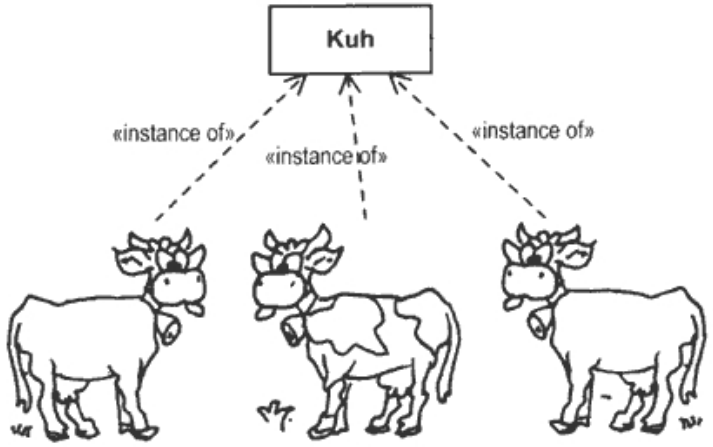
Objekte in OOP

- Zustand gespeichert in Attributwerten
- Verhalten durch Methoden
- Interaktion mit anderen Objekten durch Methoden
- Zustand ist versteckt, nur über Methoden erreichbar
- Methoden definieren Schnittstelle, über die andere Objekte mit Objekt interagieren können. (**Datenkapselung**)

Klassen und Objekte (1)

- Klasse
  - Bauplan für Objekt
  - "Idee" der Objekte
  - Definition aller Attribute und Methoden
  - Klasse allein macht noch nichts
- Objekt
  - ist konkretes Element/Instanz der Klasse

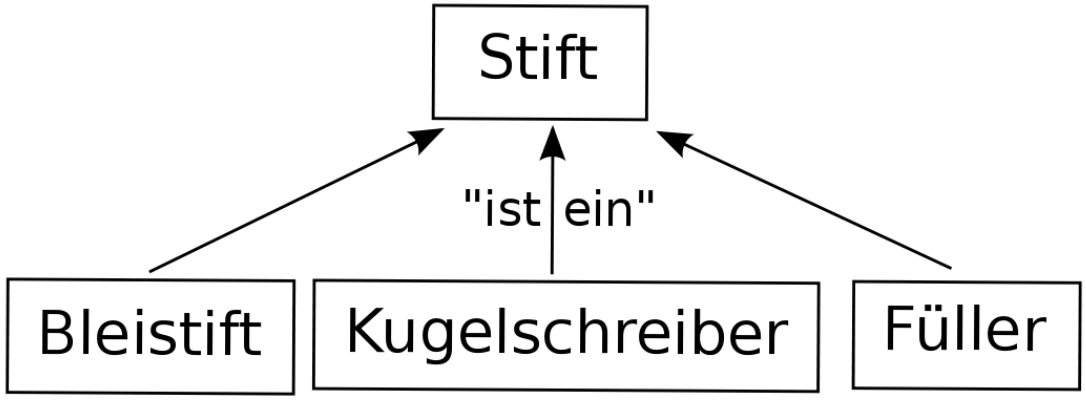
Klassen und Objekte(2)



Klasse "Kuh"	Objekt "Kuh"
• Name	• Elsa Euter
• Geburtsdatum	• 27.März 2009
• Milchleistung	• 34 l/Tag

Klassenhierarchie

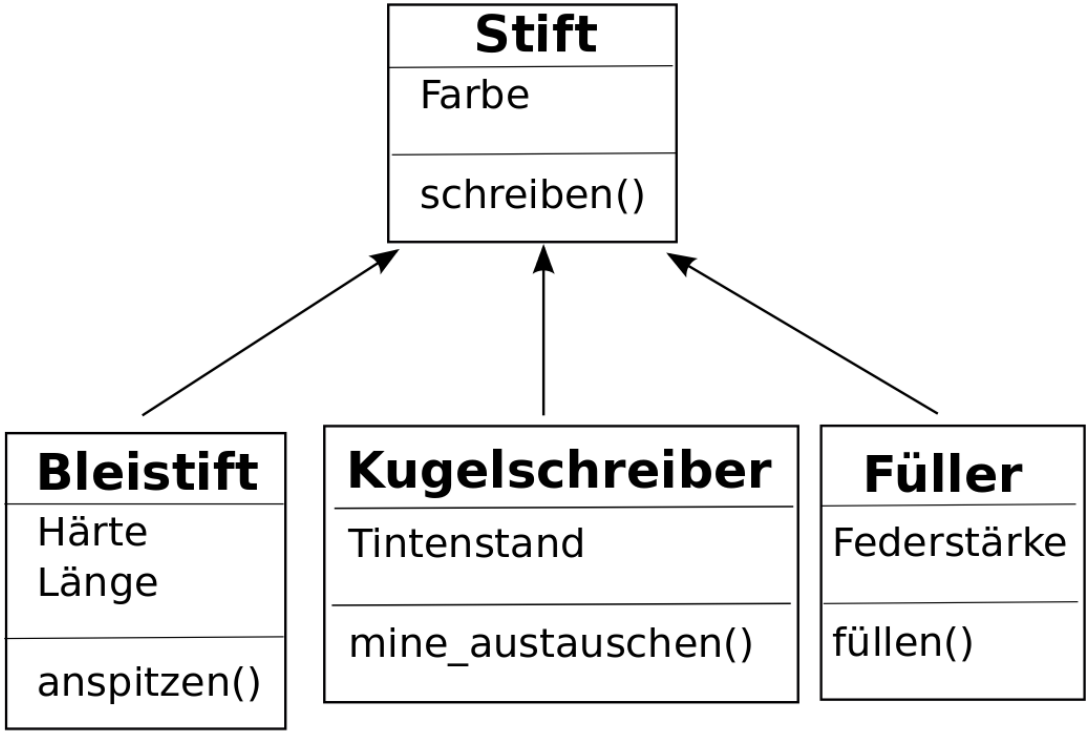
- Verschiedene Arten von Objekten haben häufig Gemeinsamkeiten
- “ist ein”-Beziehung
- Beispiel:



- Superklasse ≡ Elternklasse ≡ Oberklasse ≡ Basisklasse
- Subklasse ≡ Kindklasse ≡ Unterklasse ≡ abgeleitete Klasse

Vererbung

- Kindklassen erben alle Attribute und Methoden von Elternklassen
- haben zusätzlich eigene Attribute und Methoden
- können Attribute und Methoden der Elternklasse *überschreiben*



abstrakte Klasse

- enthält nur leere Methoden
- kann keine Instanz erzeugen
- dient zur Zusammenfassung ähnlicher Klassen
- definiert gemeinsame Attribut- und Methodennamen
- zwingt alle Kindklassen Attribute und Methoden mit entsprechendem Namen zu haben

Übersicht

Zunahme der Rechnerleistung

→ größere Programme

→ komplexere Software

→ größere Projekte

→ Modularität

Vorteile

- **Abstraktion**: Betrachtung der Objekte und ihrer Eigenschaften und Fähigkeiten, ohne Festlegung auf Implementierung
- **Datenkapselung**: Objekt interagiert nur über vordefinierte Methoden. Implementierung kann verändert werden, ohne dass andere Teile des Programms geändert werden müssen
- **Vererbung**: klarere Struktur und weniger Redundanz
- **Wiederverwendbarkeit**: Programme können einfacher erweitert und modifiziert werden. Klassen können auch in anderen Programmen verwendet werden.

Nachteile

- **Formulierung**: natürliche Sprache hat keine feste Bindung von Substantiv (Objekt) und Verb (Methode).
- **Klassenhierarchie**: ist in der realen Welt nicht immer so klar. (Kreis-Ellipse-Problem)
- **Transparenz**: Kontrollfluss nicht im Quelltext
- **Laufzeit- und Energieeffizienz**: OOP-Anwendungen benötigen häufig mehr Energie und längere Laufzeit

Übersicht

Klassen in Python

- Klasse:

  ```
  class KlassenName:
      def method1(self, ):
      def method2(self, ):
  ```

- Konstruktor:

  erzeugt ein Objekt (Instanz) der Klasse

  ```
      def __init__(self, ):
          ...
  ```

- Verwendung:
  - `obj1 = KlassenName( )`
  - `obj1.method1( )`

Variablen/Attribute

- Klassenvariablen:
  - wird von allen Instanzen einer Klasse geteilt
  - mit `<KlassenName>.<VariablenName>` innerhalb und außerhalb der Klasse erreichbar
- Objektvariable:
  - existiert allein für dieses Objekt (Instanz der Klasse)
  - mit `<ObjektName>.<VariablenName>` innerhalb der Klasse erreichbar (evtl. auch außerhalb).

public, protected, private

Name	Bezeichnung	Bedeutung
`name`	public	sowohl innerhalb einer Klasse, als auch von außen les- und schreibbar
`_name`	protected	von außen les- und schreibbar, Attribute und Methoden sollten nicht benutzt werden
`__name`	private	von außen weder sichtbar, noch nutzbar

Beispiel

Prozedural ⇔ Objektorientiert

prozedurale Programmierung	objektorientierte Programmierung
• Ansammlung von Variablen, Datenstrukturen und Funktionen, bzw. Unterprogrammen. • Prozeduren oder Funktionen operieren direkt auf Datenstrukturen • Funktionen und Daten haben keinen Zusammenhalt	• Datentypen (Klassen), welche Verhalten (Methoden) mit Daten (Attribute) verbinden. • Instanz einer Klassen (Objekt) operiert auf seiner "eigenen" Datenstruktur • Funktionen (Methoden) und Daten (Attribute) sind fest miteinander verbunden.