Objektorientierte Programmierung OOP

Ronja Düffel
Alina Stürck
WS2016/17

11. Oktober 2016

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

Was ist das?

- ein Programmierparadigma

Was ist das?

- ein Programmierparadigma (Programmierstil)
- Art und Weise ein Problem zu modellieren
- Beschreibung eines Systems anhand des Zusammenspiels kooperierender Objekte

Was sind Objekte?

- Objekte sind überall
- werden von uns als solche wahrgenommen
- Begriff eher unscharf $\Rightarrow$ kann auch abstrakter sein

In der realen Welt	OO-Programmierung
Zustand	Attribute
Verhalten	Methoden

Datenkapselung

- Zustand gespeichert in Attributwerten
- Verhalten festgelegt durch Methoden
- Interaktion mit anderen Objekten durch Methoden

Datenkapselung

- Zustand gespeichert in Attributwerten
- Verhalten festgelegt 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

Überblick

Klassen und Objekte

- Klasse
  - definiert für eine Menge von Objekten deren Struktur (**Attribute**), Verhalten(**Methoden**) und Beziehungen
  - Bauplan für Objekt
  - Definition aller Attribute und Methoden
  - Besitzt Mechanismus zur Erzeugung eines Objekts

Klassen und Objekte

- Klasse
  - definiert für eine Menge von Objekten deren Struktur (**Attribute**), Verhalten(**Methoden**) und Beziehungen
  - Bauplan für Objekt
  - Definition aller Attribute und Methoden
  - Besitzt Mechanismus zur Erzeugung eines Objekts
- Klasse allein macht noch nichts

Klassen und Objekte

- Klasse
  - definiert für eine Menge von Objekten deren Struktur (**Attribute**), Verhalten(**Methoden**) und Beziehungen
  - Bauplan für Objekt
  - Definition aller Attribute und Methoden
  - Besitzt Mechanismus zur Erzeugung eines Objekts
- Klasse allein macht noch nichts
- Objekt $\rightarrow$ ist konkrete Ausprägung (**Instanz**) der Klasse
- Jedem Objekt ist genau eine Klasse zugeordnet

Objekte der Klasse Kuh

Klasse „Kuh“	Objekt „Kuh Elsa“
Name	Elsa
Geburtsdatum	01.05.2012
Milchleistung	20 Liter/Tag

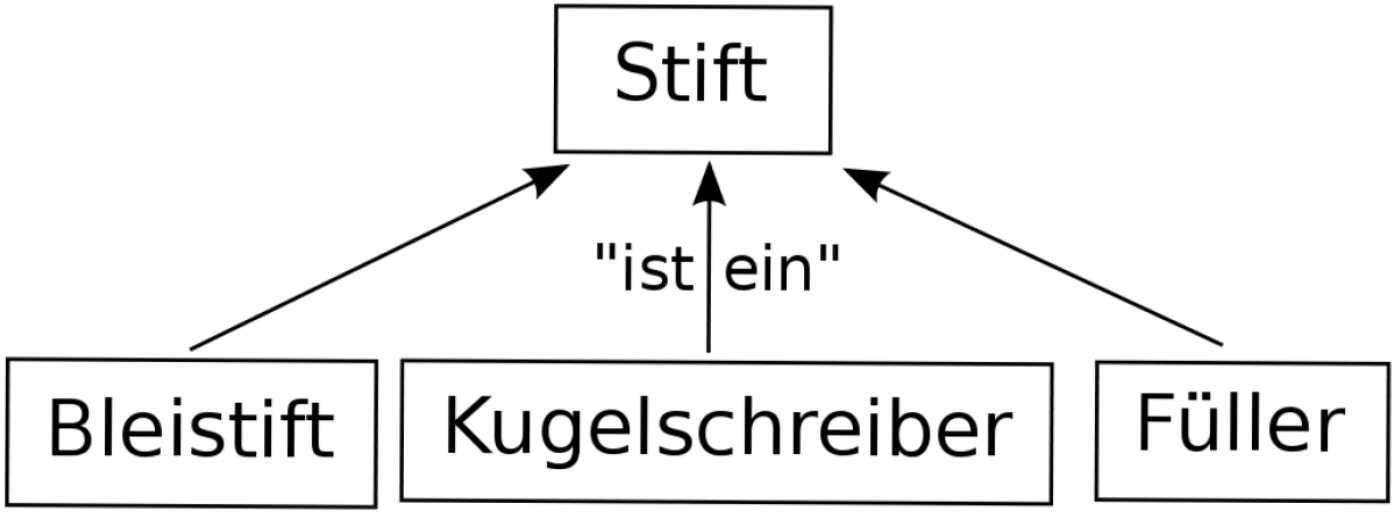
Klassenhierarchie

- Verschiedene Arten von Objekten haben haben Gemeinsamkeiten
- Zusammenfassung verschieder Klassen möglich
- „ist ein“-Beziehung

Klassenhierarchie

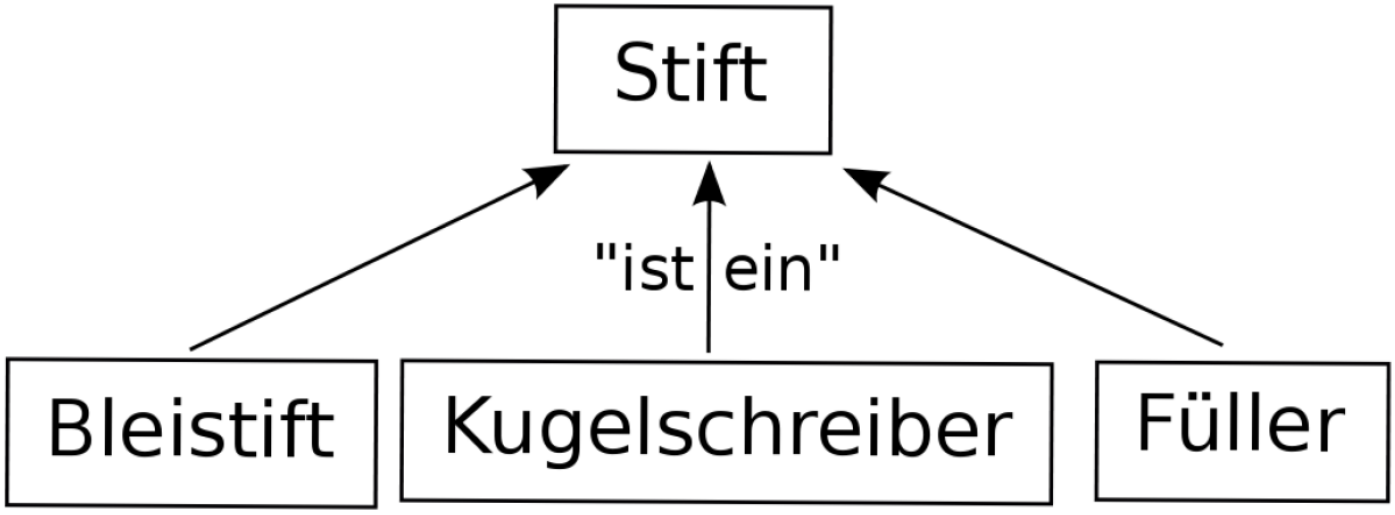
- Verschiedene Arten von Objekten haben haben Gemeinsamkeiten
- Zusammenfassung verschieder Klassen möglich
- „ist ein“-Beziehung

Beispiel:



Klassenhierarchie

- Übergeordnete Klasse:
  - Superklasse = Elternklasse = Oberklasse = Basisklasse
- Untergeordnete Klasse:
  - Subklasse = Kindklasse = Unterklasse = abgeleitete Klasse

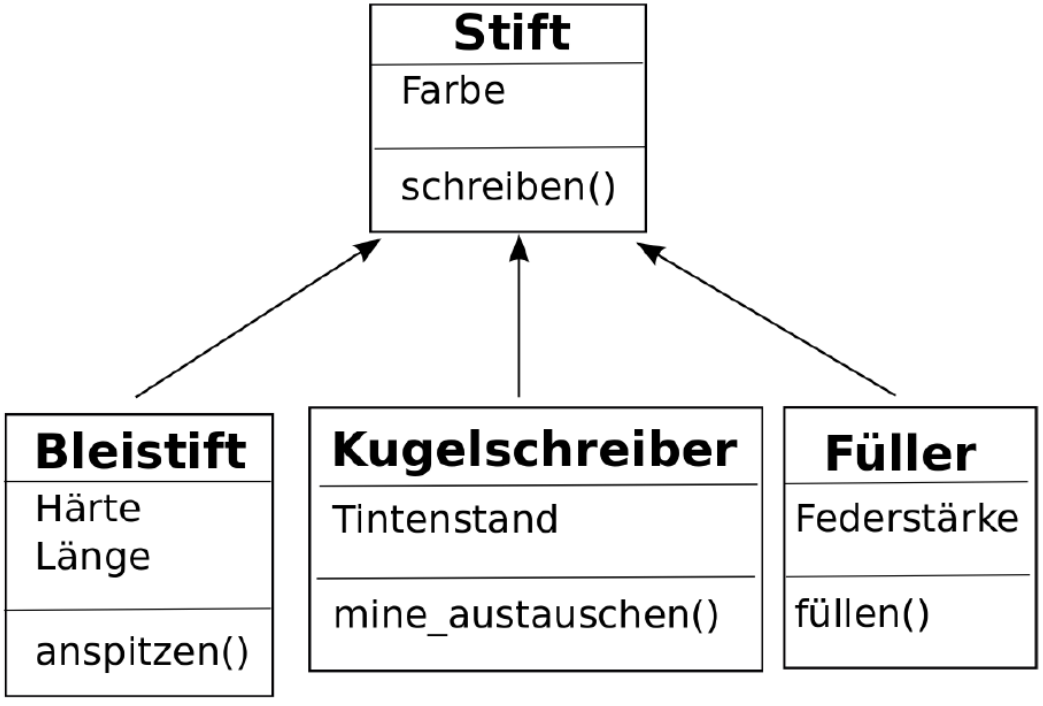


Vererbung

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

Vererbung

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



Überblick

Zunahme der Rechnerleistung

- größere Programme
- komplexere Software
- große Projekte
- Mehr Übersicht nötig

$\Rightarrow$ Modularität (Aufteilung in kleinere Komponenten)

Vorteile

- **Abstraktion:** Betrachtung der Objekte und ihrer Eigenschaften und Fähigkeiten, ohne Festlegung auf Implementierung

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

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

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).

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 langere Laufzeit

Überblick

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