FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

DIE HOCHSCHULE, DIE MEHR KANN.

ILV Einführung in das Programmieren

Objektorientiertes Programmieren (Teil 1)

Mohamed Goha, BSc. | WS 2024/25

APPLIED LIFE SCIENCES | MASTERSTUDIENGANG BIOINFORMATIK

Motivation OOP

- **Bis jetzt Lösung aller Probleme durch prozedurale Programmierung**
  - Stößt bei komplexen Problemen aber an ihre Grenzen, deshalb natürliche Entwicklung hin zur OOP
  - Bisher schon unbewusste Verwendung von Klassen/Objekten
  - Grundidee: SW kann reale Welt mit Hilfe von Objekten besser abbilden

- **Objektorientierte Programmierung hat folgende Ziele:**
  - Fehlerquellen verringern
  - Das EntwicklerInnenleben erleichtern
  - Wiederverwendbarkeit von Code erhöhen

Motivation OOP

Vor allem unter dem Kontext, dass Softwareentwicklung Teamarbeit ist

- Wie kann ich den Umgang mit unbekanntem Code erleichtern?
- Wie kann ich andere dazu bringen, meinen Code richtig zu verwenden?
- Was für Mechanismen erlauben mir, fremden Code einfach zu erweitern bzw. in meinen Code zu integrieren?

- **Vorteile der Objektorientierung**
  - Der Mensch begreift die reale Welt (zum Teil) als Ansammlung physischer Objekte
  - Diese Objekte werden über ihre Eigenschaften und ihr Verhalten definiert, z.B. ein Molekül, ein Organ, ein Lebewesen, …
  - Objekte können eine Hierarchie aufweisen, z.B. ein Molekül besteht aus Atomen, ein Säugetier ist ein Lebewesen
  - Wenn sich Objekte sehr ähneln, werden sie in dieselbe „Schublade" gesteckt
    - FH Campus Wien ist eine Hochschule, genauso wie die Hochschule Salzburg
    - Gemeinsamkeit: Hochschule, aber bestimmte Eigenschaften sind anders

Konzept der Abstraktion

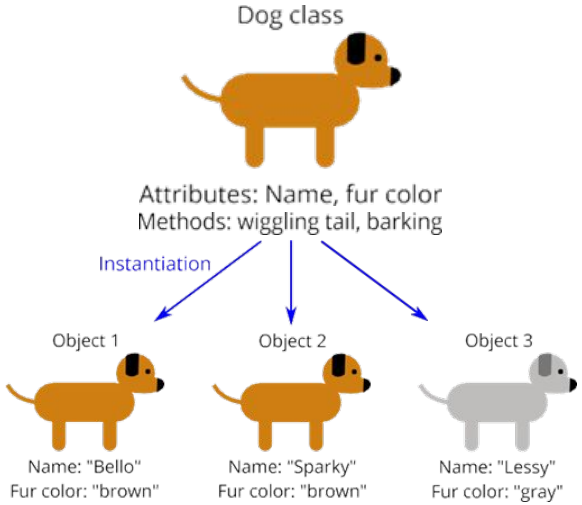
- Abstraktion kann als das Finden von Gemeinsamkeiten einer Menge gleichartiger Objekte gesehen werden
  - Was für gemeinsame Eigenschaften haben die Objekte?
  - Was für gemeinsames Verhalten haben die Objekte?
  - Reduktion auf das Wesentliche!
- Ergebnis ist Vorlage für Erzeugung weiterer Objekte
  - Bauplan gibt Eigenschaften und Verhalten der Objekte vor
  - In OO-Programmiersprachen werden diese Baupläne **Klassen** genannt
    - Eigenschaften werden durch **Attribute** definiert
    - Verhalten wird durch dazugehörige **Methoden** (funktionen) definiert

Klassen und Objekte

- Klassen definieren **Objekte** (ähnlich wie ein **Bauplan**). Das Erstellen einer neuen Klasse bedeutet, einen neuen Typ zu erschaffen.
- Ein Objekt ist eine **Instanz** einer Klasse bzw. eines Typs und existiert einzigartig unter allen anderen Objekten.
- **Beispiel:** Angenommen, wir möchten mehrere Hunde beschreiben.
  - Wir würden zuerst eine Hundeklasse erstellen.
  - Diese Hundeklasse würde **Attribute** und **Methoden** enthalten, die verwendet werden, um einen Hund zu beschreiben.
  - Wenn wir einen bestimmten Hund beschreiben möchten, erstellen wir eine **Instanz** unserer Hundeklasse (ein neues **Objekt**).

Klassen und Objekte

- Beispiel: Angenommen, wir möchten mehrere Hunde beschreiben.
  - Jede Instanz ist ein individuelles Objekt und enthält eine Kopie der Attribute und Methoden aus unserer Hundeklasse.
  - Wir können den Code für ein Hundeobjekt für jeden Hund wiederverwenden!



Klassen und Objekte (Code-Beispiel)

```python
class Dog:

    def __init__(self, name):
        self.name = name

    def communicate(self):
        communication = "bark"
        communication = f"{self.name}: {communication}"
        return communication

dog1 = Dog(name="bello")
dog2 = Dog("barky")
print(dog1.communicate()) # prints "bello: bark"
print(dog2.communicate()) # prints barky: bark
```

Klassen und Objekte

- Wir können auch neue Klassen (classes) aus bestehenden Klassen ableiten (derive), d.h., bestehende Klassen erweitern (extend):
  - Die neuen Klassen werden als Kindklassen (child classes) oder Unterklassen (**subclasses**) bezeichnet.
  - Die Klassen, von denen die Unterklassen abgeleitet werden, nennt man Elternklassen (**parent classes**), Basisklassen (**base classes**) oder Superklassen (**superclasses**).
  - Unterklassen können Attribute und Methoden von ihren Basisklassen erben (**inherit**).
  - Attribute und Methoden aus den Elternklassen sind in den Kindklassen verfügbar, können aber modifiziert oder erweitert werden.

Klassen in Python

- Wir haben bereits mit Klassen (classes) in Python gearbeitet!
  - Beispiel: Unsere Integer-Objekte (integer objects) sind Instanzen der int-Klasse, die von der object-Klasse abgeleitet ist.
- Klassen können mit der **class**-Anweisung erstellt werden.
  - Klassennamen sollten nach Konvention im CapWords-Format geschrieben werden.
  - Beispiel: MyNewClass
- Ähnlich wie Funktionen erzeugen Klassen einen namespace.
  - Attribute und Methoden existieren nur innerhalb der Klasse oder einer Instanz davon.

Initialisierung und Zugriff auf Attribute/Methoden

> Siehe Input Übung 8!

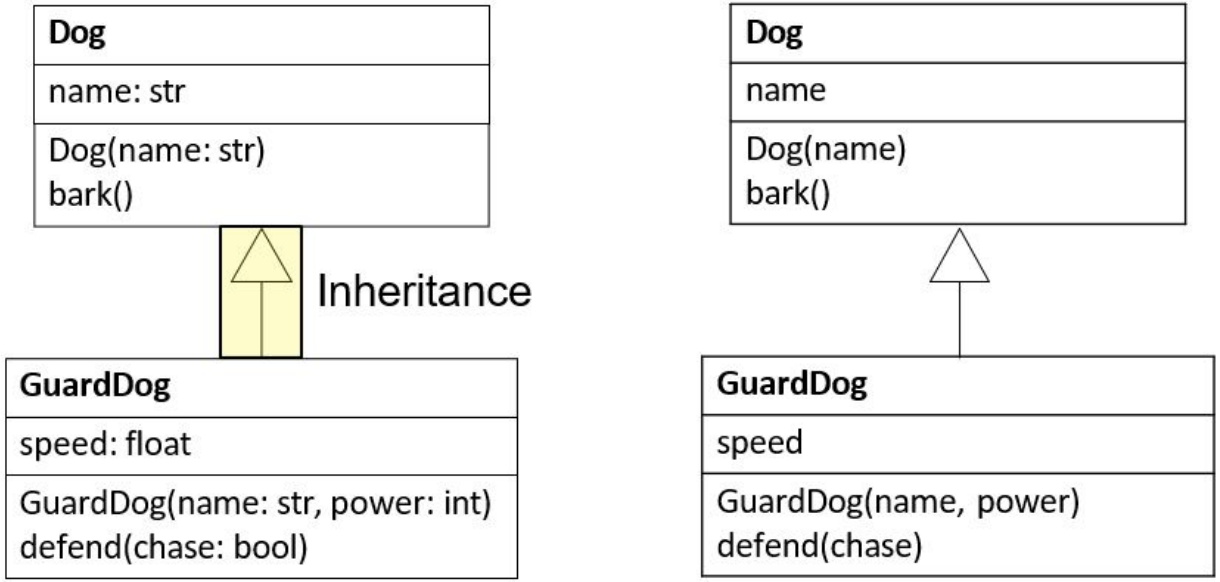
Vererbung

- Syntax zum Erweitern einer Basisklasse (base class) durch Vererbung (inheritance):
  - class MyClass(object)
  - class MySpecializedClass(MyClass)

- Alle Attribute und Methoden werden geerbt. Zusätzliche Attribute und Methoden können hinzugefügt werden, und das Verhalten vorhandener Methoden kann durch Methodenüberschreibung (**method overriding**) geändert werden.

UML (Unified Modeling Language)

> Methode, Klassen zu modellieren (mit Attributen, Methoden, Vererbung, etc)



Method Resolution Order

- Die Method Resolution Order (MRO, Methodenauflösungsreihenfolge) bestimmt, welche Methode letztendlich aufgerufen wird, wenn Sie z.B. **my_object.some_method()** aufrufen.
- Die MRO ist eng mit der Klassenhierarchie (class hierarchy) verbunden. Die Suche erfolgt von unten nach oben:
  - Bestimmen Sie die Klasse (class) von my_object.
  - Enthält die Klasse eine Implementierung (implementation) für some_method()?
  - Wenn ja, wird diese Methode ausgeführt.
  - Wenn nein, wird die Elternklasse (parent class) aufgerufen. Gehen Sie zu Schritt 2 und wiederholen Sie dies, bis die erste Implementierung gefunden wird.