Objektorientierte Programmierung

Eine kleine Einführung in wichtige Konzepte, Teil 2

Michael Greminger 2021-03-17

Grundkonzepte

- Abstraktion
- Datenkapselung
- Vererbung
- Polymorphie

Abstraktion

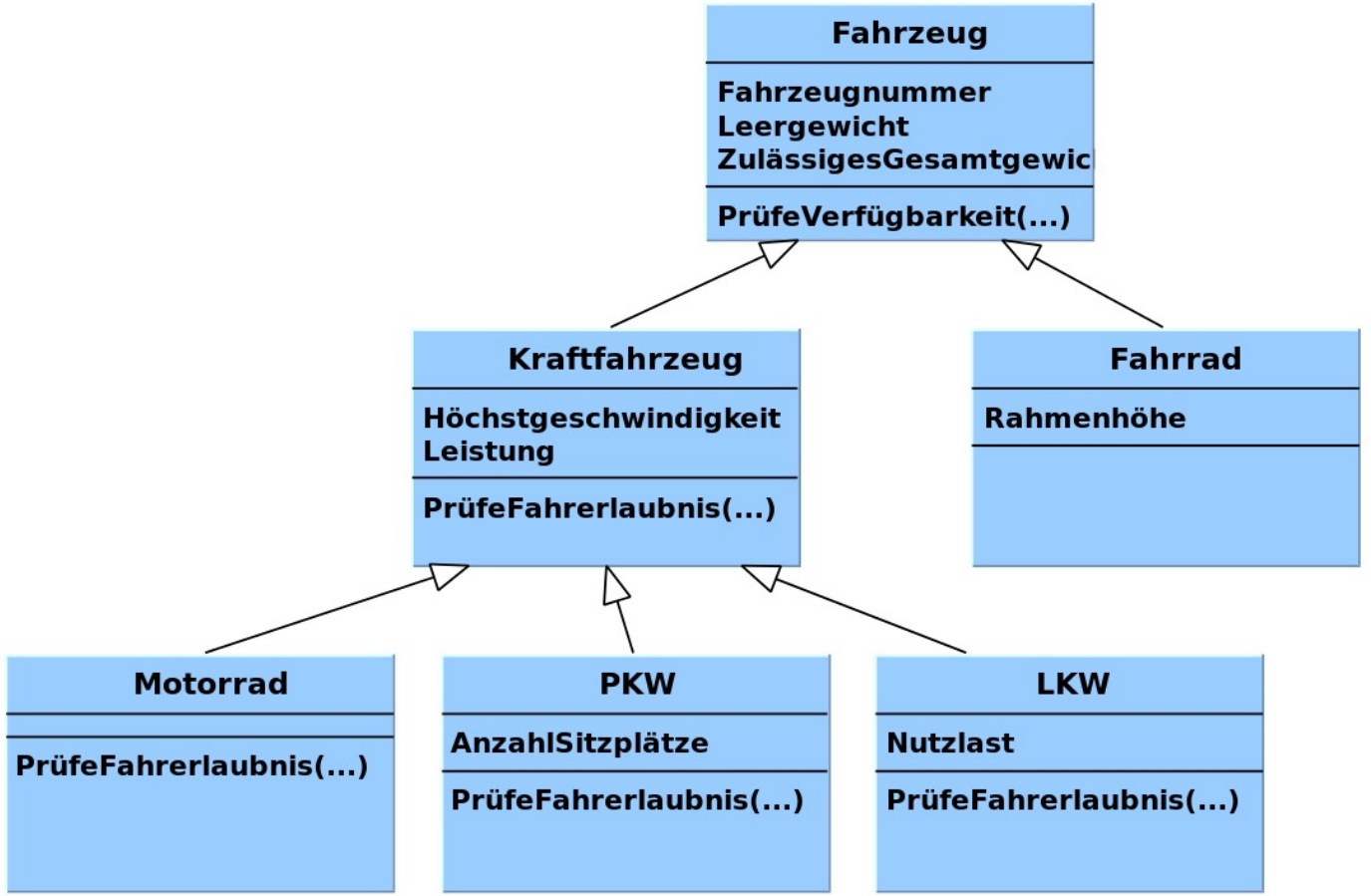
- ”Vor” OOP gibt es nur
  - Lokale und globale Variablen
  - Funktionen
- Mit OOP haben wir wesentlich bessere Gestaltungsmittel und können so die “reale Welt” besser **abstrahieren**
  - Die Strukturierung von Daten in Klassen
  - Die Zuteilung von Funktionen zu Klassen
    - Gruppierung der Funktionen
    - Es wird auch so implizit gesagt, mit welchen Daten eine Funktion arbeitet

Datenkapselung

- Der Benutzer einer Klasse kennt nur die Schnittstellen (Methoden) dieser Klasse
- Vorteile:
  - Die Internas einer Klasse können sich ändern. So lange die Schnittstellen gleich sind, bin ich (in der Regel) nicht davon betroffen
  - Ich kann Details der internen Implementierung ausblenden

Vererbung (”Inheritance”) (1)

- Es ist möglich, von einer Klasse (oft «Basisklasse» genannt) eine anderen Klasse (oft «Subclass» genannt) abzuleiten.
- Abgeleitete Klasse beinhaltet:
  - Alle Attribute der Basisklasse **plus zusätzliche eigene**
  - Die Methoden der Basisklasse (es gibt Ausnahmen) plus **plus zusätzliche eigene**



Aus “Wikipedia”

Vererbung (”Inheritance”) (2)

- Vererbung soll nur ein eingesetzt werden, wenn eine klare “ist ein” Relation gegeben ist!
- Beispiel:
  - Quadrat
    - int: breite
  - Rechteck
    - int: breite
    - int: hoehe
  - Soll Quadrat von Rechteck abgeleitet werden oder umgekehrt oder gar keine Vererbung einsetzen?

Vererbung (”Inheritance”) (3)

- Es gibt unzählige Bücher und Theorien darüber, wie man Vererbung einsetzen soll (zum Bsp. https://de.wikipedia.org/wiki/Entwurfsmuster_(Buch))
- Gerade “Anfänger” neigen dazu, Vererbung zu oft einzusetzen und zu tiefe Hierachien zu kreieren
- Bsp:
  - ”RotesAuto” als Subklasse von ”Auto”?
  - Verschiedene Varianten:

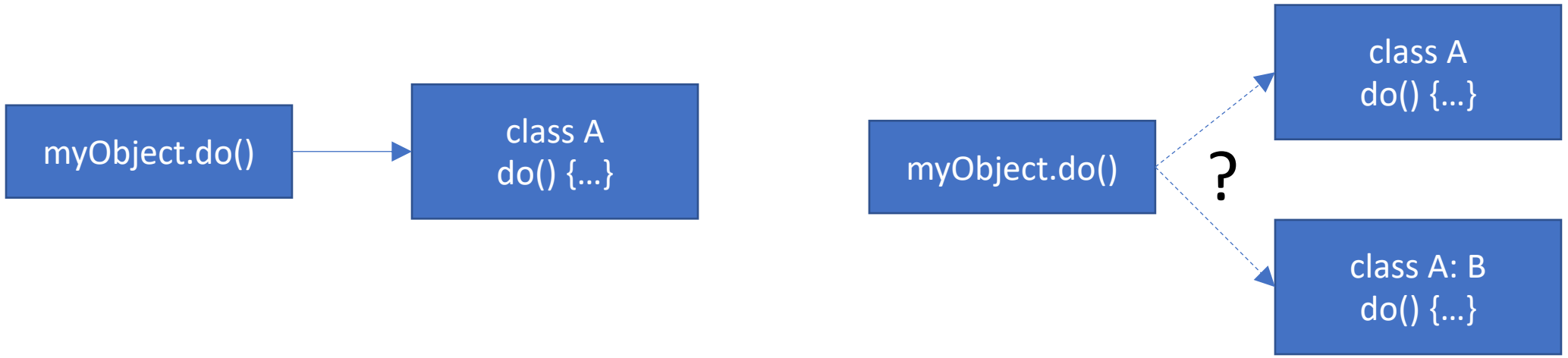
Polymorphie (1)

- Der Begriff “Polymorphie” stammt aus dem griechischen und bedeutet ”Vielgestaltigkeit”
- In der objektorientierten Programmierung steht er hauptsächlich für die Eigenschaft, dass man Methoden in einer Subklasse überschreiben kann.

```
1  class BaseClass:
2      def sayHello(self):
3          print "Hello base class"
4
5  class DerivedClass(BaseClass):
6      def sayHello(self):
7          print "Hello derived class"
8
9
10 b = BaseClass()
11 d = DerivedClass()
12 b.sayHello()
13 d.sayHello()
```

Polymorphie (2)

- Der Begriff “Polymorphie” wird in der Programmierung auch für das sogenannte “Operator Overloading” oder für Funktionen mit gleichem Namen, aber unterschiedlichen Argumenten verwendet
- In ”eher statischen” Programmiersprachen (mit Compiler) ist das Vorhandensein dieses Konstrukts bemerkenswerter.



Polymorphie (3)

```
3     class Animal { // Base class (parent)
4         public virtual void sound() {
5             Console.WriteLine("The animal makes a sound");
6         }
7     }
8
9     class Dog : Animal { // Derived class (child)
10        public override void sound() {
11            Console.WriteLine("The dog says: bow wow");
12        }
13    }
14
15    class Program{
16        static void Main(string[] args) {
17            Animal myAnimal = new Animal();  // Create a Animal object
18            Animal myDog = new Dog();  // Create a Dog object
19
20            myAnimal.sound();
21            myDog.sound();  // this is the real miracle
22        }
23    }
```