12. OOP Intro

12.1. Definitionen

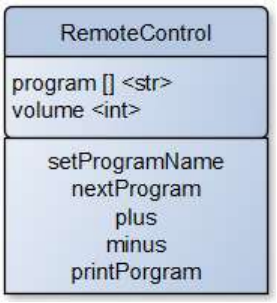
- Klassen sind Baupläne
- ein Objekt (Instanz) ist eine konkrete Ausprägung einer Klasse, d.h.
  Objekte sind Instanzen, die man von den Bauplänen erzeugen kann
- jedes Objekt hat seine Attribute
- die Attribute haben Werte, sie beschreiben den Zustand eines Objektes
- Attributwert kann man mit Hilfe von Methoden setzen, lesen oder verändern
- jede Klasse ist für die Manipulation ihrer Attribute selbst verantwortlich
- eine Klasse wird definiert mit dem Schlüsselwort **class** *Name der Klasse***:**
  Klassennamen sollten mit Großbuchstaben beginnen

- eingerückt folgt die Definition der Attribute und

- ebenfalls eingerückt die Definition der Methoden jeweils mit **def** *Name der Methode(…)***:**
  mit dem Schlüsselwort **def**, dem Namen der Methode
  und in runden Klammern umschlossen die Parameter

- jede Methode hat mindestens einen Parameter, eine Referenz auf das Objekt
  sie wird formuliert mit **self**

12.2. Beispiel Klasse RemoteControl



- Objekte der Klasse RemoteControl sollen die folgenden Attribute (Eigenschaften) besitzen bzw. Methoden beherrschen:
  - Attribute
    - program — eine Liste mit Programmnamen - die Anzahl der Plätze wird bei der Instanziierung festgelegt
    - volume — Lautstärke wird bei der Instanziierung auf 0 gesetzt
  - Methoden
    - setProgramName — Methode zur Benennung des aktuellen Programms
    - nextProgram — gehe zum nächsten Programm
    - plus — Methode, um die Lautstärke zu erhöhen
    - minus — Methode, um die Lautstärke zu verringern
    - printProgram — Methode zum Ausgeben des aktuellen Programms
- Beispiel:

```
# Klasse RemoteControl definieren
class RemoteControl:

    # Konstruktor, dem die maximale Anzahl an Programmspeicherplätzen
    # zur Initialisierung übergeben wird
    def __init__(self, num_programs):
        self.programs = []

        # Initialisierung der Programmspeicherplätze
        for i in range(0, num_programs):
            self.programs += ["Prog "+ str(i+1)]
        self.curr_program_number = 0

        self.volume = 0

    # Methode zur Benennung des aktuellen Programms
    def setProgramName(self, name):
        self.programs[self.curr_program_number] = name

    # Methode für den Wechsel zum nächsten Programm
    def nextProgram(self):
        # Gehe zum nächsten Programm,
        # wenn noch nicht das Ende der Liste erreicht ist
        if self.curr_program_number < len(self.programs) - 1:
            self.curr_program_number += 1
```

```
            # sonst beginne wieder beim ersten Programm
            else:
                self.curr_program_number = 0

        # Methode um die Lautstärke zu erhöhen
        def plus(self):
            self.volume += 1

        # Methode um die Lautstärke zu verringern
        def minus(self):
            self.volume -= 1

        # Methode zum Ausgeben des aktuellen Programms
        # mit Nummer, Name und aktueller Lautstärke
        def printProgram(self):
            print(f"Nr {str(self.curr_program_number):>2} "
                  f"Name {str(self.programs[self.curr_program_number]):>6}"
                  f"Lautstärke {str(self.volume):>2} ")
```

12.3. Definitionen

- innerhalb einer Klasse gibt es Attribute und Methoden, die mit einem Sichtbarkeitsmodifizier versehen sind:
    - private Definition mit zwei vorangestellten Unterstrichen __name
        - auf diese Attribute und Methoden nur innerhalb der Klasse zugegriffen werden
        - sie sind außerhalb der Klasse nicht zugreifbar
    - protected Definition mit einem vorangestellten Unterstrich _name
        - ist bei der Vererbung relevant
        - diese Attribute sind nach außen geschützt, bleiben in der Vererbungshierarchie aber zugreifbar
        - dient in Python als Hinweis
    - public Definition name
        - alle Attribute und Methoden sind automatisch öffentlich (public)
        - sie bilden die Schnittstelle der Klasse nach außen
        - die Kommunikation mit der Klasse findet darüber statt

- mit diesen Sicherheitsmodifizieren
    - lassen sich Attribute kapseln und vor dem direkten Zugriff von außen schützen
    - private Methoden werden ausschließlich für den internen Gebrauch verwendet
    - öffentliche Methoden einer Klasse bilden ihre Schnittstelle zur Kommunikation nach außen

- Beispiel:

```
# Klasse RemoteControl definieren
class RemoteControl:

    # Konstruktor, dem die maximale Anzahl an Programmspeicherplätzen
    # zur Initialisierung übergeben wird
    def __init__(self, num_programs):
        self.__programs = []

        # Initialisierung der Programmspeicherplätze
        for i in range(0, num_programs):
            self.__programs += ["Prog "+ str(i+1)]
        self.__current_program_number = 0

        self.__volume = 0

    # Methode zur Benennung des aktuellen Programms
    def setProgramName(self, name):
        self.__programs[self.__current_program_number] = name

    # Methode für den Wechsel zum nächsten Programm
    def nextProgram(self):
        # Gehe zum nächsten Programm, wenn noch nicht das Ende der Liste erreicht
ist
```

```
            if self.__current_program_number < len(self.__programs) - 1:
                self.__current_program_number += 1
            # sonst beginne wieder beim ersten Programm
            else:
                self.__current_program_number = 0

        # Methode um die Lautstärke zu erhöhen
        def plus(self):
            self.__volume += 1

        # Methode um die Lautstärke zu verringern
        def minus(self):
            self.__volume -= 1

        # Methode zum Ausgeben des aktuellen Programms
        # mit Nummer, Name und aktueller Lautstärke
        def printProgram(self):
            print(f"Nr {str(self.__current_program_number):>2} "
                  f"Name {str(self.__programs[self.__current_program_number]):>6} "
                  f"Lautstärke {str(self.__volume):>2} ")


    # Klasse RemoteControl instanziieren - mit 5 Programmplätzen
    rc = RemoteControl(5)

    # aktuellen Programmplatz: Programmname setzen
    rc.setProgramName("ARD")

    # aktuellen Programmplatz: Programmnummer und Programmname ausgeben
    rc.printProgram()

    # gehe 1 Sender weiter
    rc.nextProgram()

    # aktuellen Programmplatz: Programmnummer und Programmname ausgeben
    rc.printProgram()

    # aktuellen Programmplatz: Programmname setzen
    rc.setProgramName("ZDF")

    # aktuellen Programmplatz: Programmnummer und Programmname ausgeben
    rc.printProgram()


    # rc.__program[3] = "RTL"
    # führt zu AttributeError: 'RemoteControl' object has no attribute '__program'
    rc.plus()
    rc.plus()
    # aktuellen Programmplatz: Programmnummer und Programmname ausgeben
    rc.printProgram()
```

Aufgaben

1. Klasse Person
   es ist die Klasse Person zu erstellen
   Als Attribute besitzt sie Vorname, Name, Geburtsdatum und Wohnort, sowie eine Methode für die Namensänderung (name_aendern) und die Wohnortänderung (umziehen).
   Die Attribute sind als „private“ zu definieren, so dass der Zugriff von außerhalb der Klasse nicht erfolgen kann (z.B. p.name = **"Lehmann"** ist dann nicht gültig)