

Informatik I

17. Nochmals Veränderlichkeit

Jan-Georg Smaus

Albert-Ludwigs-Universität Freiburg

20. Januar 2011

- Jetzt, da wir verlinkte Listen kennen, können wir ein etwas größeres Beispiel betrachten, in dem es um die Veränderlichkeit von Objekten geht.

Kammermusikensembles

- Jetzt, da wir verlinkte Listen kennen, können wir ein etwas größeres Beispiel betrachten, in dem es um die Veränderlichkeit von Objekten geht.
- In dem Beispiel geht es um Kammermusikensembles.



- Mitspieler in Ensembles können ausgetauscht werden, und mitunter wird ein Quartett zu einem Quintett erweitert o.Ä.: ein Paradebeispiel für veränderliche Objekte.

Informatik I

Jan-Georg
Smaus

Die Klasse
Musician

Die Klasse
ChamberEnsemb

Die Klasse
Musician

Attribute
Gleichheit von
Musikern
Stringausgabe

Die Klasse
ChamberEnsemb

Die Klasse Musician

Zunächst brauchen wir eine Klasse für einen Musiker.

```
chamber_ensemble.py
```

```
class Musician:  
    def __init__(self, name, instrument):  
        self.name = name  
        self.instrument = instrument
```

Ein Musiker hat zwei Attribute: sein Name und sein Instrument; beide sind Strings.

Einige Musiker konstruieren

Python-Interpreter

```
>>> ed = Musician("Eugene Drucker", "Violine 1")
>>> ps = Musician("Philip Setzer", "Violine 2")
>>> gf = Musician("Guillermo Figueroa", "Viola")
>>> ew = Musician("Eric Wilson", "Cello")
>>> ld = Musician("Lawrence Dutton", "Violine 1")
>>> df = Musician("David Finckel", "Cello")
>>> mr = Musician("Mstislaw Rostropowitsch",
... "Cello 2")
```

Die ersten vier Musiker sind die Erstbesetzung des Emerson-Quartetts. Die nächsten zwei sind später nachgerückt. Zu Mstislaw Rostropowitsch kommen wir später.

Einige Musiker konstruieren

Python-Interpreter

```
>>> ed = Musician("Eugene Drucker", "Violine 1")
>>> ps = Musician("Philip Setzer", "Violine 2")
>>> gf = Musician("Guillermo Figueroa", "Viola")
>>> ew = Musician("Eric Wilson", "Cello")
>>> ld = Musician("Lawrence Dutton", "Violine 1")
>>> df = Musician("David Finckel", "Cello")
>>> mr = Musician("Mstislaw Rostropowitsch",
... "Cello 2")
```

Die ersten vier Musiker sind die Erstbesetzung des Emerson-Quartetts. Die nächsten zwei sind später nachgerückt. Zu Mstislaw Rostropowitsch kommen wir später.

Übrigens: Klammern erlauben Zeilenumbrüche!

Gleichheit von Musikern

```
chamber_ensemble.py
```

```
class Musician:
    def __init__(self, name, instrument):
        self.name = name
        self.instrument = instrument

    def __eq__(self, oth):
        return(self.name == oth.name and
               self.instrument == oth.instrument)
```

Im richtigen Leben ist es unsinnig, von "gleichen" Menschen zu sprechen (am ehesten noch bei Zwillingen), aber für die Programmierung ist das sinnvoll.

Informatik I

Jan-Georg
Smaus

Die Klasse
Musician

Attribute
Gleichheit von
Musikern
Stringausgabe

Die Klasse
ChamberEnsemb

Einen Musiker als String ausgeben

```
chamber_ensemble.py
```

```
class Musician:
    def __init__(self, name, instrument):
        self.name = name
        self.instrument = instrument
        ...

    def __str__(self):
        return (" " + self.instrument +
                ": " + self.name + ".")
```

Die Methode `__str__` ist wie `__init__` und `__eq__` eine spezielle (so genannte **magische** Methode). Sie wird verwendet, wo immer eine String-Darstellung des Objekts benötigt wird, z.B. für die Funktion `print`.

Informatik I

Jan-Georg
Smaus

Die Klasse
Musician

Attribute
Gleichheit von
Musikern

Stringausgabe

Die Klasse
ChamberEnsemb

Verwendung von `__str__`

Python-Interpreter

```
>>> print(ed)
```

```
Violine 1: Eugene Drucker.
```

Informatik I

Jan-Georg
Smaus

Die Klasse
Musician

Attribute
Gleichheit von
Musikern

Stringausgabe

Die Klasse
ChamberEnsemb

Die Klasse ChamberEnsemble

Informatik I

Jan-Georg
Smaus

Die Klasse
Musician

Die Klasse
ChamberEnsemble

Attribute
prettyprint
add_musician
remove_musician
replace_musician
Zuweisung und
Zustand

Ein Kammermusikensemble

chamber_ensemble.py

```
class ChamberEnsemble:
    def __init__(self, name, members):
        self.name = name
        self.members = members
```

Ein Kammermusikensemble besteht aus seinem Namen und der verlinkten Liste seiner Mitglieder.

Informatik I

Jan-Georg
Smaus

Die Klasse
Musician

Die Klasse
ChamberEnsemb

Attribute
prettyprint
remove_musician
replace_musician
Zuweisung und
Zustand

Ein Ensemble konstruieren

Python-Interpreter

```
>>> musicians =  
empty.cons(ew).cons(gf).cons(ps).cons(ed)  
>>> quartett = ChamberEnsemble("Emerson-Quartett",  
musicians)
```

Informatik I

Jan-Georg
Smaus

Die Klasse
Musician

Die Klasse
ChamberEnsemb

Attribute
prettyprint
add_musician
remove_musician
replace_musician
Zuweisung und
Zustand

prettyprint

```
chamber_ensemble.py
```

```
class ChamberEnsemble:  
    def __init__(self, name, members):  
        self.name = name  
        self.members = members  
  
    def prettyprint(self):  
        print(self.name + ":")  
        self.members.prettyprint()
```

Der Aufruf von `prettyprint` bezieht sich auf die für verlinkte Listen definierte Methode. Sie funktioniert für Listen von Musikern, weil

Informatik I

Jan-Georg
Smaus

Die Klasse
Musician

Die Klasse
ChamberEnsemb

Attribute
prettyprint
add_musician
remove_musician
replace_musician
Zuweisung und
Zustand

prettyprint

```
chamber_ensemble.py
```

```
class ChamberEnsemble:  
    def __init__(self, name, members):  
        self.name = name  
        self.members = members  
  
    def prettyprint(self):  
        print(self.name + ":")  
        self.members.prettyprint()
```

Der Aufruf von `prettyprint` bezieht sich auf die für verlinkte Listen definierte Methode. Sie funktioniert für Listen von Musikern, weil `print` für Musiker definiert ist (s.o.).

Informatik I

Jan-Georg
Smaus

Die Klasse
Musician

Die Klasse
ChamberEnsemb

Attribute

prettyprint

add_musician

remove_musician

replace_musician

Zuweisung und
Zustand

Verwendung von prettyprint

Python-Interpreter

```
>>> quartett.prettyprint()
```

```
Emerson-Quartett
```

```
cons Violine 1: Eugene Drucker. Violine 2:
```

```
Philip Setzer. Viola: Guillermo Figueroa.
```

```
Cello: Eric Wilson.
```

Naja, es geht sicher noch hübscher ...

Einen Musiker hinzufügen

Wenn das Emerson-Quartett das Streichquintett C-Dur op. post. 163, D 956 von Franz Schubert spielen möchte, braucht es natürlich einen zweiten Cellisten. Dafür fragt es keinen geringeren als Mstislaw Rostropowitsch.



Informatik I

Jan-Georg
Smaus

Die Klasse
Musician

Die Klasse
ChamberEnsemb

Attribute
prettyprint
add_musician
remove_musician
replace_musician
Zuweisung und
Zustand

chamber_ensemble.py

```
class ChamberEnsemble:
    def __init__(self, name, members):
        self.name = name
        self.members = members
        ...

    def add_musician(self, musician):
        self.members = self.members.cons(musician)
```

Verwendung von `add_musician`

Python-Interpreter

```
>>> quartett.add_musician(mr)
```

```
>>> quartett.prettyprint()
```

Emerson-Quartett:

```
cons Cello 2:  Mstislaw Rostropowitsch.  Violine  
1:  Eugene Drucker.  Violine 2:  Philip Setzer.  
Viola:  Guillermo Figueroa.  Cello:  Eric Wilson.
```

Einen Musiker entfernen

Natürlich will das Emerson-Quartett den zweiten Cellisten auch wieder loswerden:

chamber_ensemble.py

```
class ChamberEnsemble:
    def __init__(self, name, members):
        self.name = name
        self.members = members
        ...

    def remove_musician(self, mus):
        self.members = self.members.without(mus)
```

Informatik I

Jan-Georg
Smaus

Die Klasse
Musician

Die Klasse
ChamberEnsemb

Attribute
prettyprint
add_musician
remove_musician
replace_musician
Zuweisung und
Zustand

Verwendung von `remove_musician`

Python-Interpreter

```
>>> x = Musician("Mstislaw Rostropowitsch", "Cello 2")
>>> quartett.remove_musician(x)
>>> quartett.prettyprint()
Emerson-Quartett:
cons Violine 1: Eugene Drucker. Violine 2:
Philip Setzer. Viola: Guillermo Figueroa.
Cello: Eric Wilson.
```

`x` ist nicht der **Selbe** wie `mr`, sondern nur der **Gleiche**.
Für die Benutzerin ist es intuitiv, dass wenn sie ein Musikerobjekt mit Namen Mstislaw Rostropowitsch und Instrument "Cello 2" erzeugt und dieses aus dem Ensemble entfernen, dass dann dieses auch funktioniert.

Ersetzen eines Musikers

Im Laufe der Jahre kommt es vor, dass Musiker sterben oder aus sonstigen Gründen aus einem Ensemble austreten. Dann müssen sie ersetzt werden:

chamber_ensemble.py

```
class ChamberEnsemble:
    def __init__(self, name, members):
        self.name = name
        self.members = members
        ...

    def replace_musician(self, old, new):
        return self.members.replace(old, new)
```

Informatik I

Jan-Georg
Smaus

Die Klasse
Musician

Die Klasse
ChamberEnsemb

Attribute
prettyprint
add_musician
remove_musician
replace_musician
Zuweisung und
Zustand

Verwendung von `replace_musician`

Python-Interpreter

```
>>> quartett.prettyprint()
Emerson-Quartett:
cons Violine 1: Eugene Drucker. Violine 2:
Philip Setzer. Viola: Guillermo Figueroa.
Cello: Eric Wilson.
>>> quartett.replace_musician(gf, ld)
True
>>> quartett.prettyprint()
Emerson-Quartett:
cons Violine 1: Eugene Drucker. Violine 2:
Philip Setzer. Viola: Lawrence Dutton. Cello:
Eric Wilson.
```

Verwendung von `replace_musician` II

Python-Interpreter

```
>>> quartett.replace_musician(ew, df)
True
>>> quartett.prettyprint()
Emerson-Quartett:
cons Violine 1: Eugene Drucker. Violine 2:
Philip Setzer. Viola: Lawrence Dutton. Cello:
David Finckel.
>>> quartett.replace_musician(mr, df)
False
```

Informatik I

Jan-Georg
Smaus

Die Klasse
Musician

Die Klasse
ChamberEnsemb

Attribute
prettyprint
add_musician
remove_musician
replace_musician
Zuweisung und
Zustand

Vergleich zur funktionalen Programmierung

(Wiederholung)

- Die Beispiele des Kontos und des Streichquartetts stammen aus Kapitel 12 des Scheme-Buchs [KS07]: *Zuweisung und Zustand*.
- Zuweisungen und Veränderlichkeit sind dem funktionalen Paradigma eigentlich fremd und werden deshalb im Buch spät als besonderes, mit Vorsicht zu genießendes Konzept behandelt.
- Im imperativen bzw. objektorientierten Paradigma sind Zuweisungen und Veränderlichkeit hingegen eine Selbstverständlichkeit.
- Trotzdem sind nicht alle Objekte veränderlich! Unveränderliche Objekte spielen auch eine wichtige Rolle.

Informatik I

Jan-Georg
Smaus

Die Klasse
Musician

Die Klasse
ChamberEnsemb

Attribute
prettyprint
add_musician
remove_musician
replace_musician
Zuweisung und
Zustand



Herbert Klaeren and Michael Sperber.
Die Macht der Abstraktion.
Teubner Verlag, 2007.

Informatik I

Jan-Georg
Smaus

Die Klasse
Musician

Die Klasse
ChamberEnsemb

Attribute
prettyprint
add_musician
remove_musician
replace_musician

Zuweisung und
Zustand