Schritte zur Vererbung

Vererbung im Raumplaner-Projekt

kritische Betrachtung
der Schritte

Schritte zur Vererbung

Schritt 1 umgesetzt

- Geklärt, welche Attribute und Methoden in den Klassen *`stuhl`* und *`tisch`* (und …) identisch sind und daher nach *`moebel`* verschoben werden können.

File Edit Format Run Options Window He

```
class Stuhl():
    """Klasse Stuhl
    ermoeglicht das Zeichnen und Bearbeiten
    Stuhl-Symbols fuer den Raumplaner"""

    def __init__(self,
                 xPos=20,
                 yPos=20,
                 breite=40,
                 tiefe=40,
                 winkel=0,
                 farbe="blue",
                 sichtbar=False):
        """Konstruktor mit vordefinierten P
        self.x=xPos
        self.y=yPos
        self.b=breite
        self.t=tiefe
        self.w=winkel
        self.f=farbe
        self.s=sichtbar
        if sichtbar: self.Zeige()

    def GibFigur(self):
        """definiert und transformiert die
        gc = Zeichenflaeche.GibZeichenflaec
        path = gc.CreatePath()

        path.MoveToPoint(0, 0)
        path.AddLineToPoint(self.b, 0)
        path.AddLineToPoint(self.b*1.1, sel
        path.AddLineToPoint(-self.b*0.1, se
        path.AddLineToPoint(0, 0)
        path.AddLineToPoint(0, -self.t*0.1)
        path.AddLineToPoint(self.b, -self.t
        path.AddLineToPoint(self.b, 0)

        gc.PushState()
        gc.Translate(self.x+self.b/2, self.
        gc.Rotate(radians(self.w))
        gc.Translate(-self.b/2, -self.t/2)
        transformation = gc.GetTransform()
        gc.PopState()
        path.Transform(transformation)
        return path

    def GibFarbe(self):
        """Get-Methode fuer die Farbe"""
        return self.f

    def GibSichtbar(self):
        """Get-Methode fuer die Sichtbarkei
        return self.s

    def BewegeHorizontal(self, weite):
        """Veraendernde Methode fuer die x-
        self.verberge()
        self.x += weite
        self.Zeige()

    def BewegeVertikal(self, weite):
        """Veraendernde Methode fuer die y-
        self.verberge()
        self.y += weite
        self.Zeige()

    def Drehe(self, winkel):
        """Veraendernde Methode fuer die Or
        self.verberge()
        self.w += winkel
        self.Zeige()

    def verberge(self):
        """Veraendernde Methode fuer die Si
        self.s = False
        Zeichenflaeche.GibZeichenflaeche().

    def Zeige(self):
        """Veraendernde Methode fuer die Si
        self.s = True
```

File Edit Format Run Options Window Help

```
### ------------------------------------------------
class Tisch():
    """Klasse Tisch
    ermoeglicht das Zeichnen und Bearbeiten eines
    Tisch-Symbols fuer den Raumplaner"""

    def __init__(self,
                 xPos=60,
                 yPos=10,
                 breite=120,
                 tiefe=60,
                 winkel=0,
                 farbe='red',
                 sichtbar=False):
        """Konstruktor mit vordefinierten Parametern"""
        self.x=xPos
        self.y=yPos
        self.b=breite
        self.t=tiefe
        self.w=winkel
        self.f=farbe
        self.s=sichtbar
        if sichtbar: self.Zeige()

    def GibFigur(self):
        """definiert und transformiert die zu zeichnende Figur"""
        gc = Zeichenflaeche.GibZeichenflaeche().GibGC()
        path = gc.CreatePath()

        path.AddRectangle(0, 0, self.b, self.t)

        gc.PushState()
        gc.Translate(self.x+self.b/2, self.y+self.t/2)
        gc.Rotate(radians(self.w))
        gc.Translate(-self.b/2, -self.t/2)
        transformation = gc.GetTransform()
        gc.PopState()
        path.Transform(transformation)
        return path

    def GibFarbe(self):
        """Get-Methode fuer die Farbe"""
        return self.f

    def GibSichtbar(self):
        """Get-Methode fuer die Sichtbarkeit"""
        return self.s

    def BewegeHorizontal(self, weite):
        """Veraendernde Methode fuer die x-Position"""
        self.Verberge()
        self.x += weite
        self.Zeige()

    def BewegeVertikal(self, weite):
        """Veraendernde Methode fuer die y-Position"""
        self.Verberge()
        self.y += weite
        self.Zeige()

    def Drehe(self, winkel):
        """Veraendernde Methode fuer die Orientierung [Winkel]"""
        self.Verberge()
        self.w += winkel
        self.Zeige()

    def Verberge(self):
        """Veraendernde Methode fuer die Sichtbarkeit mit Wert False"""
        self.s = False
        Zeichenflaeche.GibZeichenflaeche().Entferne(self)

    def Zeige(self):
        """Veraendernde Methode fuer die Sichtbarkeit mit Wert True"""
        self.s = True
        Zeichenflaeche.GibZeichenflaeche().Zeichne(self)
```

File Edit Format Run Options Window Help

```
class Sessel():
    """Klasse Sessel
    ermoeglicht das Zeichnen und Bearbeiten eines
    Sessel-Symbols fuer den Raumplaner"""

    def __init__(self,
                 xPos=20,
                 yPos=20,
                 breite=40,
                 tiefe=40,
                 winkel=0,
                 farbe="blue",
                 sichtbar=False):
        """Konstruktor mit vordefinierten Parametern"""
        self.x=xPos
        self.y=yPos
        self.b=breite
        self.t=tiefe
        self.w=winkel
        self.f=farbe
        self.s=sichtbar
        if sichtbar: self.Zeige()

    def GibFigur(self):
        """Definiert den Pfad und transformiert ihn."""
        path = Zeichenflaeche.GibZeichenflaeche().GibGC().CreatePath()
        # Umrandung
        path.AddRectangle(0, 0, self.b, self.t)
        # linke Lehne
        path.AddRectangle(0, self.t/6, self.b/6, 5*self.t/6)
        # rechte Lehne
        path.AddRectangle(5*self.b/6, self.t/6, self.b/6, 5*self.t/6)
        # Rueck-Lehne
        path.AddRectangle(0, 0, self.b, self.t/6)

        gc = Zeichenflaeche.GibZeichenflaeche().GibGC()
        gc.PushState()
        gc.Translate(self.x+self.b/2, self.y+self.t/2)
        gc.Rotate(radians(self.w))
        gc.Translate(-self.b/2, -self.t/2)
        transformation = gc.GetTransform()
        gc.PopState()
        path.Transform(transformation)
        return path

    def GibFarbe(self):
        """Get-Methode fuer die Farbe"""
        return self.f

    def GibSichtbar(self):
        """Get-Methode fuer die Sichtbarkeit"""
        return self.s

    def BewegeHorizontal(self, weite):
        """Veraendernde Methode fuer die x-Position"""
        self.Verberge()
        self.x += weite
        self.Zeige()

    def BewegeVertikal(self, weite):
        """Veraendernde Methode fuer die y-Position"""
        self.Verberge()
        self.y += weite
        self.Zeige()

    def Drehe(self, winkel):
        """Veraendernde Methode fuer die Orientierung [Winkel]"""
        self.Verberge()
        self.w += winkel
        self.Zeige()

    def Verberge(self):
        """Veraendernde Methode fuer die Sichtbarkeit mit Wert False"""
        self.s = False
        Zeichenflaeche.GibZeichenflaeche().Entferne(self)

    def Zeige(self):
        """Veraendernde Methode fuer die Sichtbarkeit mit Wert True"""
        self.s = True
        Zeichenflaeche.GibZeichenflaeche().Zeichne(self)
```

Schritte zur Vererbung

Schritt 2 umgesetzt:

- Im Dateibrowser mit copy-and-paste beispielsweise die Datei *`tisch.py`* kopiert und zu *`moebel.py`* umbenannt.
- Im Text alle auftretenden *`Tisch`* durch *`Moebel`* ersetzt
- Im Text alle auftretenden *`tisch`* durch *`moebel`* ersetzt

Schritte zur Vererbung

moebel.py - /home/pool/LI/OO/2019-LI-OO/Python-Projekte/Raumplaner-Vererbung ohne Kapselung/moebel.py (2.7.12)

File Edit Format Run Options Window Help

```
from grafikfenster import *
from math import radians

### ------------------------------------------------
class Moebel():
    """Klasse Moebel
    Oberklasse für das Zeichnen und Bearbeiten von
    Möbel-Symbolen fuer den Raumplaner"""

    def __init__(self,
                 xPos,
                 yPos,
                 breite,
                 tiefe,
                 winkel,
                 farbe,
                 sichtbar):
        """Konstruktor mit vordefinierten Parametern
        ist bei Moebel nicht sinnvoll"""
        self.x=xPos
        self.y=yPos
        self.b=breite
        self.t=tiefe
        self.w=winkel
        self.f=farbe
        self.s=sichtbar
        if sichtbar: self.Zeige()
```

Ln: 3 Col: 43

Schritte zur Vererbung

Schritt 3 umgesetzt

- Im Text von allen anderen Möbelklassen alle gemeinsamen Methoden gelöscht
- Wir sind bei Tisch "vorsichtig" vorgegangen, indem wir zunächst diese gemeinsamen Abschnitte auskommentieren
- In den anderen Dateien sind sie gelöscht

Schritte zur Vererbung

```
    def GibFigur(self):
        """Definiert die Figur <path>"""
        gc = Zeichenflaeche.GibZeichenflaeche().GibGC()
        path = gc.CreatePath()

        # Hilfsvariable zur Vereinfachung
        b,t = self.__b, self.__t

        path.AddRectangle(0, 0, b, t)

        x,y,w = self.__x, self.__y, self.__w
        gc.PushState()
        gc.Translate(x+b/2, y+t/2)
        gc.Rotate(radians(w))
        gc.Translate(-b/2, -t/2)
        transformation = gc.GetTransform()
        gc.PopState()
        path.Transform(transformation)
        return path

##    def GibX(self):
##        """Get-Methode fuer die x-Position"""
##        return self.__x
##
##    def GibY(self):
##        """Get-Methode fuer die y-Position"""
##        return self.__y
##
##    def GibBreite(self):
##        """Get-Methode fuer die Breite"""
```

Ln: 70 Col:

Schritte zur Vererbung

Schritt 4

- Wir importieren Moebel und
- teilen dem Programm mit, dass es von der Klasse Moebel abgeleitet wird

Schritte zur Vererbung

Schritt 4 testen:

- Ein Starten der TischApp schlägt nach diesen Änderungen erstaunlicherweise nicht fehl!
- Warum ist das erstaunlich?

Schritte zur Vererbung

Problem:

- wenn tisch.py ausgeführt wird:
  - Welcher Klassentext wurde denn nun dafür ausgeführt?
  - Welche Methode GibFigur() wurde denn nun ausgeführt?
  - Auf welche Attribute wurde zugegriffen?

Schritte zur Vererbung

tisch.py - /home/nutzer/Dokumente/LI/2021-LI-OO/Projekte/02/Raumplaner-Moebel-Schritt... ×

File Edit Format Run Options Window Help

```
# tisch.py

from grafikfenster import *
from math import radians
from moebel import Moebel

### ------------------------------------------------
class Tisch(Moebel):
    """Klasse Tisch
    ermoeglicht das Zeichnen und Bearbeiten eines
    Tisch-Symbols fuer den Raumplaner"""

    def __init__(self,
                 xPos=60,
                 yPos=10,
                 breite=120,
                 tiefe=60,
                 winkel=0,
                 farbe='red',
                 sichtbar=False):
        """Konstruktor mit vordefinierten Parametern"""
        self.x=xPos
        self.y=yPos
        self.b=breite
        self.t=tiefe
        self.w=winkel
        self.f=farbe
        self.s=sichtbar
        if sichtbar: self.Zeige()

    def GibFigur(self):
        """definiert und transformiert die zu zeichnende Figur"""
```

Schritte zur Vererbung

moebel.py - /home/nutzer/Dokumente/LI/2021-LI-OO/Projekte/02/Raumplaner-Moebel-Sch... ×

File Edit Format Run Options Window Help

```
# moebel.py

from grafikfenster import *
##from math import radians # unnoetig, da nur bei konkreten M.

### ------------------------------------------------
class Moebel():
    """Klasse Moebel
    Oberklasse fuer Moebel-Symbole zum Raumplaner"""

    def __init__(self,
                 xPos=20,
                 yPos=20,
                 breite=40,
                 tiefe=40,
                 winkel=0,
                 farbe="blue",
                 sichtbar=False):
        """Konstruktor mit vordefinierten Parametern ???"""
        self.x=xPos
        self.y=yPos
        self.b=breite
        self.t=tiefe
        self.w=winkel
        self.f=farbe
        self.s=sichtbar
        if sichtbar: self.Zeige()

    def GibFigur(self):
        """Definiert den Pfad und transformiert ihn."""
```

Schritte zur Vererbung

Schritt 4 testen:

- Beide Klassendefinitionen enthalten gleichnamige Attribute
- Welche werden verwendet?

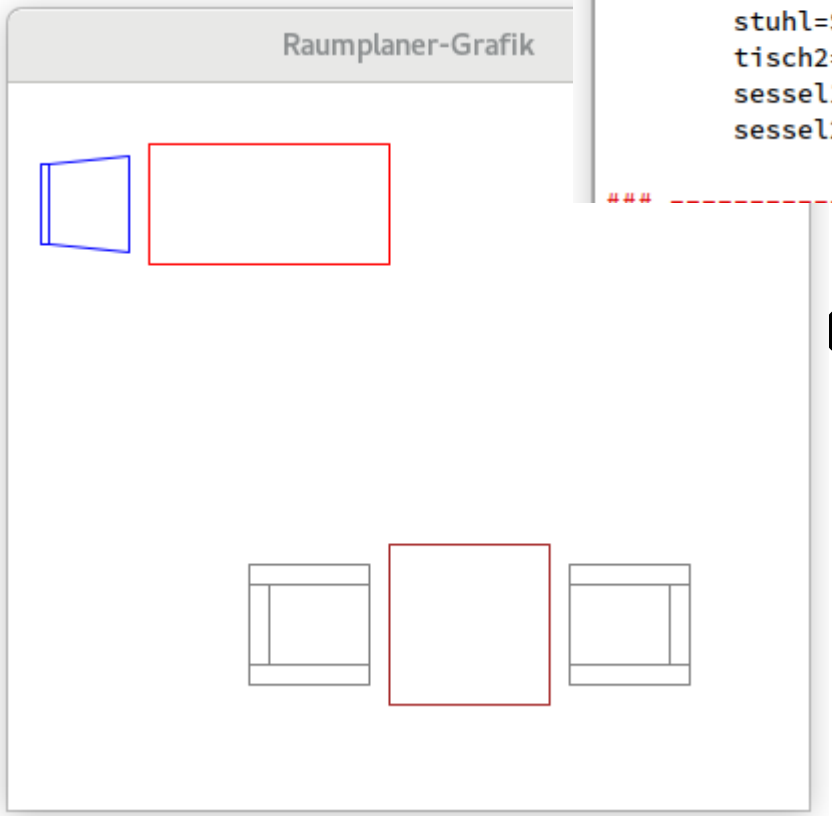
Eine Antwort gibt das Starten der Testanwendung in Raumplaner.py

- Alle Objekte lassen sich richtig positionieren!

Schritte zur Vererbung

Schritt 4 testen

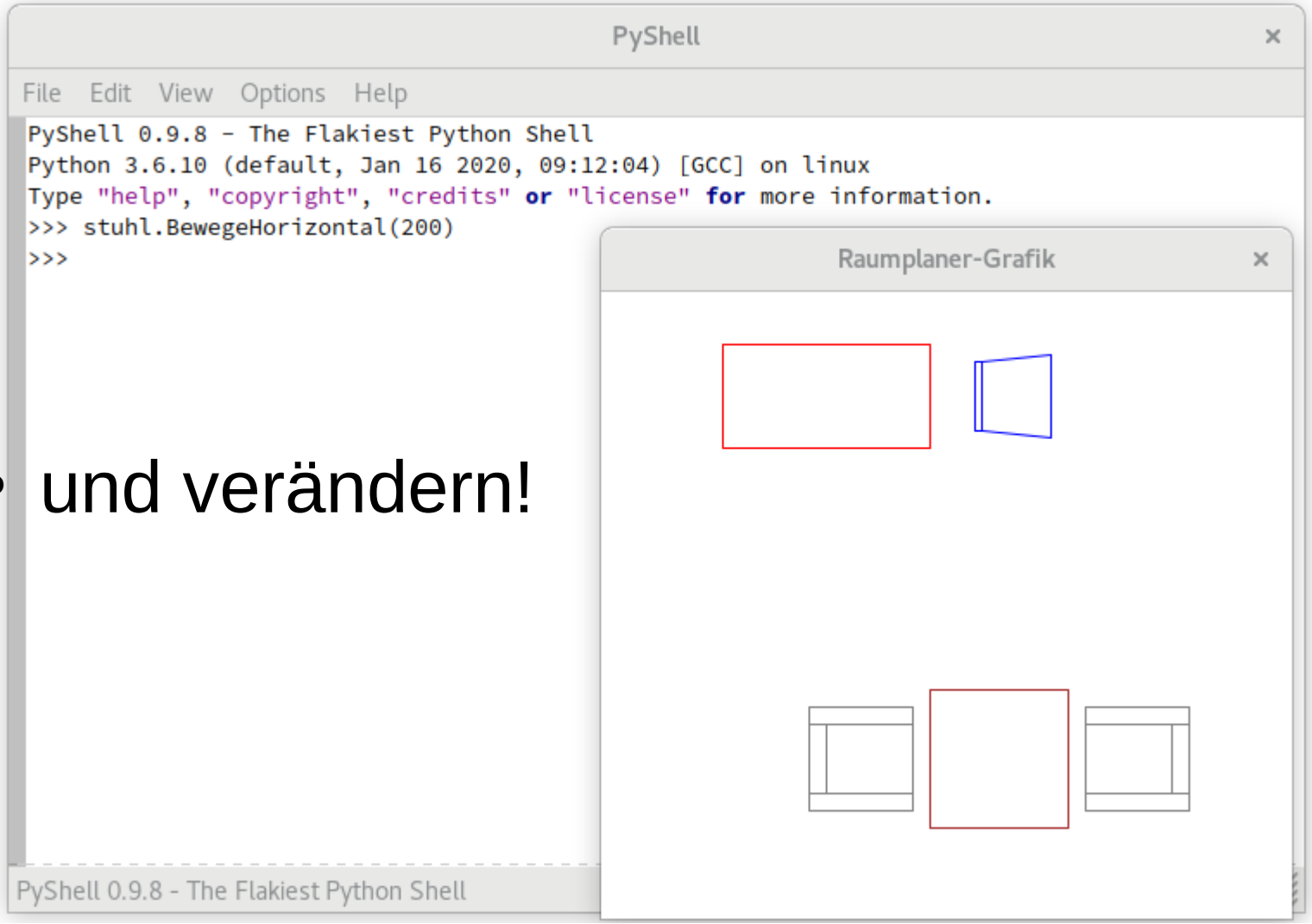
```
    def TestAnwendung(self, sesselvariante=1):
        """Allein fuer die Testanwendung:"""
        global tisch1, stuhl, tisch2, sessel1, sessel2
        print('global: tisch1, stuhl, tisch2, sessel1, sessel2')
        tisch1=Tisch(70, 30, 120, 60, 0, 'red', True)
        stuhl=Stuhl(20, 40, 40, 40, 270, 'blue', True)
        tisch2=Tisch(190, 230, 80, 80, 0, 'brown', True)
        sessel1=Sessel(120, 240, 60, 60, 270, 'gray', True)
        sessel2=Sessel(280, 240, 60, 60, 90, 'gray', True)

### ------------------------------------------------
```



en sich richtig darstellen!

Schritte zur Vererbung

Schritt 4 testen

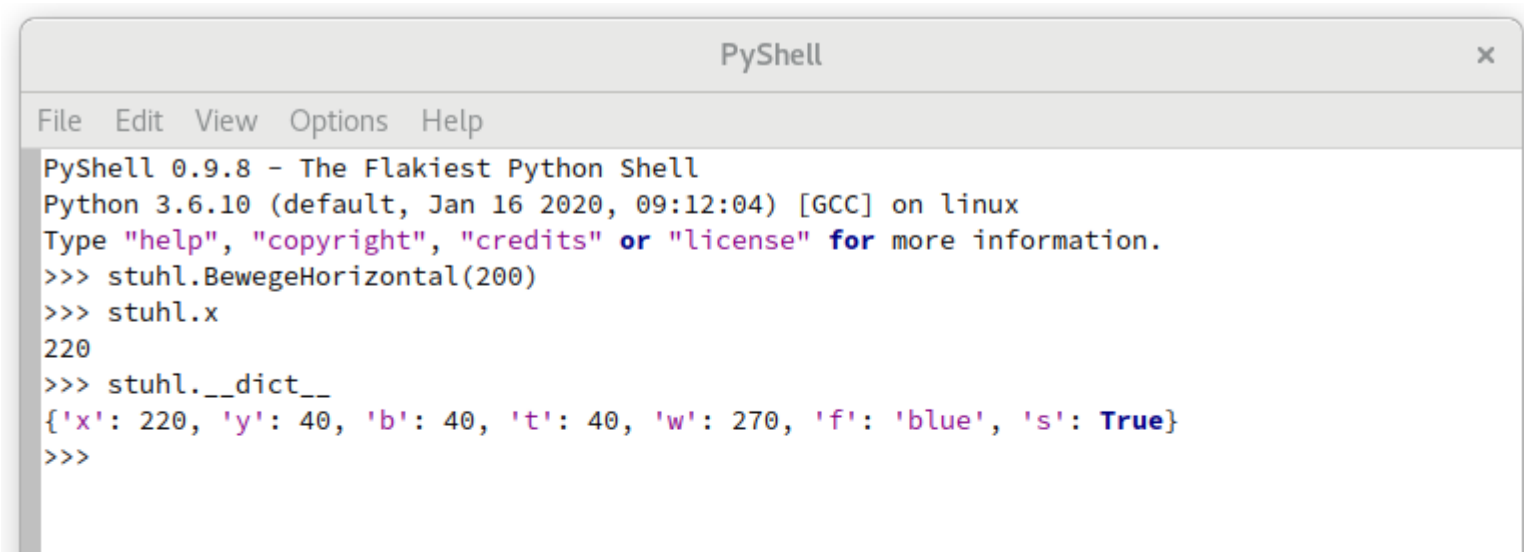


- und verändern!

Schritte zur Vererbung

Schritt 4 testen:

- Die Zugriffe erfolgen also auf die in den Tochterklassen definierten Attribute,
- obwohl sie aus den in Moebel definierten Methoden erfolgen!



Schritte zur Vererbung

Vererbung im Raumplaner-Projekt

machen wir es besser!!!