Vererbung

Im Raumplaner-Projekt

Vererbung einsetzen

(kurze Version ohne UML)

Vererbung

Schritt 1

- Klären Sie, welche Attribute und Methoden in den Klassen *`stuhl`* und *`tisch`* (und …) identisch sind und daher nach *`moebel`* verschoben werden können.
- Das folgende Bild zeigt den wesentlichen Unterschied.

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)
```

Vererbung

Schritt 2

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

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

Vererbung

Schritt 3

- Im Text von allen anderen Möbelklassen alle gemeinsamen Methoden löschen
- Wir bleiben hier bei Tisch und gehen "vorsichtig" vor, indem wir zunächst diese gemeinsamen Abschnitte auskommentieren

Vererbung

tisch.py - /home/nutzer/Dokumente/LI/2021-LI-OO/Projekte/02/Moebel-erarbeiten-aus-Anf...

File Edit Format Run Options Window Help

```
        self.f='red'
        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()
```

Vererbung

Schritt 4

- Ein Starten von **tisch.py** schlägt natürlich fehl, da ihm wichtige Teile fehlen.
- Moniert wird self.Zeige(), der Aufruf in der letzten Zeile des Konstruktors.
- Das Programm muss wissen, dass es die Methoden von Moebel verwenden soll.
  - Wir ...

Vererbung

Schritt 4

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

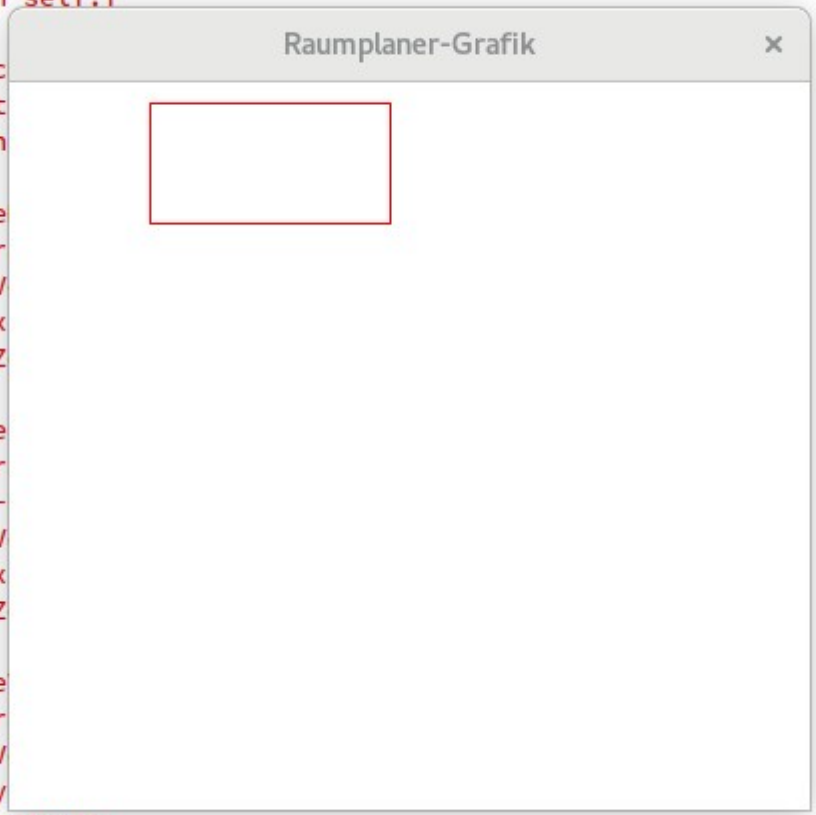
Vererbung

```
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"""
        gc = Zeichenflaeche.GibZeichenflaeche().GibGC()
        path = gc.CreatePath()
```

Vererbung

- Ein Starten des Programms schlägt nicht mehr fehl!

```
##    def GibFarbe(self):
##        """Get-Methode fuer die Farbe"""
##        return self.f
##
##    def GibSic
##        """Get
##        return
##
##    def Bewege
##        """Ver
##        self.V
##        self.x
##        self.Z
##
##    def Bewege
##        """Ver
##        die y-
##        self.V
##        self.x
##        self.Z
##
##    def Bewege
##        """Ver
##        self.V
##        self.y
##        self.Zeige()
##
```



Vererbung

Schritt 5:

- tisch.py erfolgreich ausgeführt → aber:
  - Welcher Klassentext wurde denn nun dafür ausgeführt?
  - Welche Methode GibFigur() wurde denn nun ausgeführt?
- Und:
  - Auf welche Attribute wird zugegriffen?
- Überlegen Sie sich Tests, mit denen diese Fragen beantwortet werden können.

Vererbung

Schritt 6:

- Diese Fragen müssen klar beantwortet werden können.
- Wir verbessern die Lösung:
  - Attribute nur an einer Stelle definieren
  - Werte an die Oberklasse explizit übergeben

Vererbung

File Edit Format Run Options Window Help

```
from moebel import Moebel

### ------------------------------------------------
class Stuhl(Moebel):
    """Klasse Stuhl
    ermoeglicht das Zeichnen und Bearbeiten eines
    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 Parametern
        Konstruktor der Oberklasse aufrufen!"""
        Moebel.__init__(self, xPos, yPos, breite, tiefe, winkel, farbe, sichtbar
        if sichtbar: self.Zeige()

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

        # Hilfsvariable zur Vereinfachung
        b,t = self.b, self.t
        path.MoveToPoint(0, 0)
        path.AddLineToPoint(b, 0)
        path.AddLineToPoint(b*1.1, t)
        path.AddLineToPoint(-b*0.1, t)
        path.AddLineToPoint(0, 0)
        path.AddLineToPoint(0, -t*0.1)
```

Ln: 1 Col: 0