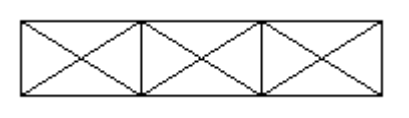
Wiederholung Schrankwand

Zu Termin 5 / 2021

Eine Klasse für
eine Schrankwand
erstellen

Wiederholung Schrankwand

- Anlass ist die Aufgabe, eine Klasse zu schreiben, die eine Schrankwand aus drei Schränken darstellt.

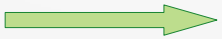


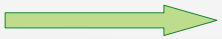
- Im ersten Schritt speichern wir schrank.py als schrankwand.py ab und benennen intern passend um.
- GibFigur(self) bleibt unverändert.

```
def GibFigur(self):
    """Definiert den Pfad."""
    gc = Zeichenflaeche.GibZeichenflaeche().GibGC()
    path = gc.CreatePath()

    # lokale Variable zur Schreibvereinfachung
    b, t = self.GibBreite(), self.GibTiefe()
    path.AddRectangle(0, 0, b, t)
    path.MoveToPoint(0, 0)
    path.AddLineToPoint(b, t)
    path.MoveToPoint(b, 0)
    path.AddLineToPoint(0, t)

    return self.Transformiere(path)
```

Wiederholung Schrankwand

- Wir
  - kopieren den Abschnitt zur Definition eines Schranks,
  - fügen ihn zweimal ein und
  - editieren die relativen Positionen.





```
# lokale Variable zur Schreibvereinfachung
b, t = self.GibBreite(), self.GibTiefe()
path.AddRectangle(0, 0, b, t)
path.MoveToPoint(0, 0)
path.AddLineToPoint(b, t)
path.MoveToPoint(b, 0)
path.AddLineToPoint(0, t)
# zweiter Shrank
path.AddRectangle(b, 0, b, t)
path.MoveToPoint(b, 0)
path.AddLineToPoint(2*b, t)
path.MoveToPoint(2*b, 0)
path.AddLineToPoint(b, t)
# dritter Schrank
path.AddRectangle(2*b, 0, b, t)
path.MoveToPoint(2*b, 0)
path.AddLineToPoint(3*b, t)
path.MoveToPoint(3*b, 0)
path.AddLineToPoint(2*b, t)
```

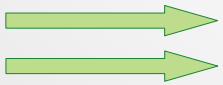
Wiederholung Schrankwand

- Die Darstellung gelingt wie gewünscht.
- Das Vorgehen wird im Kurs hinterfragt und statt dessen 1. vorgeschlagen:

In der Methode GibFigur()
soll der Programmblock
für je einen der Schränke
wiederholt (Schleife!) aufgerufen werden

Wiederholung Schrankwand

- Wir erarbeiten, dass
  - eine Wiederholungsstruktur notwendig ist,
  - dazu eine Zählschleife verwendet wird und
  - deren Zählindex zur Berechnung der relativen Position verwendet wird



```
    def GibFigur(self):
        """definiert die zu zeichnende Figur (Pfad)"""
        path = self.GibZeichenPfad()

        b,t=self.GibBreite()/3, self.GibTiefe() # Anpassung !
        for i in range(3): # Zaehlschleife
            path.AddRectangle(i*b, 0, b, t)
            path.MoveToPoint(i*b, 0)
            path.AddLineToPoint((i+1)*b, t)
            path.MoveToPoint((i+1)*b, 0)
            path.AddLineToPoint(i*b, t)

        return self.Transformiere(path)

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

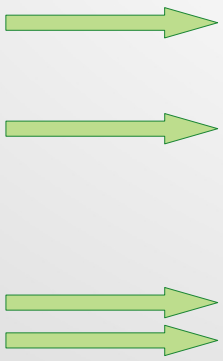
Wiederholung Schrankwand

- Die Darstellung gelingt wie gewünscht.
- Das Vorgehen wird im Kurs hinterfragt und statt dessen 2. vorgeschlagen:

Mit der Klasse Schrankwand
sollen Schrankwände
aus einer beliebigen Anzahl
von Schränken erstellt werden können.

Wiederholung Schrankwand

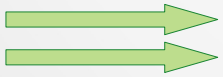
- Wir erarbeiten, dass dazu
  - der Konstruktor der Schrankwandklasse einen Parameter für die Anzahl haben muss,
  - nicht mehr die Breite der Schrankwand, sondern eines Schranks übergeben werden sollte, so dass die Breite *(notwendig für den Aufruf des Konstruktors von **Moebel**)* berechnet werden muss.



```
class Schrankwand(Moebel):
    def __init__(self,
                 anzahl=3,
                 xPos=0,
                 yPos=0,
                 schrankbreite=60,
                 tiefe=37,
                 winkel=0,
                 farbe="black",
                 sichtbar=False):
        self.__anzahl=anzahl
        Moebel.__init__(self, xPos, yPos, anzahl*schrankbreite, tiefe,
```

Wiederholung Schrankwand

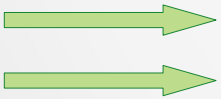
- Wir erarbeiten, dass
  - im Konstruktor zwar der Parameter **`anzahl`** verwendet werden kann,
  - aber ein Attribut **`self.anzahl`** benötigt wird, da dieser Wert auch in der Methode **`GibFigur()`** benötigt wird.



```
def GibFigur(self):
    """definiert die zu zeichnende Figur (Pfad)"""
    path = self.GibZeichenPfad()

    b,t=self.GibBreite()/self.anzahl, self.GibTiefe() # Anpassung !
    for i in range(self.anzahl): # Zaehlschleife
        path.AddRectangle(i*b, 0, b, t)
        path.MoveToPoint(i*b, 0)
        path.AddLineToPoint((i+1)*b, t)
        path.MoveToPoint((i+1)*b, 0)
        path.AddLineToPoint(i*b, t)

    return self.Transformiere(path)
```

Wiederholung Schrankwand

- Wir bringen im nächsten Schritt beide Ansätze zusammen und erarbeiten, dass dazu
  - eine Sammlungsstruktur verwendet werden muss
  - statt *(einer möglichen statischen Struktur entsprechend Arrays)* verwenden wir eine Liste.



```
self.anzahl = anzahl
self.schraenke = []
for i in range(anzahl): # Zaehlschleife
    self.schraenke.append(Schrank(i*schrankbreite, 0, schrankbreite, tie
Moebel.__init__(self, xPos, yPos,
                anzahl*schrankbreite, tiefe, winkel, farbe, sichtbar)
```

Wiederholung Schrankwand

- Wir bringen im nächsten Schritt beide Ansätze zusammen und erarbeiten, dass dazu
  - in **`GibFigur()`** statt einer Zählschleife *(auskommentiert)*
  - auch eine for-each-Schleife verwendet werden kann.

```
    def GibFigur(self):
        """definiert die zu zeichnende Figur (Pfad)"""
        path = self.GibZeichenPfad()

        for schrank in self.schraenke:
            path.AddPath(schrank.GibFigur())

##          for i in range(self.anzahl): # Zaehlschleife
##              path.AddPath(self.schraenke[i].GibFigur())

        return self.Transformiere(path)
```