Umbau zu Aggregation

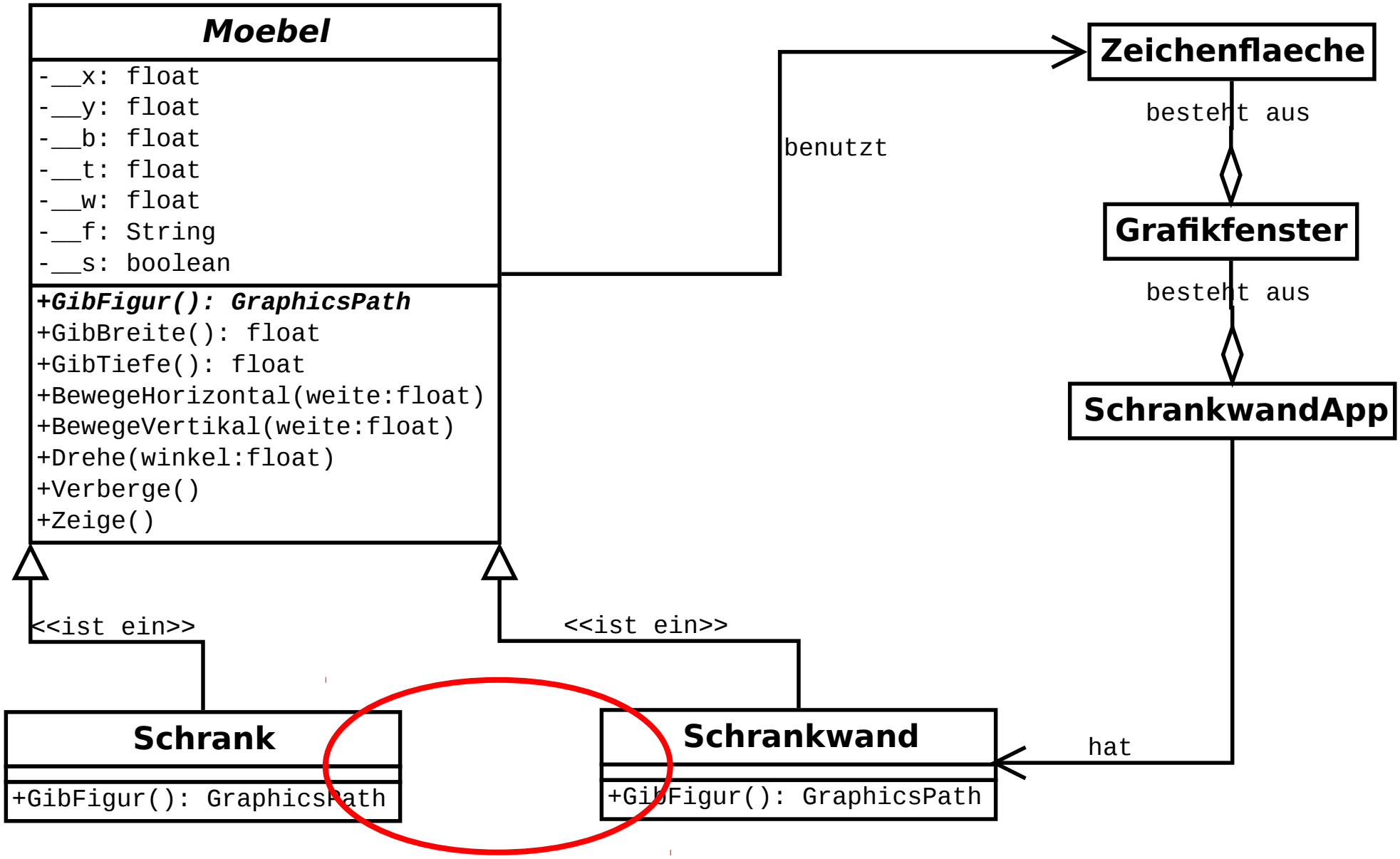
Modellierung
umsetzen

Umbau zu Aggregation

Ausgangspunkt ist das Programm mit den beiden getrennten Klassen Schrank und Schrankwand.

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

    # lokale Hilfsvariable
    b,t=self.__schrankbreite, self.GibTiefe()

    for i in range(self.__anzahl):
        path.AddRectangle(i*b, 0, b, t)
        path.MoveToPoint(i*b, 0)
        path.AddLineToPoint(i*b+b, t)
        path.MoveToPoint(i*b+b, 0)
        path.AddLineToPoint(i*b, t)

    return self.Transformiere(path)
```

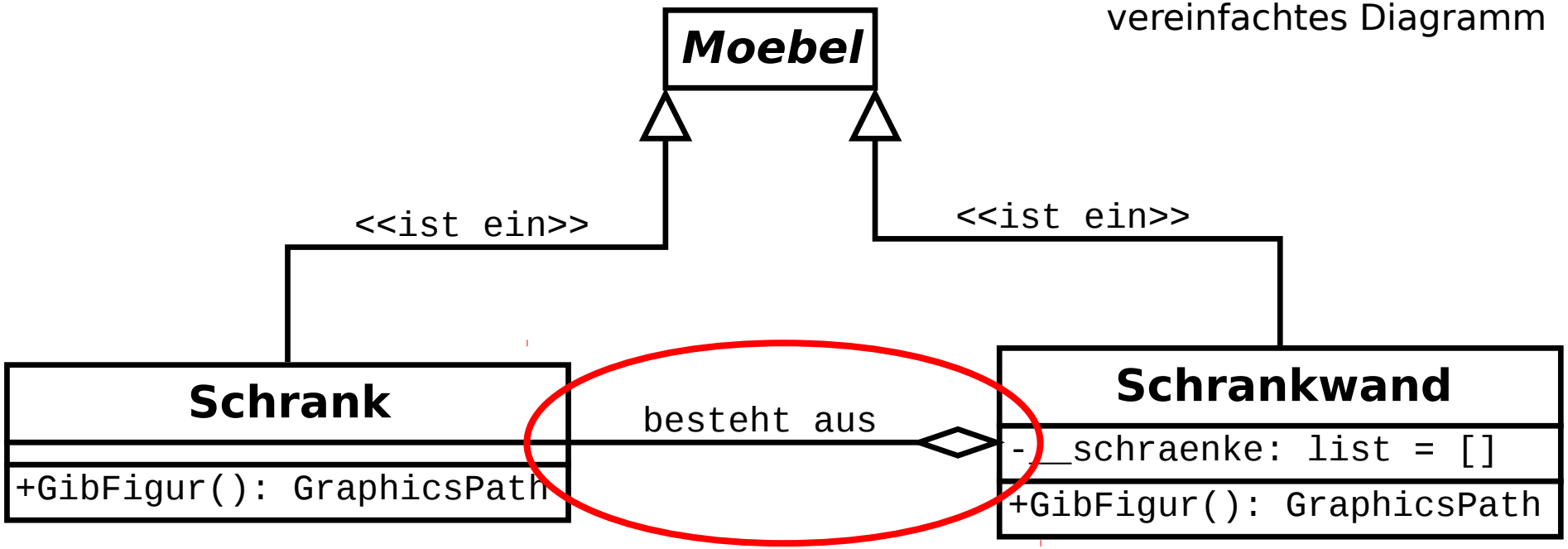
Umbau zu Aggregation



Umbau zu Aggregation

Änderung der Modellierung

vereinfachtes Diagramm



Umbau zu Aggregation

Änderung des Programms:

- In der Klasse Schrankwand Import der Klasse Schrank, da wir Schrankobjekte erzeugen wollen

```
from schrank import Schrank
```

Umbau zu Aggregation

Änderung des Programms:

- Der Ansatz mit der Definition von drei Schrankobjekten im Konstruktor wird verworfen

```
def __init__(self,
             anzahl=3,  # Ergaenzung
             xPos=0,
             yPos=0,
             schrankbreite=60,
             tiefe=37,
             winkel=0,
             farbe="black",
             sichtbar=False):
    self.__anzahl=anzahl
    self.__schrankbreite=schrankbreite
    schrank1=Schrank(xPos, yPos, schrankbreite, tiefe, winkel, farbe, sichtbar)
    schrank2=Schrank(xPos+schrankbreite, yPos, schrankbreite, tiefe, winkel, farbe, sichtbar)
    schrank3=Schrank(xPos+2*schrankbreite, yPos, schrankbreite, tiefe, winkel, farbe, sichtbar)
    Moebel.__init__(self, xPos, yPos, anzahl*schrankbreite, tiefe, winkel, farbe, sichtbar)
```

Umbau zu Aggregation

Änderung des Programms:

- Liste einbauen

```
def __init__(self,
             anzahl=3,  # Ergaenzung
             xPos=0,
             yPos=0,
             schrankbreite=60,
             tiefe=37,
             winkel=0,
             farbe="black",
             sichtbar=False):
    self.__anzahl=anzahl
    self.__schrankbreite=schrankbreite
    schraenke=[]

    Moebel.__init__(self, xPos, yPos, anzahl*schrankbreite, tiefe, winkel, farbe, sichtbar)
```

Umbau zu Aggregation

Änderung des Programms:

- Liste mit Hilfe einer Zählschleife füllen

```
def __init__(self,
             anzahl=3,  # Ergaenzung
             xPos=0,
             yPos=0,
             schrankbreite=60,
             tiefe=37,
             winkel=0,
             farbe="black",
             sichtbar=False):
    self.__anzahl=anzahl
    self.__schrankbreite=schrankbreite
    schraenke=[]
    for i in range(anzahl):
        schraenke.append(Schrank(xPos+i*schrankbreite, yPos, schrankbreite, ...)
    Moebel.__init__(self, xPos, yPos, anzahl*schrankbreite, tiefe, winkel, farbe, sichtbar)
```

Umbau zu Aggregation

Änderung des Programms:

- In GibFigur() von Schrankwand auf GibFigur() von jedem Schrank zugreifen

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

    # lokale Hilfsvariable
    b,t=self.__schrankbreite, self.GibTiefe()

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

    return self.Transformiere(path)
```

ACHTUNG!
schraenke muss Attribut sein,
sonst haben wir hier
keinen Zugriff

Umbau zu Aggregation

Änderung des Programms:

- **ein Attribut Liste** mit Hilfe einer Zählschleife füllen

```
def __init__(self,
             anzahl=3,  # Ergaenzung
             xPos=0,
             yPos=0,
             schrankbreite=60,
             tiefe=37,
             winkel=0,
             farbe="black",
             sichtbar=False):
    self.__anzahl=anzahl
    self.__schrankbreite=schrankbreite
    self.__schraenke=[]
     for i in range(anzahl):
         self.__schraenke.append(Schrank(xPos+i*schrankbreite, yPos, ...)
    Moebel.__init__(self, xPos, yPos, anzahl*schrankbreite, tiefe, winkel, farbe, sichtbar)
```

Umbau zu Aggregation

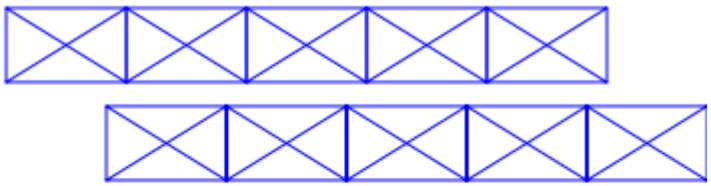
Änderung des Programms:

- In GibFigur() von Schrankwand auf GibFigur() von jedem Schrank zugreifen

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

    # lokale Hilfsvariable
    b,t=self.__schrankbreite, self.GibTiefe()

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

    return self.Transformiere(path)
```

Umbau zu Aggregation

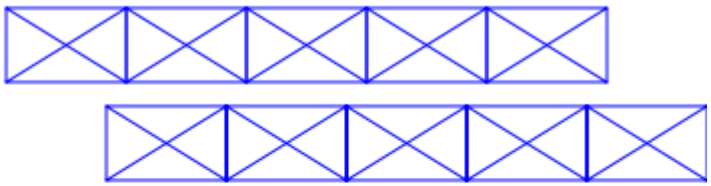
- Warum 2 Schrankwände?



Umbau zu Aggregation

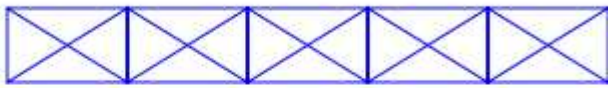
Schrankobjekte selbst dürfen nicht sichtbar sein

- Attributwert von sichtbar ändern



Umbau zu Aggregation

- Warum nach rechts und unten verschoben?



Umbau zu Aggregation

Warum nach rechts und unten verschoben?

- relative Koordinaten verwenden

```
self.__schraenke.append(Schrank(i*schrankbreite, 0, schrankbreite, tiefe, winkel, farbe, False))
```