# Das Singleton Entwurfsmuster

# Singleton

Begründung

- Die Anwendung greift zum Zeichnen mehrfach auf das Zeichenflaeche-Objekt zu.
- Ziel ist dabei, alle Möbelobjekte in der selben Zeichenfläche darzustellen.

# Singleton

Realisierung bei **Java**:

- Versehe den Konstruktor der Klasse Zeichenflaeche mit dem Sichtbarkeitsattribut private.
- Stelle in der Klasse eine Klassenmethode gibZeichenflaeche() bereit, die genau einmal intern den Konstruktor aufruft, danach immer das vorhandene Objekt zurück gibt.

# Singleton

Realisierung bei **Python**:

- Python kennt kein Sichtbarkeitsprädikat *private*.
- Eine Möglichkeit ist, einen Fehler auszulösen, wenn der Konstruktor extern aufgerufen wird.
- Stelle in der Klasse eine Klassenmethode GibZeichenflaeche() bereit, die genau einmal intern den Konstruktor aufruft, danach immer das vorhandene Objekt zurück gibt.

# Singleton

eine *Klassenmethode* (?)

- Das Problem ist, dass es beim ersten Aufruf des Konstruktors noch keine Instanz von der Zeichenflaeche gibt.
- Es muss also eine Methode sein, die nicht auf ein Objekt angewiesen ist, sondern direkt der Klasse zugeordnet ist.
- Sie muss dazu gekennzeichnet sein mit `@staticmethod` und kann nicht `self` als Parameter haben.

# Singleton

Python Beispielcode in Zeichenflaeche:

- Attribut und Methode:

```
 __zeichenflaeche=None
@staticmethod
def GibZeichenflaeche ( parent=None ):
   if Zeichenflaeche.__zeichenflaeche == None:
      if parent==None:
         return None
      else:
         Zeichenflaeche.__zeichenflaeche=Zeichenflaeche(parent)
   return Zeichenflaeche.__zeichenflaeche
```

# Singleton

Python Beispielcode in Zeichenflaeche:

- Konstruktor:

```
def __init__(self, parent):
    """Konstruktor
    !nicht direkt aufrufen!"""
    if Zeichenflaeche.__zeichenflaeche != None:
            raise Exception('Konstruktor nicht direkt aufrufen')

    wx.Panel.__init__(self, parent, -1)
    ...
```

# Singleton

- Testaufrufe in der Shell
  (*Konstruktor ohne die Exception-Zeile*)

```
zf0=Zeichenflaeche(app.fenster)
```

so bitte nicht!

```
zf1=Zeichenflaeche.GibZeichenflaeche(app.fenster)
zf2=Zeichenflaeche.GibZeichenflaeche(app.fenster)
print(zf0)
<grafikfenster.Zeichenflaeche object at 0x7f29b8e9a0d8>
print(zf1)
<grafikfenster.Zeichenflaeche object at 0x7f29bbc5a318>
print(zf2)
<grafikfenster.Zeichenflaeche object at 0x7f29bbc5a318>
```

# Singleton

- Testaufrufe in der Shell (*Konstruktor mit Exception-Zeile*)

```
zf0=Zeichenflaeche(app.fenster)
Traceback (most recent call last):
  File "<input>", line 1, in <module>
  File
"/home/nutzer/Dokumente/LI/OO/2025-LI-OO/Kursprojekte/Moebelgruppe_Kompositum_P
ersistenz_Kurs/grafikfenster.py", line 39, in __init__
    raise Exception('Konstruktor nicht direkt aufrufen')
Exception: Konstruktor nicht direkt aufrufen
zf1=Zeichenflaeche.GibZeichenflaeche(app.fenster)
print(zf1)
<grafikfenster.Zeichenflaeche object at 0x7fb7b0ae5708>
zf2=Zeichenflaeche.GibZeichenflaeche(app.fenster)
print(zf2)
<grafikfenster.Zeichenflaeche object at 0x7fb7b0ae5708>
```