Einbau einer
Controller-Klasse

Controller

Anlass:

- Wo speichert man sinnvollerweise die erzeugten Objekte? (sicher nicht in Gui)
- Wohin gehen Mausereignisse der Zeichenflaeche? (sicher nicht zu Objekten)
- Wer sorgt für einen konsistenten Zustand?

Controller

Umsetzen des

- Model – View – Controller – Konzepts
- Ziel:
  - Verringern der Kopplung
  - Abhängigkeiten verringern
  - klare Verantwortlichkeiten

Controller

Controller

- Unsere Klasse `raumplanerModell.py` bereitet eine Controllerklasse bereits vor.
- Der bisherige Begriff betrachtet die Software allein aus Sicht des Modells, für das es eine Hülle sein sollte.
- Die Anforderungen, die bereits angesprochen worden sind, verstärken eine andere Sicht auf diese Klasse: Sie ist bereits weitgehend eine Controllerklasse!

Controller

Controller

- Der primitive erste Schritt zur Umwandlung der Klasse `RaumplanerModell` in eine Klasse `RaumplanerController` besteht allein in einer Umbenennung.
- Was ist noch zu tun?

Controller

Die Controllerklasse

- Es kann nur ein Controller-Objekt geben.
- Das bedeutet eine Modellierung als Singleton.
- Die ist in unserem Projekt umgesetzt mit
  - Klassenvariable und
  - statischer Zugriffsmethode

Controller

```
__raumplanerController = None

@staticmethod
def GibRaumplanerController():
    if RaumplanerController.__raumplanerController == None:
        RaumplanerController.__raumplanerController
                                        =RaumplanerController()
    return RaumplanerController.__raumplanerController
```

Controller

Controllerobjekt in der RaumplanerApp erzeugen und die Objekte jeweils bekannt machen

```
self.controller=RaumplanerController.
                    GibRaumplanerController()
self.controller.SetzeGui(self.gui)
self.gui.SetzeController(self.controller)
self.zf=Zeichenflaeche.GibZeichenflaeche()
self.zf.SetzeController(self.controller)
self.controller.SetzeZeichenflaeche(self.zf)
self.gui.SetzeZeichenflaeche(self.zf)
```

Controller

Hinweis

- Mausereignisse der Zeichenflaeche werden nicht dort verarbeitet

```
self.Bind(wx.EVT_LEFT_DOWN, self.OnLeftDown)
self.Bind(wx.EVT_LEFT_UP, self.OnLeftUp)
self.Bind(wx.EVT_LEFT_DCLICK, self.OnLeftDoubleClick)
self.Bind(wx.EVT_RIGHT_DOWN, self.OnRightDown)
self.Bind(wx.EVT_RIGHT_UP, self.OnRightUp)
self.Bind(wx.EVT_MOTION, self.OnMotion)

def OnLeftDown(self, event):
    print('angeklickt',event.GetX(),event.GetY())
    event.Skip()
```

Controller

Beispiel

- Mausereignisse der Zeichenflaeche werden an die Controller-Klasse weiter gereicht:

```
def OnLeftDown(self, event):
    Zeichenflaeche.__controller.Angeklickt(
        event.GetX(),event.GetY())
    event.Skip()

def OnLeftUp(self, event):
    Zeichenflaeche.__controller.LinksLosgelassen(
        event.GetX(),event.GetY())
    event.Skip()
```

(was bringt das?)

Controller

Beispiel

- Mausereignisse werden in der Controller-Klasse "verarbeitet":

```
def Angeklickt(self, x, y):
    ...

def LinksLosgelassen(self, x, y):
    ...

usw
```

Controller

Vorteil

- Wenn der Controller alle Moebel kennt, kann er untersuchen, wem der Klick gegolten hat.

```
def Angeklickt(self, moebel, x, y):
    return self.__zf.Angeklickt(moebel.GibFigur(),x,y)

def LinksKlick(self, x, y):
    for moebel in self.__alleMoebel:
        if self.Angeklickt(moebel,x,y):
        self.Waehle(self.__alleMoebel.index(moebel))
            self.__clickPosition=(x,y)
            return moebel
    self.Waehle(-1)
    self.__clickPosition=None
    return None
```