RaumplanerModell

Eine Hülle
für die
Moebelklassen

RaumplanerModell

- Eine eigene Modellklasse für die Moebel-Klassen übernimmt die gesamte Verwaltung der erzeugten Möbel
  - einschließlich Speichern und Laden,
  - sowie der Auswahl eines einzelnen Objekts
- Die Klasse ist als Singleton modelliert

Hinweis: Auf Möbelgruppen wird an dieser Stelle verzichtet

RaumplanerModell

- Die Klasse ist als Singleton modelliert

```
__raumplanerModell = None

@staticmethod
def GibRaumplanerModell():
    if RaumplanerModell.__raumplanerModell == None:
        RaumplanerModell.__raumplanerModell =
                                        RaumplanerModell()
    return RaumplanerModell.__raumplanerModell
```

RaumplanerModell

- Die Klasse ist als Singleton modelliert

```
def __init__(self):
    if RaumplanerModell.__raumplanerModell != None:
        raise Exception('Konstruktor nicht direkt
                         aufrufen')
    self.__alleMoebel = []
    self.__ausgewaehlt = -1
```

RaumplanerModell

- Erzeugen eines neuen Objekts

```
def Neu(self, klasse, *par, **kwargs):
    if klasse.__class__==str:
        klasse=eval(klasse)
    neu=klasse(*par, **kwargs)
    self.__alleMoebel.append(neu)
    return neu
```

RaumplanerModell

- Speichern aller Objekte (Laden entsprechend)

```
def Speichere(self, dateiname):
    """Objektdaten werden in Datei geschrieben"""
    try:
        with open(dateiname, 'wb') as objekt_datei:
            pickle.dump(self.__alleMoebel,
                        objekt_datei)
            return 'OK'
    except IOError as ioerr:
        return 'Dateifehler: ' + str(ioerr)
    except pickle.PicklingError as perr:
        return 'Pickle-Fehler: ' + str(perr)
```

RaumplanerModell

- Auswählen eines Objekts
- arbeitet indexbasiert, die Aufrufvarianten sind:
  - index ist nicht None
  - index ist None -> weiterzählen

RaumplanerModell

- Auswählen eines Objekts
- arbeitet indexbasiert, die Aufrufvarianten sind:
  - index ist nicht None
  - index ist None -> weiterzählen

```
if index==-1: # Beenden der Auswahl
    ...
# Indexfehler pruefen
if index<-1: return
if index>=len(self.__alleMoebel): return
## sonst auswaehlen und markieren
   ...
```

RaumplanerModell

- Auswählen eines Objekts
- arbeitet indexbasiert, die Aufrufvarianten sind:
  - index ist nicht None
  - index ist None -> weiterzählen

```
...
self.__ausgewaehlt+=1
...
## am Ende Auswahl löschen
if self.__ausgewaehlt==len(self.__alleMoebel):
    self.__ausgewaehlt=-1
```

RaumplanerModell

- Das Moebelobjekt sollte wissen, ob es ausgewählt ist oder nicht

```
# im Konstruktor von Moebel
self.__ausgewaehlt=False
...
def GibFarbe(self):
    """Get-Methode fuer die Farbe"""
    if self.__ausgewaehlt: return "gray"
    return self.__f
...
def Waehle(self, auswaehlen):
    if auswaehlen: self.__ausgewaehlt=True
    else: self.__ausgewaehlt=False
    self.Update()
```