Klasse RaumplanerModell

Die Aufgaben für eine Klasse
RaumplanerModell
und ihre Umsetzung

Klasse RaumplanerModell

Die Aufgaben

- *gesamte Verwaltung der erzeugten Moebel*
- *Speichern*
- *Laden*
- *Auswahl eines speziellen Objekts, um es für eine Bearbeitung herauszuheben*
- *Bearbeitung von Objekten ermöglichen*

Beachten: RaumplanerModell als Singleton modellieren!

Klasse RaumplanerModell

Verwaltung der erzeugten Moebel

- *Attribut für alle erzeugten Moebel-Objekte:*

```
self.__alleMoebel = []
```

- *Get-Methode dafür*

```
def GibAlleMoebel(self):
    return self.__alleMoebel
```

- *Methode zum Erzeugen der Moebel-Objekte:*

```
def Neu(self, klasse, *par, **kwargs):
    ...
    self.__alleMoebel.append(neu)
```

Klasse RaumplanerModell

Methode zum Erzeugen der Moebel-Objekte:

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

Erläuterungen

- ***klasse** kann als Name übergeben werden*
- *damit Konstruktor arbeitet, muss es aber die Klasse sein*
- ***eval** wertet String zu Objekt → Klasse aus!*

Klasse RaumplanerModell

Speichern

```
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)
```

Wichtig: Fehler abfangen und Meldung zurück geben !

Klasse RaumplanerModell

Laden, Schritt 1
vorhandene Möbelobjekte verbergen und löschen

```
def Lade(self, dateiname):
    """Objektdaten werden aus der Datei gelesen"""
    for moebel in self.__alleMoebel:
      moebel.Verberge()
      del(moebel)
    self.__ausgewaehlt = -1
    ...
```

siehe spätere Folie

Klasse RaumplanerModell

Laden, Schritt 2

```
def Lade(self, dateiname):
    ...
    try:
      with open(dateiname, 'rb') as objekt_datei:
        self.__alleMoebel = pickle.load(objekt_datei)
        for moebel in self.__alleMoebel:
          if moebel.GibSichtbar():
            moebel.Zeige()
        return 'OK'
    ...
```

Klasse RaumplanerModell

Laden, Schritt 3

```
def Lade(self, dateiname):
    ...
    except IOError:
       return 'Datei nicht auffindbar!'
    except AttributeError as ae:
       return 'AttributeError: ' + str(ae)
    except pickle.UnpicklingError as perr:
       return 'Pickle-Fehler: ' + str(perr)
```

Wichtig: Auch hier Fehler abfangen und Meldung zurück geben !

Klasse RaumplanerModell

Auswahl eines speziellen Objekts,
um es für eine Bearbeitung herauszuheben

- *Zwei Aspekte*
  - *Auswahl umsetzen*
  - *Auswahl darstellen*

Klasse RaumplanerModell

Auswahl umsetzen

- *Zwei Ansätze:*
  - *nur RaumplanerModell kennt das ausgewählte Objekt*
  - *auch das Objekt weiß, dass es ausgewählt ist*

Klasse RaumplanerModell

RaumplanerModell kennt das ausgewählte Objekt

- *Attribut von RaumplanerModell im Konstruktor initialisieren*

```
self.__ausgewaehlt = -1
```

  - *merkt sich Index des ausgewählten Objekts,*
  - *-1 kennzeichnet „keines“*

- *Methodenkopf*

```
def Waehle(self, index=None):
  '''wählt das Moebel aus; None zum Weitersetzen der Auswahl,
 -1 zum Beenden der Auswahl,
  nach dem letzten wird die Auswahl ebenfalls beendet'''
```

Klasse RaumplanerModell

auch das Objekt weiß, dass es ausgewählt ist

- *in der Methode (beispielsweise)*

```
def Waehle(self, index=None):
  ...
    self.__alleMoebel[self.__ausgewaehlt].Waehle(True)
  ...
```

Klasse RaumplanerModell

Auswahl darstellen

- *in der Methode (beispielsweise)*

```
def Waehle(self, index=None):
    ...
      self.__alleMoebel[self.__ausgewaehlt].Waehle(True)
      self.__alleMoebel[self.__ausgewaehlt].Zeige()
    ...
```

- *in Moebel Attribut und Verwendung*

```
    self.__ausgewaehlt=False
     ...
def GibFarbe(self):
    """Get-Methode fuer die Farbe"""
    if self.__ausgewaehlt: return "GRAY"
    return self.__f
```

Klasse RaumplanerModell

Bearbeitung von Objekten ermöglichen

- *Methode*

```
def BearbeiteAusgewaehltes(self, **kwargs):
    """die kwargs dx,dy,dw,f erzeugen ein Dictionary"""
    if self.__ausgewaehlt>=0:
        gewaehlt=self.GibAusgewaehltes()
        if 'dx' in kwargs:
            gewaehlt.BewegeHorizontal(int(kwargs['dx']))
        if 'dy' in kwargs:
            gewaehlt.BewegeVertikal(int(kwargs['dy']))
        ...
```