Termin 3 (Stichworte)

Rückblick und etwas nachholen

- Präsentation ***Whlg-Pxx_Was_bei_run.pdf*** auf der Seite *Schritt zur Abstraktion* unter der Überschrift Termine zum zweiten Abschnitt
- Hinweis auf ***Whlg-Pxx_Wozu__main__.pdf***
- Präsentation ***OO-Python-P02-a self als Parameter.pdf***

Schrittweise Ausführung

- Was ist überhaupt das Problem? → *Demo_so_geht_es_nicht.zip*
- wozu? → Beispiel *Demo LKW und Schiff.zip*
- die Varianten vorstellen ***OO-Python-P02f schrittweise.pdf***
  - timer und Bind
  - externe Klasse Schritte
  - Threads verwenden

Wiederholung Moebel als abstrakte Klasse

- Projekt *Raumplaner-Vererbung-abstrakt-ohne-Kapselung.zip* laden und entpacken
- Warum erzeugt der Versuch, eine Instanz von Moebel zu erzeugen, einen Fehler?
- Warum bricht das Programm nicht mit einem Fehler ab? → **`try`** – **`except`**

Vererbung mit Kapselung

- Präsentation ***Arbeitsplan-05 Kapselung bei Vererbung.pdf***
- Ansatz auch hier: Codeduplizierung ist Hinweis auf Mängel der Modellierung.
- Projekt *Raumplaner-Vererbung-abstrakt-ohne-Kapselung.zip*
  jetzt zu einer Lösung mit Kapselung bearbeiten
- Problem: Attribute nur an einer Stelle, in der Oberklasse, welche die Werte beim Aufruf des Konstruktors Oberklasse übergeben bekommt.
- Klärung: Wie können die Tochterklassen auf die Attribute zugreifen?
  ```
  # Hilfsvariable zur Vereinfachung
  b,t = self.__b, self.__t
  ```
  schlägt fehl, da die Attribute gekapselt sind!
- → Bei unserem Projekt wird eine Lösung einfach, da die einzige Methode der Tochterklassen, nämlich GibFigur(), nur lesend zugreifen muss.
  Zugriff also über die Get-Methoden!
- Verändernde Methoden befinden sich alle in Moebel und haben damit vollen Zugriff auf die Attribute.

Termin-02-11

Kohäsion

- **`Transformiere`** ausgliedern → Kohäsion ***OO-Python-P02-c Kohaesion.pdf***
  Hinweis auf Text ***OO-Python-03 Kohäsion und Kopplung.pdf***

Qualität von Software

- Präsentation ***OO-Python-P02-g Qualität von Software.pdf***

Modellierung im aktuellen Schritt

- Präsentation ***OO-Python-P03-a UML Klassendiagramme.pdf***
- Download und Arbeit mit dem Modellierungstool
  Software – Objektorientierung, erster Link
- Präsentation ***Arbeitsplan 06 Klassendiagramm.pdf***
- Projekt *Raumplaner-Vererbung-abstrakt&Kohaesion.zip*
- Projekt zum Erstellen von Klassendiagrammen herunterladen und entpacken
- Starten und für die folgende Arbeit die bereits vorhandenen Lösungen ignorieren.
- Vorhandene Klassen einfügen (noch keine Attribute und Methoden)
- Klassensymbole sinnvoll verteilen
- Beziehungen hinzufügen, erst Vererbungsbeziehungen, dann Verwendungsbeziehungen
- Abspeichern von Modelldaten, Bild ~~und Code~~;
  dazu Ergebnisse ansehen
- KlassendiagrammeApp neu starten, Klasse Stuhl Klassenkarte (Attribute und Methden?), Vergrößern zeigen
- Abspeichern von Modelldaten, Bild ~~und Code~~
- Dasselbe für Klasse Moebel (nur angefangen)