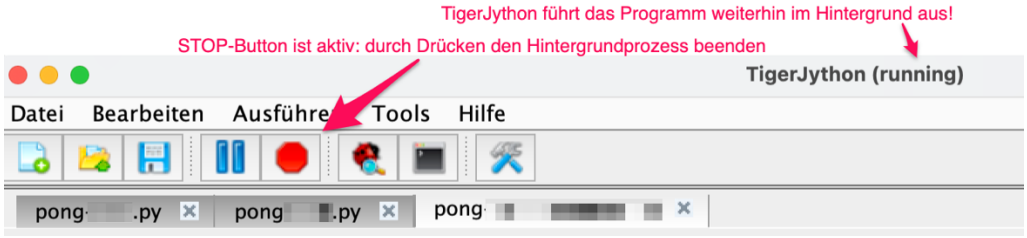
Schritt 3: Klassen in Dateien auslagern

Erwartete Endergebnisse

- Die Klasse Schlaeger ist in einer eigenen Datei abgespeichert
- Statt pong-v02.py gibt es ab sofort nur noch **eine** main.py, die für die gesamte Objektverwaltung und den Ablauf des Spiels verantwortlich ist.

WICHTIG – zur Beachtung für ein reibungsloses Arbeiten

- Sollte **nach dem Schließen des Spielfensters** in TigerJython immer noch *running* angezeigt werden und/oder der rote Stopp-Knopf aktiv sein, so unbedingt den Stopp-Button betätigen und den Prozess im Hintergrund beenden!



- Es dürfen beim Speichern von **Klassen in Dateien** beim **Dateinamen** keine Bindestriche (-), Leerzeichen oder ähnliches verwendet werden! Sonst gibt es Probleme beim import. Beispiel:

  Schlaeger-Zusatz.py

  funktioniert **nicht**, da ein Fehler auftritt beim import mit

  *from Schlaeger-Zusatz import **

- immer nur das importieren, was gebraucht wird, möglichst keinen Stern * verwenden! Beispiel:

  from gamegrid import Actor

  Hintergrund: es kann sonst zu (Run-Time)Konflikten kommen, wenn alles aus einer Klasse importiert wird und bestimmte Methoden oder Attribute schon vorhanden sind.

- sollte TigerJython abstürzen (z.B. wenn ein JGameGrid Fatal Error Fenster immer wieder erscheint), so muss unbedingt im Debugger-Modus gestartet werden. Dann lässt sich die Fehlermeldung lesen, der Fehler lokalisieren und ein Beenden über den Task-Manager ist nicht nötig.

- Weitere Fehlerquellen ausschalten: in den TigerJython Einstellungen -> Erweitert den Haken setzen bei „Direct3D-Grafik deaktivieren“. Eventuell auch den Haken setzen bei „Beans-Eigenschaften im Debugger ausschalten“
- Nach **längerer** Benutzung von TigerJython, insbesondere durch das ständige Erzeugen von GameGrid-Fenster und dessen Schließen, kann es durch Speicherüberlauf zu Fehlern kommen. Deshalb **unbedingt** in regelmäßigen Abständen nach intensivem Arbeiten TigerJython **komplett beenden** und neu starten!

- Sollte ein **Fehler** ähnlich: *„...Non-ASCII character in file...“* auftreten dann müssen alle einzelnen Klassen mit folgendem Kommentar ganz am Anfang versehen werden:

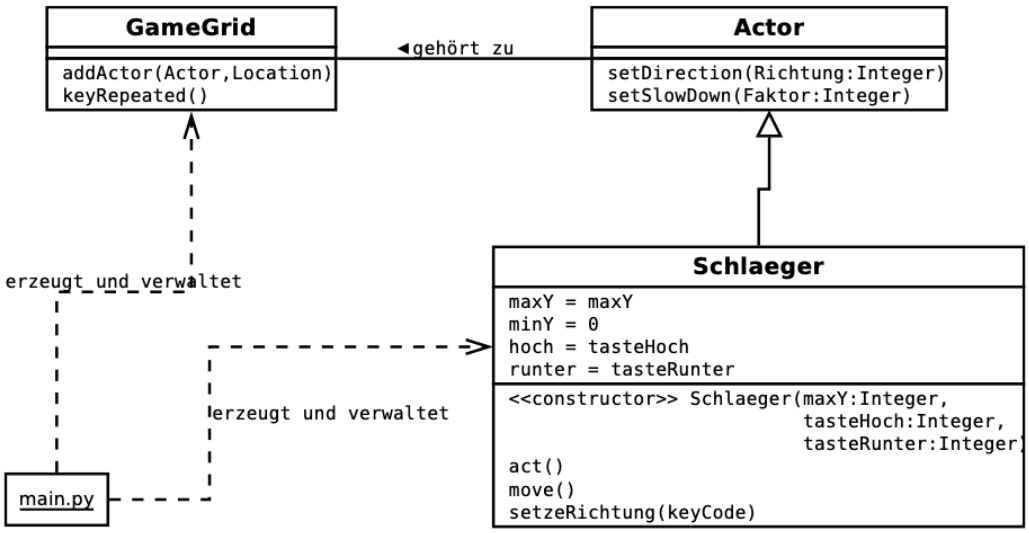
  # -- coding: utf-8 -*-*
  # utf-8 wegen Umlaute

  alternativ, wenn dies nicht funktioniert:

  # -- coding: cp1252 -*-*
  # ist der Westeuropäische Zeichensatz wegen der Umlaute

  Dies dient dazu, Fehlermeldungen bei der Darstellung von Umlauten oder „ß“ in Ausgaben oder Kommentaren zu vermeiden.

Aufgaben



1. Ergänzen Sie ihre UML-Datei aus Schritt 2 nach der o.a. Vorgabe. Hinweis: Die Datei *main.py* wird im UML als Objekt dargestellt. Die Beziehungen zu den erzeugten Objekten aus den entsprechenden Klassen sind mit Hilfe der Verbindung „Einschränkung“ modelliert. Das hat nur darstellerische Gründe, keine inhaltlichen.

2. Lagern Sie die Klasse *Schlaeger* in eine eigene Datei aus. Vergessen Sie nicht den import in main.py! Testen Sie erst auf Funktion, bevor Sie weiter fortfahren!

3. Die ausgelagerte Klasse Schlaeger soll nun selbst die Tastendrücke verarbeiten. Dazu gibt es zwei neue Attribute (vgl. UML).
   Die *keyCallback* Funktion *aus main.py* ruft jetzt nur noch die Methoden eines Schlaeger-Objektes auf*:*

   ```
   def keyCallback(keyCode):
       schlaegerLinks.setzeRichtung(keyCode)
       schlaegerRechts.setzeRichtung(keyCode)
   ```

4. Implementieren Sie die Klasse Schlaeger und main.py den Vorgaben entsprechend!