Schritt 6: Klasse Spieler und Punktezählung - Was wäre ein Spiel ohne Punktezählung?

Erwartete Endergebnisse

- Die Klassen *Spieler* und *Anzeige* sind in jeweils einer eigenen Datei abgespeichert.
- main.py erzeugt zwei Spieler-Objekte
- Eine Klasse *Anzeige* kümmert sich um die Darstellung verschiedener Spielangaben

Aufgabe

Implementieren Sie das UML-Diagramm von Schritt 6 (letzte Seite).

Hinweise

1. Es kann nur auf dem **Hintergrund**[1] des Spielfeldes gezeichnet, bzw. Text angezeigt werden!
   Deshalb benötigen wir das aktuelle GameGrid-Objekt und müssen den Aufruf in der main.py in eine Zuweisung ändern, um eine Referenz auf das GameGrid-Objekt in einer Variable spielfenster zu speichern:

   ```
   spielfenster = makeGameGrid(FENSTERBREITE, FENSTERHOEHE, 1, Color.black, False, keyRepeated = keyCallback)
   ```

   Diese **Referenz** (Objekt) *spielfenster* benötigen wir, um sie später an die Anzeige zu übergeben.

2. In der Anzeige sieht der Konstruktor entsprechend folgendermaßen aus (vgl. auch das UML-Diagramm und die Dokumentation zu GameGrid!):

   ```
   def __init__(self, spielfenster, playerLeft, playerRight):
   self.spielfenster = spielfenster # wir können nur auf dem Hintergrund des
   Fensters malen!
   self.spielfeldHintergrund = self.spielfenster.getBg()
   ...
   ```

3. In Anzeige bekommt die Methode *zeigeSpielerName* als Parameter eine *position* als String übergeben. Die Idee ist, dass damit die (grobe) Anzeigeposition des Spielers als Zeichenkette übergeben werden soll. In diesem Beispiel könnte somit „links“ bzw. „rechts“ als Postitionsangabe für den Spielernamen übergeben werden. Das Anzeigen-Objekt kümmert sich dann um die richtige Interpretation von „links“ bzw. „rechts“. Somit ist die Anzeige vollständig für die Darstellung des Spielernamen verantwortlich. Es sind hier auch anderer Möglichkeiten der Positionsangabe denkbar. Diese dürfen in Abwandlung des UML und in eigener Implementierung von Ihnen realisiert werden.

4. Für den Anfang reicht bei der Anzeige eine einfache Spielstandsausgabe auf der Textebene (Ausgabe in Konsole):

   ```
   def zeigeSpielstand(self):
         print "Spielstand:",self.playerLeft.getPunkte(),":",self.playerRight.getPunkte()
   ```

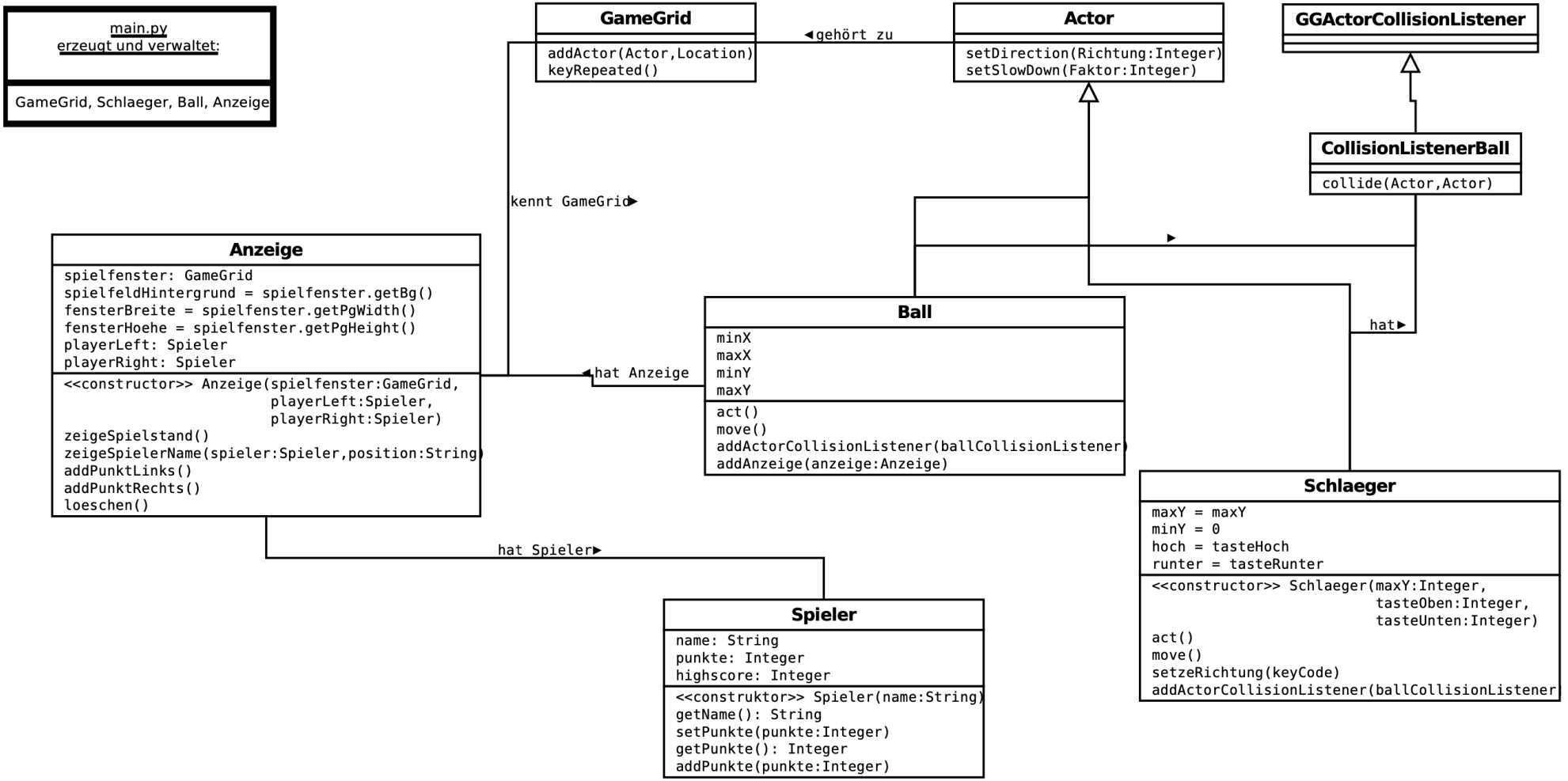
5. Der Ball überprüft beim Abprallen links oder rechts, ob ein Tor gefallen ist und ruft die entsprechende Methode

   ```
   self.anzeige.addPunktRechts()
   ```
   bzw.
   ```
   self.anzeige.addPunktLinks()
   ```

   bei seinem Anzeige-Objekt *anzeige* auf.

[1] In der GameGrid-Dokumentation als **Klasse GGBackground** bezeichnet.

UML für Schritt06



oppermann@siemens-gymnasium-berlin.de