**Modul:** **Forschungsprojekt Anwendung**
**Projekt:** **Persistenz der Scala SCROLL Implementierung**
**dieses Dokument:** **Klassendiagramm - Ansätze / Brainstorming**
**Student:** **Thorsten Seyschab <uni@todde.tv>**

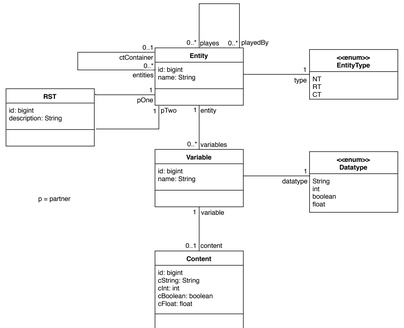
spezielle Datentypen:
bigint = int in Java, bigint in DB
text = String in Java, TEXT in DB

Verzicht auf Getter und Setter
Verzicht auf Sichtbarkeiten
Verzicht auf Methoden
Verzicht auf gerichtete Assoziationen, nur ungerichtete Assoziationen (Richtung indirekt gegeben durch Multiplizitäten)

Multiplizitäten stehen immer top ober left zu einer Assoziation
Variablen Namen der Assoziation stehen immer bottom oder right zu der Assoziation

## Ansatz 1:

NT, RT und CT möglichst gleich betrachten, um eine Vereinfachung zu realisieren.

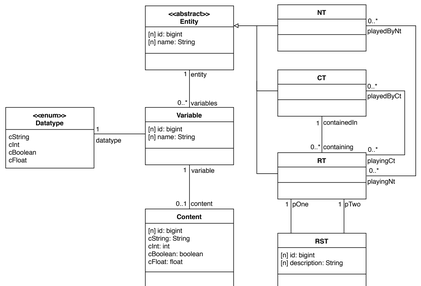



## Ansatz 2:

NT, RT und CT unter einer abstrakten Klasse oder einem Interface möglichst individuell betrachten.
Ermöglicht bessere Kontrolle über die Wohlgeformtheitsregeln.

r = role
c = Content

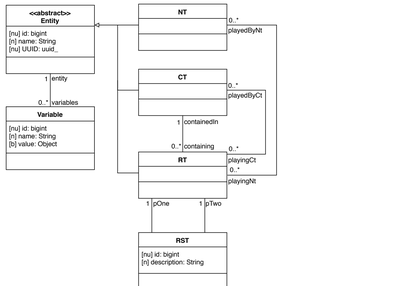
[n] = NOT NULL auf DB Ebene




## Ansatz 3:

NT, RT und CT unter einer abstrakten Klasse oder einem Interface möglichst individuell betrachten.
Ermöglicht bessere Kontrolle über die Wohlgeformtheitsregeln.

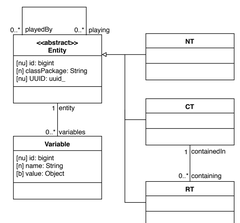
Auf Datenbank Ebene:
[n] = NOT NULL
[b] = als Blob gespeichert
[u] = UNIQUE




## Ansatz 4:

NT, RT und CT unter einer abstrakten Klasse oder einem Interface möglichst individuell betrachten.
Ermöglicht bessere Kontrolle über die Wohlgeformtheitsregeln.

Auf Datenbank Ebene:
[n] = NOT NULL
[b] = als Blob gespeichert
[u] = UNIQUE




## Ansatz 5:

NT, RT und CT unter einer abstrakten Klasse möglichst individuell betrachten.
Ermöglicht bessere Kontrolle über die Wohlgeformtheitsregeln.

Auf Datenbank Ebene:
[n] = NOT NULL
[b] = als Blob gespeichert
[u] = UNIQUE