Identificazione delle associazioni

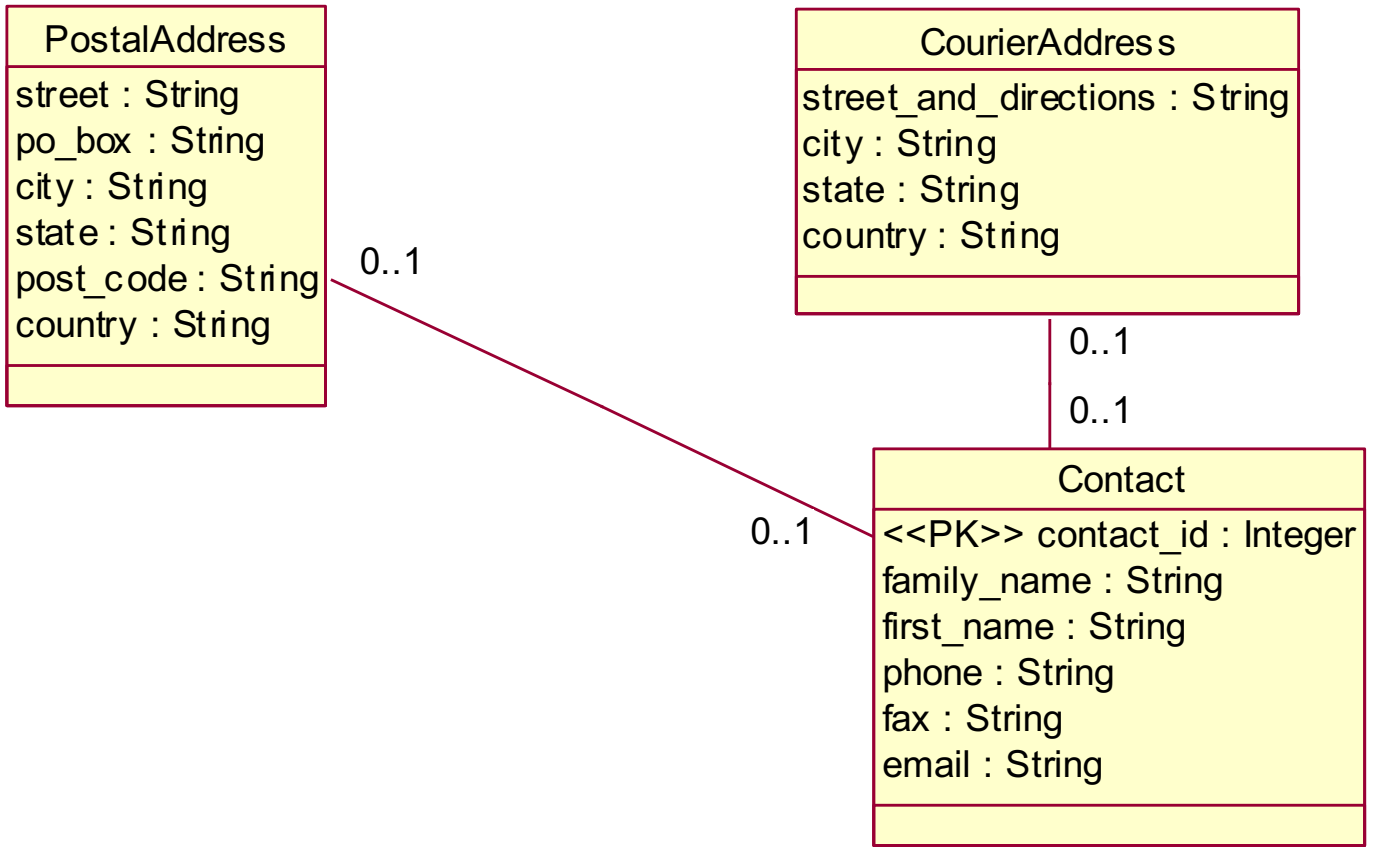
- Alcuni *attributi* identificati con le classi rappresentano associazioni (ogni attributo di tipo non primitivo dovrebbe essere modellato come un'associazione alla classe che rappresenta quel tipo di dato)
- Ogni *associazione ternaria* dovrebbe essere rimpiazzata con un *ciclo di associazioni binarie*, per evitare problemi di interpretazione
- Nei cicli di associazioni almeno un'associazione potrebbe essere eliminata e gestita come *associazione derivata*, anche se per problemi di efficienza spesso si introducono associazioni ridondanti

Specifica delle associazioni

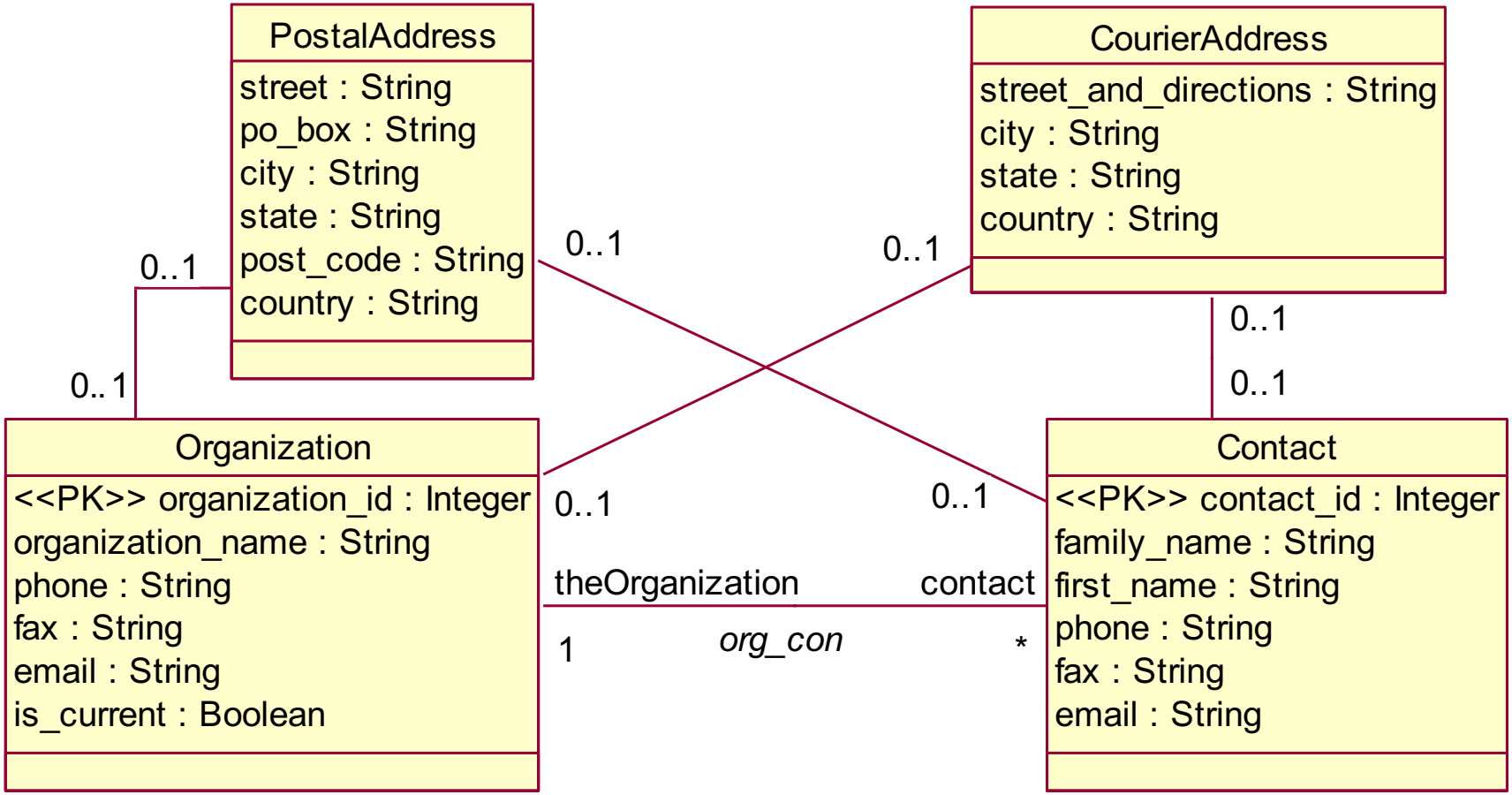
- Per *assegnare nomi alle associazioni* adottare la stessa convenzione usata per gli attributi (le parole devono essere scritte in carattere minuscolo, separate da un carattere di *underscore*)
- Assegnare nomi di ruolo (*rolename*) alle estremità dell'associazione (i *rolename* diventano i nomi degli attributi nella classe all'estremità opposta dell'associazione)
- Determinare la *molteplicità* delle associazioni (ad entrambe le estremità)

Example C.3 – Contact Management

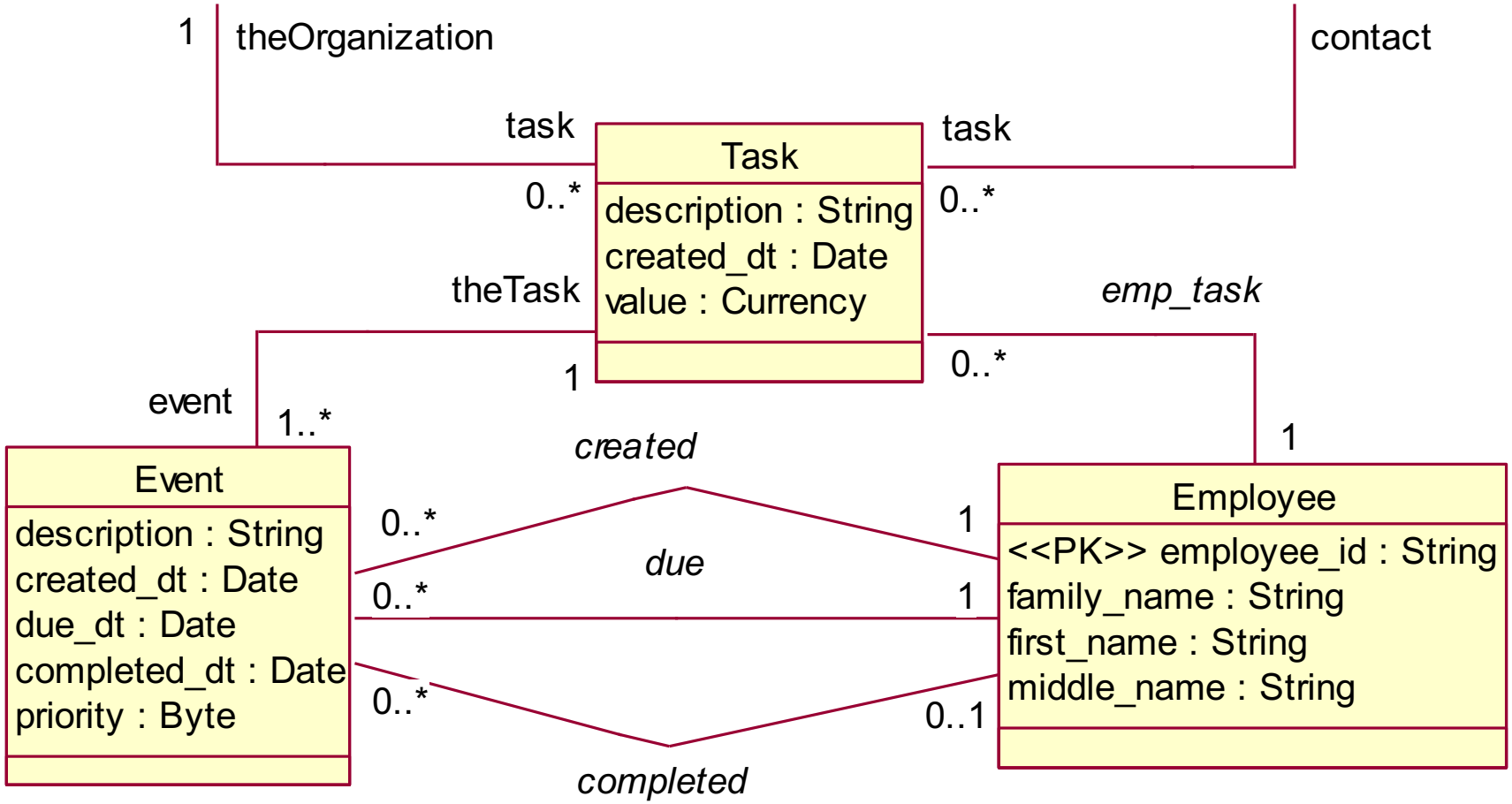
- Refer to Examples C.1 and C.2 - specify associations
- Consider, for example, the requirement:
  - The system allows producing various reports on our contacts based on postal and courier addresses



Example C.3 – Contact Management (solution – 1)



Example C.3 – Contact Management (solution – 2)



Aggregazione

- Rappresenta una relazione di tipo "*whole-part*" (contenimento) tra una classe composta (*superset class*) e l'insieme di una o più classi componenti (*subset classes*)
- Può assumere quattro differenti *significati*:
  - **ExclusiveOwns** (e.g. `Book` *`has`* `Chapter`, or `Chapter` *`is part of a`* `Book`)
    - *Existence-dependency*
    - *Transitivity*
    - *Asymmetricity*
    - *Fixed property*
  - **Owns** (e.g. `Car has Tire`)
    - No fixed property
  - **Has** (e.g. `Division has Department`)
    - No existence dependency
    - No fixed property
  - **Member** (e.g. `Meeting has Chairperson`)
    - No special properties except membership

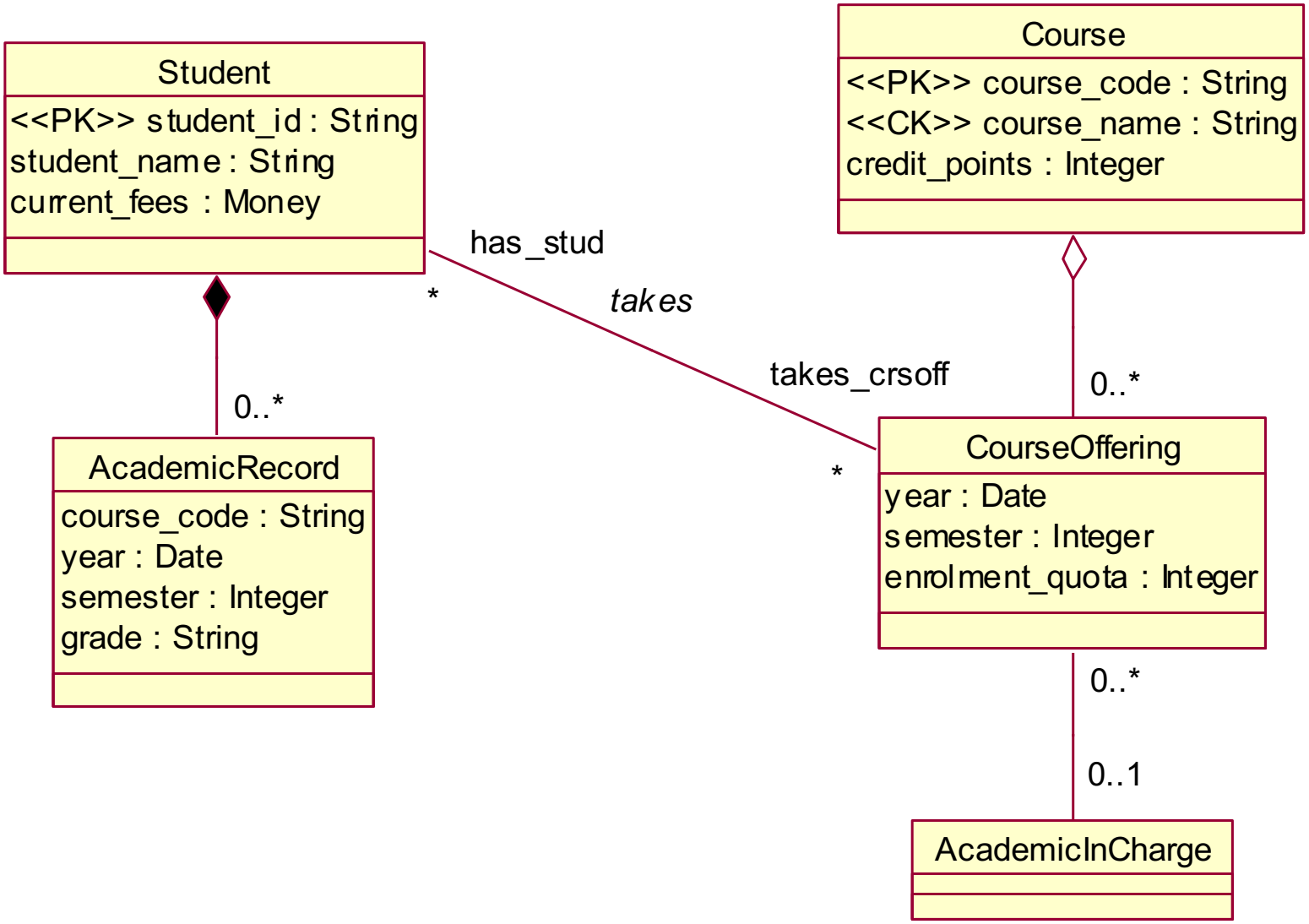
Specifica di aggregazione in UML

- **Aggregation**
  - *By-reference* semantics
  - *Hollow* diamond (◇)
  - Corresponds to *Has* and *Member* aggregations
- **Composition**
  - *By-value* semantics
  - *Solid* diamond (◆)
  - Corresponds to *ExclusiveOwns* and *Owns* aggregations

Example A.3 – University Enrolment

- Refer to Examples A.1 and A.2
- Consider the following additional requirements:
  - The student's academic record to be available on demand
  - The record to include information about the student's grades in each course that the student enrolled in (and has not withdrawn without penalty)
  - Each course has one academic in charge of a course, but additional academics may also teach in it
    - There may be a different academic in charge of a course each semester
    - There may be different academics for each course each semester

Example A.3 – University Enrolment (solution)



Ereditarietà (generalizzazione)

- Usata per rappresentare la condivisione di attributi ed operazioni tra classi
- Le caratteristiche comuni sono modellate in una classe più *generica* (**superclasse**), che viene *specializzata* nell'insieme di **sottoclassi**
- Una sottoclasse *eredita* attributi ed operazioni della superclasse
- Caratteristiche:
  - Sostituibilità: un oggetto della sottoclasse è un valore legale per una variabile avente come tipo la superclasse (es. una variabile di tipo `Frutta` può avere un oggetto di tipo `Mela` come suo valore)
  - Polimorfismo: la stessa operazione può avere differenti implementazioni nelle sottoclassi

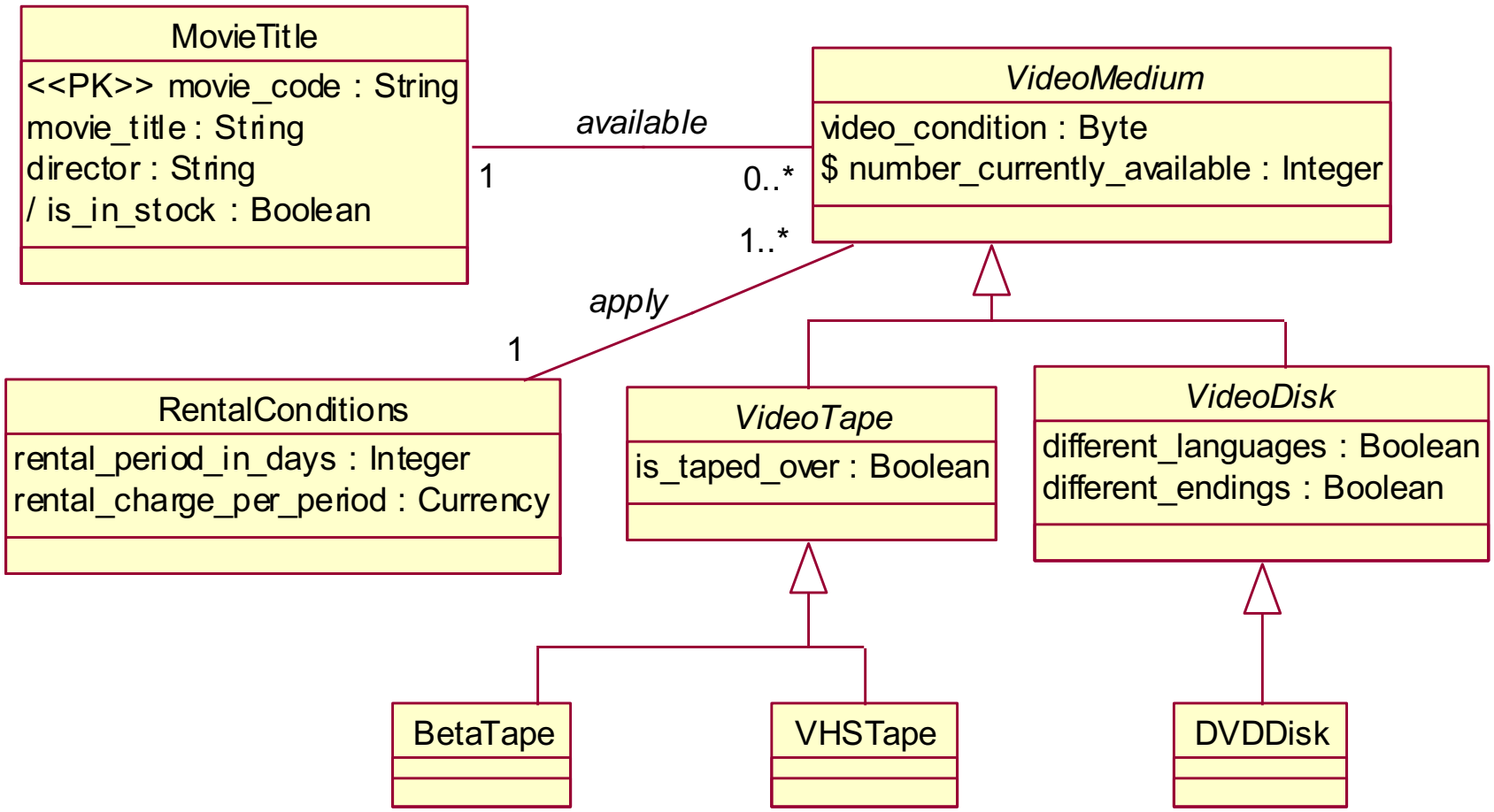
Specifica di ereditarietà in UML

- Rappresenta relazioni di tipo:
  - "can-be"
    - Es. `Student` can be a `TeachingAssistant`
  - "is-a-kind-of"
    - Es. `TeachingAssistant` is a kind of `Student`
- Supporto ad ereditarietà multipla
  - Es. `TeachingAssistant` is also a kind of `Teacher`
- Viene rappresentata in UML con una linea, che collega la sottoclasse con la superclasse, avente una freccia diretta verso la superclasse

Example B.3 – Video Store

- Refer to Examples B.1 and B.2
- The classes identified in Example 4.5 imply a generalization hierarchy rooted at the class `VideoMedium`
- Extend the model to include relationships between classes, and specify generalization relationships
- Assume that the Video Store needs to know if a `VideoTape` is a brand new tape or it was already taped over (this can be captured by an attribute `is_taped_over`)
- Assume also that the storage capacity of a `VideoDisk` allows holding multiple versions of the same movie, each in a different language or with different endings

Example B.3 – Video Store (solution)



Object Diagram

- Rappresentazione grafica di istanze di classi
- Usati per
  - modellare relazioni complesse tra classi (a scopo esemplificativo)
  - illustrare le modifiche ai singoli oggetti durante l'evoluzione del sistema
  - Illustrare la collaborazione tra oggetti durante l'evoluzione del sistema

Example A.4 – University Enrolment

- Show an object diagram with few objects representing the classes in Example A.3