33. Ereditarietà

Corso di Informatica

Corso di Laurea in Matematica (D.M. 270/04) - A.A. 2020/2021
Angelo Cardellicchio
angelo.cardellicchio@uniba.it
GG/MM/AAAA

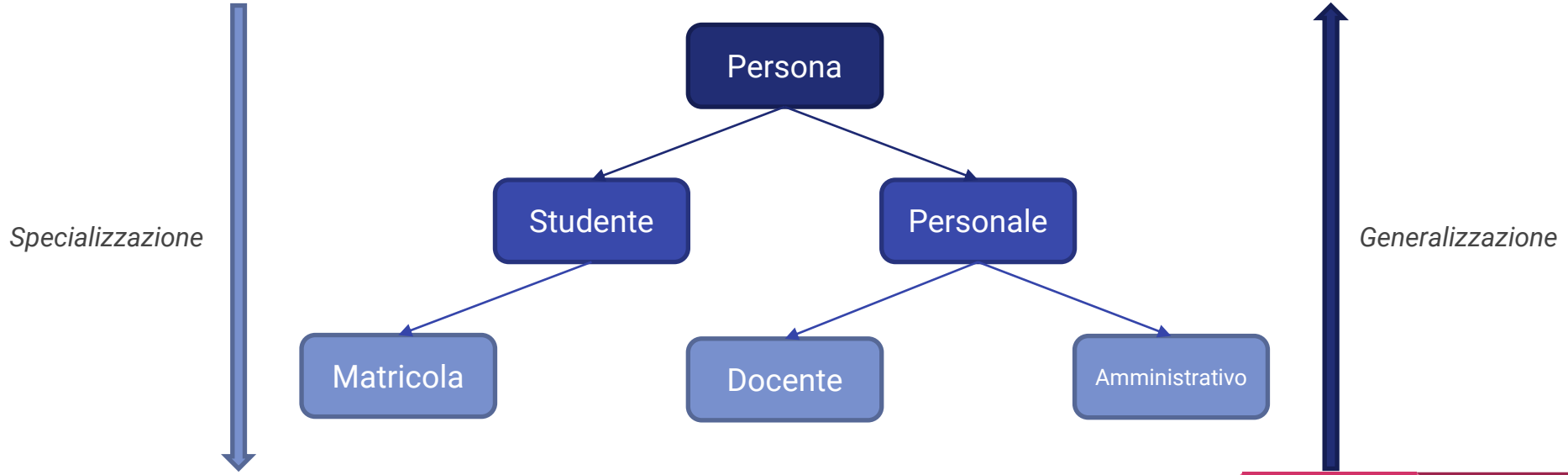
Outline

- Generalizzazione e specializzazione
- La regola **is-a**
- Definire le classi derivate

Generalizzazione e specializzazione (1)

- Il concetto di ereditarietà permette di creare una **gerarchia di classi**
- La classe più in alto nella gerarchia è quella a livello più alto di astrazione (ovvero, la più generica)
- Man mano che si scende nella gerarchia, si incontrano classi via via sempre più **specializzate**
- Facciamo un esempio usando una gerarchia di persone

Generalizzazione e specializzazione (2)



La regola is-a

- La regola **is-a** afferma che **ogni classe derivata è un tipo particolare della classe base**
  - *In altri termini, ne è sì una specializzazione, ma è anche dello stesso tipo della classe base!*
- Tornando alla nostra gerarchia:
  - *Una* `Matricola` *è uno* `Studente`*, e quindi è una* `Persona`
  - *Un* `Docente` *fa parte del* `Personale`*, e quindi è una* `Persona`
  - *Un* `Amministrativo` *fa parte del* `Personale`*, e quindi è una* `Persona`
- Le implicazioni sono importanti: infatti, tutte le classi derivate **hanno a disposizione membri e funzioni delle classi da cui deriva**

Definire le classi derivate

- Proviamo a definire le classi **`Studente`** e **`Personale`**
- Per farlo, seguiamo questa checklist:
  - *Ridefiniamo gli attributi privati come* **`protected`**
  - *Creiamo le definizioni delle nuove classi, assicurandoci che discendano da* **`Persona`** *usando al sintassi:*

```
class Persone::Studente : public Persona
```

  - *Andiamo ad implementare le nuove classi, avendo l'accortezza di richiamare, all'interno dei diversi costruttori, l'opportuno costruttore parametrizzato della classe base.*

Domande?

42