Azzolini Riccardo 2019-09-26

Ereditarietà

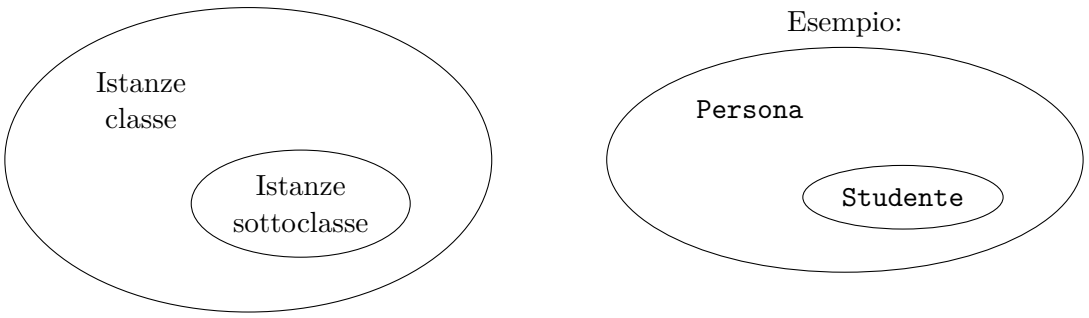
1 Ereditarietà

Gli oggetti di una classe possono essere anche oggetti di un'altra classe. Ad esempio, gli oggetti di classe `Studente` sono casi particolari degli oggetti di classe `Persona`. Allora:

- `Studente` si dice **sottoclasse** di `Persona`;
- `Persona` si dice **superclasse** di `Studente`.

In generale, la sottoclasse **eredita** tutte le proprietà della superclasse, e può inoltre averne delle altre: per questo, essa è una **specializzazione** della superclasse. Inoltre, la sottoclasse può anche **ridefinire** alcune proprietà della superclasse.

Concettualmente, la relazione tra le istanze della superclasse e quelle della sottoclasse corrisponde alla relazione tra insieme e sottoinsieme:



In alternativa, si può considerare una relazione di tipo "essere": *ogni istanza della sottoclasse è un'istanza della superclasse* (ad esempio, ogni `Studente` è una `Persona`), ma non viceversa (non è vero che ogni `Persona` è uno `Studente`).

2 Vantaggi e svantaggi

L'ereditarietà è un meccanismo di programmazione molto potente.

- Permette di non dover riscrivere codice, favorendo quindi
  - il riuso del codice;

  - la riusabilità (cioè la scrittura di codice che potrà poi essere riutilizzato);
  - l'estendibilità (perché si possono creare e modificare sottoclassi senza bisogno di apportare modifiche alla superclasse);
  - la modificabilità (perché non esistono più copie dello stesso codice).
- Aiuta a organizzare i componenti software, mostrando esplicitamente le relazioni tra di loro, e quindi favorendo la comprensibilità.

Tuttavia, proprio per la sua potenza, l'ereditarietà deve essere usata con cautela, altrimenti si rischia di commettere errori che si possono manifestare in fase di compilazione o, peggio, di esecuzione.