

lunedì 24 febbraio 2025 11:14

# Link utili Esame

venerdì 21 febbraio 2025 15:20

<https://bitbucket.org/mclab/workspace/projects/BD2>

Soluzioni esercitazioni di Analisi, SQL e alcuni progetti

# Introduzione sull'UML

lunedì 3 marzo 2025 14:27

**UML** (Unified Modeling Language) è linguaggio di modellazione basato su modelli **orientati a oggetti (Object-Oriented)**

L'UML definisce **14 tipi di diagrammi** per modellare un'applicazione sotto prospettive diverse.

I principali tipi di diagrammi sono:

- **Diagrammi strutturali:**
  - Diagramma delle **classi e oggetti** (class and object diagram)
- **Diagrammi comportamentali:**
  - Diagramma degli **use case** (use case diagram)
  - Diagramma degli **stati e transizioni** (state/transition diagram)
  - **Interaction** (Sequence e Collaboration diagram)
  - **Activity diagram**
- **Diagrammi architettonici:**
  - **Component** diagram
  - **Deployment** diagram

Il corso si concentra sugli aspetti base e sui diagrammi più importanti, e verranno usati solo questi diagrammi (e solo le caratteristiche più importanti):

- **Diagrammi strutturali:**
  - Diagramma delle **classi e oggetti** (class and object diagram)
- **Diagrammi comportamentali:**
  - Diagramma degli **use case** (use case diagram)
  - Diagramma degli **stati e transizioni** (state/transition diagram)

## Obiettivi

Per **progettare applicazioni** di base di dati reali e di grandi dimensioni è necessario spendere gran parte del tempo per **comprendere** i dati di interesse e le loro **interrelazioni**. Non esiste una soluzione unica o una ricetta base da applicare ma bisogna fare delle **scelte ragionate**. È necessario quindi:

- Ragionare logicamente
- Considerare le conseguenze delle nostre scelte
- Valutare le alternative
- Scomporre i problemi complessi in sottoproblemi

## Esempio

Se dovessimo creare un'applicazione che permetta di mantenere informazioni su un insieme di contatti (telefonici e mail) è facile definire uno schema di relazione per la base dei dati. I requisiti richiedono di mantere il nome, cognome, numero e indirizzo dei contatti e permettere all'utente di effettuare delle azioni per la creazione/modifica/cancellazione/ricerca di un contatto o aggiungerli/rimuoverli da un gruppo.

Se invece si vuole sviluppare un sistema che permette ad una banca di gestire i conti correnti dei clienti, i loro investimenti e la propria rete di promotori finanziari, allora la creazione di uno **schema di relazione** risulta **complesso**.

Il sistema deve **tenere traccia** di tutti gli acquisti e vendite di azioni, obbligazioni, ecc. effettuati dai clienti, e poter calcolare la valorizzazione del loro portafoglio. Inoltre deve assistere i promotori finanziari nella scelta degli strumenti finanziari più adeguati da proporre ai clienti, e permettere ai responsabili di agenzia di controllare la professionalità dei promotori.

È impossibile scrivere direttamente lo **schema relazionale** e il **programma** per l'applicazione.

**Progetto software complesso:**

- pool di Ingegneri del SW, progettisti, programmati: 1.5 anni (spannometricamente...)
- tempo per capire il problema: 6 mesi (33%)
- tempo per la progettazione: 9 mesi (50%)
- tempo per la realizzazione: 3 mesi (solo il 17%)
- più: tempo per test&verifica, ecc

## Contesto Organizzativo

Bisogna essere in grado di distinguere le **figure o i ruoli** coinvolti nelle varie fasi del **ciclo di vita del software (progettazione, sviluppo ed esercizio)**:

- **Committente:** Chi finanzia e commissiona il progetto, definendo requisiti e obiettivi
- **Esperti del dominio:** Specialisti del settore di applicazione del software, forniscono conoscenze e specifiche
- **Analisti:** Raccolgono e analizzano i requisiti, traducendoli in specifiche tecniche e funzionali
- **Progettisti:** Definiscono l'architettura e la struttura del software
- **Programmati:** Scrivono il codice sulla base delle specifiche fornite
- **Utenti finali:** Coloro che utilizzeranno il software e ne valuteranno l'**usabilità**
- **Manutentori:** Si occupano di aggiornamenti, correzione bug e miglioramenti post-rilascio

## Esempio

Il Comune di XYZ vuole automatizzare la gestione delle info relative alle contravvenzioni elevate sul suo territorio, in particolare vuole dotare ogni vigile di una app per smartphone che gli consente di comunicare al sistema informatico il veicolo, il luogo e la natura della contravvenzione.

Il sistema informatico notificherà, tramite posta ordinaria, la contravvenzione al cittadino interessato.

Il Comune bandisce gare per la progettazione, realizzazione e manutenzione del sistema, che vengono vinte da ITSolutions s.p.a., Develop s.r.l. e Ops s.r.l.

Quali sono i ruoli coinvolti in questa progettazione del software?

- **Committente:** Comune di XYZ
- **Esperti del dominio:** Funzionario del Comune o altro professionista designato, esperto del Codice della Strada
- **Analisti:** Personale ITSolutions s.p.a.
- **Progettisti:** Personale ITSolutions s.p.a.
- **Programmati:** Personale Develop s.r.l.
- **Utenti finali:** Polizia locale XYZ
- **Manutentori:** Personale Ops s.r.l.

## Ciclo di Vita del Software

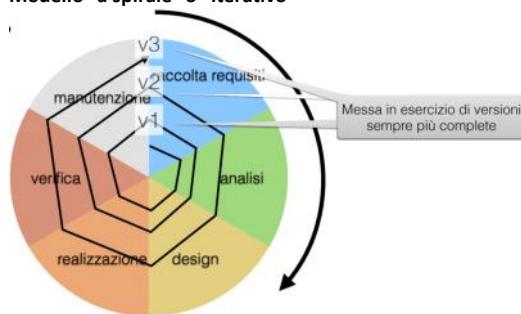
### Macro-fasi Principali

1. **Studio di fattibilità**
  - Comprendere i **requisiti di alto livello**
  - Valutare **costi e benefici**
  - Pianificare le **attività** e le **risorse** del progetto
  - Individuare l'**ambiente di programmazione** (hardware/software)
2. **Raccolta dei requisiti**
  - Raccolta dei **requisiti** presso i diversi ruoli
  - Stesura e sintesi iniziali
  - **Raffinamento** dei requisiti
3. **Analisi concettuale dei requisiti**
  - **Obiettivo:** produrre lo **schema concettuale** dell'applicazione, che definisce in dettaglio **cosa** l'applicazione dovrà realizzare, indipendentemente dal **come**.
  - Lo **schema concettuale**:
    - Modella i **dati di interesse**, le loro **articolazioni, interellazioni ed evoluzioni possibili**
    - Specifica i **servizi computazionali** che l'applicazione dovrà offrire ai diversi utenti
    - Lo schema concettuale è un modello logico-matematico dell'applicazione, e sarà la base da cui partire per le successive attività di progettazione
4. **Progetto (design) dell'applicazione**
  - **Obiettivo:** specificare **come** l'applicazione dovrà realizzare le sue funzioni (ora che il **cosa** è stato definito nel punto 3)
  - Scegliere il mix tecnologico ottimale per l'applicazione
  - Definire l'**architettura**
  - Definire le strutture di **rappresentazione dei dati** (in memoria centrale e di massa)
5. **Realizzazione (implementazione) dell'applicazione**
  - Scrivere il **codice**
  - Scrivere la **documentazione**
6. **Integrazione dei componenti e verifica dell'applicazione**
  - Le diverse componenti dell'applicazione, sviluppate separatamente, **vengono integrate**
  - Si valuta se l'applicazione svolge correttamente, completamente ed efficientemente i suoi compiti
7. **Messa in esercizio dell'applicazione**
  - L'applicazione viene **messa in esercizio** ed inizia a funzionare
8. **Manutenzione dell'applicazione**
  - L'applicazione viene **monitorata** durante l'esercizio
  - Correzioni ed aggiornamenti vengono prodotti ove e quando necessario

Al termine di ogni fase, se necessario, si può **tornare indietro**

### Modelli

- **Modello "a cascata" ("waterfall model")**
  - Ogni attività inizia quando termina la precedente
  - Ha un interesse esclusivamente didattico
- **Modello "a spirale" o "iterativo"**



# Classi, Oggetti e Link

lunedì 3 marzo 2025 16:36

Nella fase di **analisi** (punto 3 del ciclo di vita del software) ci si **concentra di più sulle classi** che sugli **oggetti**. Questi ultimi servono principalmente per descrivere elem. singoli particolarmente significativi oppure per descrivere esempi.

## Oggetti in UML

In UML un oggetto modella un elem. del dominio di analisi che:

- ha "vita propria"
- è **identificato univocamente** dall'identificatore di oggetto
- è **istanza di una classe**
- in alcuni casi di più classi, ma comunque tra le classi di cui un oggetto è istanza, esiste sempre la classe più specifica

div_comm: Libro

### Esempio

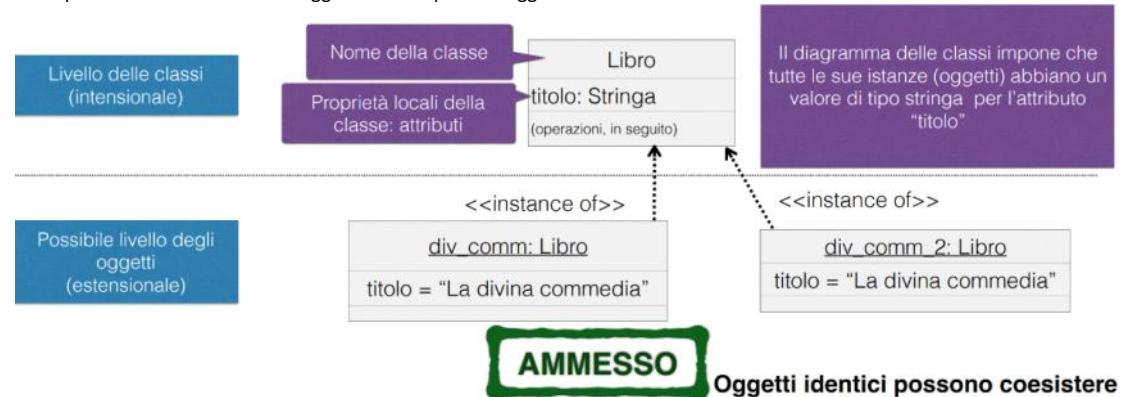
- **div\_comm** è l'ID di oggetto
- **Libro** è la classe più specifica di cui l'oggetto è istanza
- L'oggetto si segna con la **sottolineatura**

## Classi

Una **classe** modella un insieme di **oggetti omogenei** (le istanze della classe) ai quali sono associate **proprietà statiche (attributi)** e **dinamiche (operazioni)**. Ogni classe è descritta da:

- **Un nome**
- **Un insieme di proprietà statiche o dinamiche** (astrazioni delle proprietà degli oggetti che sono istanze delle classi)

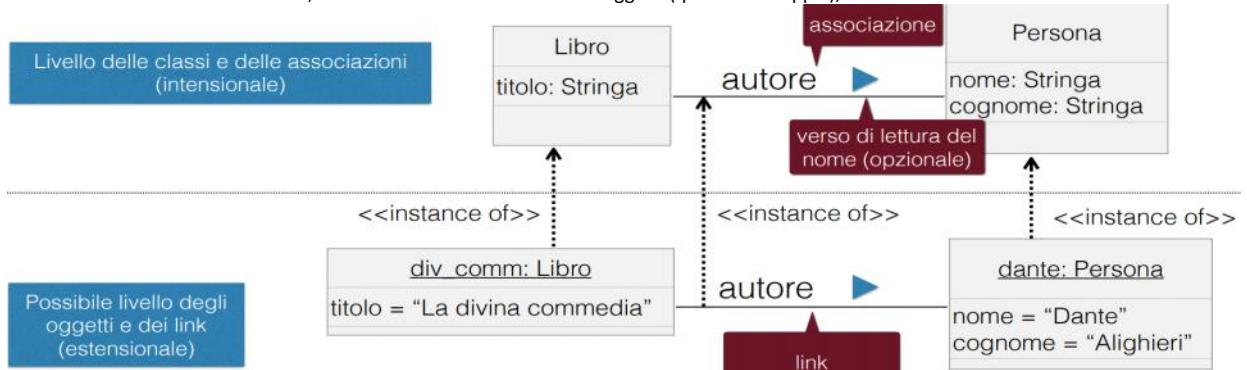
Il diagramma delle classi permette la **coesistenza** di oggetti identici poiché l'oggetto è identificato dal suo ID univoco.



## Link e Associazioni

Una **associazione** modella la possibilità che gli oggetti di più classi abbiano **dei legami**. Le **istanze di associazione** si chiamano **link**.

Es. Se A è una associazione tra le classi C1 e C2, una istanza di A è un link tra due oggetti (quindi una coppia), uno della classe C1 e l'altro della classe C2



I link quindi sono **istanze delle associazioni**. Quindi se l'associazione è un **insieme di coppie** (Es. Libro - Persona), il link è una di quelle coppie (es. div\_comm - Dante). Però al contrario degli oggetti, i **link non hanno ID univoci**, ma sono **implicitamente identificati** dalla coppia di oggetti che rappresenta. Quindi non è ammesso (più che ammesso, non ha concettualmente senso) usare più di un link per collegare due istanze

Tra le stesse classi possono essere definite **più associazioni**, che modellano **legami di natura diversa**



Vediamo come Libro e Persona hanno due associazioni, però nel livello degli oggetti **non è obbligatorio** usare entrambi i link per definire l'autore e l'editore (si può anche usare un solo link ad esempio per l'autore)

## Association Class

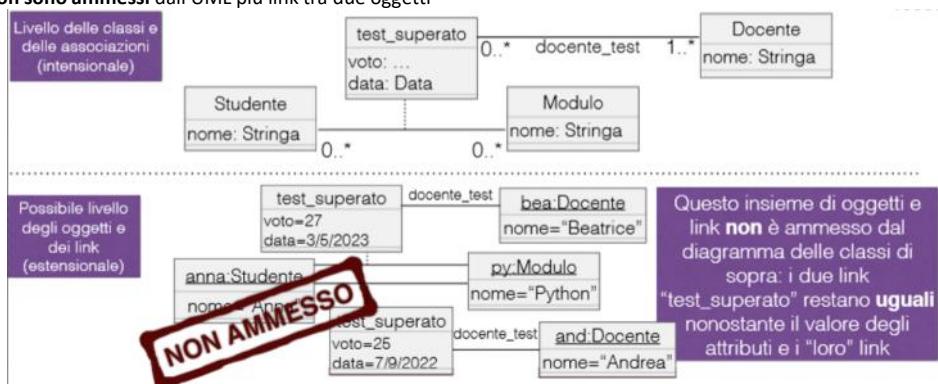
Se non possiamo aggiungere un attributo intrinseco ad una coppia di classi associate allora è probabile che quell'attributo sia una proprietà dell'associazione stessa. Si può quindi definire una **associazione con attributi (association class)** a cui diamo gli attributi necessari intrinseci all'associazione.

Ad es. se un sistema deve gestire gli esiti (voto e data) dei moduli superati dagli studenti, allora **voto** non è proprietà locale né di Studente, né di Modulo, ma è una proprietà del **legame** dell'oggetto Studente e Modulo.

E poiché l'associazione con attributi è essa stessa una **classe** possiamo collegarla ad altre classi e può essere **superclasse/sottoclasse** di un'altra classe.

Ad esempio il test\_superato si può collegare col Docente con cui si è effettuato il test

Ovviamente, come prima, **non sono ammessi** dall'UML più link tra due oggetti

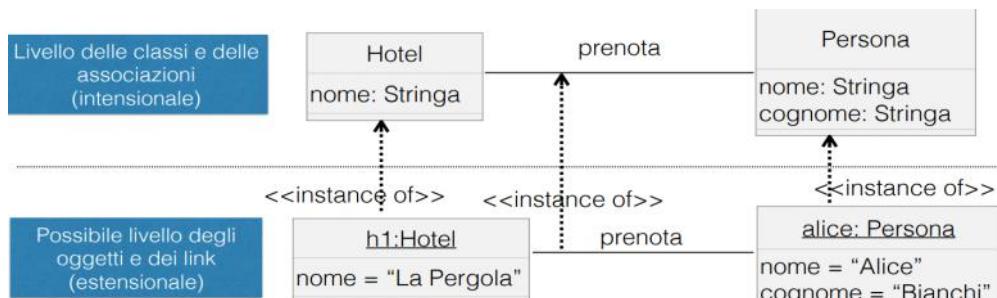


L'associazione con attributi inizia con la **lettera minuscola** ed ha una **linea tratteggiata** che la lega con l'associazione tra le classi.

## Esempio

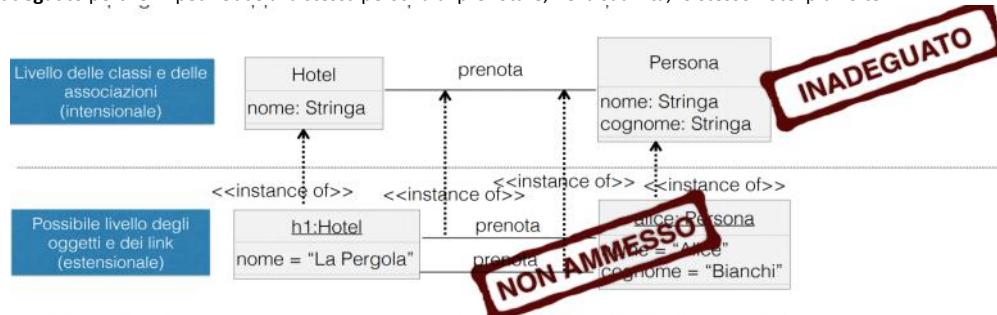
Si vuole progettare un'applicazione che permette ai clienti di prenotare l'hotel via web.

### Esempio 1 delle classi



Però se provassimo a rappresentare una seconda prenotazione di "alice" presso "h1" non potremmo perché non è ammesso.

Quindi il diagramma è **inadeguato** perché impedirebbe alla stessa persona di prenotare, nella sua vita, lo stesso hotel più volte.



Possiamo risolvere inserendo la classe **Prenotazione** permettendo ad ogni singola prenotazione (oggetto) di avere "vita propria"

## ADEGUATO



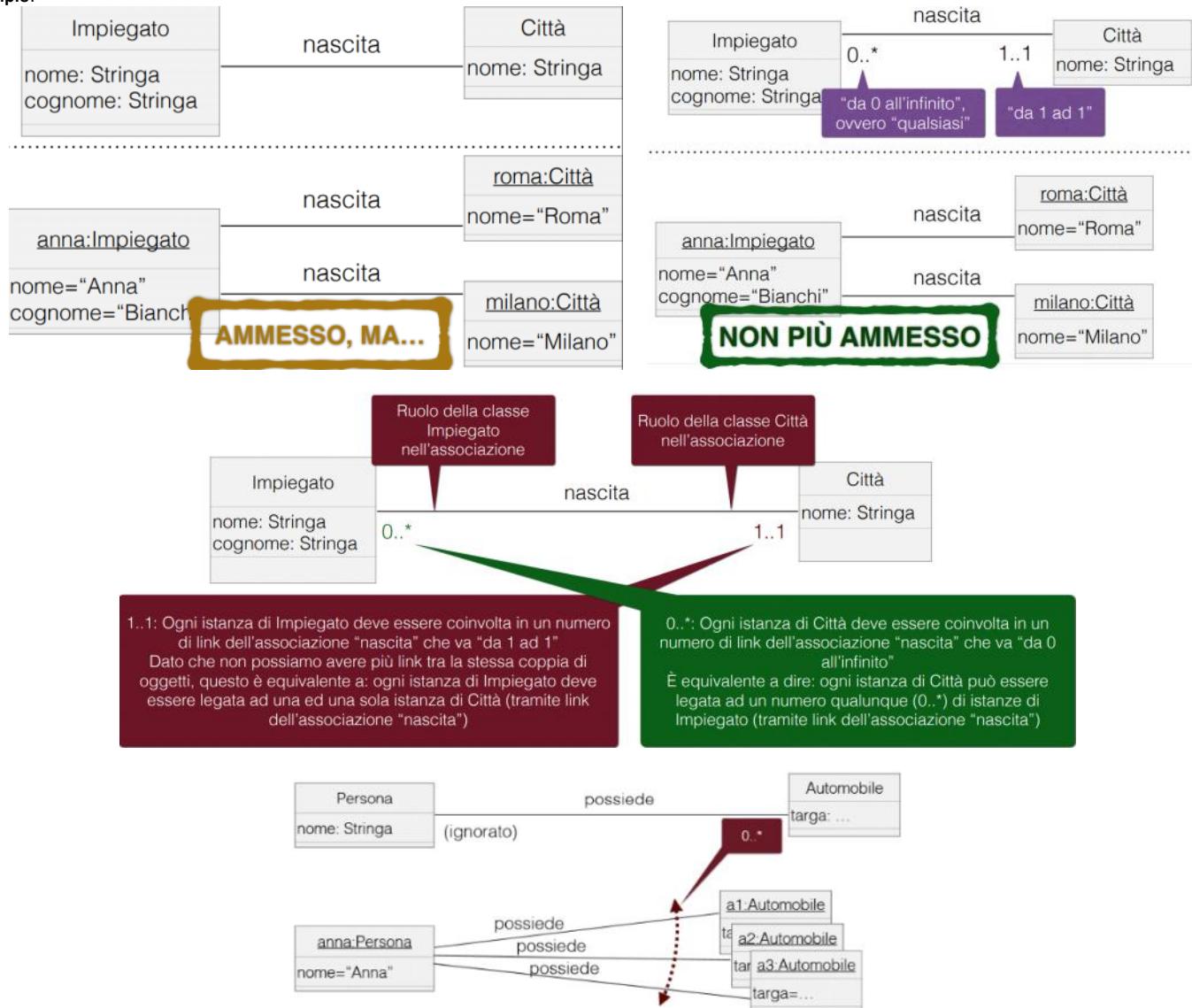
# Vincoli di molteplicità

lunedì 3 marzo 2025 17:46

Per soddisfare i **vincoli** del mondo reale bisogna impostare dei **vincoli di molteplicità** sulle associazioni e sugli attributi.

UML permette di definire sul diagramma delle classi dei **vincoli di integrità**, che impongono ulteriori **restrizioni** sui livelli estensionali ammessi.

**Esempio:**



Ad esempio sull'esercizio sull'Hotel ([Es. Prenotazione Hotel](#)) avremmo questi vincoli di molteplicità sui ruoli

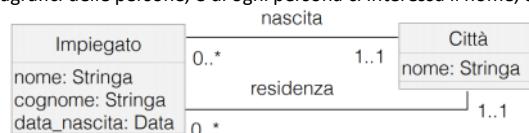


**DA IMPARARE A MEMORIA** (perché il prof senno ti mela):

- "A quanti link **hotel\_prenotato** ogni istanza di **hotel** può partecipare?"
- "Ogni istanza della classe **Hotel** a quanti link **hotel\_prenotato** può partecipare?"
- "Un'istanza di **Hotel** può partecipare da 0 a \* link dell'associazione **hotel\_prenotato**" oppure
- "Ogni istanza di **Hotel** deve essere coinvolta in un numero di link dell'associazione **hotel\_prenotato** che va da 0 a \*" e allo stesso modo
- "Un'istanza di **Prenotazione** può partecipare ad esattamente 1 link dell'associazione **hotel\_prenotato**" oppure
- "Ogni istanza di **Prenotazione** deve essere coinvolta in un numero di link dell'associazione **hotel\_prenotato** che va da 1 a 1"

## Esercizio

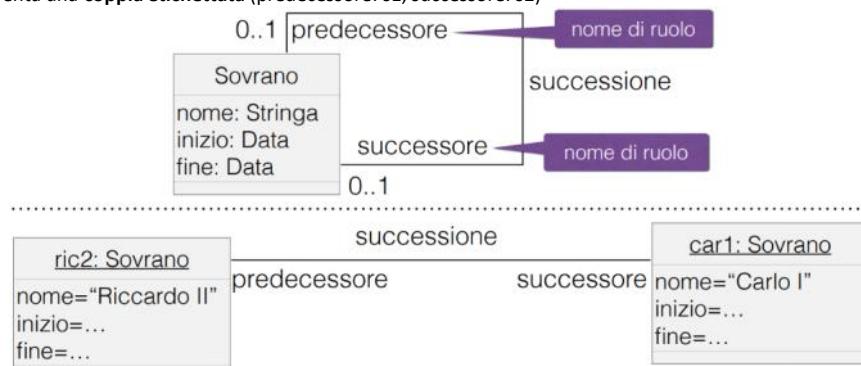
Si vuole progettare un sistema che gestisca i dati anagrafici delle persone, e di ogni persona ci interessa il nome, cognome, data di nascita e città di nascita e residenza



## Associazioni con la classe stessa

Se volessimo modellare i sovrani di un regno per cui ci interessa il nome, il periodo in cui ha regnato e il suo **predecessore**, possiamo fare in modo che la classe sovrano stessa si associa con se stessa per indicare sia un **successore** e sia un **predecessore**.

Una istanza dell'associazione diventa una **coppia etichettata** (predecessore: s1, successore: s2)



Quando si usa un associazione sulla classe stessa è **obbligatorio** inserire i nomi dei ruoli (es. predecessore/Successore)

# Tipi di Dati

martedì 4 marzo 2025 17:24

Nell'analisi bisogna scegliere dei dati che siano separati dalle scelte tecnologiche, quindi dobbiamo usare **tipi di dato concettuali**, che siano facilmente realizzabili con qualsiasi tecnologia informatica.

## Tipi base

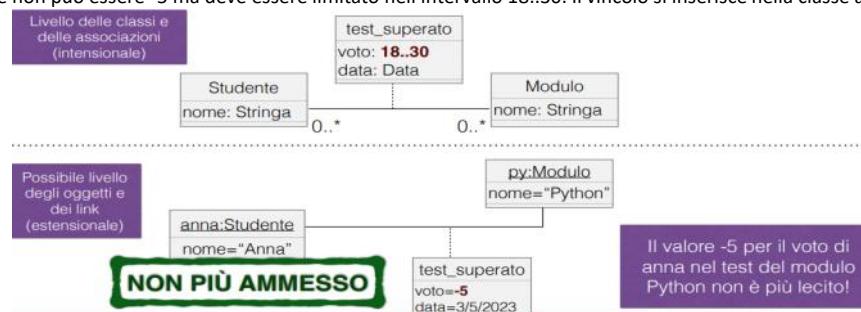
- **Intero** (1, 2, ...)
- **Reale** (1.15134, 15.51353, ecc)
- **Stringa** ("Anna", "Roma", ecc)
- **Booleano** (Vero, Falso)
- **Data** (5/10/2025)
- **Ora** (13:50)
- **DataOra** (5/10/2025 13:50)

## Dati Specializzati

Per assicurarsi che i valori siano leciti nel diagramma dobbiamo impostare dei **vincoli** sul valore dei dati, usiamo quindi dei **tipi di dati specializzati**:

Esempio: Intero > 0, Reale ≤ 0, ecc, tipo di intervallo (Intero): 0..100, 18..30, ecc

Ad esempio il voto di un esame non può essere -5 ma deve essere limitato nell'intervallo 18..30. Il vincolo si inserisce nella classe al posto del tipo dell'attributo



## Tipi Enumerativi

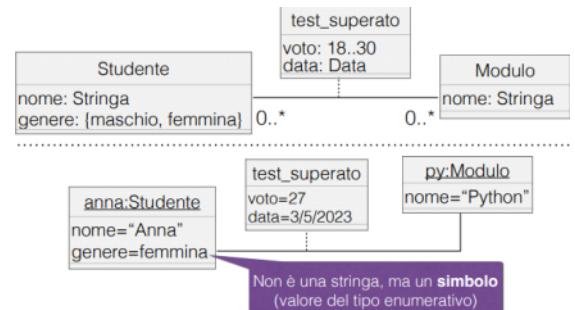
Se abbiamo un insieme di val. **finito e piccolo**, possiamo usare un **tipo di dato enumerativo** che definisce esplicitamente l'insieme dei val possibili per l'attributo.

Bisogna fare molta attenzione sull'uso del tipo enumerativo

poiché bisogna essere **sicuri** che i dati siano quelli e **non mutino nel tempo**.

Il dominio enumerativo si usa con le **parentesi graffe** e i val. sono **etichette** e non **stringhe**, quindi non bisogna usare le virgolette per indicare i valori.

Esempio: {maschio, femmina}, {Africa, America, Antartide, Asia, Europa, Oceania}, ecc



## Dati definiti dall'Utente e Dati Composti

È permesso anche definire **nuovi tipi di dati**, nel caso ad esempio in cui un tipo di dato viene utilizzato spesso nelle classi.

È necessario definire il tipo di dato dandogli un nome e il suo valore.

Si possono inoltre usare **tipi di dato composti** da più campi, chiamati **tipi record** (o record)

Esempio:

- Tipo **Genere** = {maschio, femmina} (enumerativo)
- Tipo **Indirizzo** = (via: Stringa, civico: Intero > 0, CAP: Intero > 0) (record con 3 val)

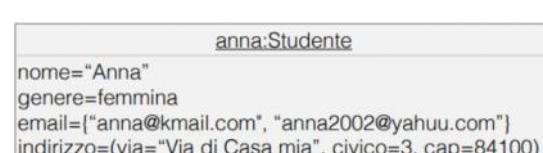
## Vincoli di Molteplicità sugli attributi

Si possono impostare dei vincoli di molteplicità anche sugli attributi stessi (il default 1..1 non viene scritto)

I vincoli di **molteplicità sugli attributi** sono segnati con le **parentesi quadre**.

Ad es.

- Ogni studente ha associato un solo nome e Genere
- Ogni studente ha associato **1 o più** email
- Ogni studente ha associato **al più un** indirizzo di casa



# Vincoli di Identificazione di Classe

mercoledì 5 marzo 2025 12:24

Oltre ai:

- **Vincolo (di integrità)**: affermazione che impone restrizioni all'insieme degli oggetti e link ammessi ulteriori a quelli strutturali dell'UML
- **Vincolo di molteplicità sulle associazioni**

È necessario impostare ulteriori vincoli per le classi, il **vincolo di identificazione di classe**

Esso impone che **non possono coesistere** oggetti di una classe che coincidono nel val. di un insieme di attributi e/o sono collegati tramite link agli stessi oggetti di altre classi.

**Esempio:**

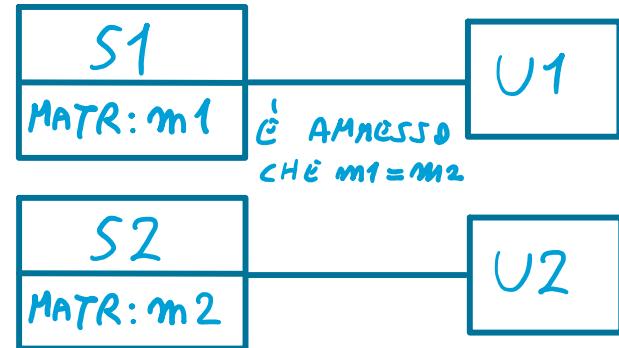
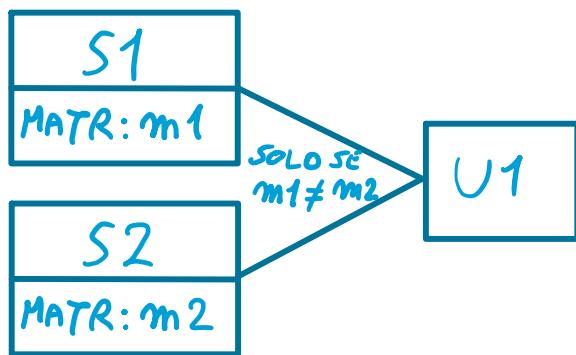
- **Non possono esistere** due persone con lo **stesso CF** ("{id1}")
- **Non possono esistere** due persone con, simultaneamente, lo **stesso nome, cognome, data di nascita** ("{id2}")

Un **vincolo di identificazione di classe** può anche coinvolgere **ruoli della classe**

**Esempio:** Non possono esistere due studenti con la stessa matricola nella stessa università



**ATTENZIONE:** Il vincolo di identificazione di classe può coinvolgere solo **attributi a molteplicità [1..1]** e/o **ruoli della classe con molteplicità 1..1**



# Generalizzazione

mercoledì 5 marzo 2025 12:43

## Ereditarietà

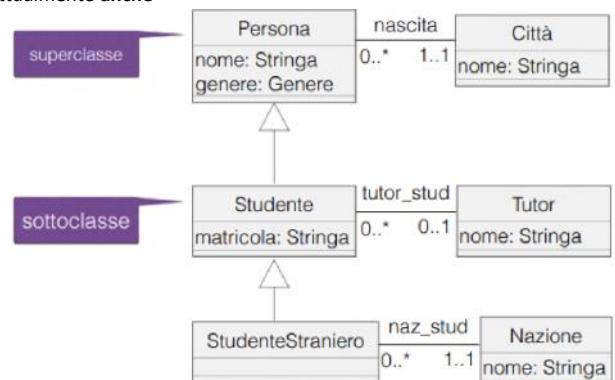
In molte situazioni si ha che **classi diverse** sussista una **relazione di sottoinsieme**. L'UML permette di definire il concetto di **relazione is-a tra classi**:

- **Relazione is-a ("è anche un")**: ogni istanza della classe Studente (**sotto-classe**) è concettualmente **anche** un'istanza della classe Persona (**super-classe**)
- Ovviamente **non vale il viceversa**: non tutte le istanze di Persona sono per forza istanze di Studente
- L'ereditarietà può essere anche a **più livelli (transitività)**, passando le classi associate e gli attributi alle sottoclassi.
- Una superclasse può avere **più sottoclassi**.

### Esempio

In presenza di relazioni is-a, vige il **meccanismo dell'ereditarietà**:

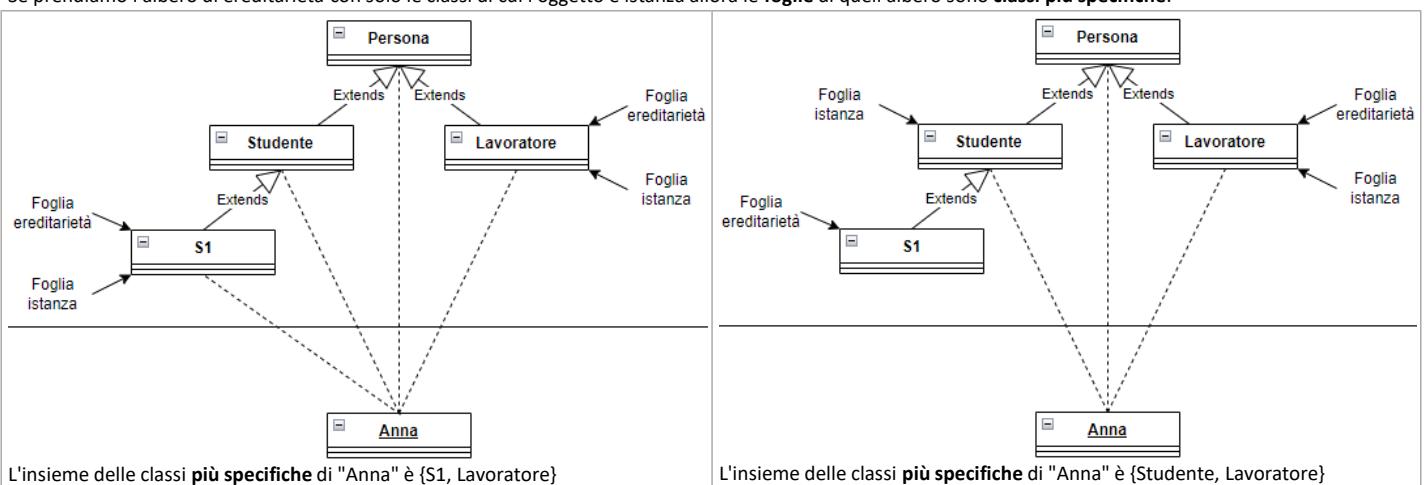
- Di tutte le Persona rappresentiamo nome, genere, città di nascita
- Di tutti gli Studenti rappresentiamo matricola e eventuale Tutor
- Di tutti gli StudentiStranieri rappresentiamo la loro Nazione d'origine
- Tutti gli Studenti sono Persone (relazione is-a) → **di conseguenza**, di tutti gli Studenti rappresentiamo anche nome, genere e città di nascita (con le loro molteplicità 1..1)
- Tutti gli StudentiStranieri sono Studenti e quindi Persone → **di conseguenza**, di tutti gli StudentiStranieri rappresentiamo anche nome, cognome, città di nascita e matricola



## Classi più Specifiche

Le classi **più specifiche** di un oggetto sono un **insieme delle classi** di cui l'oggetto è istanza che **non sono a loro volta superclassi** di altre classi dell'oggetto.

Se prendiamo l'albero di ereditarietà con solo le classi di cui l'oggetto è istanza allora le **foglie** di quell'albero sono **classi più specifiche**.



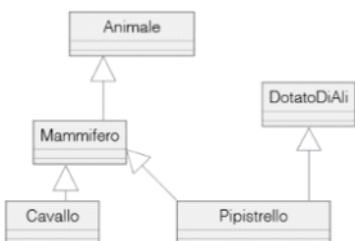
Dobbiamo contare solo l'albero di ereditarietà togliendo le classi di cui l'oggetto non è istanza (nel secondo esempio la classe S1).

## Generalizzazione

La **generalizzazione** è un costrutto **più complesso** della relazione is-a, poiché permette di definire che gli oggetti possono essere istanze di **più classi figlie** secondo **uno stesso criterio concettuale**.

### Esempio

- Studente/Lavoratore is-a Persona
- Uomo/Donna is-a Persona
- Secondo il **criterio del genere**, le Persone sono considerate Uomini e/o Donne
- Secondo il **criterio (indipendente) dell'occupazione**, le Persone sono considerate Studenti e/o Lavoratori
- È ancora possibile per un oggetto di essere istanza sia di Lavoratore che di Studente o entrambi
- Una classe può anche essere superclasse di **più generalizzazioni distinte (criteri diversi)**
- Una classe può essere **sottoclasse di più classi** avendo quindi una **ereditarietà multipla**



Il problema è che una istanza della classe base (Persona) può essere istanza di **più di una sottoclasse** di una stessa generalizzazione (sia Uomo che Donna).

Oppure una istanza della classe base (Persona) può essere istanza di **nessuna delle sottoclassi** di una stessa generalizzazione (né Uomo né Donna)

Bisogna quindi evitare questa situazione impostando dei **vincoli**.

### Vincolo Disjoint

La generalizzazione **disgiunta non permette** ad un'istanza di essere **istanza di entrambe** le sottoclassi.

E:

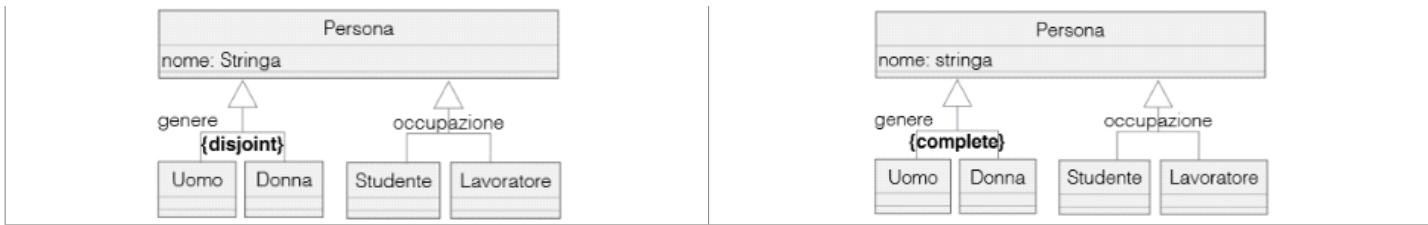
Una istanza di Persona **non può più essere istanza sia di Uomo che di Donna**.  
Quindi un'istanza di Uomo **non può essere anche istanza di Donna**  
Però un'istanza di Studente **può essere anche istanza di Lavoratore**

### Vincolo Complete

La generalizzazione **complete non permette** ad un'istanza di non essere **istanza di nessuna delle sottoclassi**.

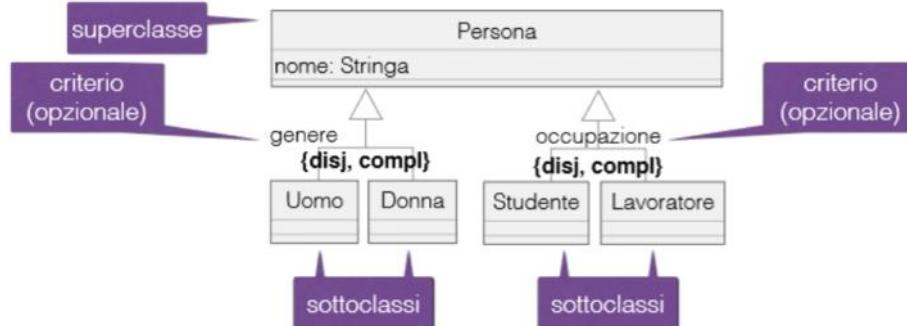
E:

Una istanza di Persona **non può più essere istanza né di Uomo né di Donna**.  
Quindi un'istanza di Persona deve **essere per forza** anche istanza di Uomo/Donna  
Però un'istanza di Persona **può non essere** di Studente e Lavoratore



### Vincolo Disjoint e Complete

Una generalizzazione può essere sia disgiunta sia completa, quindi ogni istanza della superclasse è anche istanza di esattamente una sottoclasse della generalizz.



In questo esempio si ha:

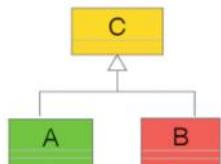
- Ogni Istanza di Persona deve essere anche istanza di esattamente una sottoclasse tra **Uomo** e **Donna**
- Ogni istanza di Persona deve essere anche istanza di esattamente una sottoclasse tra **Studente** e **Lavoratore**

- Una istanza di Persona può essere istanza sia di **Uomo** che di **Studente**? Si
- Una istanza di Persona può essere istanza né di **Uomo** né di **Donna**? Non più
- Una istanza di Persona può essere istanza sia di **Uomo** che di **Donna**? Non più

## Riepilogo

### Generalizzazione Normale

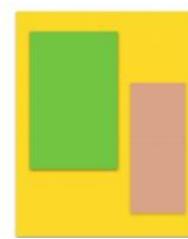
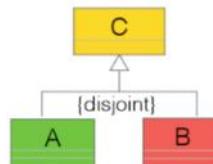
Ereditarietà classica con la relazione **is-a** tra superclassi e sottoclassi.  
Il criterio della generalizz. è **opzionale**



Un'istanza di C può essere anche istanza di A o di B (o entrambe) o niente

### Generalizzazione Disjoint

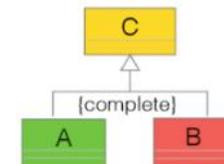
Il vincolo **disjoint** non permette ad un oggetto di essere istanza di più sottoclassi della stessa generalizzazione



Un'istanza di A non può essere anche istanza di B

### Generalizzazione Complete

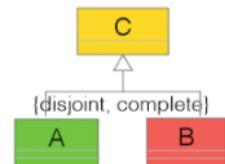
Il vincolo **complete** non permette ad un oggetto di non essere istanza di nessuna sottoclasse della stessa gener.



Un'istanza di C deve essere per forza anche istanza di A o B (o entrambe)

### Disjoint e Complete

Entrambi i vincoli impongono che ogni oggetto sia istanza esattamente di una sola sottoclasse della stessa generalizz.



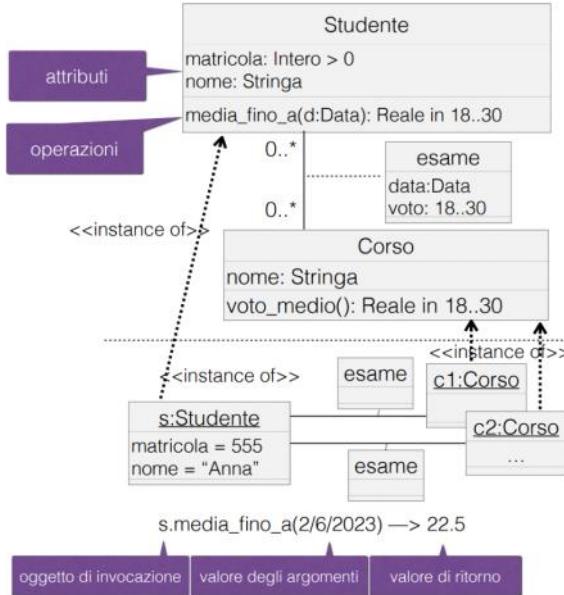
Una istanza di C deve essere per forza anche istanza di solo una classe tra A e B

# Operazioni di Classe

venerdì 14 marzo 2025 11:02

UML, per le classi, può definire proprietà:

- **Statiche:** quelle viste in precedenza, dove i val. **cambiano** dopo la **modifica** da parte degli **utenti** del sistema
- **Dinamiche:** dove i val. vengono **calcolati ogni volta** che servono, a **partire** dai **valori** di altre proprietà



## Operazioni di Classe

Una **op. della classe C** indica che su ogni oggetto di quella classe si può eseguire un calcolo per:

- **Calcolare** un val. a partire da altri val/operazioni
- **Cambiare lo stato dell'oggetto** (modificare le sue proprietà), **dei link** in cui è coinvolto e/o degli oggetti collegati a esso.

## Sintassi

**nome\_ogg.nome\_op(argomenti): tipo Ritorno**

- "argomenti": lista di elem. in forma **nome\_argomento: tipo\_argomento**
- "tipo Ritorno": tipo del val. **restituito** dall'operazione

I tipi degli argomenti/ritorno possono essere tipi di **dato concettuali o classi del diagramma**.

L'op. di classe si può **invocare solo dall'oggetto** della classe.

## Esempio:

**s.media\_fino\_a(2/6/2023) → 22.5**

- "s" è l'oggetto della classe **Studente** che esegue la chiamata dell'operazione
- "media\_fino\_a" è l'operazione della classe
- "2/6/2023" è il val. di tipo Data **mandato** come **argomento** nell'operazione
- "22.5" è il val. di tipo Reale che viene **restituito** dall'operazione

L'op. di classe può essere chiamata anche da una **sottoclasse** della classe in cui vi è l'operazione

## Specifiche delle Operazioni

Il diagramma delle classi non definisce cosa calcolano/modificano le op, quindi ogni classe con op. andrà affiancata da un **documento di specifica** che entra nel dettaglio. (CREARE COLLEGAMENTO DOPO)

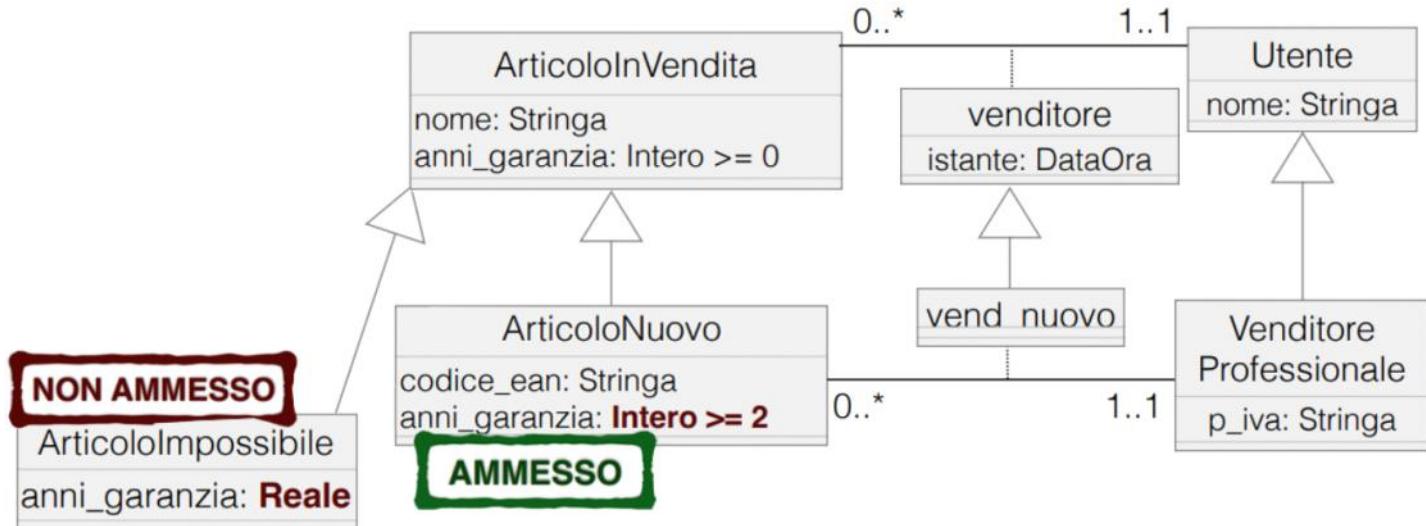
## Specializzazione di Attributi

venerdì 14 marzo 2025 13:31

Una sottoclasse, oltre a proprietà aggiuntive, può anche **specializzare** le proprietà **ereditate** dalla superclasse, **restringendone il tipo** (lasciando lo stesso tipo di dato)

Esempio:

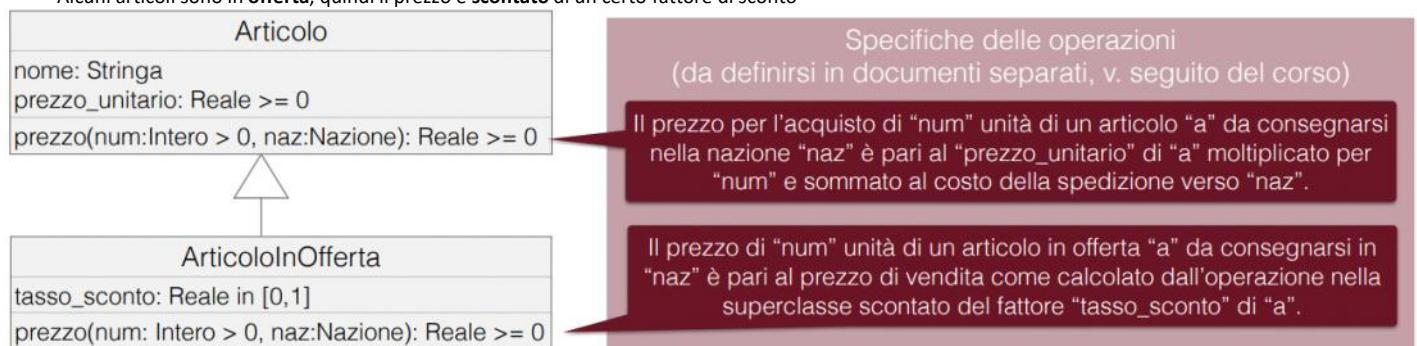
- Degli articoli in vendita si rappresenta il nome e gli anni di garanzia (anche zero)
- Alcuni articoli sono **nuovi** e per questi ci interessa l'EAN e che l'anno di garanzia deve essere **almeno due anni**
- Un utente può vendere **esattamente** un libro tra quelli in vendita
- Gli articoli nuovi li può vendere invece **solo** i venditori professionali
- **ATTENZIONE:** l'assoc. **vend\_nuovo** deve essere di tipo compatibile con il tipo dell'assoc. **venditore**



Anche le **op. di classe** possono essere oggetto di **specializzazioni nella sottoclasse**

Esempio:

- Di tutti gli articoli in vendita va calcolato il prezzo = prezzo unitario \* num. pezzi richiesto + spese di spedizione (dipende dalla nazione dove viene consegnata)
- Alcuni articoli sono in **offerta**, quindi il prezzo è **scontato** di un certo fattore di sconto



Come per gli attributi, la segnatura dell'op. nella sottoclasse deve essere **compatibile** con quella dell'operazione definita nella superclasse, quindi:

- **Stesso num. e tipo di argomenti**
- Il **tipo di ritorno** nella sottoclasse è dello stesso tipo o un sotto-tipo di quello dell'operazione nella superclasse

# Azienda 1

martedì 4 marzo 2025 15:02

## Obiettivi

Si vuole sviluppare un sistema informativo per la gestione dei dati sul personale di una certa azienda costituita da diversi dipartimenti. Durante la fase di raccolta dei requisiti è stata prodotta la specifica dei requisiti mostrata di seguito.

Si chiede di iniziare la fase di **Analisi dei requisiti** ed in particolare di:

1. raffinare la specifica dei requisiti eliminando inconsistenze, omissioni o ridondanze e produrre un elenco numerato di requisiti il meno ambiguo possibile
2. produrre un diagramma UML delle classi concettuale che modelli i dati di interesse, utilizzando solo i costrutti di classe, associazione, attributo.

## Requisiti

I dati di interesse per il sistema sono **impiegati**, **dipartimenti**, **direttori dei dipartimenti** e **progetti aziendali**.

Di ogni **impiegato** interessa conoscere il **nome**, il **cognome**, la **data di nascita** e lo **stipendio attuale**, il **dipartimento** (esattamente **uno**) al quale afferisce.

Di ogni **dipartimento** interessa conoscere il **nome**, il **numero di telefono del centralino**, e la **data di afferenza di ognuno degli impiegati che vi lavorano**.

Di ogni **dipartimento** interessa conoscere inoltre il **direttore**, che è **uno degli impiegati dell'azienda**.

Il **sistema** deve permettere di rappresentare i **progetti aziendali** nei quali sono coinvolti i diversi **impiegati**.

Di ogni **progetto** interessa il **nome** ed il **budget**.

Ogni **impiegato** può partecipare ad un **numero qualsiasi** di **progetti**.

## Analisi dei Requisiti

Per ogni ruolo dobbiamo trovare i loro attributi.

### Prova 1

Impiegato:

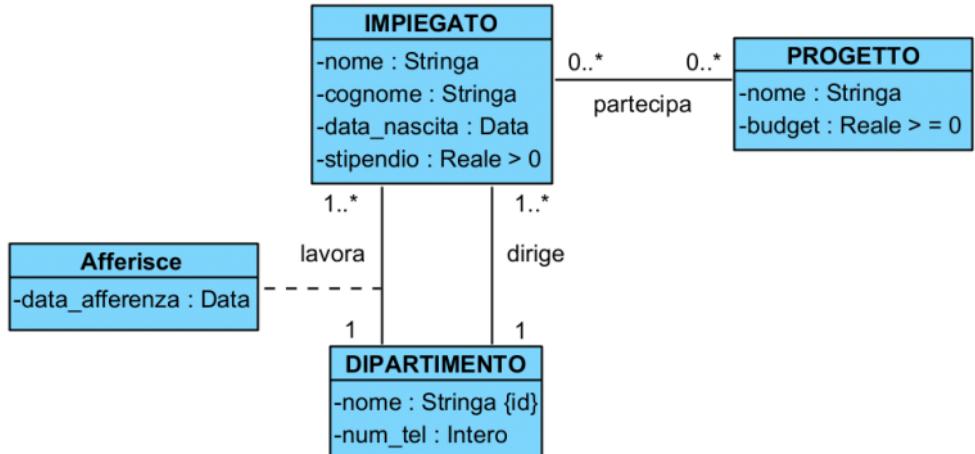
- Nome: Stringa
- Cognome: Stringa
- Data nascita: Data
- Stipendio attuale: Reale  $\geq 0$
- dipartimento (Uno): Dipartimento
  - Data di afferenza
- Progetti (infiniti): Progetto

Dipartimento:

- Nome: Stringa {univoco}
- num. tel: Intero
- Impiegati che ci lavorano: Impiegato
  - Data di afferenza
- Direttore (uno): Impiegato

Progetto:

- nome: Stringa
- budget: Reale  $\geq 0$
- Impiegati che ci lavorano (infiniti): Impiegati



# Università 1

venerdì 14 marzo 2025 17:28

## Obiettivi

Si vuole sviluppare un sistema informativo per la gestione dei dati di una università. Durante la fase di raccolta dei requisiti è stata prodotta la seguente specifica dei requisiti. Si chiede di iniziare la fase di Analisi dei requisiti ed in particolare di:

1. raffinare la specifica dei requisiti eliminando inconsistenze, omissioni o ridondanze e produrre un elenco numerato di requisiti il meno ambiguo possibile
2. produrre un diagramma UML delle classi concettuale che modelli i dati di interesse, utilizzando solo i costrutti di classe, associazione, attributo.

## Requisiti

I dati di interesse per il sistema sono studenti, facoltà, professori e corsi.

- Di ogni studente interessa conoscere il nome, il codice fiscale, il numero di matricola, la data di nascita, il luogo di nascita (città e regione), il corso di laurea a cui è iscritto (con l'anno di iscrizione), e gli insegnamenti di cui ha superato l'esame.
- Dei professori interessa il nome, la data di nascita, il codice fiscale, il luogo di nascita e gli insegnamenti erogati.
- Dei corsi di laurea interessa il nome e la o le facoltà di appartenenza.
- Di queste ultime interessa il nome.
- Di ogni insegnamento interessa il codice, il nome, il numero di ore di lezione, e i corsi di laurea a cui appartiene

## Analisi dei Requisiti

Studente:

- nome: Stringa
- CF: Stringa (univoco)
- Matricola: Stringa
- Data\_nascita: Data
- Luogo\_nascita: (Città: Stringa, Regione: Stringa)
- Corso\_laurea: → Corso (da 1 a inf) + anno iscrizione (association class)
- insegnamenti superati: Insegnamento (da 0 a inf)

Professori:

- nome: Stringa
- Data\_nascita: Data
- CF: Stringa (univoco)
- Luogo\_nascita: (Città: Stringa, Regione: Stringa)
- insegnamenti erogati: Insegnamento (da 1 a inf)

Studenti e Professori possono essere sottoclassi di Persona (con un disjoint così un prof non può essere studente), che ha:

- nome: Stringa
- CF: Stringa (univoco)
- Data\_nascita: Data
- Luogo\_nascita: (Città: Stringa, Regione: Stringa)

CORSO:

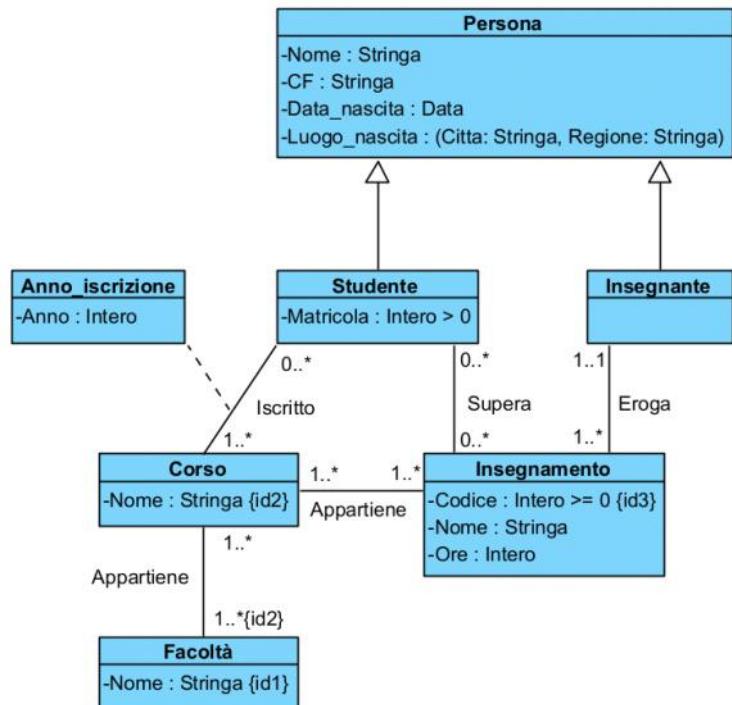
- nome: Stringa (univoco?)
- facoltà di appartenenza: Facoltà (da 1 a inf)
- Studente del corso: Studente (da 0 a inf)
- Insegnamenti: Insegnamento (da 1 a inf)

Facoltà:

- Nome (univoco)
- Corsi: Corso (da 1 a inf)

Insegnamenti:

- Codice: Intero (univoco)
- Nome: Stringa
- Ore\_lezione: Intero
- Corso\_appartenenza: Corso (da 1 a inf)
- Studenti nel corso: Studente (da 0 a inf)



# Voli Aerei 1

venerdì 14 marzo 2025 17:29

## Obiettivi

Si vuole sviluppare un sistema informativo per la gestione di dati relativi a voli aerei. Durante la fase di raccolta dei requisiti è stata prodotta la seguente specifica dei requisiti. Si chiede di iniziare la fase di Analisi dei requisiti ed in particolare di:

1. raffinare la specifica dei requisiti eliminando inconsistenze, omissioni o ridondanze e produrre un elenco numerato di requisiti il meno ambiguo possibile
2. produrre un diagramma UML delle classi concettuale che modelli i dati di interesse, utilizzando solo i costrutti di classe, associazione, attributo.

## Requisiti

I dati di interesse per il sistema sono voli, compagnie aeree ed aeroporti.

- Dei voli interessa rappresentare codice, durata, compagnia aerea ed aeroporti di partenza e arrivo.
- Degli aeroporti interessa rappresentare codice, nome, città (con nome e numero di abitanti) e nazione.
- Delle compagnie aeree interessa rappresentare nome, anno di fondazione, e la città in cui ha sede la direzione.

## Analisi dei Requisiti

Voli:

- Codice: Stringa (univoco)
- Durata: Reale
- Compagnia\_aerea: Compagnia (una)
- Aeroporto\_part: Aeroporto (uno)
- Aeroporto\_arriv: Aeroporto (uno)

Aeroporto:

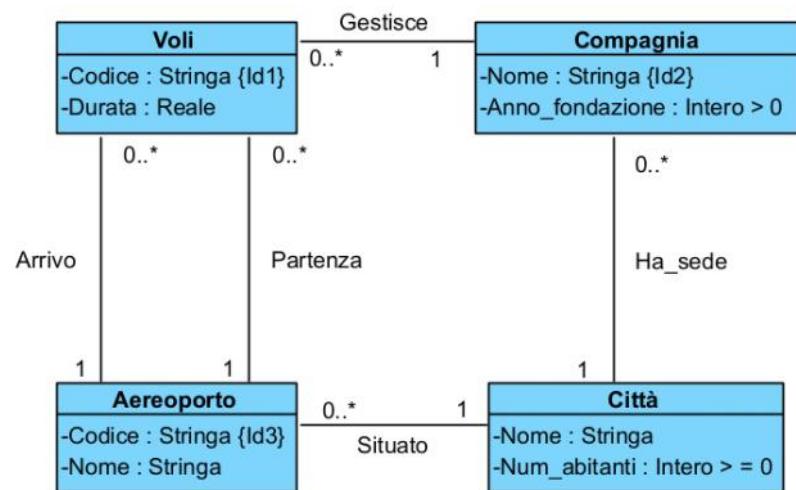
- Codice: Stringa (univoco)
- Nome: Stringa
- Città: Città (una)
- Nazione: Stringa (univoco) (non so se metterlo qua o in Città)
- Voli: Voli in uscita (da 0 a inf), voli in entrata (da 0 a inf)

Compagnia:

- Nome: Stringa (univoco)
- Anno\_fondazione: Intero > 0
- Citta\_sede: Città (una)
- Gestisce piu voli: Voli (da 0 a inf)

Città:

- Nome: Stringa
- Num\_abitanti: Intero > 0
- Aeroporti (da 0 a inf)
- Sedi\_compagnie: Compagnia: (da 0 a inf)



# Accademia 1

venerdì 14 marzo 2025 17:31

## Obiettivi

Si vuole progettare un sistema informativo per la gestione delle tabelle orarie relative al ruolo di docente universitario. Durante la fase di raccolta dei requisiti è stata prodotta la seguente specifica dei requisiti. Si chiede di iniziare la fase di Analisi Concettuale ed in particolare di:

1. raffinare la specifica dei requisiti eliminando inconsistenze, omissioni o ridondanze e produrre un elenco numerato di requisiti il meno ambiguo possibile
2. produrre un diagramma UML delle classi concettuale che modelli i dati di interesse, utilizzando solo i costrutti di classe, associazione, attributo
3. produrre la relativa specifica dei tipi di dato in caso si siano definiti nuovi tipi di dato concettuali.

## Requisiti

I dati di interesse per il sistema sono i docenti universitari, i progetti di ricerca e le attività dei docenti.

- Di ogni docente interessa conoscere il nome, il cognome, la data di nascita, la matricola, la posizione universitaria (ricercatore, professore associato, professore ordinario) e i progetti ai quali partecipa.
- Dei progetti interessa il nome, un acronimo, la data di inizio, la data di fine e i docenti che vi partecipano.
- Un progetto è composto da molti Work Package (WP). Oltre al progetto a cui fa riferimento, del WP interessa sapere il nome, la data di inizio e la data di fine.
- Il sistema deve permettere ai docenti di registrare impegni di diverso tipo. Degli impegni interessa sapere il giorno in cui avvengono, la durata in ore e la tipologia di impegno con relativa motivazione.

## Analisi dei Requisiti

Docente:

- Nome: Stringa
- Cognome: Stringa
- Data\_nascita: Data
- Matricola: Intero (univoco)
- Posizione\_Uni: Enum (Ricercatore, Prof\_associato, Prof\_ordinario)
- Progetti a cui partecipa: Progetto (da 0 a inf)
- Impegni (da 0 a inf)

Progetto:

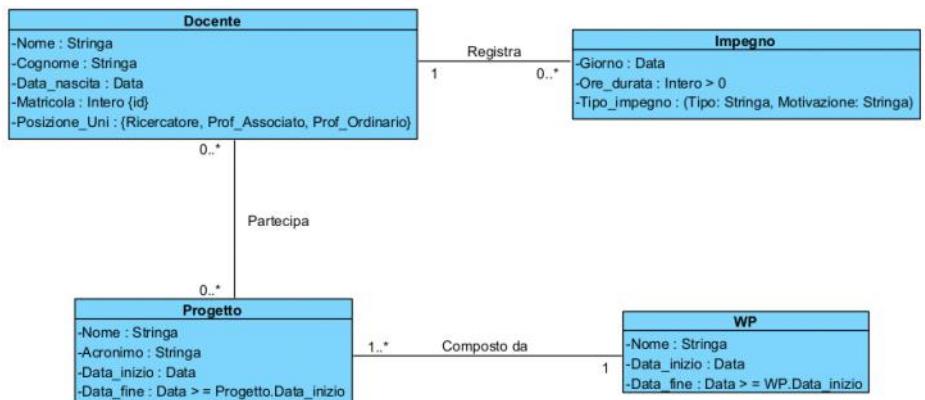
- Nome: Stringa
- Acronimo: Stringa
- Data\_inizio: Data
- Data\_fine: Data  $\geq$  Data\_inizio
- Docenti che vi partecipano: Docente (da 0 a inf)
- Work\_package: WP(da 1 a inf)

WP:

- Nome: Stringa
- Data\_inizio: Data
- Data\_fine: Data  $\geq$  data\_inizio
- Progetti di cui fa parte: Progetto (uno)

Impegno:

- Giorno\_avvenimento: Data
- Ore\_durata: Intero  $> 0$
- Tipo\_impegno: (Tipo: Stringa, Motivazione: Stringa)
- Docente (uno)



# Voli Aerei 2

venerdì 14 marzo 2025 17:32

## Obiettivi

Si vuole sviluppare un sistema informativo per la gestione di dati relativi a voli aerei. Durante la fase di raccolta dei requisiti è stata prodotta la seguente specifica dei requisiti. Si chiede di iniziare la fase di Analisi Concettuale ed in particolare di:

1. raffinare la specifica dei requisiti eliminando inconsistenze, omissioni o ridondanze e produrre un elenco numerato di requisiti il meno ambiguo possibile
2. produrre un diagramma UML delle classi concettuale che modelli i dati di interesse, utilizzando solo i costrutti di classe, associazione, attributo, generalizzazione tra classi
3. produrre la relativa specifica dei tipi di dato in caso si siano definiti nuovi tipi di dato concettuali.

## Requisiti

I dati di interesse per il sistema sono voli, compagnie aeree ed aeroporti.

- Dei voli interessa rappresentare codice, durata, compagnia aerea ed aeroporti di partenza e arrivo.
- Degli aeroporti interessa rappresentare codice, nome, città (con nome e numero di abitanti) e nazione.
- Delle compagnie aeree interessa rappresentare nome, anno di fondazione, e la città in cui ha sede la direzione.
- Un tipo particolare di voli sono voli charter. Questi possono prevedere tappe intermedie in aeroporti. Delle tappe intermedie di un volo charter interessa mantenere l'ordine con cui esse si susseguono (ad esempio, un certo volo che parte da "Milano Linate" e arriva a "Palermo Punta Raisi", prevede tappe intermedie prima nell'aeroporto di Bologna e poi in quello di Napoli). Dei voli charter interessa rappresentare anche il modello di velivolo usato.

## Analisi dei Requisiti

## Basi di Dati, Modulo 2 *Laurea in Informatica*

Parte: Analisi concettuale  
UNITÀ: A.1

Prof. Toni Mancini  
Dipartimento di Informatica

Versione del 2024-02-18



SAPIENZA  
UNIVERSITÀ DI ROMA

A.1

Analisi dei Requisiti  
Unified Modeling  
Language

## Basi di Dati, Modulo 2 *Laurea in Informatica*

Parte: Analisi concettuale  
UNITÀ: A.1

Prof. Toni Mancini  
Dipartimento di Informatica

Versione del 2024-02-18



SAPIENZA  
UNIVERSITÀ DI ROMA

A.1.1

Analisi dei Requisiti  
Unified Modeling Language  
Diagrammi UML delle classi  
e degli oggetti



# Basi di Dati, Modulo 2

## *Laurea in Informatica*

Parte: Analisi concettuale

UNITÀ: A.1

Prof. Toni Mancini  
Dipartimento di Informatica

Versione del 2024-02-18



**SAPIENZA**  
UNIVERSITÀ DI ROMA

### A.1.1.1

Analisi dei Requisiti

Unified Modeling Language

Diagrammi UML delle classi e degli  
oggetti

Oggetti e classi



SAPIENZA  
UNIVERSITÀ DI ROMA

## Diagramma delle classi e degli oggetti per l'analisi

- Nella fase di analisi ci si concentra sulle classi più che sugli oggetti
- Gli oggetti servono essenzialmente per descrivere elementi singoli particolarmente significativi oppure per descrivere esempi
- Approccio simile al paradigma della programmazione object-oriented, ad es. in Java:
  - Un programma si scrive definendo un insieme definito di class(i), non di oggetti
  - Gli oggetti vengono creati, modificati e distrutti durante l'esecuzione del programma
- Come detto in precedenza, faremo riferimento solo ad un sottoinsieme dei meccanismi previsti in UML per descrivere il diagramma delle classi

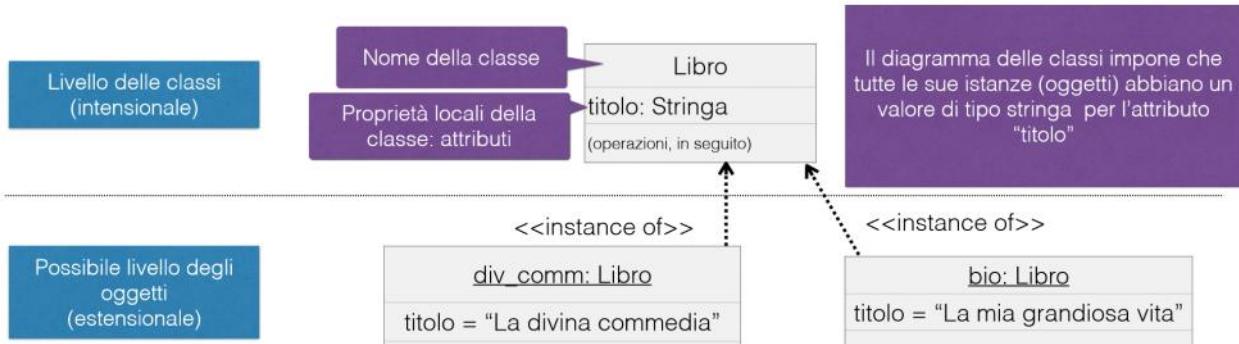


- Un oggetto in UML modella un elemento del dominio di analisi che:
  - ha "vita propria"
  - è identificato univocamente mediante l'identificatore di oggetto
  - è istanza di una classe (la cosiddetta classe più specifica – vedremo che, in determinate circostanze, un oggetto è istanza di più classi, ma in ogni caso, tra le classi di cui un oggetto è istanza, esiste sempre la classe più specifica)
- `div_comm` è l'identificatore di oggetto (scelto dall'analista per potersi riferire all'oggetto nello schema concettuale)
- Libro è la classe più specifica di cui l'oggetto è istanza
- Si noti la sottolineatura



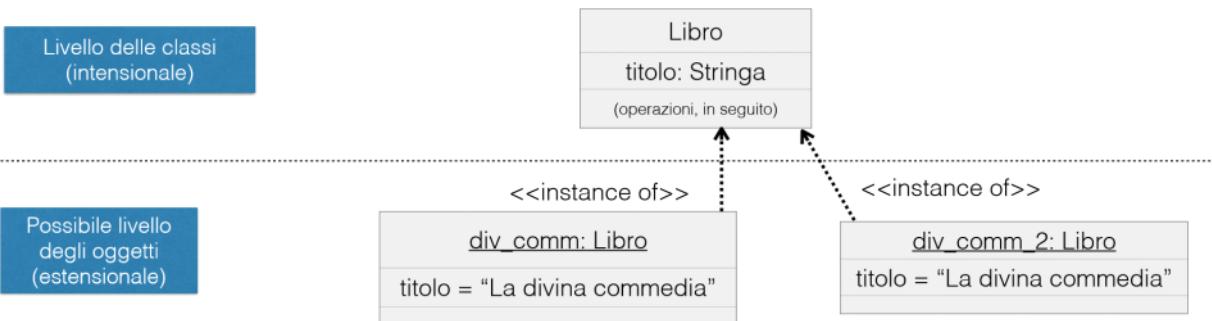
- Una classe modella un insieme di oggetti omogenei (le istanze della classe) ai quali sono associate proprietà statiche (attributi) e dinamiche (operazioni, le vedremo in seguito).
- Ogni classe è descritta da:
  - un nome
  - un insieme di proprietà (astrazioni delle proprietà comuni degli oggetti che sono istanze delle classi)

N



- Un oggetto (istanza di una classe) è identificato da un identificatore univoco
- Un diagramma delle classi in generale permette la coesistenza di oggetti identici

I due oggetti ri



mangono due oggetti distinti anche se hanno lo stesso val in titolo.

degli oggetti  
(estensionale)

div\_comm: Libro

titolo = "La divina commedia"

div\_comm\_2: Libro

titolo = "La divina commedia"

AMMESSO

Oggetti identici possono coesistere

## Basi di Dati, Modulo 2 *Laurea in Informatica*

Parte: Analisi concettuale  
UNITÀ: A.1

Prof. Toni Mancini  
Dipartimento di Informatica

Versione del 2024-02-18



SAPIENZA  
UNIVERSITÀ DI ROMA

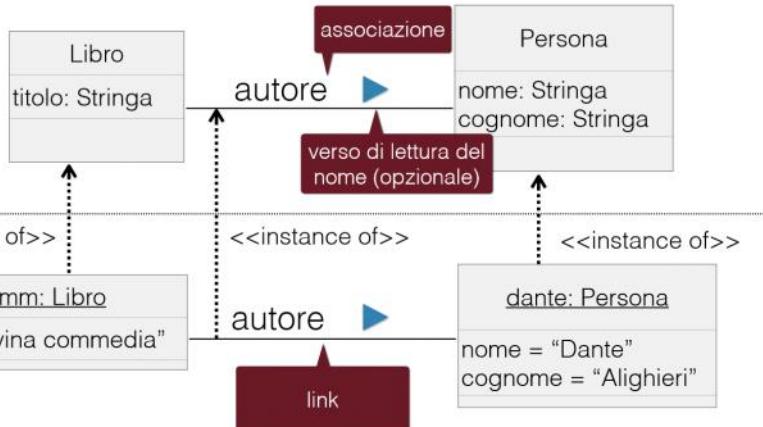
### A.1.1.2

Analisi dei Requisiti  
Unified Modeling Language  
Diagrammi UML delle classi e degli  
oggetti  
Link e associazioni

## Associazioni e link

- Una associazione modella la possibilità che oggetti di due (o più classi) abbiano dei legami
- Le istanze di associazioni si chiamano link: se A è una associazione tra le classi C1 e C2, una istanza di A è un link tra due oggetti (in altre parole, una coppia), uno della classe C1 e l'altro della classe C2

Livello delle classi e delle associazioni  
(intensionale)



## Associazioni e link (2)

- Come gli oggetti sono istanze delle classi, così i link sono istanze delle associazioni (gli archi <<instance of>> non sono necessari)
- Al contrario degli oggetti, però, i **link non hanno identificatori esplicativi**: un link è **implicitamente identificato** dalla coppia (o in generale dalla ennupla, v. seguito) di oggetti che esso rappresenta
- Ciò implica, ad esempio, che **il seguente diagramma degli oggetti non è ammesso dal diagramma delle classi**

Livello delle classi e delle associazioni

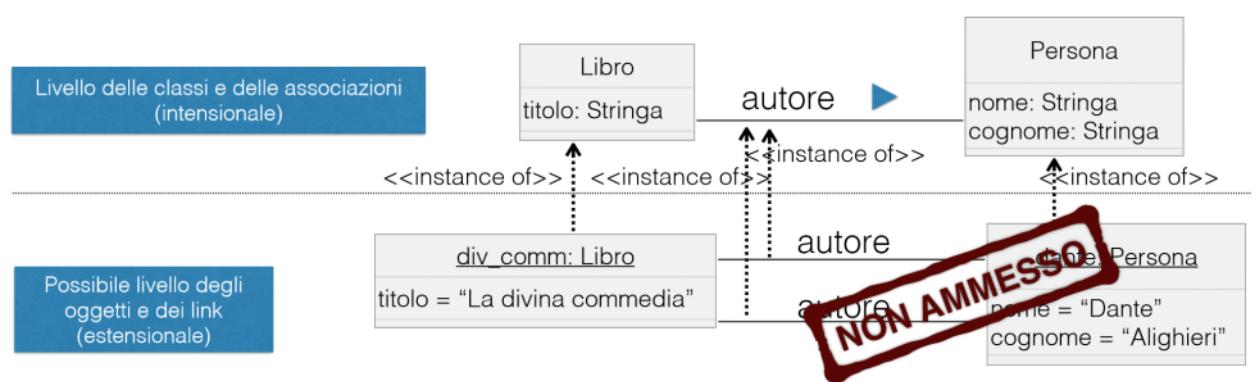


za di associazione

è un insieme di coppie (es. libro persona)

quelle coppie

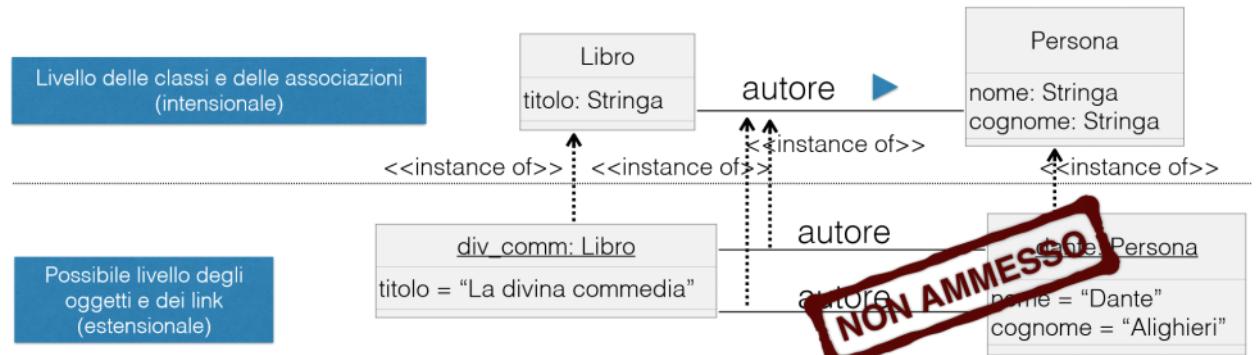
ificati dagli oggetti che vi partecipano



**Non ammesso:** link uguali **non** possono coesistere

## Associazioni e link (3)

- ...il seguente diagramma degli oggetti non è ammesso dal diagramma delle classi
- Una associazione tra le classi Libro e Persona definisce la **possibilità** che le istanze (oggetti) della classe Libro siano in relazione con istanze (oggetti) di classe Persona (la relazione si chiama "autore").
- Aver modellato la possibilità di tali legami mediante una associazione implica che, per l'analista, **non ha concettualmente senso** il fatto che un libro abbia "più volte lo stesso autore".



## Esempio

Si vuole progettare un'applicazione che permetta ai clienti di prenotare hotel via web

## Esempio

Si vuole progettare un'applicazione che permetta ai clienti di prenotare hotel via web



## Esempio

Si vuole progettare un'applicazione che permetta ai clienti di prenotare hotel via web

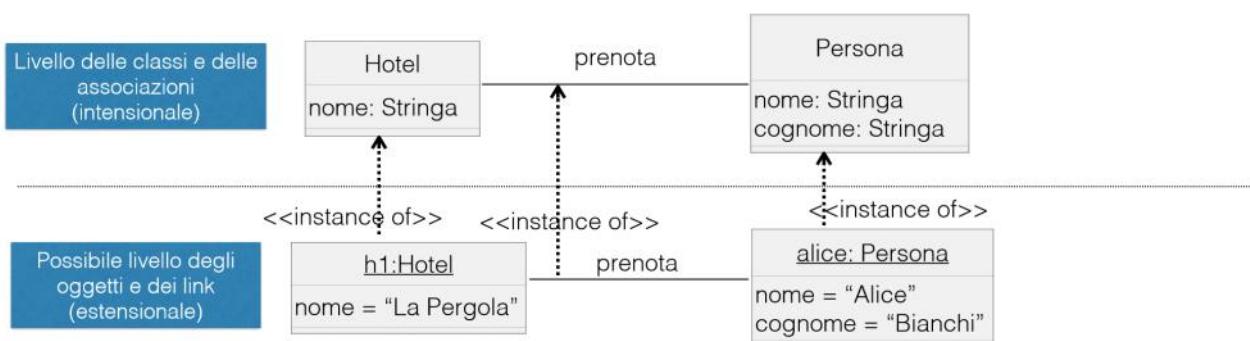


Prof. Toni Mancini – Basi di Dati, Modulo 2 – Laurea in Informatica

A.1. 12

## Esempio

Si vuole progettare un'applicazione che permetta ai clienti di prenotare hotel via web

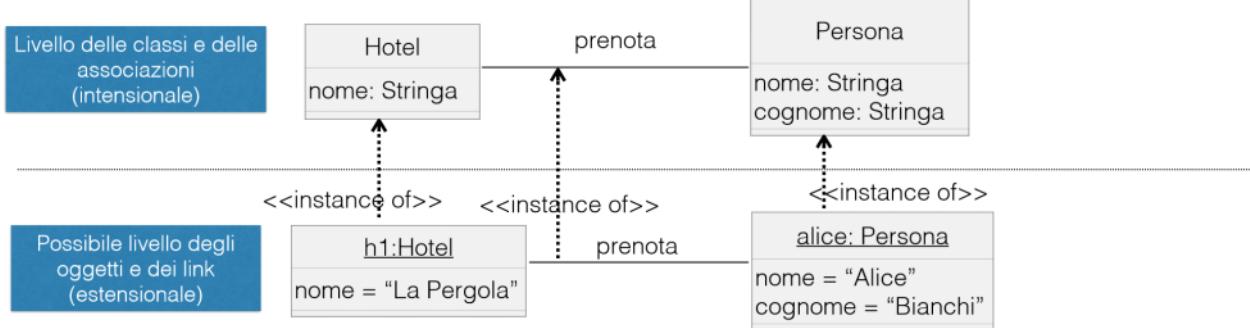


Prof. Toni Mancini – Basi di Dati. Modulo 2 – Laurea in Informatica

A.1. 12

## Esempio

Si vuole progettare un'applicazione che permetta ai clienti di prenotare hotel via web.



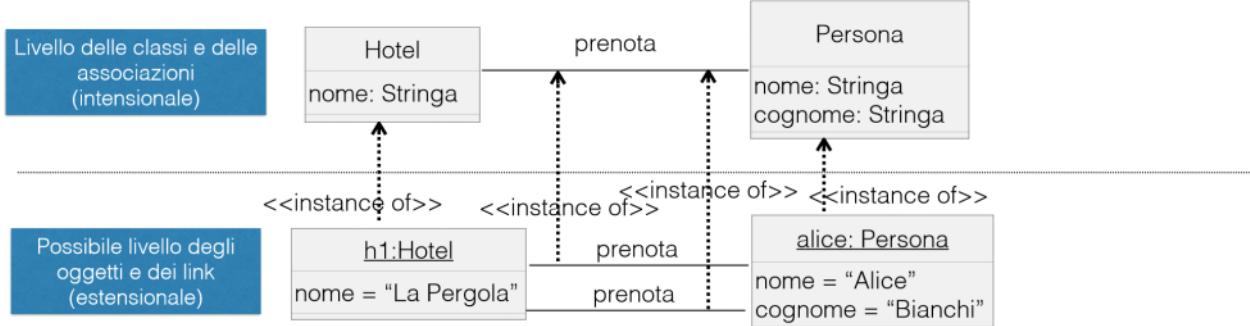
- E se volessimo rappresentare una seconda prenotazione di 'alice' presso 'h1'?



- E se volessimo rappresentare una seconda prenotazione di 'alice' presso 'h1'?

## Esempio

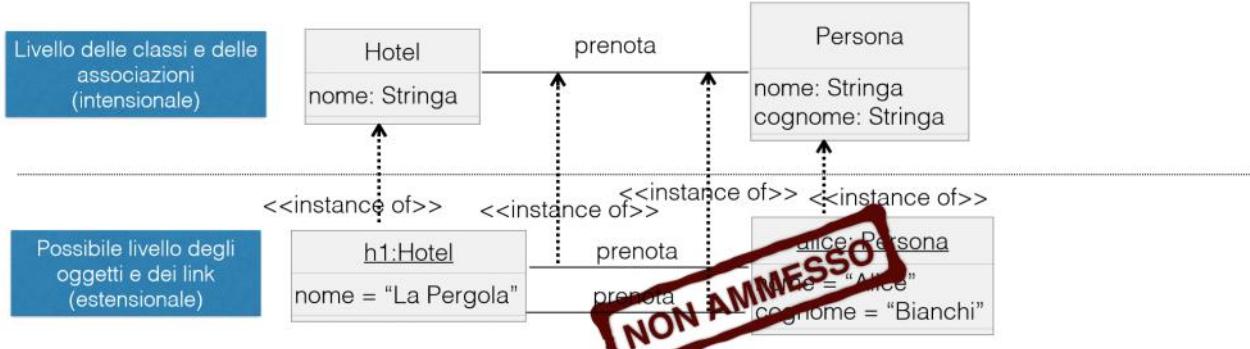
Si vuole progettare un'applicazione che permetta ai clienti di prenotare hotel via web



- E se volessimo rappresentare una seconda prenotazione di 'alice' presso 'h1'?

## Esempio

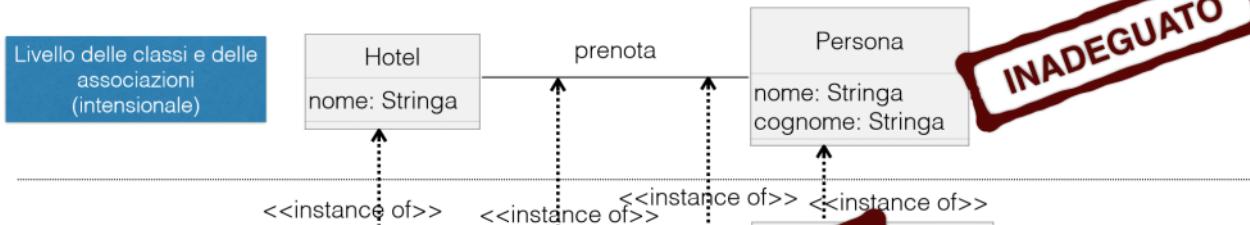
Si vuole progettare un'applicazione che permetta ai clienti di prenotare hotel via web



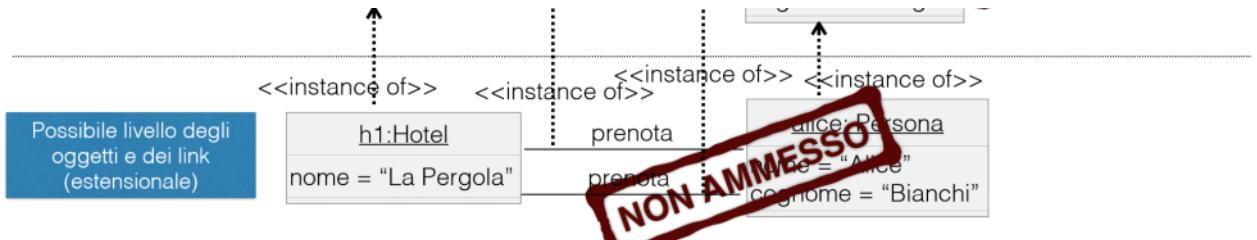
- E se volessimo rappresentare una seconda prenotazione di 'alice' presso 'h1'?

## Esempio

Si vuole progettare un'applicazione che permetta ai clienti di prenotare hotel via web



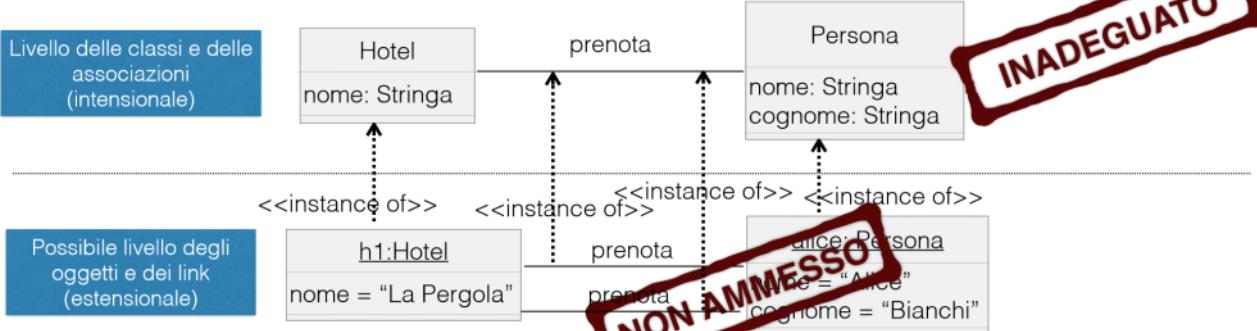




- E se volessimo rappresentare una seconda prenotazione di 'alice' presso 'h1'?
- Il diagramma qui sopra impedirebbe ad una stessa persona di prenotare, nella sua vita, lo stesso hotel più volte!

## Esempio

Si vuole progettare un'applicazione che permetta ai clienti di prenotare hotel via web



- E se volessimo rappresentare una seconda prenotazione di 'alice' presso 'h1'?
- Il diagramma qui sopra impedirebbe ad una stessa persona di prenotare, nella sua vita, lo stesso hotel più volte!
- Come risolvere?

## Esempio

Si vuole progettare un'applicazione che permetta ai clienti di prenotare hotel via web

- Il diagramma precedente impedirebbe ad una stessa persona di prenotare, nella sua vita, lo stesso hotel più volte!
- Come risolvere?



- Abbiamo fatto in modo che ogni singola prenotazione abbia "vita propria"  
→ le prenotazioni sono a loro volta oggetti, istanze della nuova classe Prenotazione

## Associazioni multiple tra due classi

- Tra le stesse classi possono essere definite più associazioni, che modellano **legami di natura diversa**



- Tra le stesse classi possono essere definite più associazioni, che modellano **legami di natura diversa**

Livello delle classi e delle associazioni (intensionale)



Non è obbligatorio l'autore)

div\_comm: Libro

titolo = "La divina commedia"

**AMMESSO**

Possibile livello degli oggetti e dei link (estensionale)

autore

dante: Persona

nome = "Dante"  
cognome = "Alighieri"

editore

feltr: Persona

nome = "Faustino"  
cognome = "Feltrinelli"

bio: Libro

titolo = "La mia grandiosa vita"

autore

editore

modestino: Persona

nome = "Modestino"  
cognome = "Rossi"

**AMMESSO**

Prof. Toni Mancini – Basi di Dati, Modulo 2 – Laurea in Informatica

A.1. 14

## Basi di Dati, Modulo 2 *Laurea in Informatica*

Parte: Analisi concettuale  
UNITÀ: A.1

Prof. Toni Mancini  
Dipartimento di Informatica

Versione del 2024-02-18



SAPIENZA  
UNIVERSITÀ DI ROMA

### A.1.1.3

Analisi dei Requisiti  
Unified Modeling Language  
Diagrammi UML delle classi e degli oggetti  
Vincoli di molteplicità sulle associazioni e sugli attributi

- I diagrammi delle classi visti fino ad ora sono modelli **molto laschi** della realtà di interesse

Livello delle classi e delle associazioni (intensionale)



Possibile livello degli oggetti e dei link (estensionale)



Questo insieme di oggetti è ammesso dal diagramma delle classi di sopra, ma **non** soddisfa i vincoli del

orio usare entrambi i link per gli oggetti (posso anche collegare anche solo

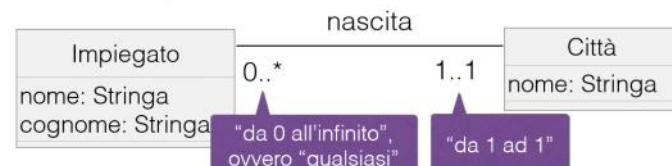
Possibile livello degli oggetti e dei link (estensionale)



## Associazioni: vincoli di molteplicità (2)

- UML permette di definire **vincoli di integrità** in un **diagramma delle classi**
- Un vincolo di integrità impone ulteriori restrizioni (oltre quelle strutturali imposte dal diagramma) sui livelli estensionali ammessi
- Vediamo ora i **vincoli di molteplicità sulle associazioni**

Livello delle classi (intensionale)

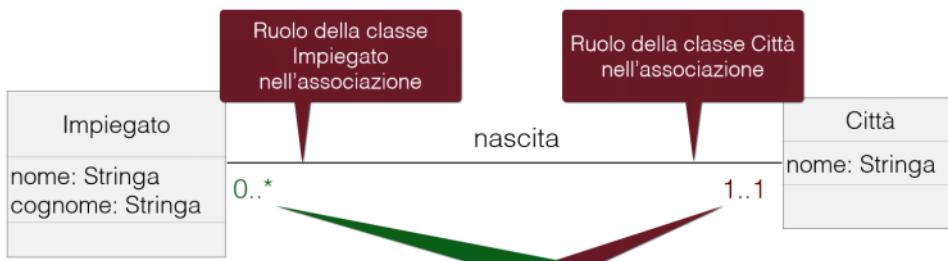


Possibile livello degli oggetti e dei link (estensionale)



## Associazioni: vincoli di molteplicità (3)

- Semantica dei vincoli di molteplicità sui ruoli delle associazioni



1..1: Ogni istanza di Impiegato deve essere coinvolta in un numero di link dell'associazione "nascita" che va "da 1 ad 1".  
Dato che non possiamo avere più link tra la stessa coppia di oggetti, questo è equivalente a: ogni istanza di Impiegato deve essere legata ad una ed una sola istanza di Città (tramite link dell'associazione "nascita")

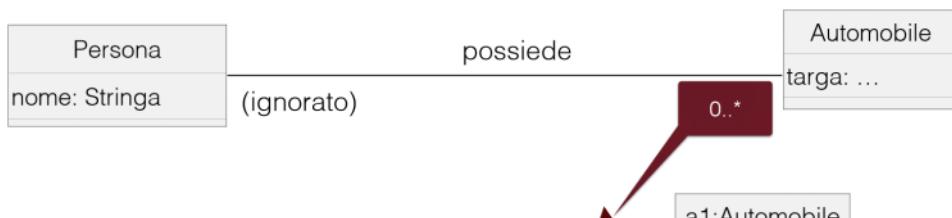
0..\*: Ogni istanza di Città deve essere coinvolta in un numero di link dell'associazione "nascita" che va "da 0 all'infinito".  
È equivalente a dire: ogni istanza di Città può essere legata ad un numero qualunque (0..\*) di istanze di Impiegato (tramite link dell'associazione "nascita")

Da imparare a r

ogni oggetto cit

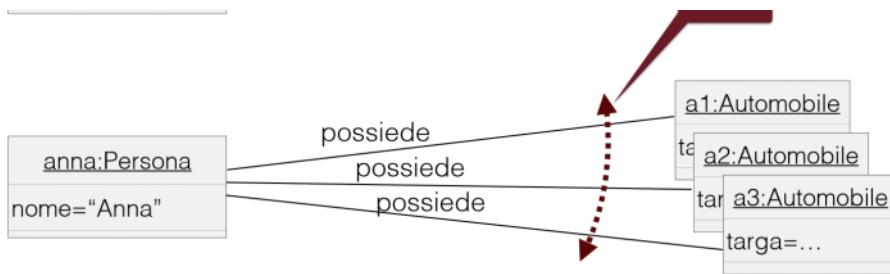
## Associazioni: vincoli di molteplicità (4)

- Semantica dei vincoli di molteplicità sui ruoli delle associazioni (cont.)



memoria la definizione dei vincoli "ogni istanza di impiegato ecc che va da 1 a 1"

ta è legato a



## Esercizio: prenotazioni di hotel

- Definire i vincoli di molteplicità sui ruoli delle associazioni del seguente diagramma delle classi



Ogni oggetto ist...  
A quanti link ho...

Ogni oggetto de...

A quanti link cli...

## Esercizio: prenotazioni di hotel

- Definire i vincoli di molteplicità sui ruoli delle associazioni del seguente diagramma delle classi
- Soluzione:



## Esercizio

- Si vuole progettare un sistema che gestisca i dati anagrafici delle persone
- Di ogni persona interessa il nome, il cognome, la data di nascita, la città di nascita e quella di

istanza della classe hotel a quanti link hotel\_prenotato può partecipare?  
hotel\_prenotato ogni istanza di hotel può partecipare? 0..n

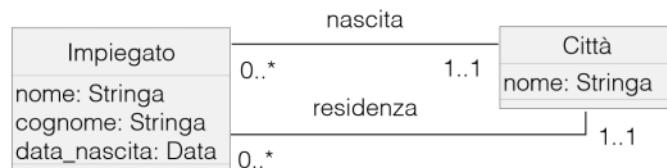
della classe prenotazione a quanti link cliente\_prenotazione può partecipare?

ente\_prenotazione può partecipare ogni istanza di persona?

- Si vuole progettare un sistema che gestisca i dati anagrafici delle persone
- Di ogni persona interessa il nome, il cognome, la data di nascita, la città di nascita e quella di residenza
- Si definisca un diagramma delle classi concettuale per l'applicazione

## Esercizio

- Si vuole progettare un sistema che gestisca i dati anagrafici delle persone
- Di ogni persona interessa il nome, il cognome, la data di nascita, la città di nascita e quella di residenza
- Si definisca un diagramma delle classi concettuale per l'applicazione



## Associazioni che insistono più volte sulla stessa classe

- Supponiamo di voler modellare i sovrani di un regno ormai scomparso
- Di ogni sovrano interessa il nome, il periodo in cui ha regnato, ed il predecessore



un sovrano non  
essere predecessore

- Come possiamo rappresentare il concetto di **predecessore**?

è predecessore/successore per sua natura, in relazione un altro sovrano può  
essere/essere successore

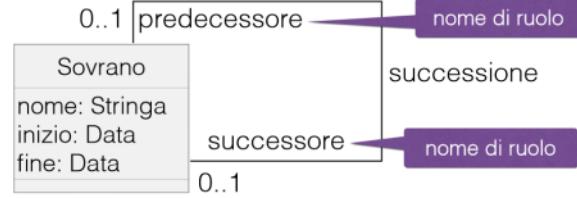


## Associazioni che insistono più volte sulla stessa classe (2)

- Supponiamo di voler modellare i sovrani di un regno ormai scomparso
- Di ogni sovrano interessa il nome, il periodo in cui ha regnato, ed il predecessore

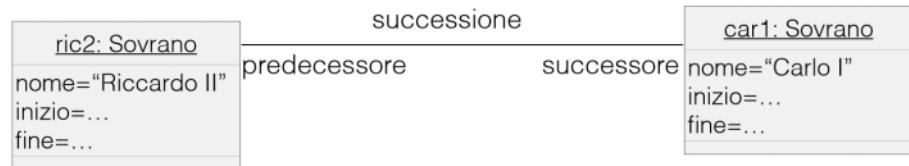
Livello delle classi e delle associazioni (intensionale)

- Sappiamo che le istanze dell'associazione sono coppie (s1, s2) di oggetti (istanze) di classe Sovrano.
- La classe Sovrano gioca due ruoli nell'associazione → UML ci obbliga a dare esplicitamente nomi distinti ai due ruoli
- Una istanza dell'associazione diventa una **coppia etichettata**: (predecessore:s1, successore:s2)
- L'ambiguità è sparita



sono obbligatori  
ogni oggetto deve avere  
che va da 0 a 1

Possibile livello degli  
oggetti e dei link  
(estensionale)



se ci fosse stato  
ogni oggetto sarebbe  
Ogni oggetto sarebbe

## Basi di Dati, Modulo 2 *Laurea in Informatica*

Parte: Analisi concettuale  
UNITÀ: A.1

Prof. Toni Mancini  
Dipartimento di Informatica

Versione del 2024-02-18



### A.1.1.4

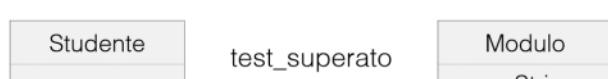
Analisi dei Requisiti  
Unified Modeling Language  
Diagrammi UML delle classi e degli oggetti  
Associazioni con attributi  
(association class)

**ri** i nomi dei ruoli quando si usa un associazione sulla classe stessa  
ella classe sovr partecipa Tipicon ruolo successore con un num. di link successione

o 1.\* sotto a successore  
avrano partecipa un num. di link successione con ruolo di predec che va da 1 a  $\infty$   
ovrano partecipa con un num. di link come successore che va da 0 a 1

## Associazioni con attributi

- Si vuole progettare un sistema che gestisca gli esiti (voti in trentesimi) dei test superati dagli studenti di un corso.
- Il corso è diviso in moduli e uno studente può superare il test di ogni modulo al più una volta.
- Si definisca un diagramma delle classi concettuale per l'applicazione



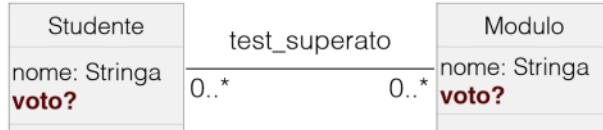
test\_superato non possono essere superati più di una volta

Voto non può essere superato più di una volta

modella il fatto che ogni studente può superare una certa volta  
esistere più link distinti che legano lo stesso studente allo stesso modulo

essere intrinseco ne con studente ne con modulo  
~~dal legame della cassetta studente a modulo~~

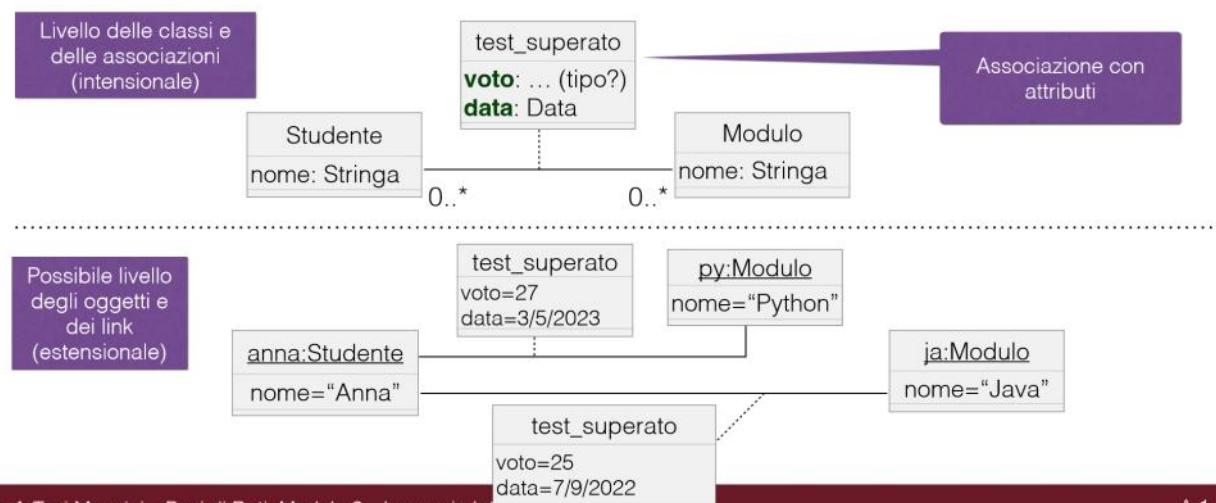
Voto non può essere una proprietà



- Come rappresentare i voti conseguiti dagli studenti nei test dei diversi moduli?
- Non possiamo aggiungere un attributo "voto" né nella classe Studente né nella classe Modulo!
- In effetti, un voto non è una proprietà locale di uno studente, né di un modulo, ma è una proprietà del legame tra uno studente ed un modulo... ovvero è una **proprietà dell'associazione**

## Associazioni con attributi (2)

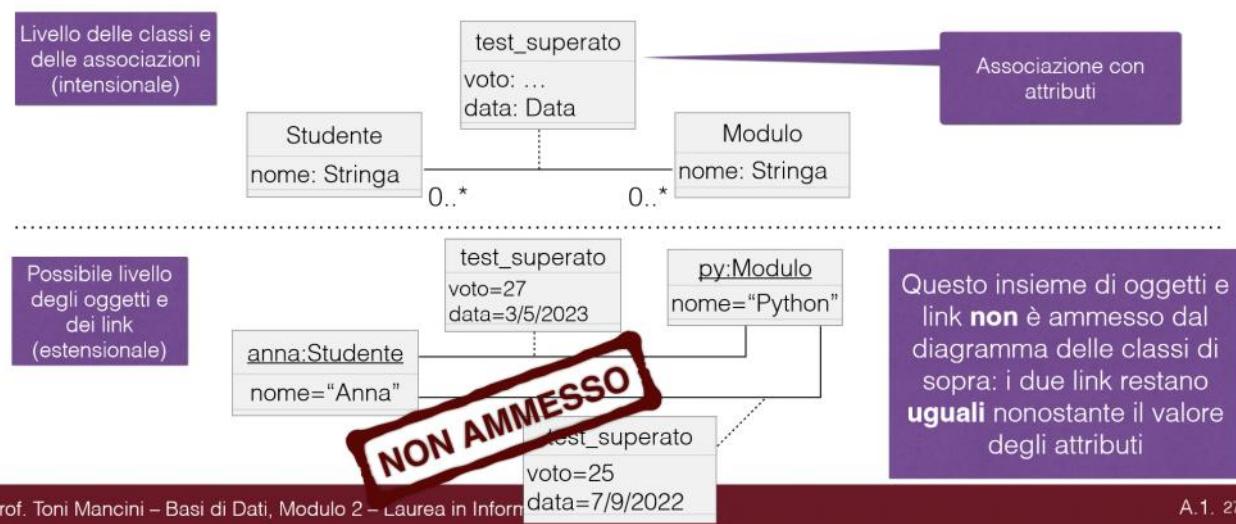
- Si vuole progettare un sistema che gestisca gli esiti (data e voto in trentesimi) dei test svolti dagli studenti di un corso.
- Il corso è diviso in moduli e uno studente può sostenere il test di ogni modulo al più una volta.



I'associazione con attributi lega con l'associazione

## Associazioni con attributi (3)

- **Attenzione:** anche in presenza di attributi, **la natura dell'associazione non cambia**:
  - il diagramma permette ad una stessa coppia di oggetti di formare **al più un link** dell'associazione



ovviamente, ciò

## Associazioni con attributi, anche dette association class

- Un'associazione UML con attributi è anche chiamata **association class**, in quanto può essere a sua volta terminale di ulteriori associazioni
- **La natura dell'associazione non cambia**:

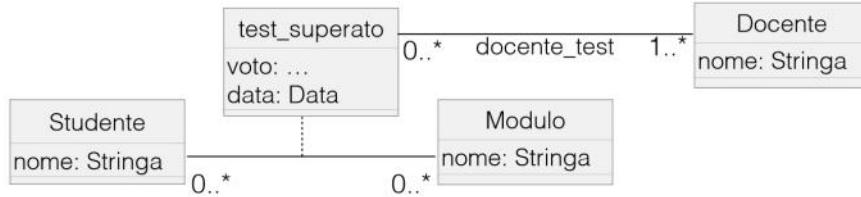
essere intrinseco ne con studente ne con modulo  
del legame dello oggetto studente e modulo

on gli attributi inizia con la lettera minuscola e ha una linea tratteggiata che la  
iazione

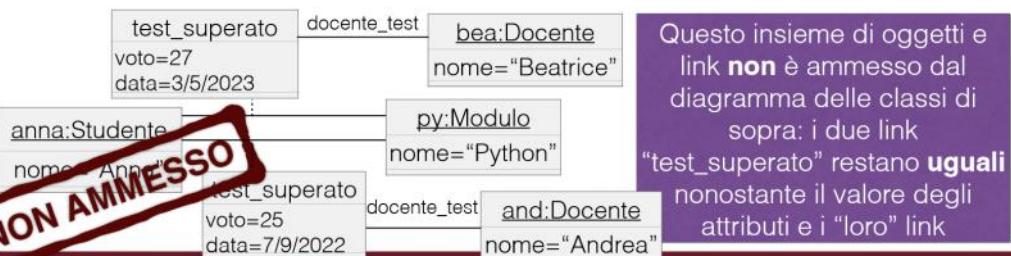
me prima, non si possono collegare più link tra due oggetti

- Un'associazione UML con attributi è anche chiamata **association class**, in quanto può essere a sua volta terminale di ulteriori associazioni
- **La natura dell'associazione non cambia:**
  - il diagramma permette ad una stessa coppia di oggetti di formare **al più un link** dell'associazione

Livello delle classi e delle associazioni (intensionale)



Possibile livello degli oggetti e dei link (estensionale)



Prof. Toni Mancini – Basi di Dati, Modulo 2 – Laurea in Informatica

A.1. 28

## Basi di Dati, Modulo 2 *Laurea in Informatica*

Parte: Analisi concettuale  
UNITÀ: A.1

Prof. Toni Mancini  
Dipartimento di Informatica

Versione del 2024-02-18



## Tipi di dato concettuali: tipi base

- Durante l'analisi concettuale dobbiamo definire il tipo di ogni attributo, di classe e di associazione
- Ricordiamoci che, in analisi, non vogliamo effettuare scelte tecnologiche (ad es., sul linguaggio di programmazione)
- Dunque, vogliamo utilizzare **tipi di dato concettuali**, che siano facilmente realizzabili con qualsivoglia tecnologia informatica (Python, Java, sistemi di gestione di basi di dati, etc.)
- **Tipi base:**
  - Intero, Reale, Booleano, Data, Ora, DataOra
- Vogliamo però anche essere certi di modellare la realtà in modo accurato!



SAPIENZA  
UNIVERSITÀ DI ROMA

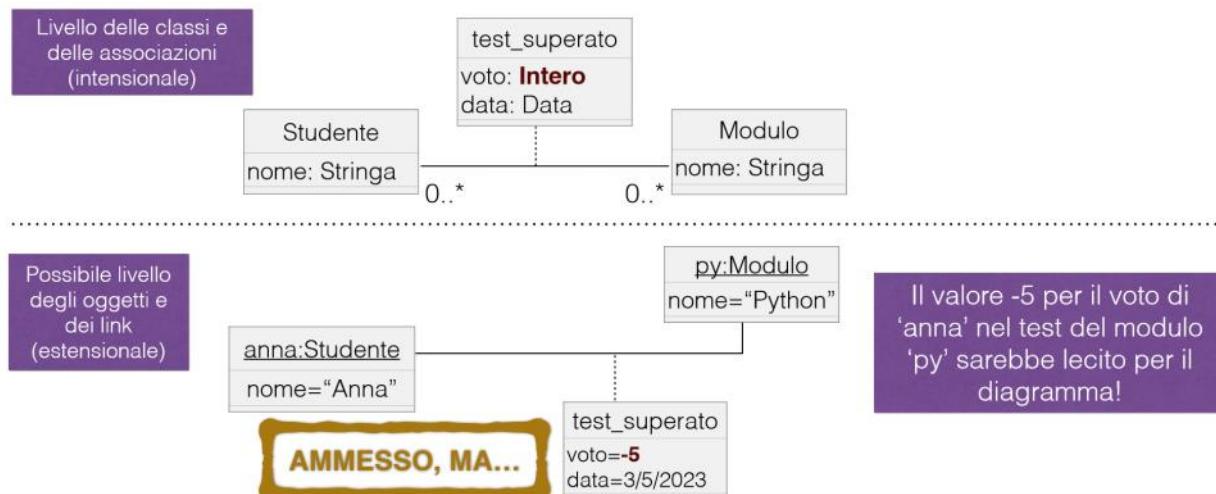
### A.1.1.5

Analisi dei Requisiti  
Unified Modeling Language  
Diagrammi UML delle classi e degli oggetti  
Tipi di dato concettuali



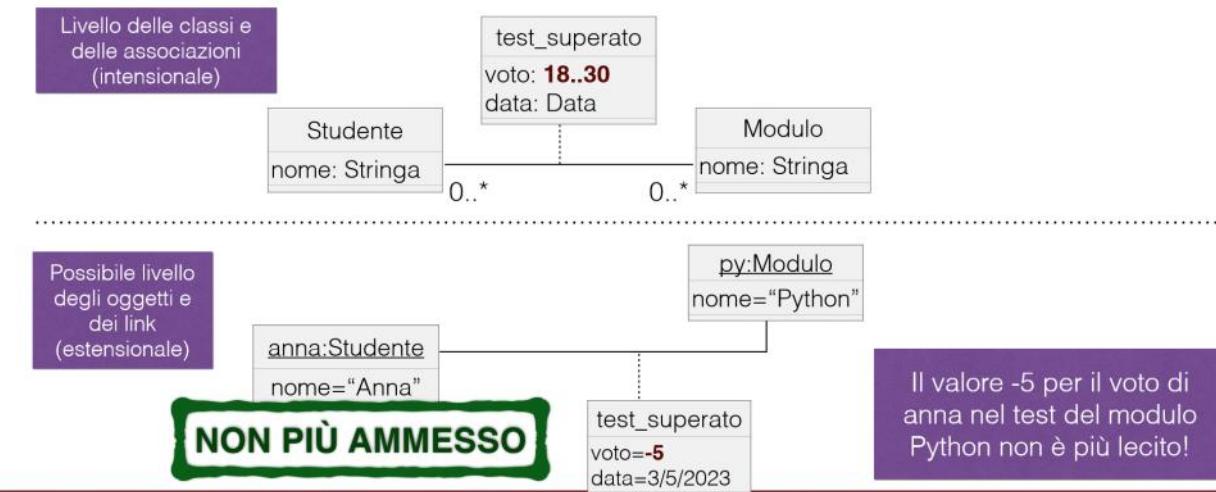
## Tipi di dato concettuali: tipi specializzati

- ...Vogliamo però anche essere certi di modellare la realtà in modo accurato!



## Tipi di dato concettuali: tipi specializzati (2)

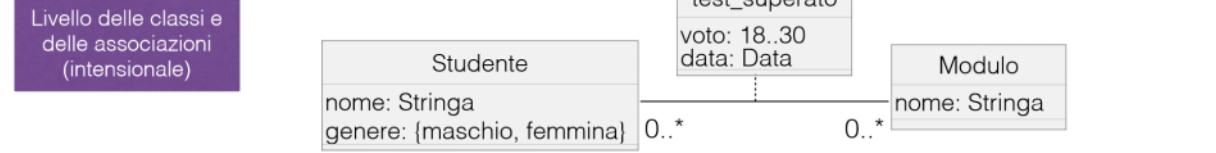
- Vogliamo però anche essere certi di modellare la realtà in modo accurato!
- Tipi di dato specializzati:
  - Intero > 0, Reale <= 0, etc., tipo intervallo (di interi): 18..30, 0..100, etc.



## Tipi di dato concettuali: tipi enumerativi

- Quando l'insieme dei possibili valori di un attributo è finito e piccolo, possiamo usare un **tipo di dato enumerativo**, che definisce esplicitamente e completamente l'insieme dei valori possibili per l'attributo.  
Ad esempio:

- {maschio, femmina}, {Africa, America, Antartide, Asia, Europa, Oceania}, etc.

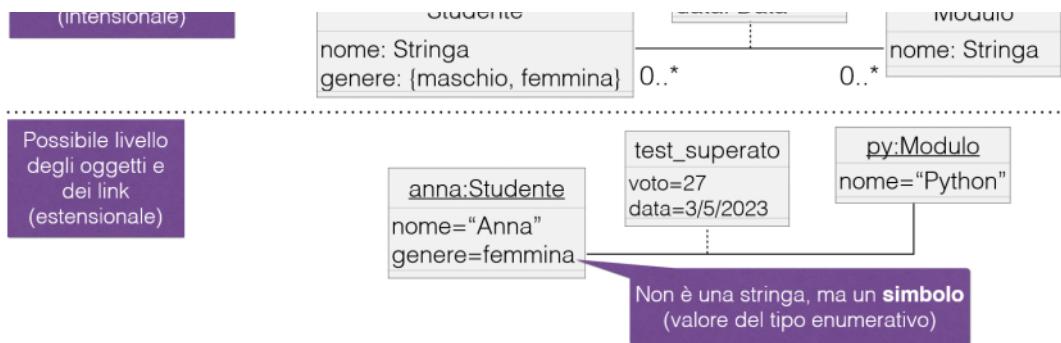


fare molta attenzione  
bisogna essere

zione sull'uso del tipo enumerativo  
sicuri che i tipi siano quelli e non mutino nel tempo

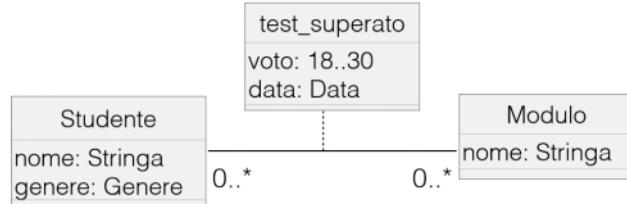
tare molto atten  
bisogna essere

sono etichette  
il dominio enum



## Tipi di dato definiti dall'utente

- UML consente all'analista di definire nuovi tipi di dato, che potranno essere usati liberamente nello schema concettuale.

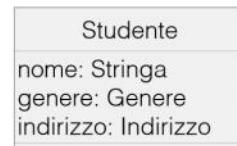


- Tipo Genere = {maschio, femmina}

## Tipi di dato composti

- UML consente all'analista di definire anche **tipi di dato composti** da più campi: **tipi record**

- Tipo Genere = {maschio, femmina}
- Tipo Indirizzo = (via:stringa, civico:intero>0, cap:intero>0)  
(Attenzione: la scelta di usare il tipo 'intero>0' per i campi 'civico' e 'cap' non è affatto adeguata: è solo un semplice esempio!)



## Vincoli di molteplicità sugli attributi

- Anche gli attributi di classe e associazione possono avere vincoli di molteplicità (default: 1..1)

i vincoli di molte

nzione sull'uso del tipo enumerativo

sicuri che i tipi siano quelli e non mutino nel tempo

non stringhe "non hanno le virgolette"

enumerativo usa le graffe

implicità sugli attributi sono con le quadre

- Anche gli attributi di classe e associazione possono avere vincoli di molteplicità (default: 1..1)

- Ogni studente ha associato esattamente un nome e un genere
- Ogni studente ha associato **uno o più** indirizzi email
- Ogni studente ha associato **al più un** indirizzo

Studente
nome: Stringa
genere: Genere
email: Stringa [1..*]
indirizzo: Indirizzo [0..1]

<u>anna:Studente</u>
nome="Anna"
genere=femmina
email={"anna@kmail.com", "anna2002@yahoo.com"}
indirizzo={}

## Basi di Dati, Modulo 2 *Laurea in Informatica*

Parte: Analisi concettuale  
UNITÀ: A.1

Prof. Toni Mancini  
Dipartimento di Informatica

Versione del 2024-02-18



**SAPIENZA**  
UNIVERSITÀ DI ROMA

### A.1.6

Analisi dei Requisiti  
Unified Modeling Language  
Diagrammi UML delle classi e degli oggetti  
Vincoli di identificazione di classe

- In alcuni casi, per modellare correttamente il dominio applicativo, è necessario imporre alcuni ulteriori vincoli oltre a quelli naturalmente imposti dal diagramma delle classi
- Vincolo (di integrità): una asserzione che impone restrizioni all'insieme dei livelli estensionali ammessi (ovvero agli insiemi di oggetti e link che possono coesistere) ulteriori a quelle strutturali che provengono dal diagramma delle classi
  - Abbiamo già visto i vincoli di molteplicità sulle associazioni
- Ulteriore tipologia di vincolo:  
**vincolo di identificazione di classe**
  - Impone che non possono coesistere oggetti di una classe che coincidono nel valore di un insieme di attributi e/o sono collegati tramite link agli stessi oggetti di altre classi
  - Esempio: Non possono esistere due persone con lo stesso codice fiscale ("{id1}") e non possono esistere persone con, simultaneamente, lo stesso nome, cognome e data di nascita ("{id2}")

Persona
cf: CodiceFiscale {id1}
nome: Stringa {id2}
cognome: Stringa {id2}
nascita: Data {id2}

{id2} definisce un unico  
vincolo su tre attributi

i vincoli di molti  
Ogni oggetto de

Non possono esistere due persone con lo stesso codice fiscale  
Però possono esistere persone con lo stesso nome, cognome e data di nascita

esistere due persone con stesso CF o con la stessa coppia nome,cognome,data  
e-mail

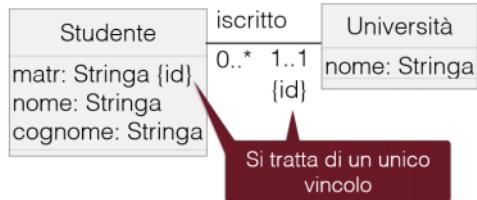
esistere due persone con lo stesso nome o con lo stesso cognome (solo se hanno  
cognome o nome diverso)

codice fiscale ("{id1}") e non possono esistere persone con, simultaneamente, lo stesso nome, cognome e data di nascita ("{id2}")

{id2} definisce un unico vincolo su tre attributi

- Un vincolo di identificazione di classe può coinvolgere anche **ruoli** della classe

- Esempio: Non possono esistere due studenti con la stessa matricola nella stessa università



- **Attenzione:** un vincolo di identificazione di classe può coinvolgere solo **attributi a molteplicità [1..1]** e/o **ruoli della classe a molteplicità 1..1**

## Basi di Dati, Modulo 2 Laurea in Informatica

Parte: Analisi concettuale  
UNITÀ: A.1

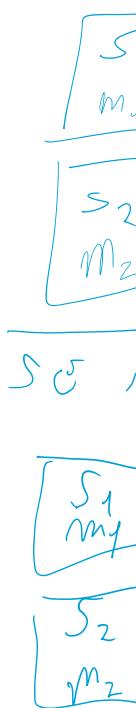
Prof. Toni Mancini  
Dipartimento di Informatica

Versione del 2024-02-18



### A.1.1.7

Analisi dei Requisiti  
Unified Modeling Language  
Diagrammi UML delle classi e degli oggetti  
Generalizzazione



## Relazioni is-a tra classi

- Fino ad ora abbiamo implicitamente assunto che **classi diverse non hanno istanze in comune**
- In molte situazioni, vogliamo rappresentare il fatto che tra due classi sussista una relazione di **sottoinsieme**
- I diagrammi delle classi permettono di definire il concetto di **relazione is-a tra classi**



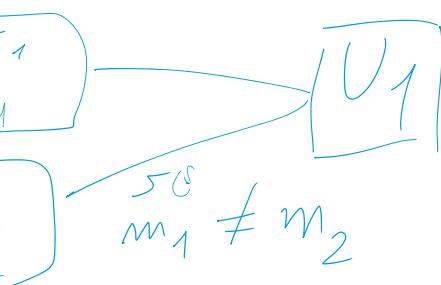
- **Relazione is-a ("è anche un"):** ogni istanza della classe Studente (sotto-classe) è (concettualmente!) **anche** un'istanza della classe Persona (super-classe)

licano allo stesso vincolo {id}

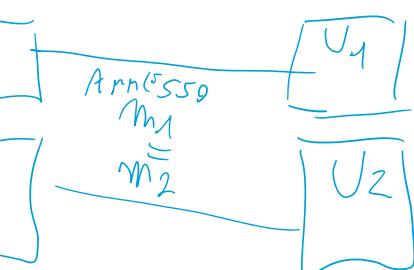
o esistere due istanze di studente che hanno lo stesso val di attributi matricola e che  
link iscritto ai quali partecipa lo stesso oggetto di università

o esistere due matricole uguali per due università diverse

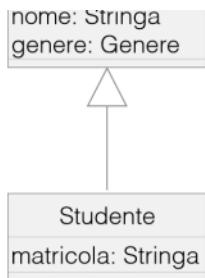
o esistere due matricole degli studenti uguali per la stessa università



$m_1 = m_2$  SONO 2 UNI DISTINTE



superclasse

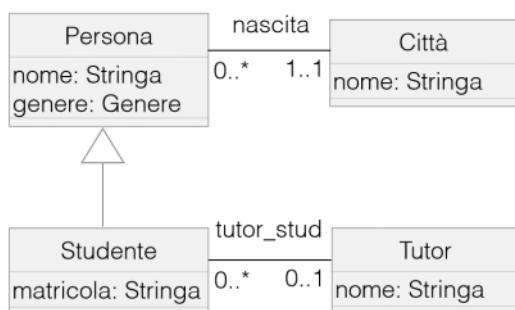


**Relazione is-a** è una relazione di sottoinsieme. Ogni istanza della classe Studente (sotto-classe) è (concretamente!) **anche** un'istanza della classe Persona (super-classe)

- Ovviamente **non vale il viceversa**: non tutte le istanze di Persona devono per forza essere anche istanze di Studente

## Relazioni is-a tra classi: ereditarietà

- Fino ad ora abbiamo implicitamente assunto che classi diverse non hanno istanze in comune
- In molte situazioni, vogliamo rappresentare il fatto che tra due classi sussista una relazione di sottoinsieme
- I diagrammi delle classi permettono di definire il concetto di **relazione is-a** tra classi

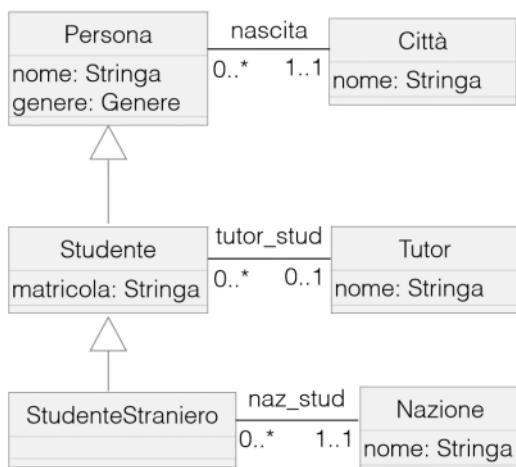


In presenza di relazioni is-a, vige il **meccanismo dell'ereditarietà** (ricorda il **sillogismo aristotelico**)

- Di tutte le persone di interesse vogliamo rappresentare nome, genere e città di nascita
- Di tutti gli studenti vogliamo rappresentare la matricola e l'eventuale tutor
- Tutti gli studenti sono persone (relazione is-a)  
→ **Di conseguenza**, di tutti gli studenti **stiamo rappresentando anche** nome, genere, e città di nascita (con le loro molteplicità 1..1)

## Relazioni is-a tra classi: ereditarietà (2)

- Fino ad ora abbiamo implicitamente assunto che classi diverse non hanno istanze in comune
- In molte situazioni, vogliamo rappresentare il fatto che tra due classi sussista una relazione di sottoinsieme
- I diagrammi delle classi permettono di definire il concetto di relazione is-a tra classi



Ovviamente possiamo avere **relazioni is-a a più livelli: transitività!**

- Di tutte le persone di interesse vogliamo rappresentare nome, genere e città di nascita
- Di tutti gli studenti vogliamo rappresentare la matricola e l'eventuale tutor
- Di tutti gli studenti stranieri vogliamo rappresentare la nazione di provenienza
- Tutti gli studenti sono persone (relazione is-a)  
→ **Di conseguenza**, di tutti gli studenti **stiamo rappresentando anche** nome, genere, ed città di nascita (con le loro molteplicità 1..1)
- Tutti gli studenti stranieri sono studenti (relazione is-a)  
→ **Di conseguenza**, degli studenti stranieri **stiamo rappresentando anche** nome, genere, matricola, città di nascita (con le loro molteplicità 1..1)

## Classi più specifiche di un oggetto

- Classi più specifiche** di un oggetto O:
  - L'insieme delle classi di cui O è istanza che non sono a loro volta superclassi di altre classi di O
  - Esempio:

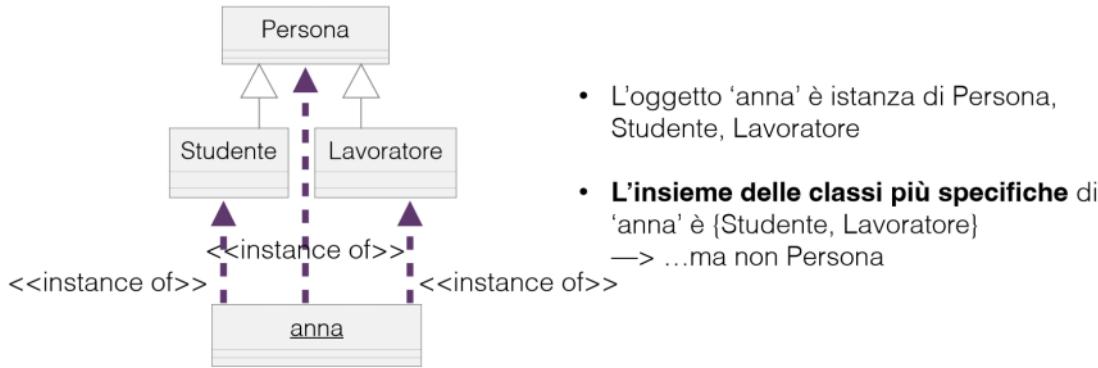
È l'insieme delle classi di cui l'oggetto

e classi di cui l'oggetto è istanza che non sono a loro volta superclassi di altre  
oggetto è istanza

- L'insieme delle classi di cui O è istanza che non sono a loro volta superclassi di altre classi di O
- Esempio:

È l'insieme delle classi di cui l'oggetto

Se c'è un eredità

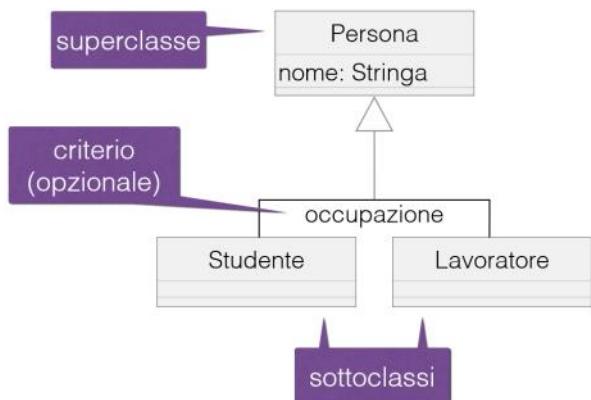


## Generalizzazioni

- I diagrammi delle classi UML offrono un costrutto più complesso della relazione is-a: il costrutto della **generalizzazione**
- Permette di definire che le istanze di una classe possono essere istanze di più classi figlie **secondo uno stesso criterio concettuale**

POWA / S<sub>1</sub>

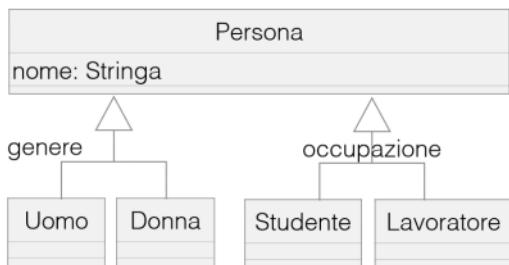
quindi il fatto d  
allora S<sub>1</sub> (che è :  
Questo ragiona



- Significato:
  - Studente **is-a** Persona
  - Lavoratore **is-a** Persona
  - Il **criterio** secondo il quale le Persone sono considerate studenti e/o lavoratori è **lo stesso**: quello dell'“occupazione”
- È ancora possibile per un oggetto essere istanza sia di Studente che di Lavoratore

## Generalizzazioni (2)

- La stessa classe può essere superclasse di generalizzazioni distinte (criteri diversi)

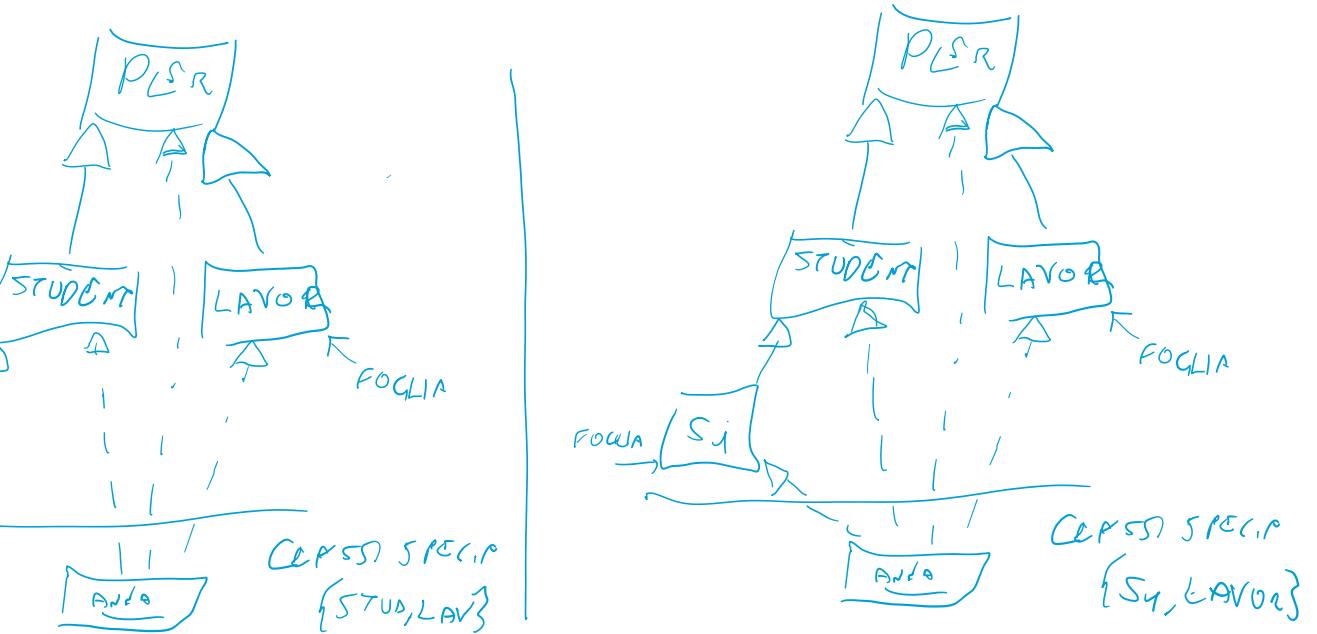


- Significato:
  - Secondo il **criterio** del genere, le Persone sono considerate uomini e/o donne
  - Secondo il **criterio (indipendente!)** dell'occupazione, le persone sono considerate studenti e/o lavoratori

- Una istanza di Persona può essere sia istanza di Uomo che di Studente? Sì
- Una istanza di Persona può essere istanza né di Uomo né di Donna? Sì
- Una istanza di Persona può essere istanza sia di Uomo che di Donna? Sì

e classi di cui l'oggetto è istanza che non sono a loro volta superclassi di altre  
oggetto è istanza

aerietà ad albero, le classi più specifiche sono le foglie dell'albero (CAZZATA)

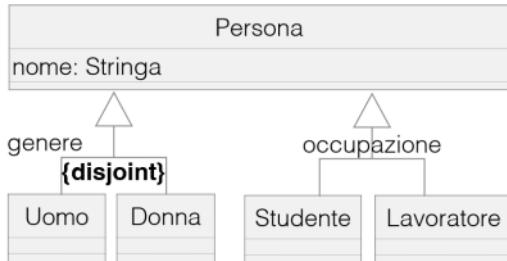


elle foglie dell'ereditarietà non è valido perché se anna è istanza di studente  
(foglia) non è una classe specifica di anna ma Studente si  
mento funziona solo se andiamo a selezionare le classi di cui l'oggetto è istanza

- Una istanza di Persona può essere istanza né di Uomo né di Donna? Sì
- Una istanza di Persona può essere istanza sia di Uomo che di Donna? Sì

## Vincoli sulle generalizzazioni: {disjoint}

- In linea di principio, una istanza della classe base può essere istanza di più di una sottoclassificazione di una stessa generalizzazione
- Spesso, nell'ottica di modellare accuratamente i requisiti, vorremmo evitarlo: **generalizzazioni disgiunte**.

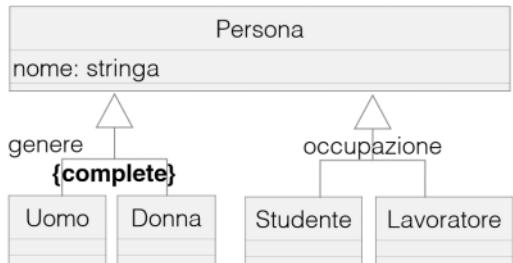


- Significato:
  - La generalizzazione sul genere è disgiunta: una istanza di Uomo non può essere anche istanza di Donna
  - La generalizzazione sull'occupazione non è disgiunta: una istanza di Studente può essere anche istanza di Lavoratore (ma...)

- Una istanza di Persona può essere sia istanza di Uomo che di Studente? Sì
- Una istanza di Persona può essere istanza né di Uomo né di Donna? Sì
- Una istanza di Persona può essere istanza sia di Uomo che di Donna? **Non più**

## Vincoli sulle generalizzazioni: {complete}

- In linea di principio, una istanza della classe base può essere istanza di nessuna delle sottoclassificazioni di una stessa generalizzazione
- Spesso, nell'ottica di modellare accuratamente i requisiti, vorremmo evitarlo: **generalizzazioni complete**.

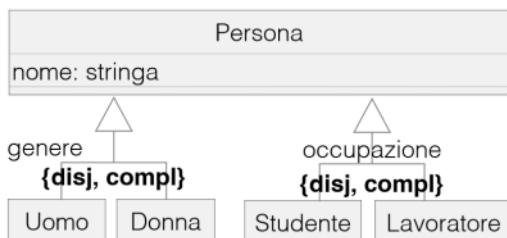


- Significato:
  - La generalizzazione sul genere è completa: ogni istanza di Persona deve essere anche istanza di almeno una sottoclassificazione tra Uomo e Donna
  - La generalizzazione sull'occupazione non è completa: possono esistere istanze di Persona che non sono né istanze di Studente né istanze di Lavoratore

- Una istanza di Persona può essere sia istanza di Uomo che di Studente? Sì
- Una istanza di Persona può essere istanza né di Uomo né di Donna? **Non più**
- Una istanza di Persona può essere istanza sia di Uomo che di Donna? Sì

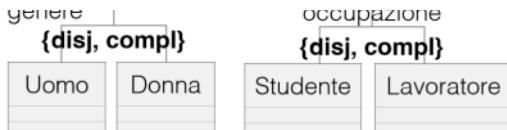
## Vincoli sulle generalizzazioni: {disjoint, complete}

- Una generalizzazione può essere **sia disgiunta che completa**
- Ogni istanza della superclasse è anche istanza di **esattamente una** sottoclassificazione della generalizzazione



- Significato:
  - ogni istanza di Persona deve essere anche istanza di esattamente una sottoclassificazione tra Uomo e Donna
  - ogni istanza di Persona deve essere anche istanza di esattamente una sottoclassificazione tra Studente e Lavoratore

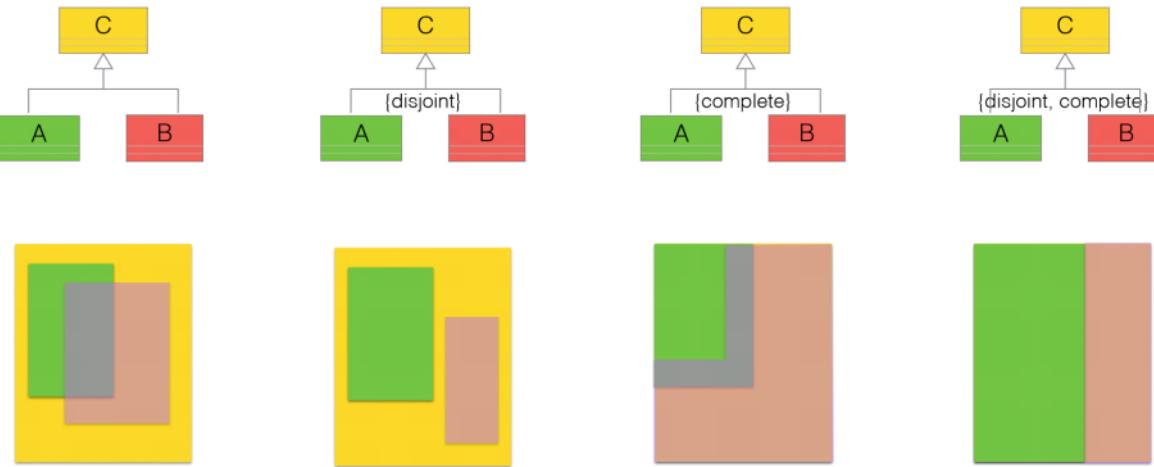




- ogni istanza di Persona deve essere anche istanza di esattamente una sottoclasse tra Studente e Lavoratore

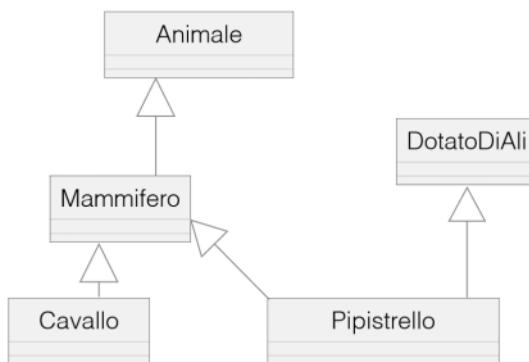
- Una istanza di Persona può essere sia istanza di Uomo che di Studente? Sì
- Una istanza di Persona può essere istanza né di Uomo né di Donna? **Non più**
- Una istanza di Persona può essere istanza sia di Uomo che di Donna? **Non più**

## Vincoli sulle generalizzazioni: riepilogo



## Ereditarietà multipla

- Una classe può essere sottoclasse di più classi. Il meccanismo dell'ereditarietà vale come sempre.





# Basi di Dati, Modulo 2

## Laurea in Informatica

Parte: Analisi concettuale  
UNITÀ: A.1

Prof. Toni Mancini  
Dipartimento di Informatica

Versione del 2024-02-18



SAPIENZA  
UNIVERSITÀ DI ROMA

### A.1.1.8

Analisi dei Requisiti  
Unified Modeling Language  
Diagrammi UML delle classi e degli oggetti  
Operazioni di classe



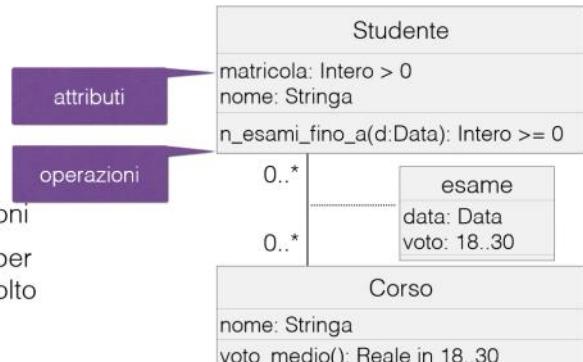
## Operazioni di classe



- Finora abbiamo fatto riferimento solo a proprietà statiche (attributi e associazioni) di classi.
  - Proprietà statiche: i valori cambiano a seguito di esplicita modifica dei dati da parte degli utenti del sistema (o da parte di altre parti del sistema)
- Una classe UML può definire anche proprietà dinamiche, che si chiamano operazioni.
  - Proprietà dinamiche: i valori vengono calcolati ogni volta che servono, a partire dai valori di altre proprietà

### Operazione di classe

- Una operazione della classe C indica che su ogni oggetto (istanza) della classe C si può eseguire un calcolo per:
  - calcolare un valore a partire da altri dati e operazioni
  - effettuare cambiamenti di stato dell'oggetto (cioè per modificare le sue proprietà), dei link in cui è coinvolto e/o degli oggetti a questo collegati



Prof. Toni Mancini – Basi di Dati, Modulo 2 – Laurea in Informatica

A.1. 53

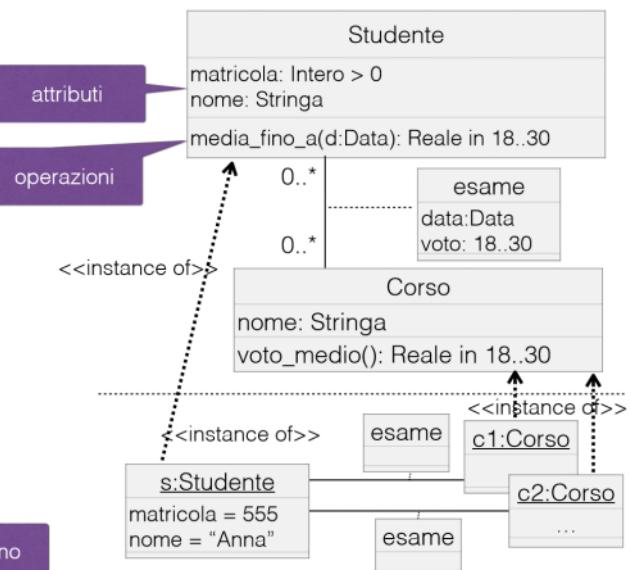


## Operazioni: sintassi

Sintassi per la segnatura dell'operazione:

- nome\_operazione(argomenti) : tipo\_ritorno**  
dove:
  - “**argomenti**” è una lista di elementi della forma:  
**nome\_argomento : tipo\_argomento**
  - “**tipo\_ritorno**” è il tipo del valore restituito dall'operazione
  - I tipi degli argomenti e del valore di ritorno possono essere tipi di dato concettuali oppure classi del diagramma
- Una operazione di classe può essere invocata **solo** su un oggetto della classe, che è un ulteriore argomento (non indicato nella segnatura)

s.media\_fino\_a(2/6/2023) —> 22.5  
oggetto di invocazione | valore degli argomenti | valore di ritorno



Prof. Toni Mancini – Basi di Dati, Modulo 2 – Laurea in Informatica

A.1. 54



## Operazioni: ereditarietà

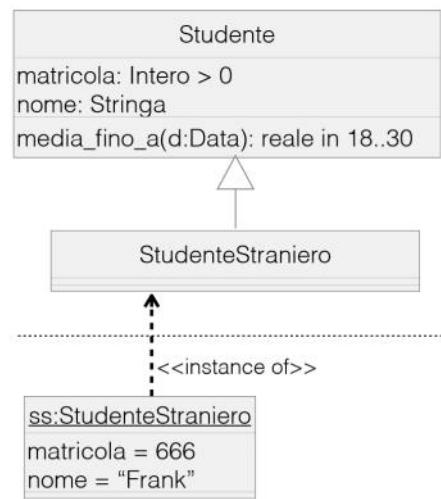
- Il meccanismo dell'ereditarietà si applica anche alle operazioni di classe
- Dunque, ad es., possiamo invocare:

ss.media\_fino\_a(2/6/2023) —> 28.3

oggetto di invocazione (della sottoclasse)

valore degli argomenti

valore di ritorno



## Operazioni: specifiche

- Il diagramma delle classi non definisce cosa calcolano le operazioni, né se e come modificano i dati
- Ogni classe del diagramma con operazioni andrà affiancata da un documento di specifica che entra nel dettaglio (v. seguito)





## A.1.1.9

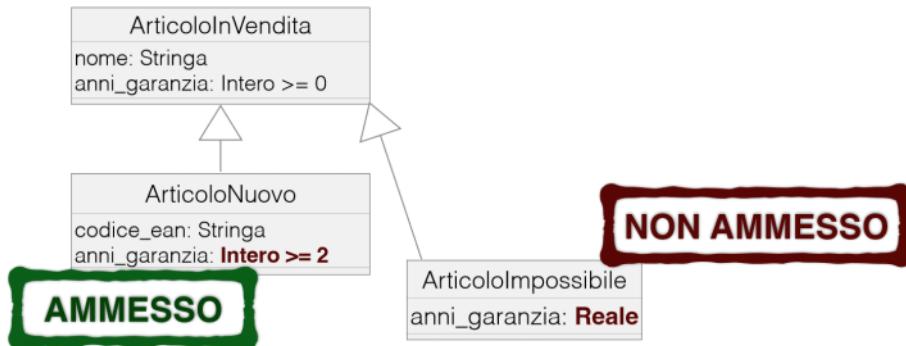
Analisi dei Requisiti  
 Unified Modeling Language  
 Diagrammi UML delle classi e degli oggetti  
**Specializzazioni di attributi,  
 associazioni ed operazioni**



## Specializzazione di attributi

- In una generalizzazione, una sottoclasse non solo può avere proprietà aggiuntive rispetto alla superclasse, ma può anche **specializzare** le proprietà ereditate dalla superclasse, **restringendone il tipo**
- Degli articoli in vendita vanno rappresentati: nome e numero di anni di garanzia (anche zero)
- Alcuni articoli sono nuovi
- Degli articoli nuovi interessa anche il codice EAN
- Inoltre, per gli articoli nuovi, la garanzia deve essere di **almeno due anni**

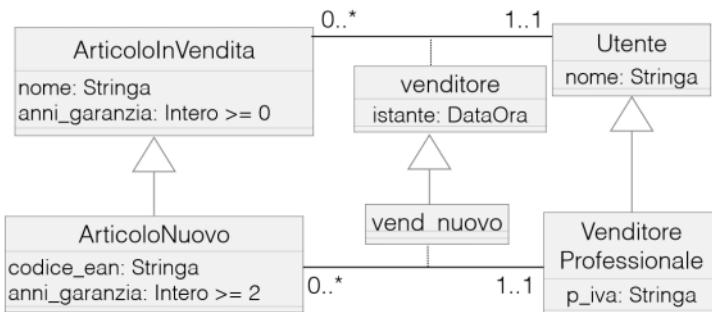
**Attenzione:** il tipo dell'attributo specializzato (ArticoloNuovo.anni\_garanzia) deve essere **più ristretto** del tipo dell'attributo che specializza



## Specializzazione di associazioni

- Ricordiamo che un'associazione con attributi si chiama “association class” e dunque... è anche una classe, e dunque... può a sua volta essere terminale di associazioni
  - Una association class, in quanto classe, può anche essere radice di relazioni is-a e generalizzazioni!
- Ogni articolo è venduto da **esattamente** un utente
- Gli articoli nuovi possono essere venduti **solo** da venditori professionali

**Attenzione:** l'associazione vend\_nuovo deve essere di tipo compatibile con il tipo dell'associazione venditore



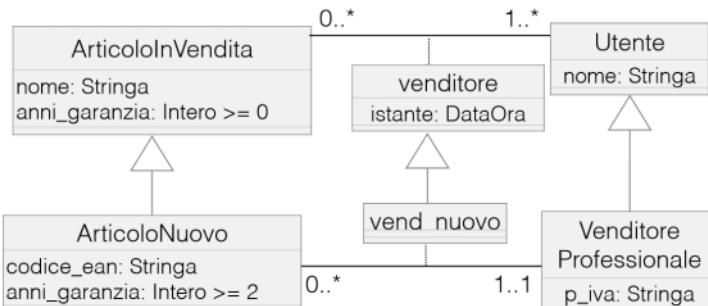
## Specializzazione di associazioni (2)

- Ricordiamo che un'associazione con attributi si chiama “association class” e dunque... è anche una classe, e dunque... può a sua volta essere terminale di associazioni
  - Una association class, in quanto classe, può anche essere radice di relazioni is-a e generalizzazioni!



- Una associazione class, in quanto classe, può anche essere radice di relazioni is-a e generalizzazioni!
- Ogni articolo è venduto da **almeno un** utente
- Gli articoli nuovi devono essere venduti da **almeno un** venditore professionale

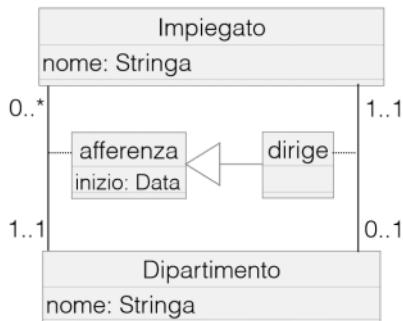
**Attenzione:** l'associazione *vend\_nuovo* deve essere di tipo compatibile con il tipo dell'associazione *venditore*



## Specializzazione di associazioni (3)

Esempio:

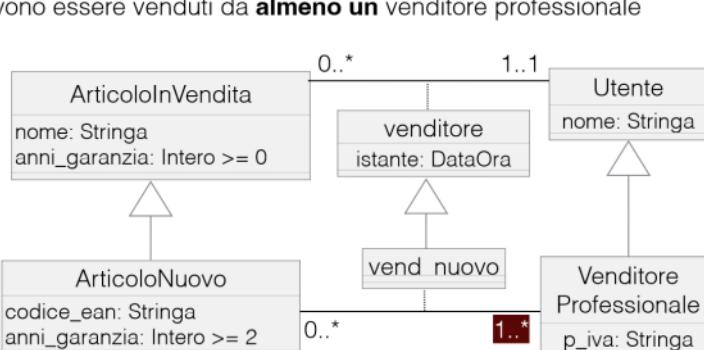
- Il sistema deve rappresentare impiegati, il dipartimento a cui afferisce ogni impiegato (con data di inizio afferenza) e direttore (un impiegato).
- I direttori devono afferire al dipartimento che dirigono.



## Specializzazione di associazioni: condizioni per la correttezza

Esempio:

- Ogni articolo è venduto da **esattamente un** utente
- Gli articoli nuovi devono essere venduti da **almeno un** venditore professionale



Il vincolo **1..\*** è **fouorviante**. In realtà diagramma implica che ogni articolo nuovo è venduto da **un solo** venditore professionale!

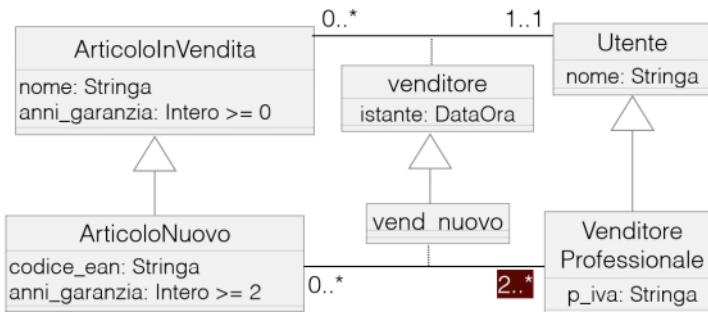




Le specializzazioni di associazioni necessitano di particolare attenzione relativamente ai vincoli di molteplicità.

Esempio:

- Ogni articolo è venduto da **esattamente un** utente
- Gli articoli nuovi devono essere venduti da **almeno due** venditori professionali



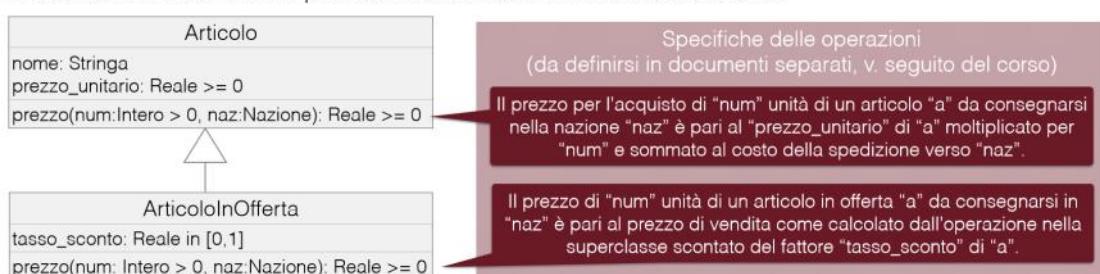
Il vincolo 2..\* comporta che non possano esistere articoli nuovi!



Anche le operazioni di classe possono essere oggetto di specializzazione nelle sottoclassi

Esempio:

- Di tutti gli articoli in vendita va calcolato il prezzo, che è uguale al prezzo unitario moltiplicato per il numero di pezzi richiesto, ed incrementato delle spese di spedizione (che dipendono dalla nazione dove la merce sarà consegnata)
- Alcuni articoli sono in offerta. Il loro prezzo è scontato di un certo fattore di sconto.



Come per la specializzazione di attributi, la segnatura dell'operazione nella sottoclasse deve essere **compatibile** con quella dell'operazione definita nella superasse, ovvero:

- Stesso numero e tipo di argomenti
- Il tipo di ritorno dell'operazione nella sottoclasse è dello stesso tipo o un sotto-tipo di quello dell'operazione nella superclasse.





## A.1.2

### Analisi dei Requisiti Unified Modeling Language Diagrammi degli use-case

## Diagramma UML degli use-case

Modellano le **funzionalità** che il sistema deve realizzare, in termini di **use-case** (scenari di utilizzo)

### Use-case

Cattura un **insieme omogeneo di funzionalità** accedute da un **gruppo omogeneo di utenti**.

Tipicamente coinvolge concetti rappresentati da più classi e associazioni del diagramma delle classi.

### Attore

Ruolo che un utente (umano o sistema esterno) gioca interagendo con il sistema.

Lo stesso utente può essere rappresentato da più attori (può giocare più ruoli).

Più utenti possono essere rappresentati dallo stesso attore.

## Diagramma UML degli use-case

### Un diagramma UML degli use-case è un grafo in cui:

- i **nodi** rappresentano attori e use-case
- gli **archi** rappresentano:
  - la possibilità per un attore di invocare uno use-case
  - la possibilità per uno use-case di invocare un altro use-case
  - la generalizzazione tra attori e tra use-case

Definizione: cosa è un grafo?

Una struttura molto comune in informatica. Rappresenta una **rete** di elementi, chiamati **nodi**, collegati a coppie da **archi**

- Esempio: mappa di una rete di metropolitane:

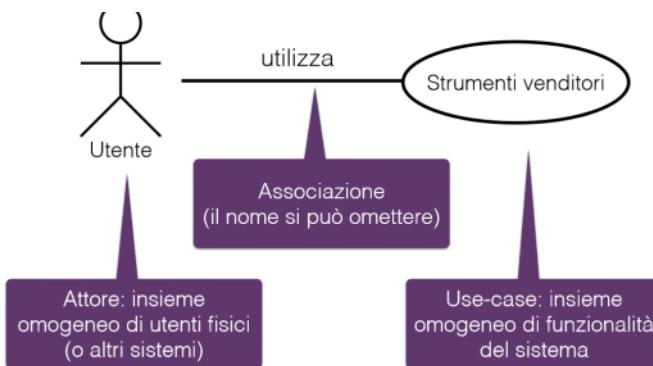


## Diagramma degli use-case: associazione

- Modella la possibilità di accesso, da parte di un attore, alle funzionalità di uno use-case



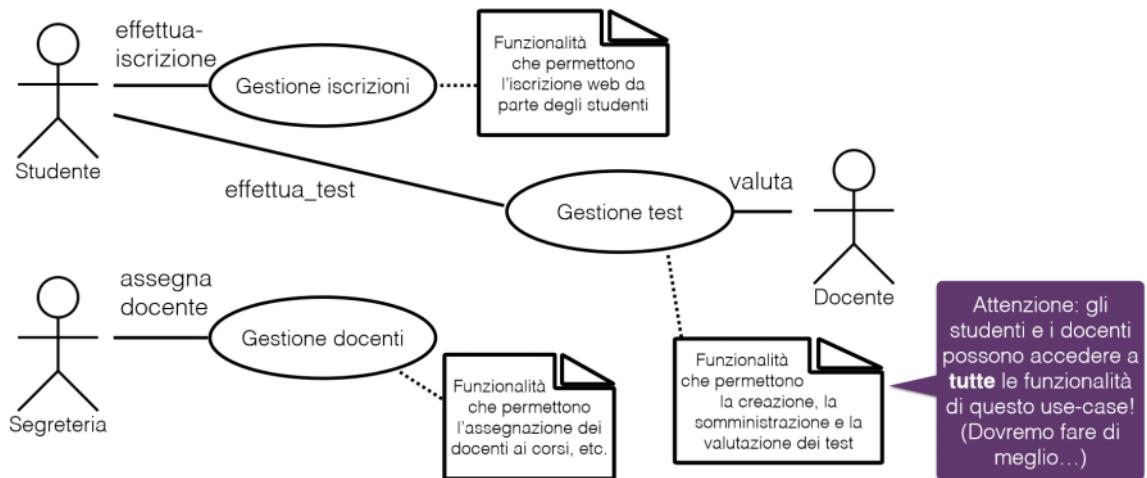




**Attenzione.** L'esistenza dell'attore Utente non implica l'esistenza della classe Utente nel diagramma delle classi. Avremo la classe Utente **solo se** il sistema deve rappresentare **dati** sugli utenti.

## Esempio

Il sistema deve permettere agli studenti di iscriversi, via web, ai corsi offerti. La segreteria deve poter assegnare i docenti ai singoli corsi. I docenti devono poter inserire i risultati dei test degli studenti: tali test sono somministrati agli studenti utilizzando il sistema.

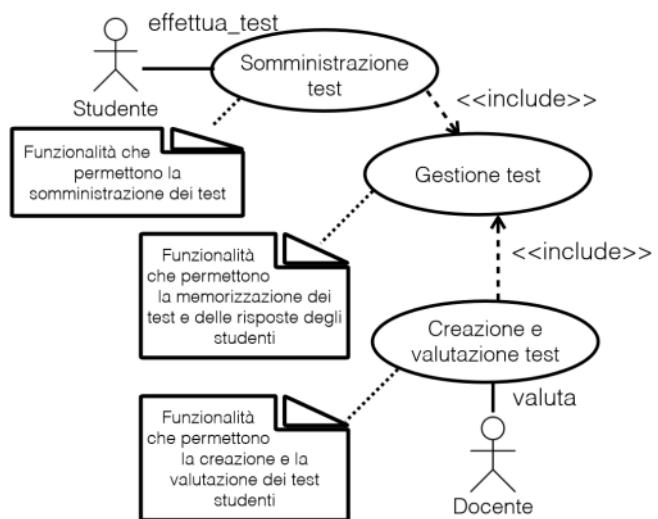


## Dipendenze tra use-case: inclusione

- Alcune funzionalità dello use-case A hanno bisogno di usare alcune funzionalità dello use-case B

### Esempio:

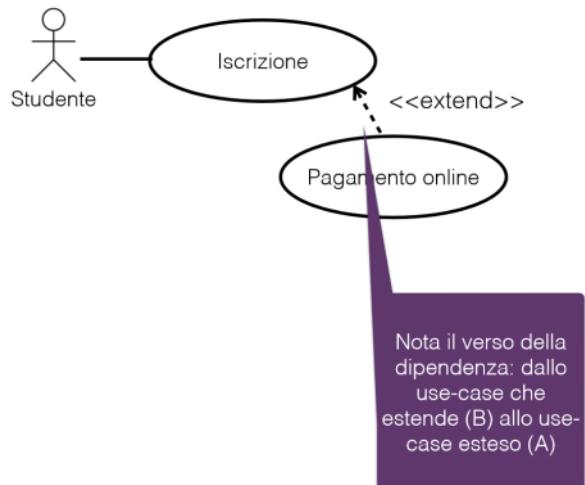
- i docenti possono creare e valutare i test degli studenti
- gli studenti possono rispondere ai test
- i test e le risposte degli studenti vanno memorizzati nel sistema





## Dipendenze tra use-case: estensione

- Alcune funzionalità dello use-case A, solo in alcuni casi particolari, sono estese con le funzionalità dello use-case B

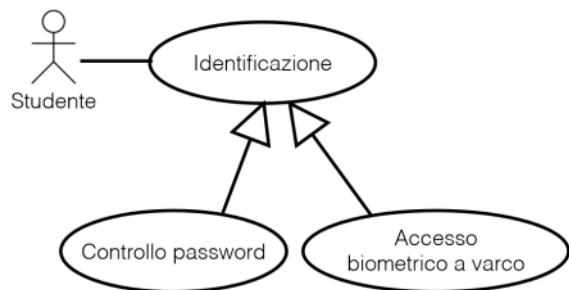


## Generalizzazione tra use-case

- Alcune funzionalità dello use-case A, solo in alcuni casi particolari, sono rimpiazzate con le funzionalità dello use-case B

**Esempio:**

- Gli studenti devono potersi identificare
- L'identificazione online avviene tramite password
- La registrazione delle presenze ai corsi avviene tramite impronta digitale/iride scansionata dal lettore del tornello

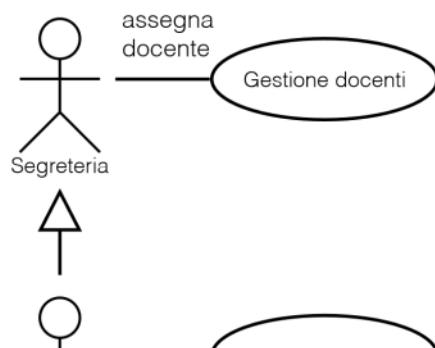


Nota: nei diagrammi degli use-case non possiamo usare generalizzazioni uniche che coinvolgono più sotto-usecase, né tantomeno vincoli {disjoint} e {complete}

## Generalizzazione tra attori

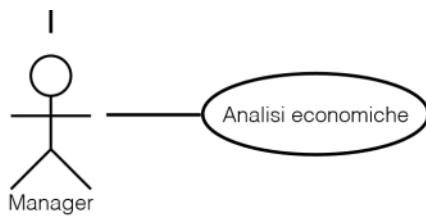
- L'attore B può fare le veci dell'attore A, e ne eredita tutte le associazioni
- Esempio: i manager possono fare le veci della Segreteria, ed accedere a tutti gli use-case accessibili dalla Segreteria

**Attenzione.** Il diagramma non implica che esistano le classi Segreteria e Manager nel diagramma delle classi, né tantomeno che la classe Manager sia una sottoclasse di Segreteria





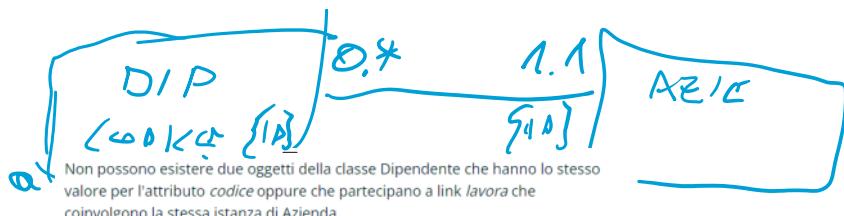
classe `iManager` sia una sottoclassificazione di `Segreteria`



## Diagramma degli use-case: specifiche

- Il diagramma degli use-case è molto semplice, e dà solo una visione di alto livello di:
  - quali attori possono usare il sistema
  - quali macro-funzionalità sono accessibili ai diversi attori
- Definisce inoltre come le diverse macro-funzionalità vadano modularizzate
- Si tratta di un diagramma facilmente comprensibile anche al committente
- Il diagramma **non** definisce le singole operazioni all'interno di ogni use-case
- Ogni use-case del diagramma andrà affiancato da un **documento di specifica** che entra nel dettaglio (v. seguito)





a) Non possono esistere due oggetti della classe Dipendente che hanno lo stesso valore per l'attributo *codice* oppure che partecipano a link *lavora* che coinvolgono la stessa istanza di Azienda.

b) Non esistono due link *lavora* con lo stesso codice e la stessa azienda.

c) Due oggetti della classe Dipendente possono avere lo stesso valore per l'attributo *codice* solamente se almeno uno dei due dipendenti non lavora presso nessuna azienda.

d) Non possono esistere due oggetti distinti della classe Dipendente che hanno lo stesso valore per l'attributo *codice* e che partecipano a link *lavora* che coinvolgono la stessa istanza di Azienda.

