Unified Modeling Language

Class Diagram

Class Diagram

- Definisce la visione statica del sistema
  - classi (di oggetti): incapsulamento di struttura dati e funzioni
  - relazioni tra classi
    - » associazione (uso)
    - » generalizzazione (ereditarietà)
    - » aggregazione (contenimento)
- È forse il modello più importante perché definisce gli elementi base del sistema

Orientamento agli oggetti (OO)

- Un software *orientato agli oggetti* è organizzato in forma di collezione di oggetti discreti che incorporano sia lo stato (struttura dei dati) che il comportamento (operazioni/ funzioni)
  - Evoluzione del concetto di tipo di dato astratto
  - Nell'approccio orientato alle procedure, struttura dati e funzioni sono connesse debolmente
  - Il termine OO e i modelli sono applicabili a tutte le fasi di sviluppo: analisi, progettazione, realizzazione; ciò che cambia è il livello di astrazione ma non i concetti e la notazione

Oggetto

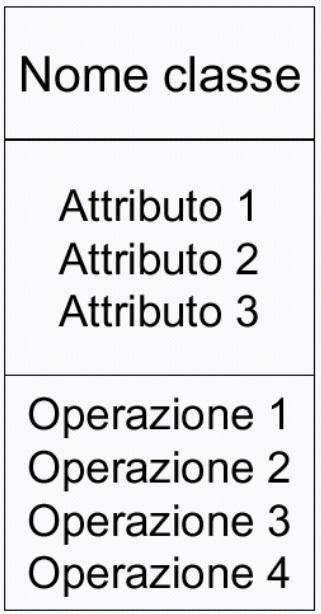
- Definizione generale
  - Un oggetto rappresenta una qualsiasi cosa, reale o astratta, che abbia un confine definito (concetto associato)
- Definizione più specifica
  - Un oggetto è un qualcosa di identificabile che ricorda il proprio stato e che può rispondere a richieste per operazioni relative al proprio stato
- Gli oggetti interagiscono tra di loro richiedendo reciprocamente servizi o informazioni
  - In risposta ad una richiesta, un oggetto può invocare un'operazione che può cambiare il suo stato

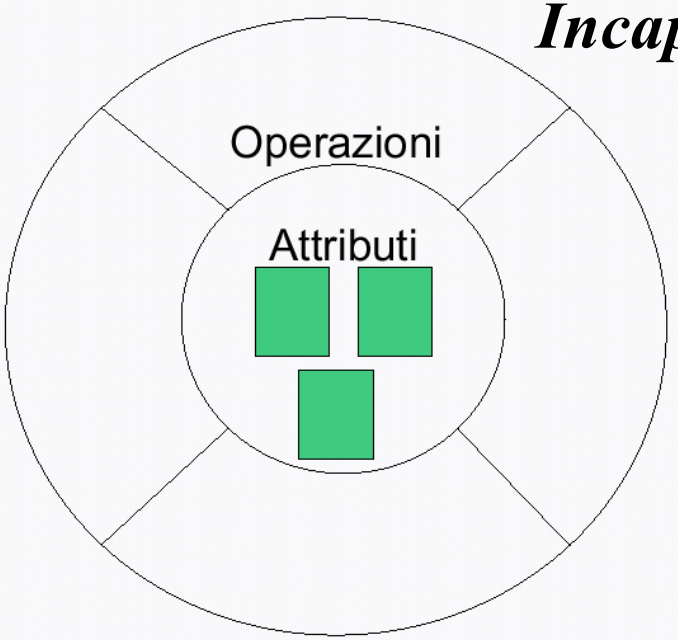
Classe (di oggetti)

- Un oggetto ha due tipi di proprietà:
  - attributi (o variabili di istanza), che descrivono lo stato
  - operazioni (o metodi), che descrivono il comportamento
- Una classe rappresenta un'astrazione di oggetti simili
  - oggetti che hanno le stesse proprietà: attributi e operazioni
- I termini oggetto e istanza (di classe) sono interscambiabili

Classi in UML

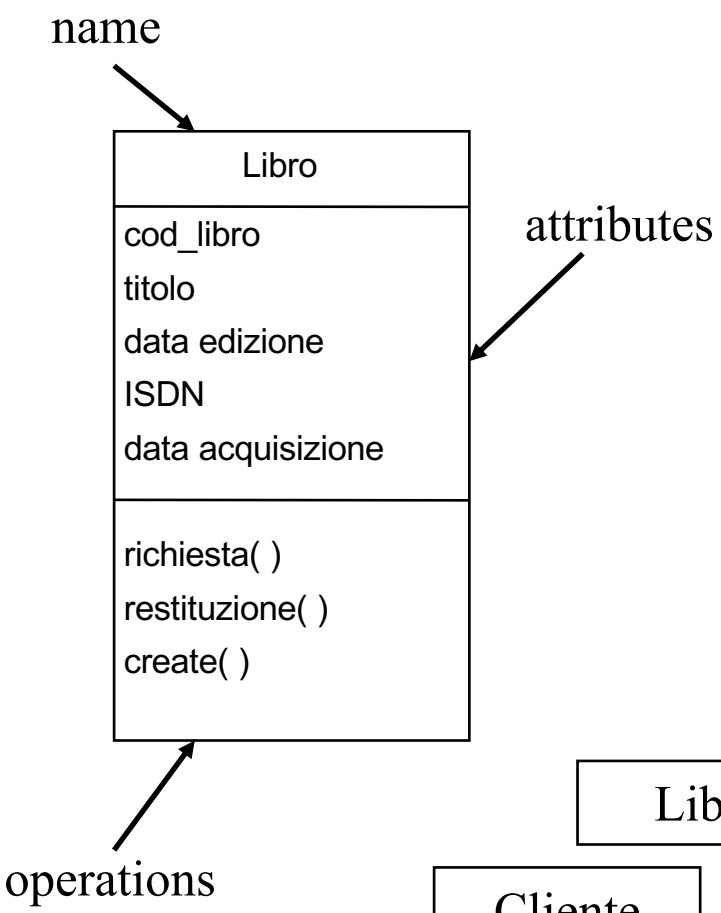
- In UML una classe è composta da tre parti
  - nome
  - attributi (lo stato)
  - metodi o operazioni (il comportamento)





Incapsulamento

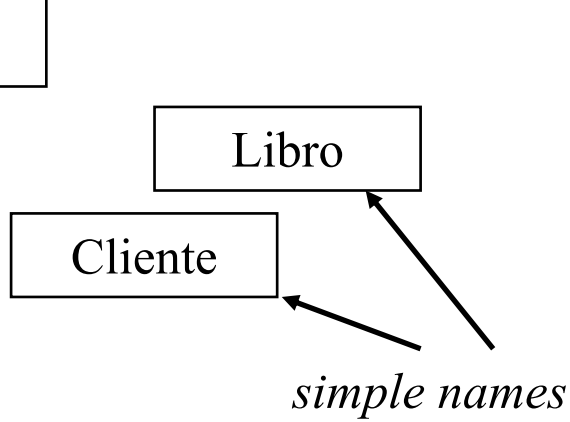
Class - names

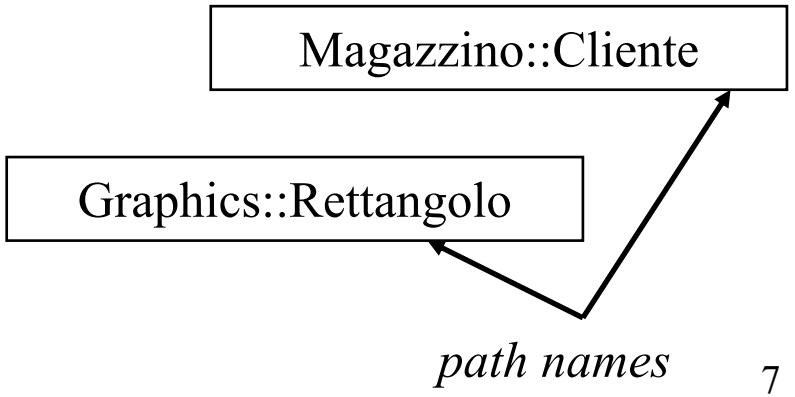


Una classe può essere rappresentata anche usando solo la sezione del nome

Un nome può essere un:

- *simple name*, il solo nome della classe
- *path name*, il nome della classe preceduto dal nome del package in cui la classe è posta



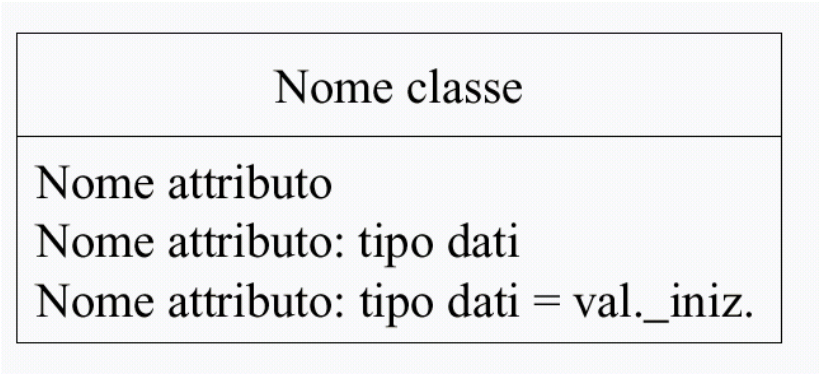


Attributi

- Un attributo è una proprietà statica di un oggetto
  - nome, età, peso sono attributi della classe Persona
  - colore, peso, anno, modello sono attributi della classe Auto
- Un attributo contiene un valore per ogni istanza
  - l'attributo età ha il valore "24" nell'oggetto Franco Lorusso
- I nomi degli attributi devono essere unici all'interno di una classe
- Il valore di un attributo non ha identità (non è un oggetto)
  - tutte le occorrenze di "24" sono indistinguibili

Attributi

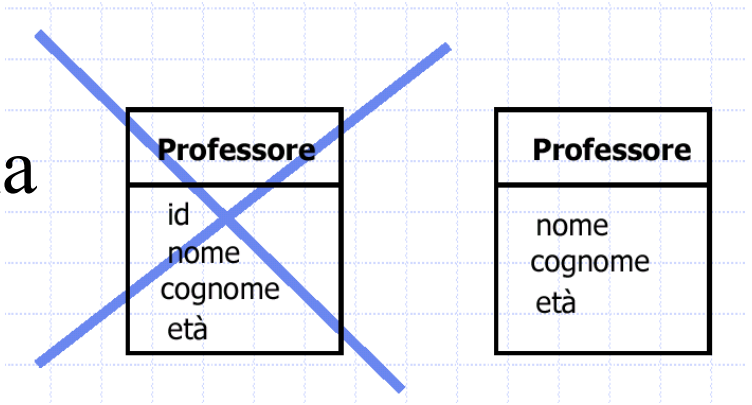
- Per ciascun attributo si può specificare il tipo ed un eventuale valore iniziale
- Tipicamente il nome di un attributo è composto da una parola o più parole
  - usare il maiuscolo per la prima lettera di ciascuna parola lasciando minuscola la lettera iniziale del nome



Scaffale
altezza: Float larghezza: Float profondità: Float numeroScansie: Int èPieno: Boolean=false

Individuazione degli attributi

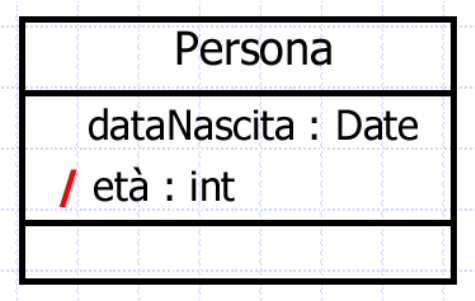
- Gli oggetti avranno una loro identità, non bisogna aggiungerla



- Candidati per attributi
  - Nomi che non sono diventati classi
- Conoscenza del dominio applicativo
  - Persona (ambito bancario)
    - » nome, cognome, codiceFiscale, numeroConto
  - Persona (ambito medico)
    - » nome, cognome, allergie, peso, altezza

Attributi derivati

- Calcolati, non memorizzati
- Si usano quando i loro valori variano frequentemente e la correttezza (precisione) del valore è importante
- Il valore viene calcolato in base ai valori di altri attributi
  - età = f(dataDiNascita, oggi)
  - area, perimetro = f(vertici)



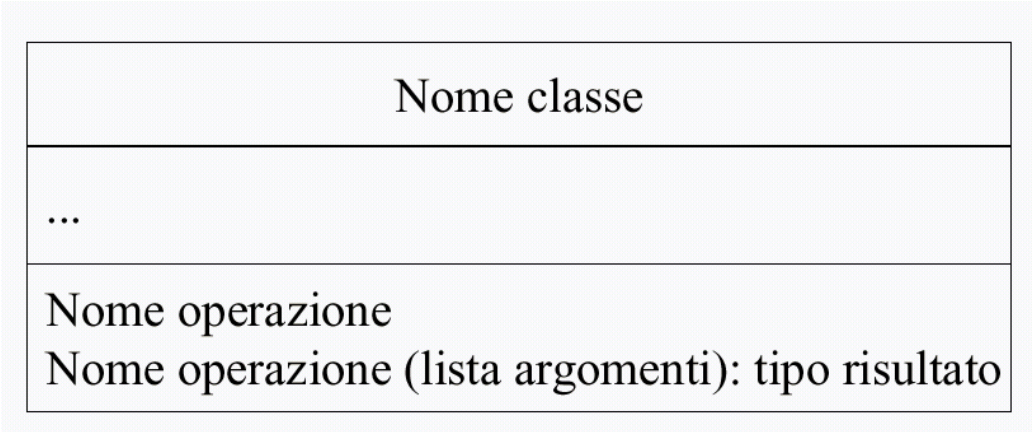
{età = oggi - dataNascita}

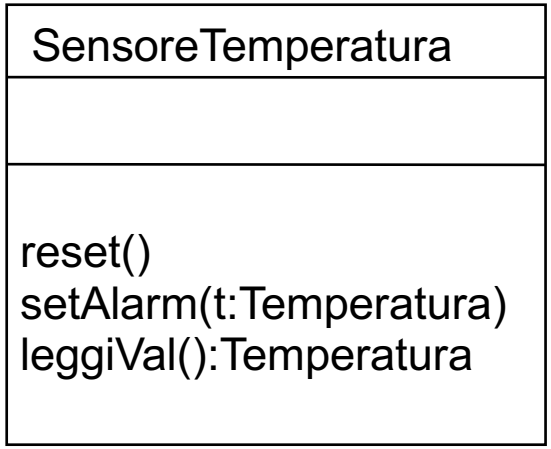
Operazioni

- Un'operazione è un'azione che un oggetto esegue su un altro oggetto e che determina una reazione
  - Le operazioni operano sui dati incapsulati dell'oggetto
- Tipi di operazione
  - selettore (query): accedono allo stato dell'oggetto senza alterarlo (es. "lunghezza" della classe coda)
  - modificatore: alterano lo stato di un oggetto (es. "appendi" della classe coda)
- Operazioni di base per una classe di oggetti (realizzate con modalità diverse a seconda dei linguaggi)
  - costruttore: crea un nuovo oggetto e/o inizializza il suo stato
  - distruttore: distrugge un oggetto e/o libera il suo stato

Operazioni

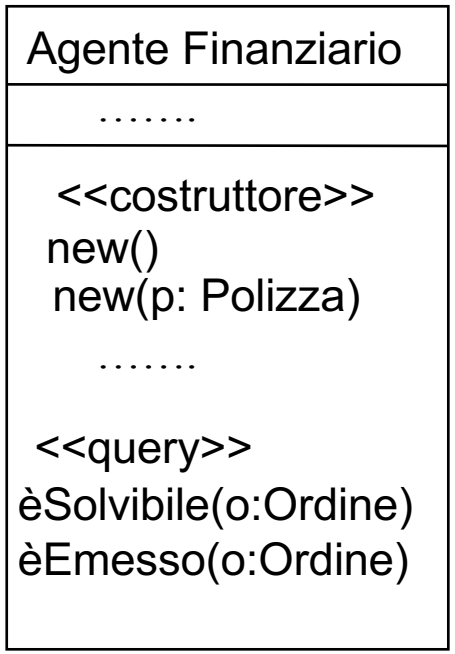
- Per ciascun operation si può specificare il solo nome o la sua *signature*, indicando il nome, il tipo, parametri e, in caso di funzione, il tipo ritornato
- Stesse convenzioni dette per attributes per l'uso di minuscolo e maiuscolo per i nomi delle operations





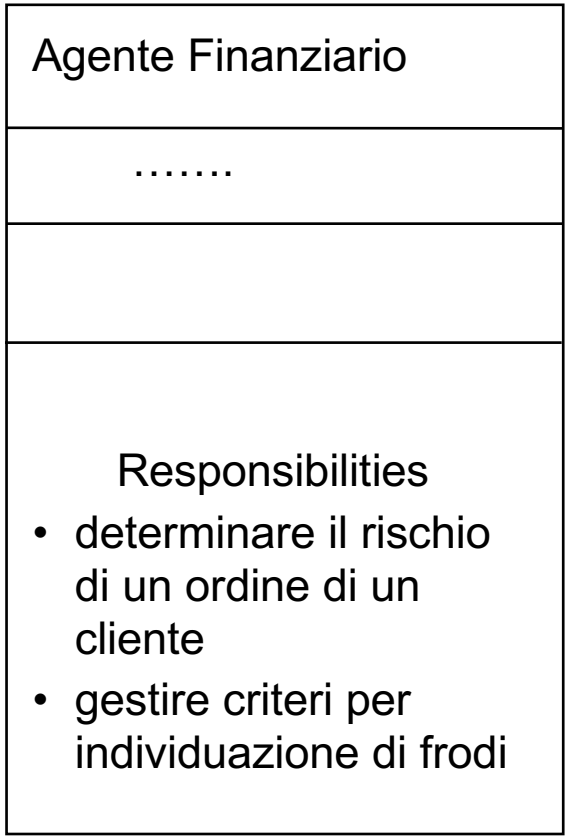
Attributi e Operazioni

- Attributi e Operazioni di una class non devono obbligatoriamente essere descritti tutti subito
- Attributi ed operazioni possono essere mostrati solo parzialmente,
  - Per indicare che esistono più attributi/operazioni di quelli mostrati usare i punti sospensivi " ……."
- Per meglio organizzare lunghe liste di attributes/operations raggrupparli insieme in categorie usando stereotypes



Responsibility

- Una responsibility è un contratto o una obbligazione di una classe
  - Questa è definita dallo stato e comportamento della classe
  - Una classe può avere un qualsiasi numero di responsabiltà, ma una classe ben strutturata ha una o poche responsabiltità
- Le responsibilità possono essere indicate, in maniera testuale, in una ulteriore sezione, sul fondo della icona della class

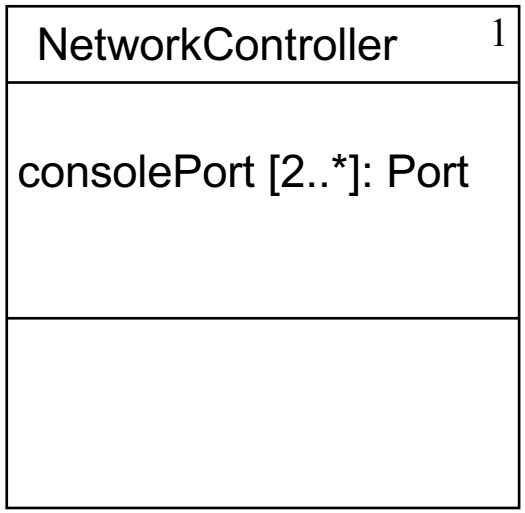


Visibilità

- E' possibile specificare la visibiltà di attributi e operazioni
- In UML è possibile specificare tre livelli di visibilità
  - + (***public***): qualsiasi altra classe con visibilità alla classe data può usare l'attributo/operazione (di default se nessun simbolo è indicato)
  - # (***protected***): qualsiasi classe discendente della classe data può usare l'attributo/operazione
  - - (***private***): solo la classe data può usare l'attributo/operazione

Molteplicità

- La *multiplicity* è usata per indicare il numero di istanze di una classe
  - una multiplicity pari a zero indicherà una classe astratta, una multiplicity pari ad uno una singleton class
  - per default è assunta una multiplicity maggiore di uno
- La multiplicity di una class è indicata con un numero intero posto nell'angolo in alto a destra del simbolo della class
- La multiplicity si applica anche agli attributi, indicandola tra […]



Specifica attributi

La sintassi completa per specificare un attribute in UML è:

[visibility] name [[multiplicity]] [: type] [= inital-value] [{property-string}]

dove property-string può assumere uno dei seguenti valori:

changeable	nessuna limitazione per la modifica del valore dell'attributo
addOnly	per attributi con molteplicità maggiore di 1 possono essere aggiunti ulteriori valori, ma una volta creato un valore non può essere né rimosso né modificato
frozen	il valore non può essere modificato dopo la sua inizializzazione

Specifica Operazioni

La sintassi completa per specificare una operation in UML è:

[visibility] name [(parameter-list)] [:return-type] [{property-string}]

con la lista dei parametri avente questa sintassi

[direction] name: type [=default-value]

dove direction può assumere uno dei seguenti valori:

in	parametro di input
out	parametro di output
inout	parametro di input/output

Specifica Operazioni: valori per property-string

isQuery	l'esecuzione dell'operazione lascia lo stato del sistema immutato

Proprietà che riguardano la concorrenza, rilevanti solo in presenza di oggetti attivi, processi o threads

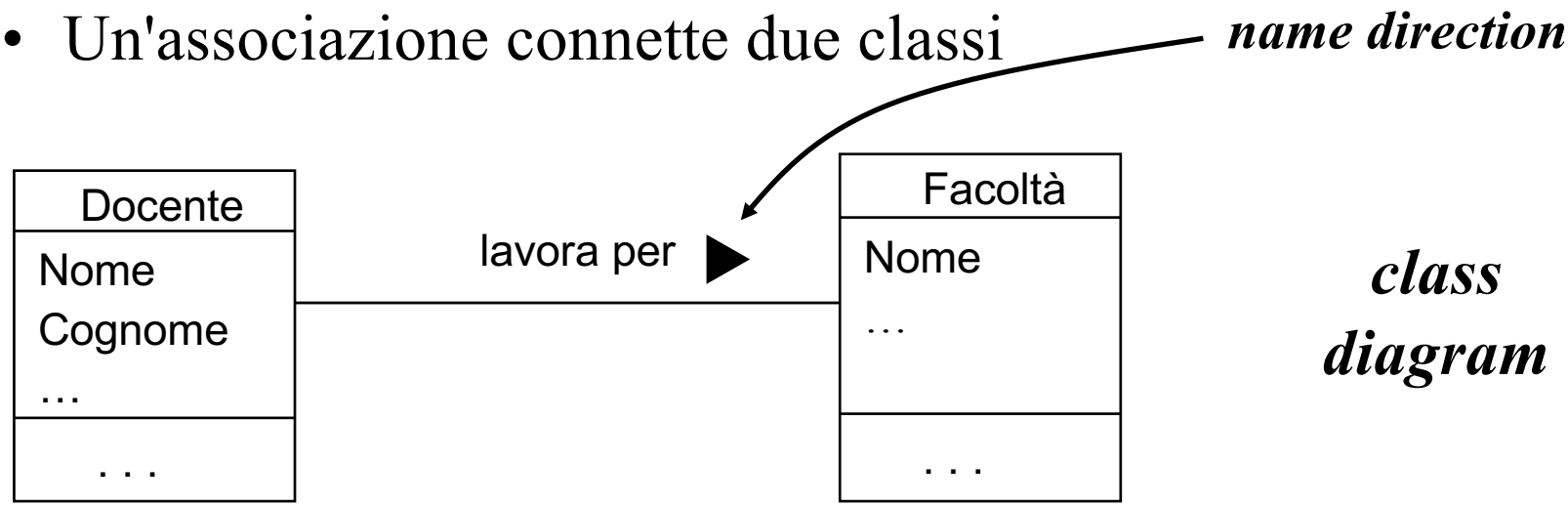
sequential	la semantica e l'integrità dell'oggetto è garantita nel caso di un solo flusso di controllo per volta verso l'oggetto
guarded	la semantica e l'integrità dell'oggetto è garantita in presenza di flussi di controllo multipli sequenzializzando le chiamate alle operazioni guarded
concurrent	la semantica e l'integrità dell'oggetto è garantita in presenza di flussi di controllo multipli, trattando la operation come atomica

Legami e associazioni

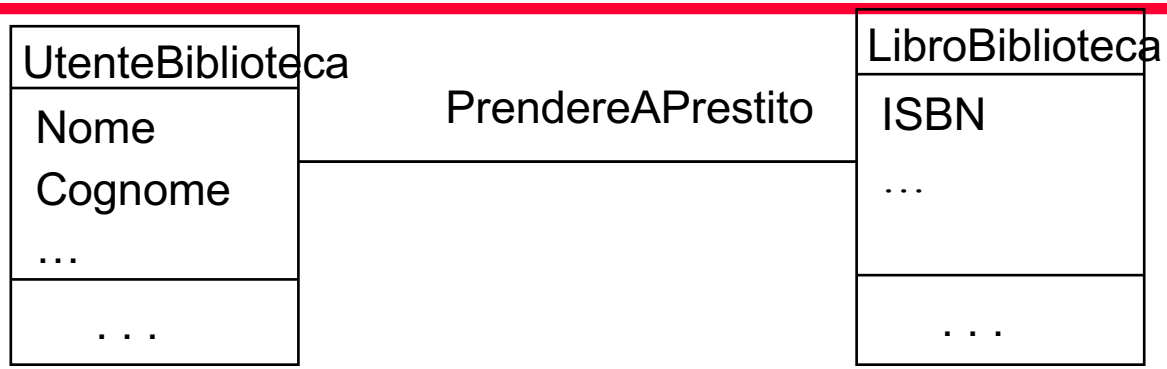
- Un legame (link) rappresenta una relazione (fisica o concettuale) tra oggetti la cui conoscenza deve essere preservata per un certo periodo di tempo
  - Es. "Filomena Ferrucci *lavora per* Facoltà di Scienze"
- Un'associazione descrive un gruppo di legami aventi struttura e semantica comuni
  - Es. "Docente *lavora per* Facoltà"
  - Un'associazione indica una relazione tra classi
- Un'associazione deve avere un nome
  - » Il nome è solitamente un verbo (un'etichetta al centro della linea che definisce l'associazione)
  - » Le associazioni sono bidirezionali sebbene al nome della relazione può essere associata una direzione

Legami e associazioni

- I link sono istanze delle associazioni
  - Un link connette due oggetti
  - Un'associazione connette due classi



Legami e associazioni

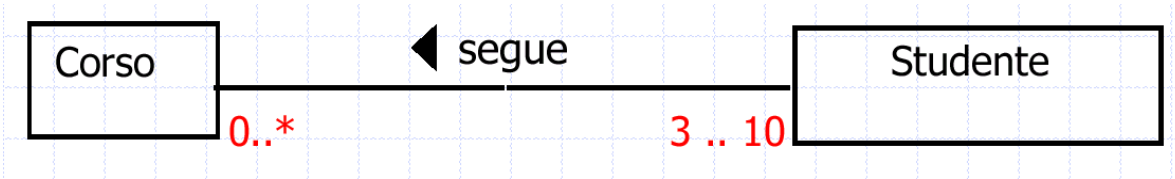


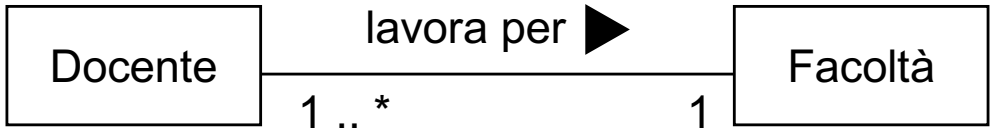
class diagram e associazione

- PrendereAPrestito, rappresenta un insieme di legami relazionali che stabilisce quale utente della Biblioteca sta attualmente prendendo a prestito un determinato libro
- Oggi potrebbe contenere due legami:
  - Alfredo sta prendendo a prestito Alice nel paese delle meraviglie
  - Carla sta prendendo a prestito Biancaneve e i sette nani
- Domani l'associazione PrendereAPrestito potrebbe contenere 8 legami
- PrendereAPrestito rappresenta un gruppo di legami il cui numero varia: Un'associazione è come una classe le cui istanze sono legami relazionali

Molteplicità delle associazioni

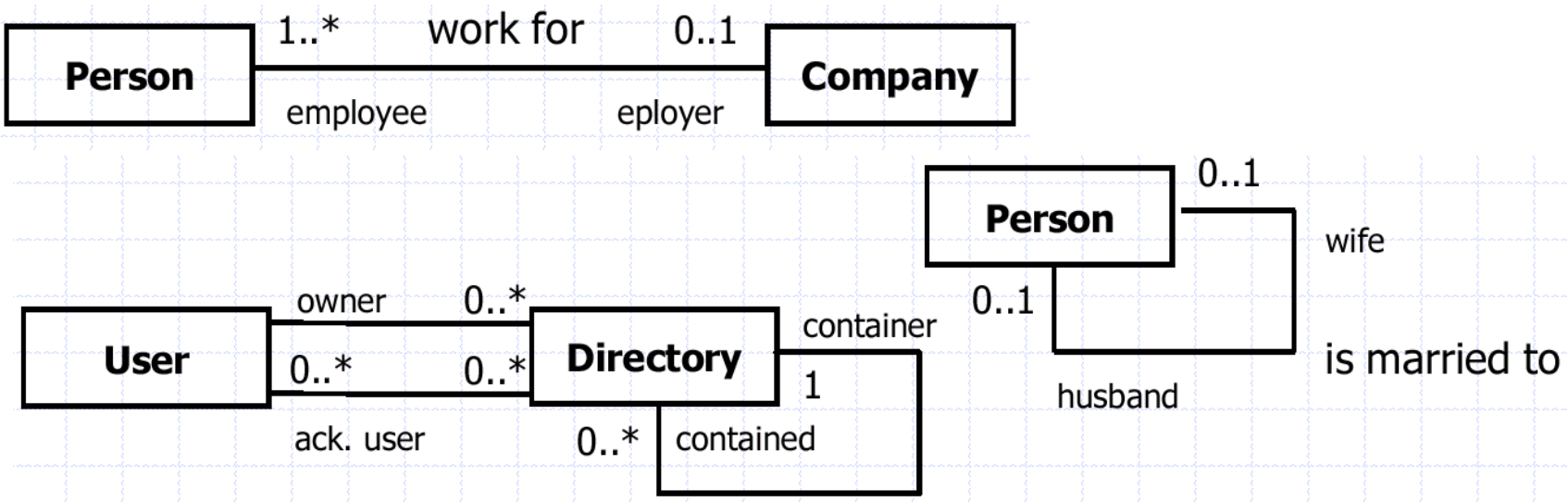
- La molteplicità dice
  - Se l'associazione è obbligatoria oppure no
  - Il numero minimo e massimo di oggetti che possono essere relazionati ad un altro oggetto
- Esempi
  - Esattamente uno: 1
  - Zero o uno: 0..1
  - Molti: 0..*
  - Uno o più: 1..*
  - Un numero specifico: 7
  - Un intervallo: 4..15
  - Lista: 0..1, 3..4, 6..*
    - » Tutti i numeri eccetto 2 e 5





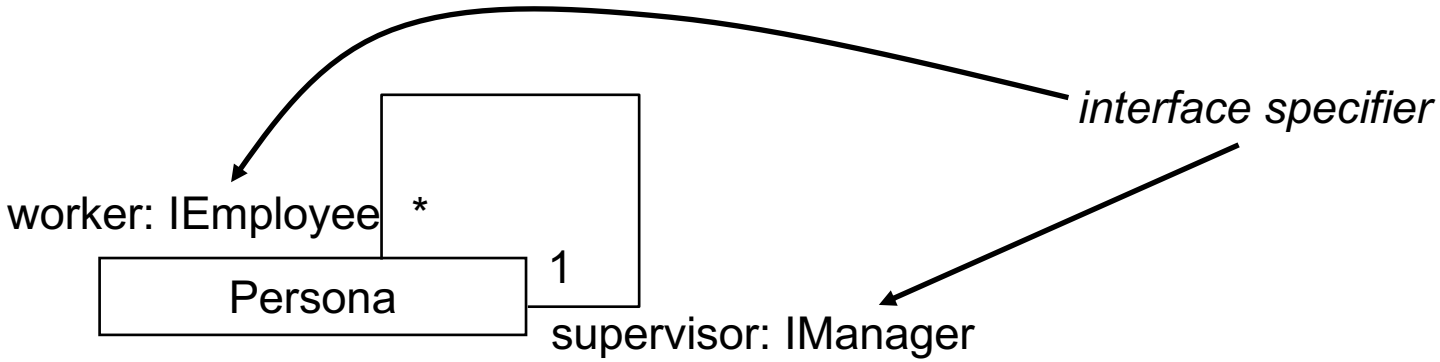
Ruoli

- I ruoli forniscono una modalità per attraversare relazioni da una classe ad un'altra
  - I nomi di ruolo possono essere usati in alternativa ai nomi delle associazioni
- I ruoli sono spesso usati per relazioni tra oggetti della stessa classe (*associazioni riflessive*)



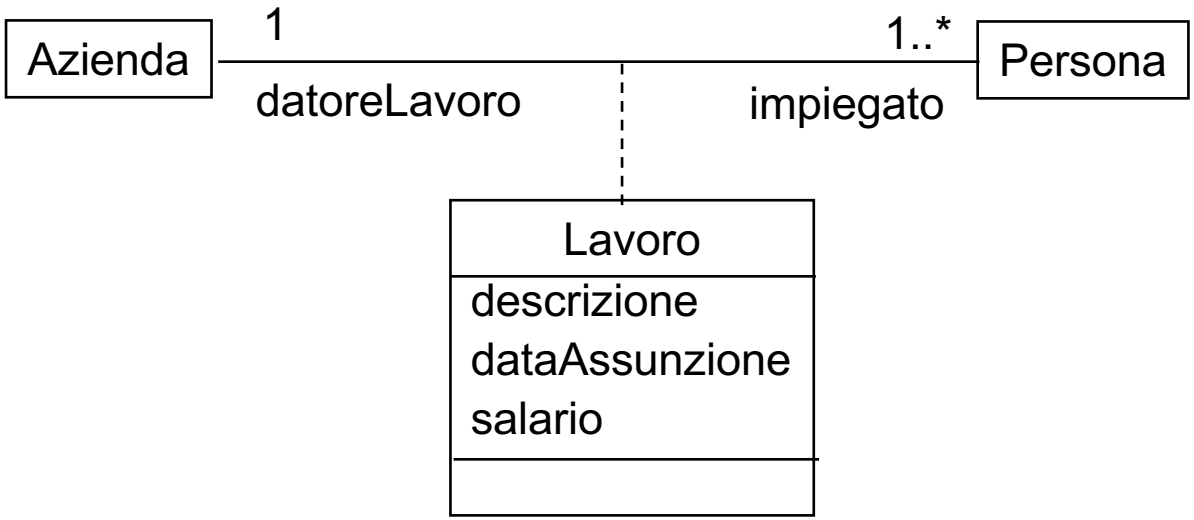
Interface Specifier

- Una association può avere un *interface specifier*, per specificare quale parte dell'interfaccia di una classe è mostrata da questa nei confronti di un'altra classe della stessa association
  - Una classe Persona nel ruolo di supervisor presenta solo la 'faccia' IManager al worker; mentre una Persona nel ruolo di worker presenta solo la 'faccia' IEmployee al supervisor



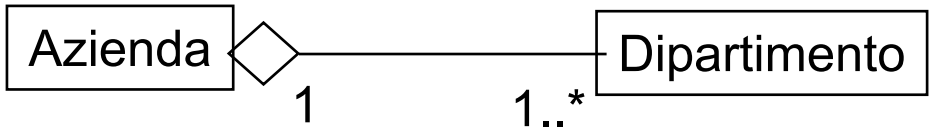
Classi associative

- Sono utilizzate per modellare proprietà delle associazioni
  - Alcune proprietà potrebbero appartenere all'associazione e non alle parti coinvolte
- Un attributo di una classe associativa contiene un valore per ogni legame



Aggregazioni

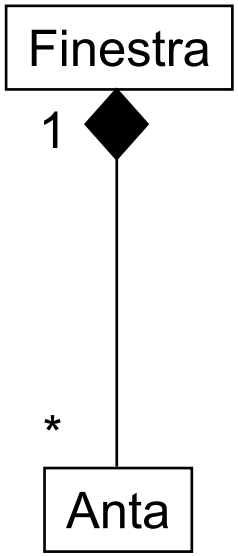
- La relazione di aggregazione è un'associazione speciale che aggrega gli oggetti di una classe componente in un unico oggetto della classe
  - la si puo' leggere come "è composto da" in un verso e "è parte di" nell'altro verso

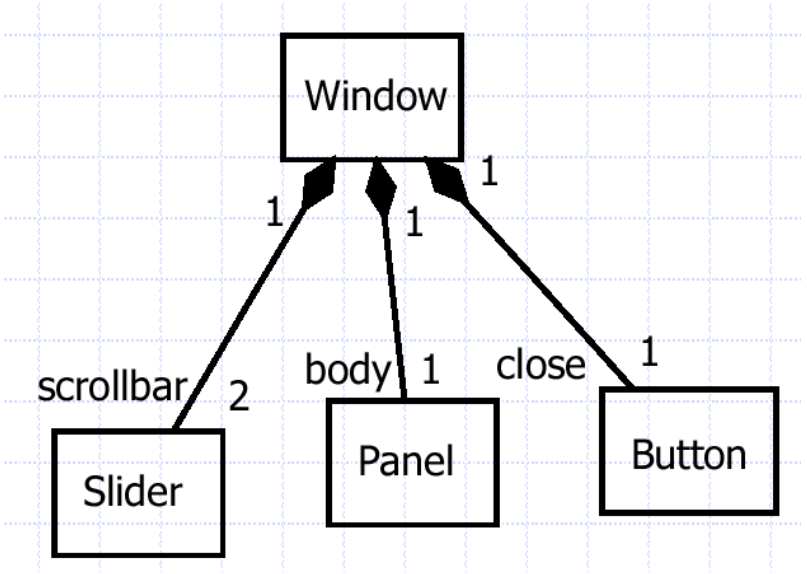


- Proprietà
  - transitività: se A è parte di B e B è parte di C allora A è parte di C
  - antisimmetria: se A è parte di B allora B non è parte di A
  - dipendenza: un oggetto contenuto potrebbe non sopravvivere senza l'oggetto contenente
- Un albero di aggregazione rappresenta una classe con più di una relazione di aggregazione

Composizione

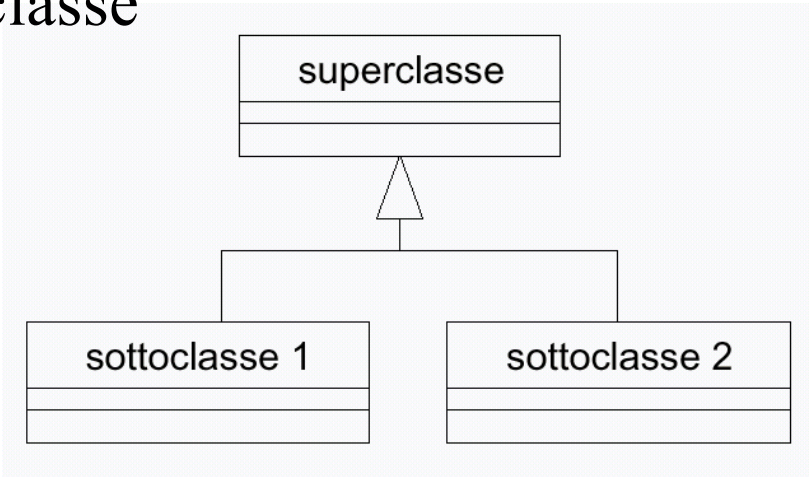
- Una relazione di composizione è un'aggregazione forte
  - Le parti componenti non esistono senza il contenitore
  - Ciascuna parte componente ha la stessa durata di vita del contenitore
  - Una parte può appartenere ad un solo tutto per volta

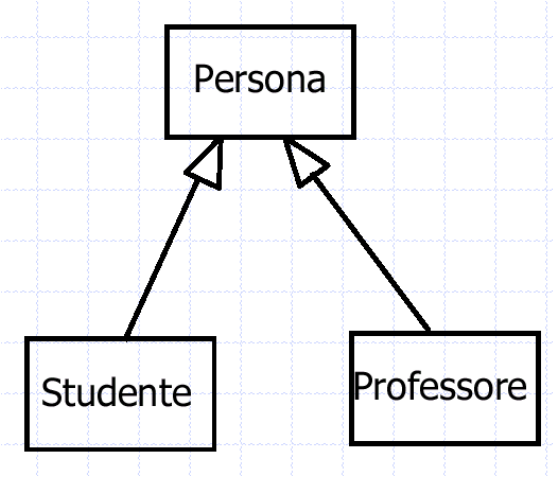




Generalizzazione

- La relazione di generalizzazione rappresenta una tassonomia delle classi
  - la classe generale è detta superclasse
  - ogni classe specializzata è detta sottoclasse
- Può essere letta come
  - “e’ un tipo di” (verso di generalizzazione)
  - “puo’ essere un” (verso di specializzazione)
- Ogni oggetto di una sottoclasse è anche un oggetto della sua superclasse



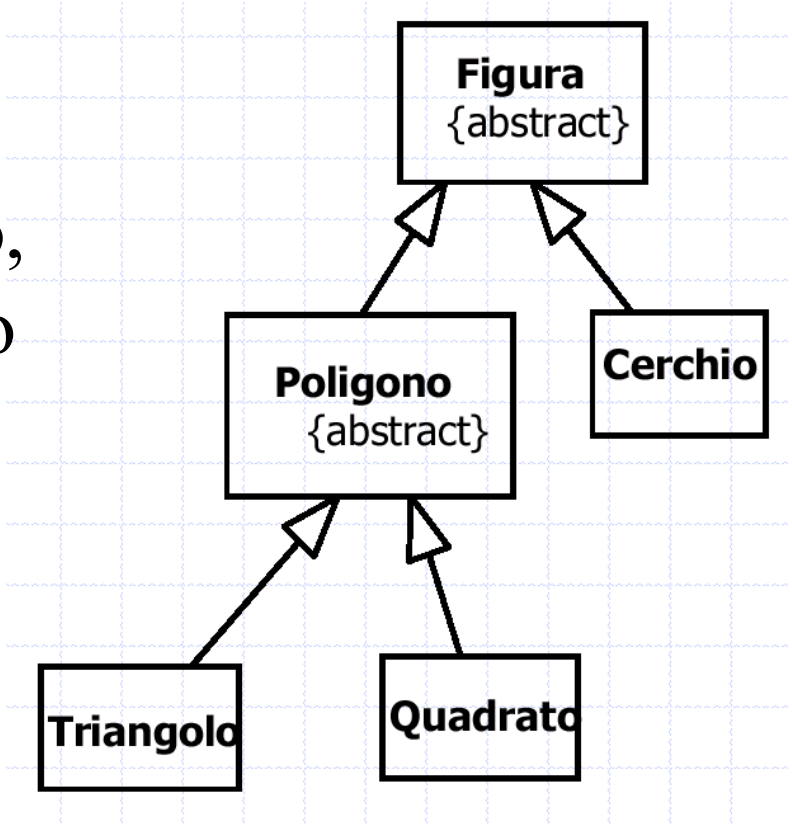


Ereditarietà

- L'ereditarietà è un meccanismo di condivisione delle proprietà degli oggetti in una gerarchia di generalizzazione
- Tutte le proprietà (attributi e operazioni) di una superclasse possono essere applicati alle sottoclassi (sono ereditati)
  - generalizzazione ed ereditarietà godono della proprietà transitiva
  - E' possibile definire nuove proprietà per le sottoclassi
  - E' possibile ridefinire le proprietà ereditate (*overriding*)
- Anche le relazioni di una superclasse valgono per le sottoclassi

Classi astratte

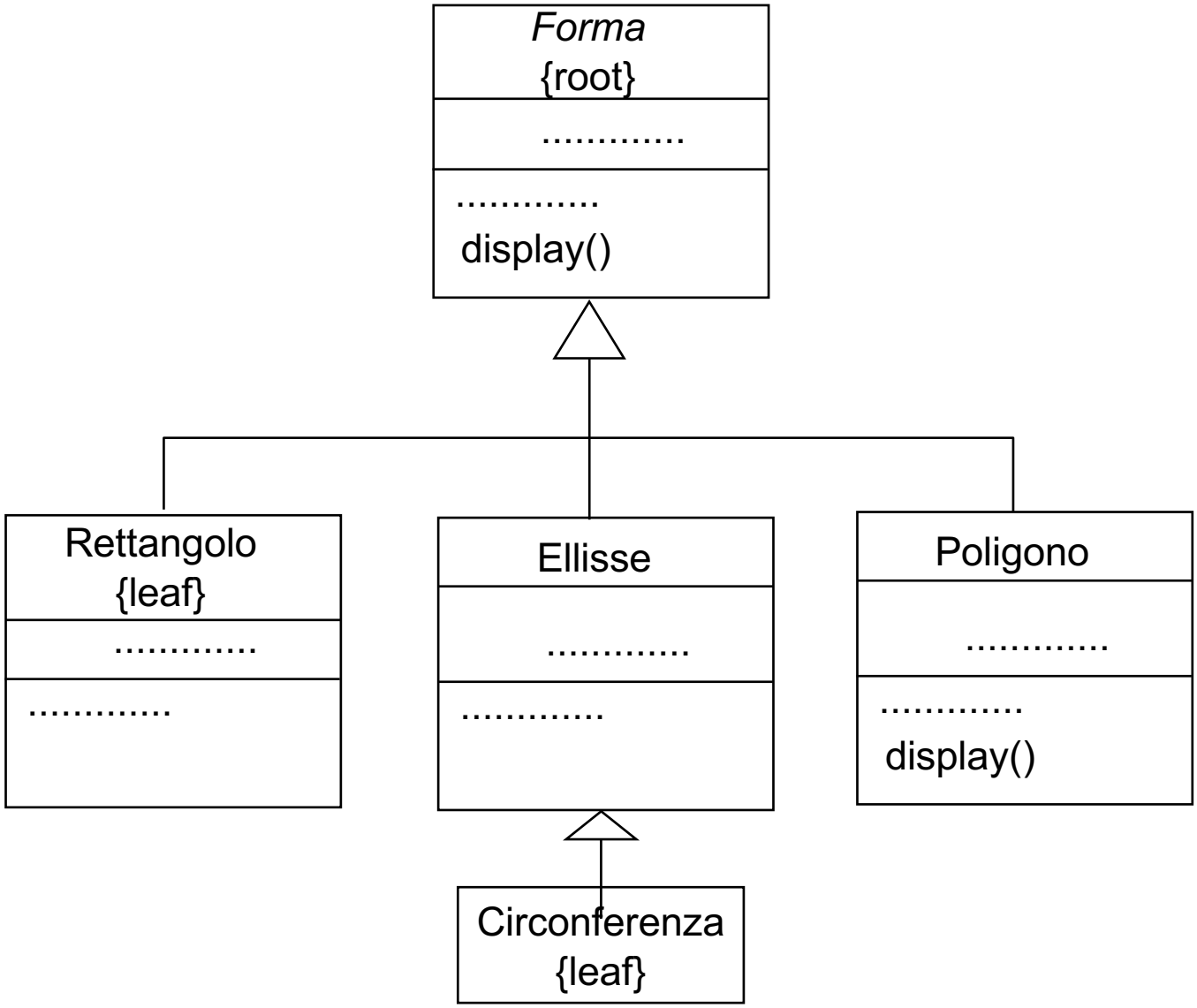
- Una classe astratta definisce un comportamento "generico".
- Definisce e può implementare parzialmente il comportamento, ma molto della classe è lasciato indefinito e non implementato. Dettagli specifici sono completati nelle sottoclassi specializzate
- Le classi astratte sono indicate ponendo il nome in corsivo



Root, leaf and polymorphic elements

- Per specificare che una class non può avere discendenti si indicherà per questa la proprietà *leaf* sotto il name della classe
- Per specificare che una classe non può avere antenati si indicherà per questa la proprietà *root* sotto il name della classe
- Per indicare che una operazione è astratta il suo nome sarà scritto in corsivo
- Una operazione per la quale esiste la stessa signature in più classi di una gerarchia è *polimorphic*

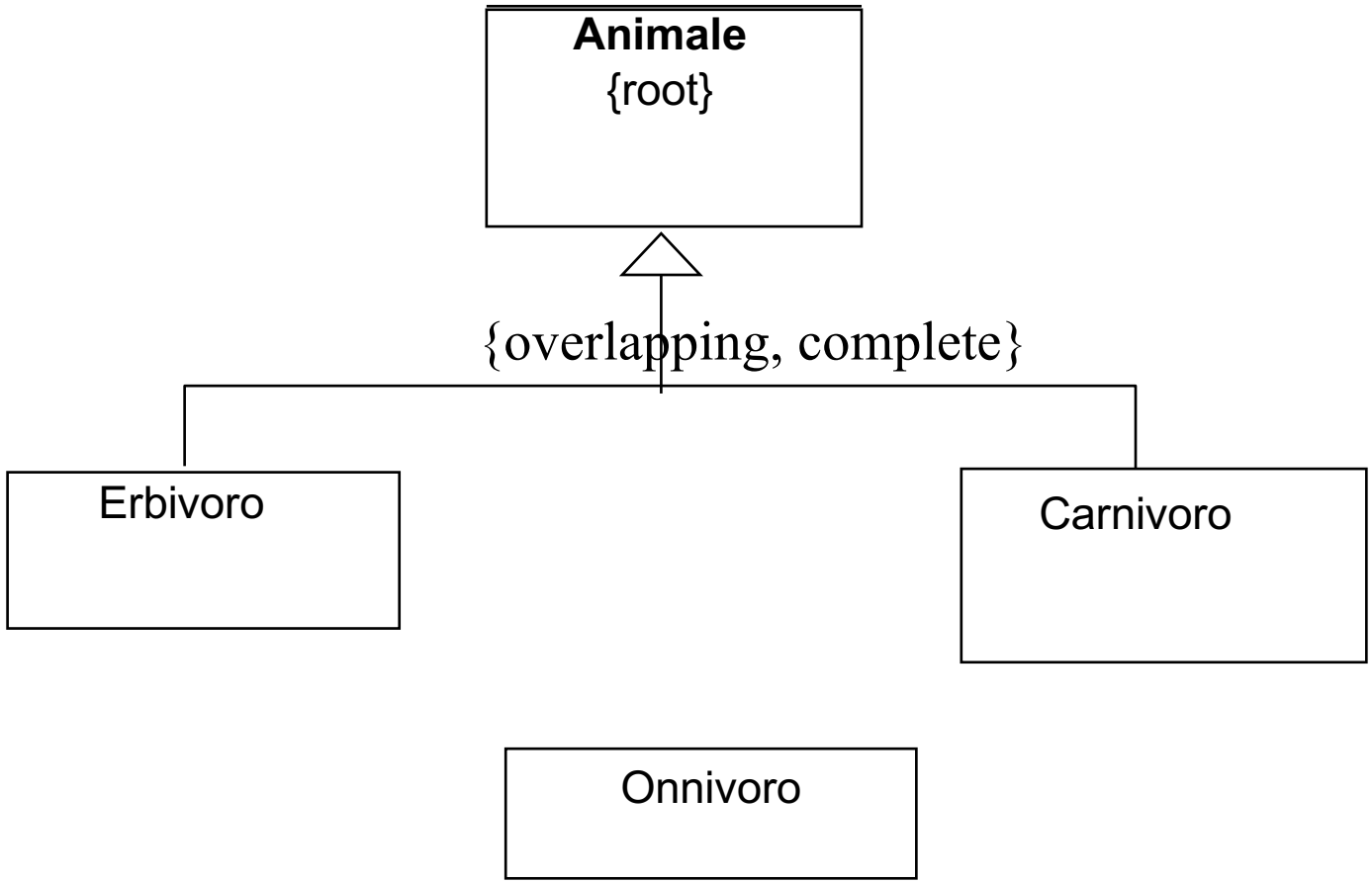
Esempio


Forma
{root}
.....
.....
display()
Rettangolo
{leaf}
.....
.....
Ellisse
.....
.....
Poligono
.....
.....
display()
Circonferenza
{leaf}

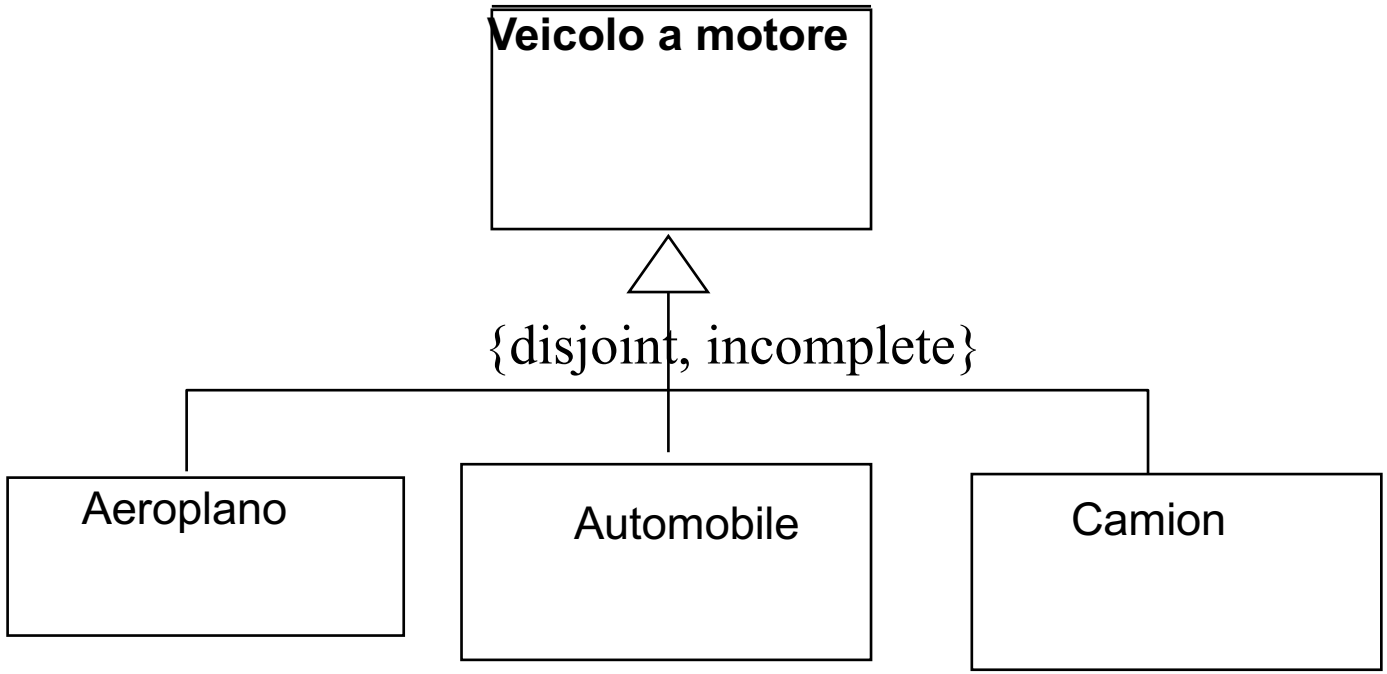
Gerarchia di classi

- UML definisce
  - 1 stereotype
    - » implementation: la sottoclasse eredita l'implementazione della superclasse ma non rende pubblica né supporta la sua interfaccia,
  - 4 constraints
    - » complete: tutte le sottoclassi sono state specificate, nessun altra sottoclasse è permessa
    - » incomplete: non tutte le sottoclassi sono state specificate, altre sottoclassi sono permesse
    - » disjoint: oggetti del genitore possono avere non più di un figlio come tipo
    - » overlapping: oggetti del genitore possono avere più di un figlio come tipo

Complete e Overlapping

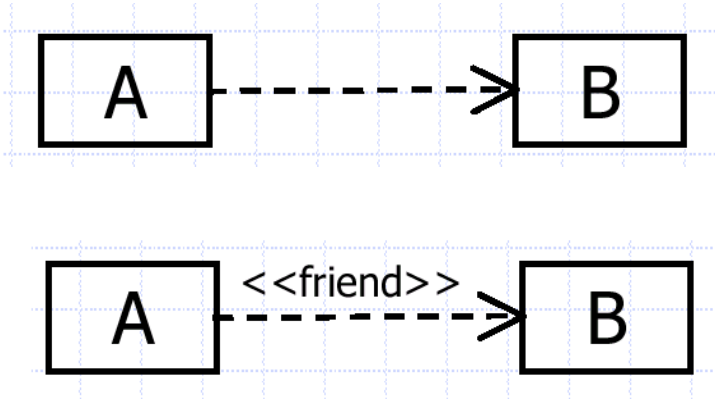


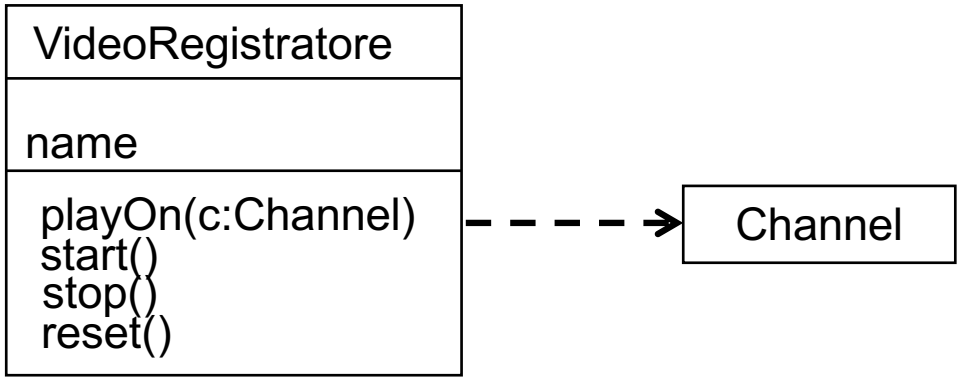
Disjoint e Incomplete



Dependency

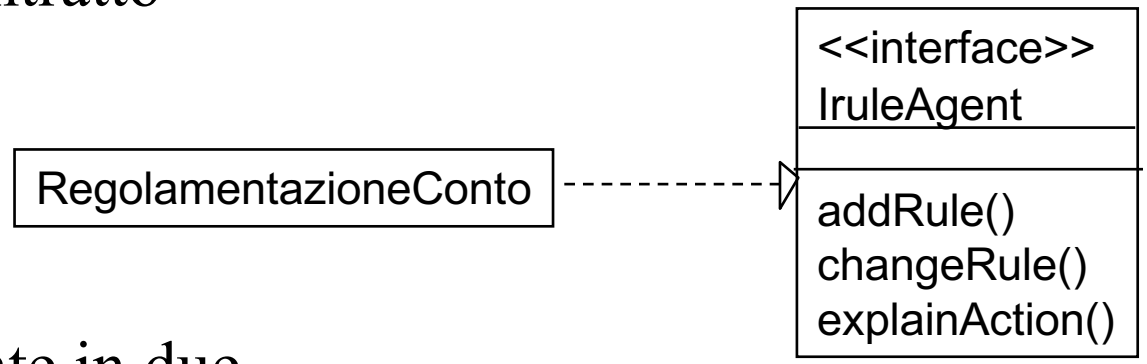
- Relazione semantica in cui un cambiamento sulla classe indipendente può influenzare la semantica della classe dipendente
  - La freccia punta verso la thing indipendente
- UML defisce 17 stereotypes (organizzati in 6 gruppi) che possono essere applicati a dependency





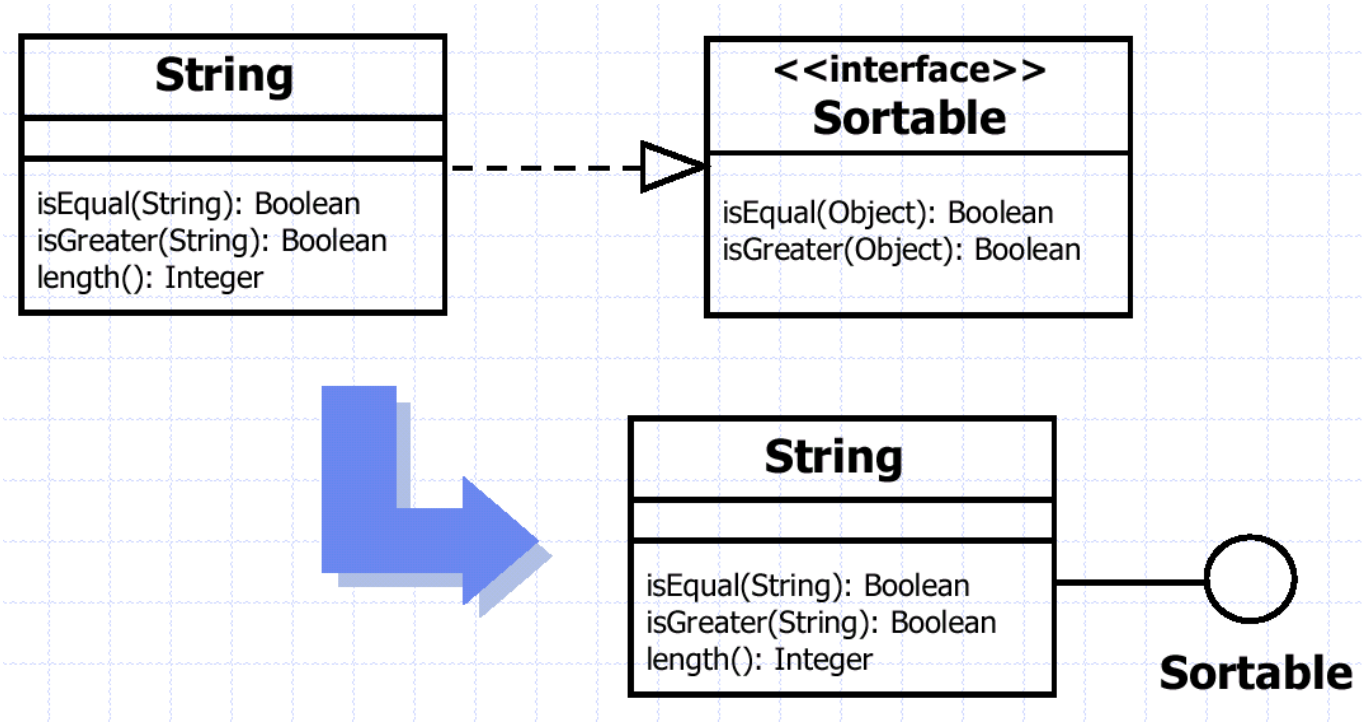
Realization

- Una relazione tra classi in cui una specifica un contratto che l'altra garantisce di compiere
  - La freccia punta alla classe che definisce il contratto
- E' un incrocio tra dependency e generalization
  - usata principalmente in due circostanze: contesto delle *interface* e delle *collaboration*



Interfaccia (Definizione)

- Specifica il comportamento di una classe senza darne l'implementazione



Interfaccia (Uso)