I diagrammi di classe vengono utilizzati per descrivere la struttura del sistema.Descrivono il sistema in termini di oggetti, classi, attributi, operazioni e relative associazioni.
Le classi sono astrazioni che specificano la struttura e il comportamento comuni di un insieme di oggetti dove questi ultimi hanno le stesse proprietà: attributi e operazioni (come in java)
Gli oggetti sono istanze di classi create, modificate e distrutte durante l'esecuzione del sistema. Un oggetto ha uno stato che include i valori dei suoi attributi, i suoi collegamenti con altri oggetti e risponde a richieste per operazioni relative al proprio stato. Quindi gli oggetti interagisco 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.
In un UML, una classe è composta da tre parti:

1. nome
2. attributi
3. metodi o operazioni (il comportamento)

Nome classe
Attributo1 Attributo2 Attributo3
Operazione1 Operazione2 Operazione3

Classe-Nomi

Una classe può essere rappresentata anche usando solo la sezione del nome. Un nome può essere un:

1. "*simple name*", il solo nome della classe --> Libro
2. "*path name*", il nome della classe preceduto dal nome del package in cui la classe è posta. —> Graphics::Rettangolo oppure Magazzino::Cliente

"Libro" oppure "Magazzino::Libro"
Cod_libro titolo data_edizione
richiesta() restituzione() create()

Classe-Attributi

Un'attributo è una proprietà statica di un oggetto:

- nome, età, peso sono attributi della classe Persona
- colore, peso, anno 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 quindi tutte le occorrenze di 24 sono indistinguibili.
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.

Nome classe
Nome attributo Nome Attributo:tipo dati Nome attributo: tipo dati=val_inz.

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

Attributi derivati

Si trattano degli attributi che vengono calcolati e non memorizzati. Si usano quando i loro valori variano frequentemente e la correttezza del valore è importante. Quindi il valore viene calcolato in base ai valori di altri attributi.

- età=f(dataDiNascita,oggi)

Persona
dataNascita:date /eta:int
{eta=oggi-dataNascita}

Operazioni

Un'operazione è un'azione che un oggetto esegue su un altro oggetto e che determina una reazione. Tali operazioni operano sui dati incapsulati dell'oggetto.
Ci sono diversi tipi di operazioni:

- Selettore (query): accedono allo stato dell'oggetto senza alterarlo.
- Modificatore: alterano lo stato di un oggetto.

Vi sono due operazioni di base per una classe di oggetti:

- Costruttore: crea un nuovo oggetto e/o inizializza il suo stato.
- distruttore: distrugge un oggetto e/o libera il suo stato.

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.

Nome Classe	
Nome Operazione Nome Operazione (lista argomenti): tipo risultato	
Sensore Temperatura	
reset() setAlarm(t:Temperatura) leggiVal():Temperatura	

Attributi e Operazioni di una classe non devono obbligatoriamente essere descritti tutti subito. Attributi ed operazioni possono essere mostrati solo parzialmente quindi per indicare che esistono più attributi/operazioni di quelli mostrati si possono usare…
Ai fini di un'organizzazione, quando abbiamo lunghe liste di attributi/operazioni, si raggruppa insieme in categorie usando stereotipi

Agente Finanziario
<<Costruttore>> new() new(p:Polizza) …. <<query>> èSolvibile(o:Ordine) èEmesso(o:Ordine)

Responsabilità

Una responsabilità è un contratto o una obbligazioni di una classe; questa è definita dallo stato e comportamento della classe. Una classe può avere un qualsiasi numero di responsabilità, ma una classe ben strutturata ha una o poche responsabilità.

Agente Finanziario
….
...
Responsibilities • Determinare il rischio di un ordine di un cliente. • Gestire criteri per individuazioni di fondi.

Visibilità

E' possibile specifica la visibilità di attributi e operazioni. In UML è possibile specificare tre livelli di pubblicità:

1. +(public): qualsiasi altra classe con visibilità alla classe data può usare l'attributo/operazioni (di default se nessun simbolo è indicato)
2. #(protected): qualsiasi classe discendente della classe data può usare l'attributo/operazione
3. -(private) solo la classe data può usare l'attributo/operazione

Libro
#Cod_libro -titolo -data_edizione
+richiesta() restituzione() +create()

Molteplicità

La molteplicità è usata per indicare il numero di istanze di una classe:

- una molteciplità pari a zero indicherà una classe astratta, una molteciplità pari ad uno una singleton class (per default è assunta una multiplicity maggiore di uno. Viene indicata con un numero interno posto nell'angolo in alto a destra del simbolo della classe. Si assegna anche agli attributi indicandolo tra […]

ControlRod	4

Sintassi completa per specificare un attribute:

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

dove property-string può assumere uno dei seguenti valori:

- changeable: nessun limitazione per la modifica del valore dell'attributo;
- addOnly: per attributi con molteciplità 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.

Sintassi completa per specificare una operation in UML:

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

con la lista dei parametri avete 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

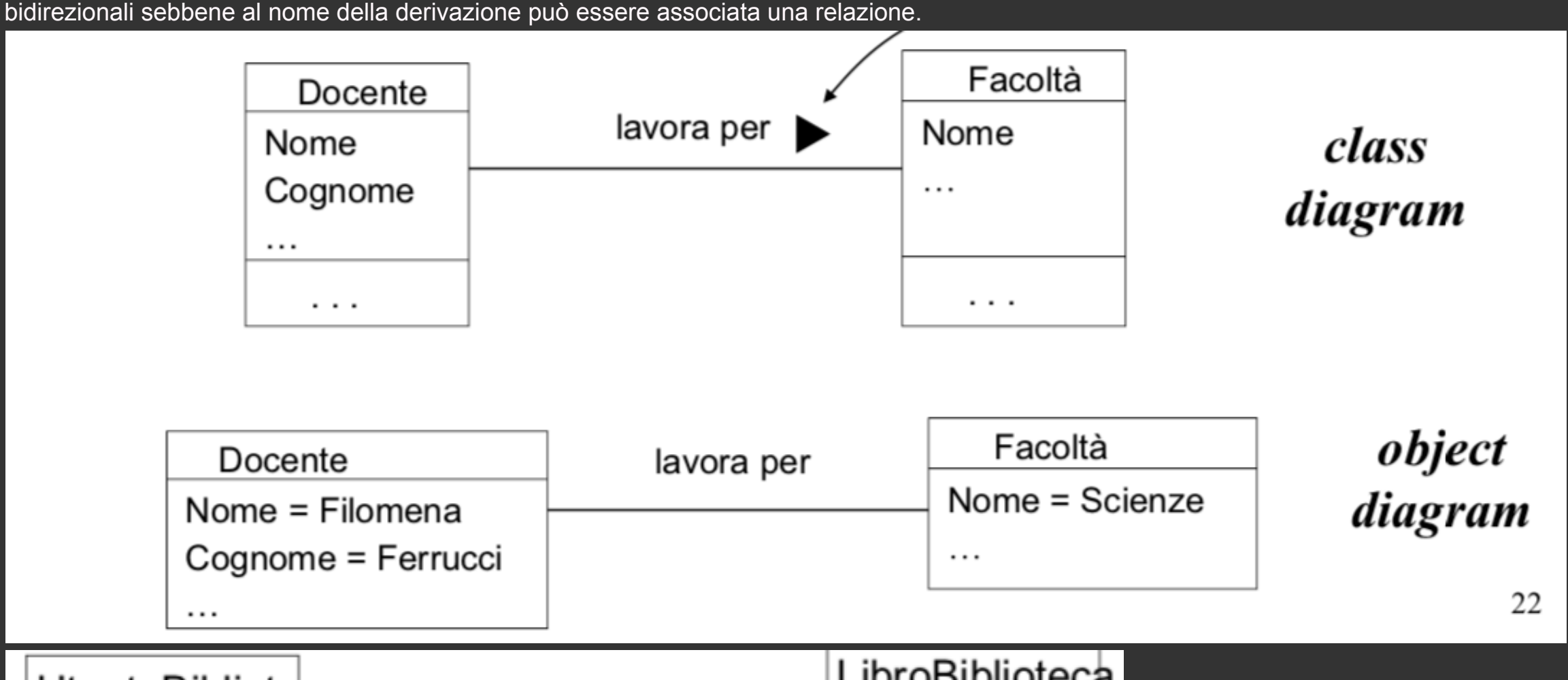
- isQuery: l'esecuzione dell'operazione lascia lo stato del sistema immutato

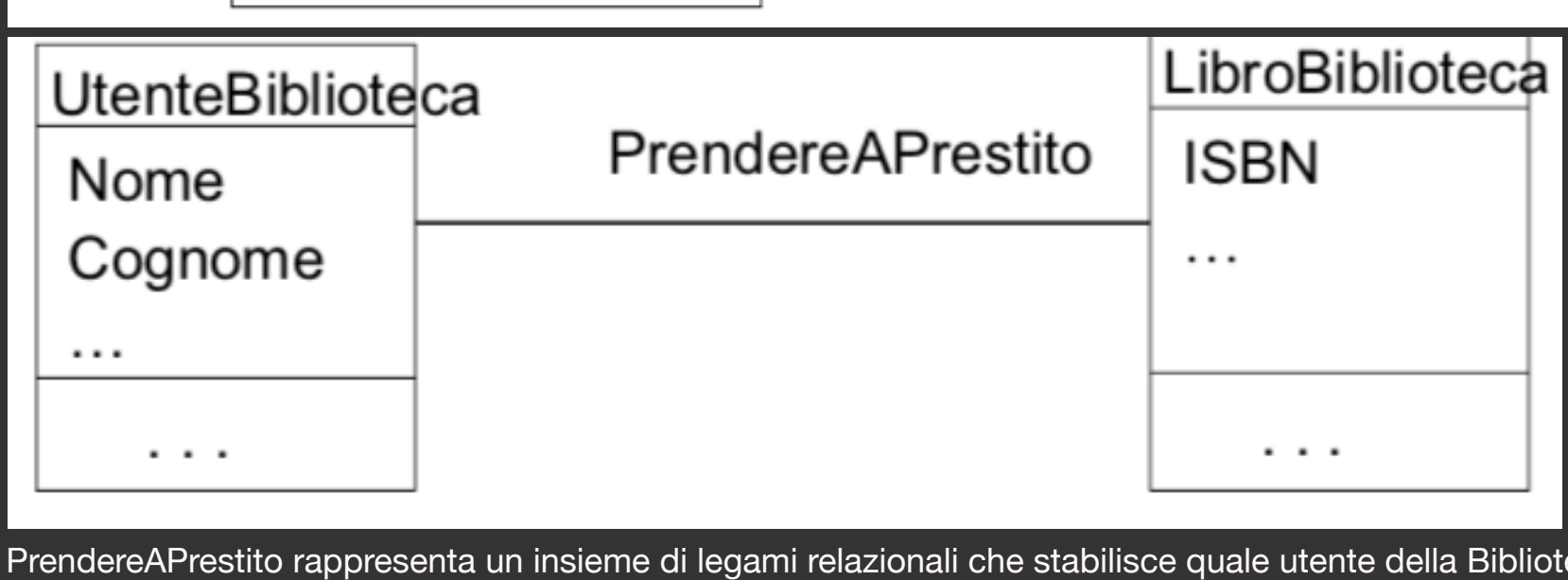
Proprietà che riguardano la concorrenza, rilevanti solo in presenza di oggetti attivi, processi o threads:

- 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 rappresenta una relazione tra "oggetti" la cui conoscenza deve essere preservata per un certo periodo di tempo. Esempio: Filomena Ferrucci lavora per Facoltà di Scienze.
Un'associazione descrive un gruppo di legami delle "classi" aventi struttura e semantica comuni. Esempio: Docente lavora per Facoltà. Esso deve avere un nome, solitamente un verbo. Sono inoltre bidirezionali sebbene al nome della derivazione può essere associata una relazione.





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, del tipo Alfredo sta prendendo a prestito Alice nel paese delle meraviglie, Carla sta prendendo a prestito Biancaneve e i sette nani, domani l'associazione potrebbe contenere 8 legami. Tutto questo per dire che l'associazione rappresenta un gruppo di legami il cui numero varia.

Molteciplità delle associazioni

La molteciplità dice:

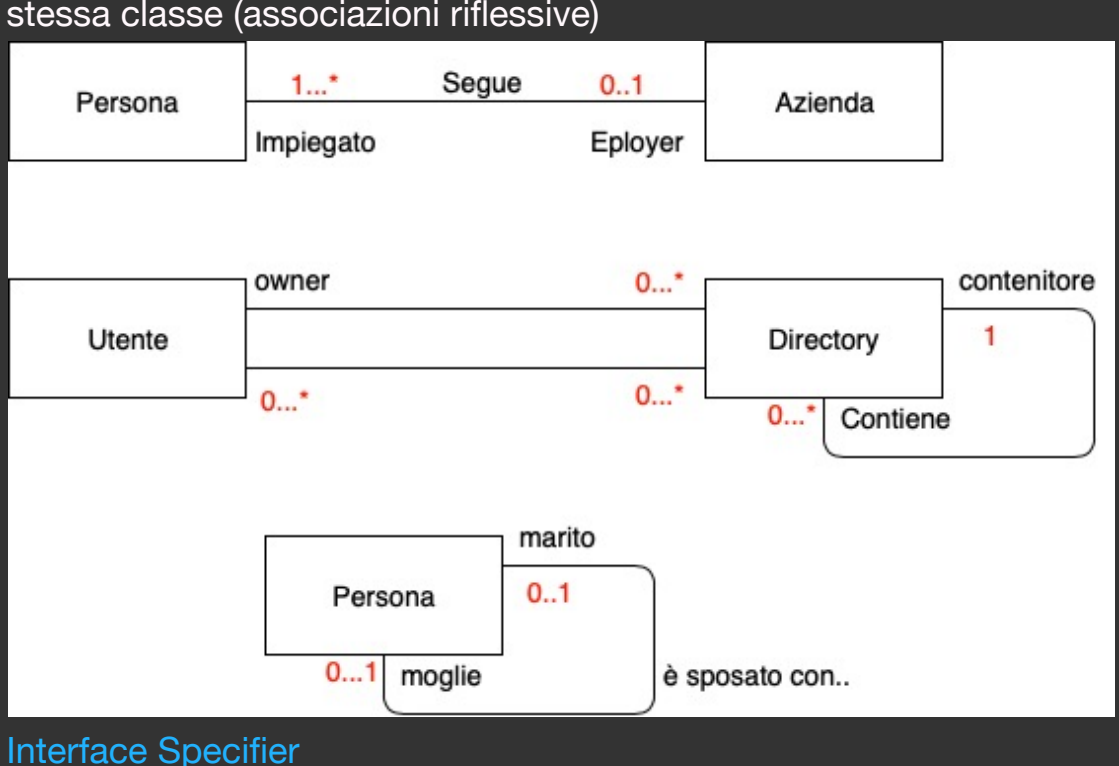
- Se l'associazione è obbligatoria oppure no;
- Il numero minimo e massimo di oggetti che possono essere relazionati ad un altro oggetto;

Esempio:

- 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)

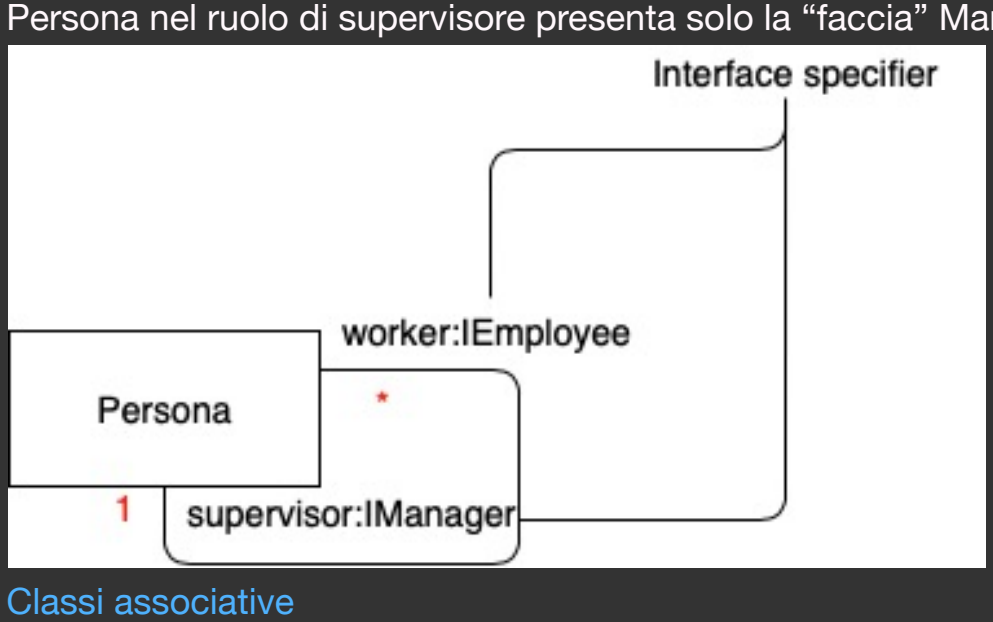
Ruoli

I ruoli forniscono una modalità per attraversare relazioni da una classe ad un'altra. Possono essere usati in alternativa ai nomi delle associazioni. Sono spesso usati per relazioni tra oggetti della stessa classe (associazioni riflessive)



Interface Specifier

Una association può avere un interface specifica, per specificare quale parte dell'interfaccia di una classe è mostrata da questa nei confronti di un'altra classe della stessa associazione; Una classe Persona nel ruolo di supervisore presenta solo la "faccia" Manager al worker; mentre una persona nel ruolo di worker presenta solo la "faccia" IEmployee al supervisor.

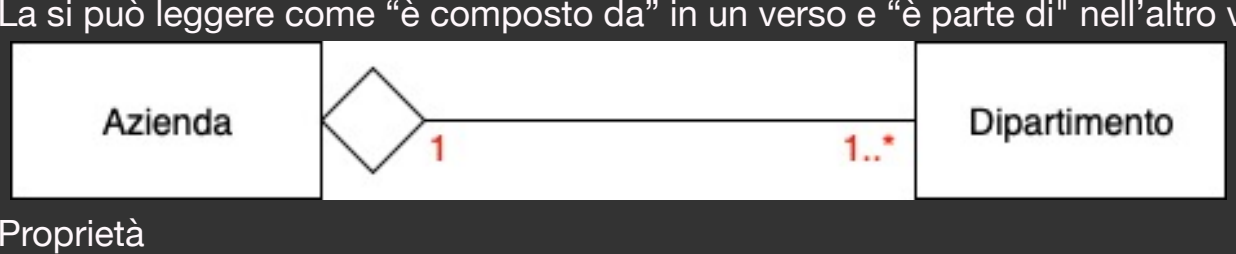


Classi associative

Sono utilizzate per modellare proprietà delle associazioni dato che vi potrebbe essere la presenta degli attributi riferenti all'associazione e non agli oggetti. 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 può leggere come "è composto da" in un verso e "è parte di" nell'altro verso.Esempio: l'azienda è composta da tot azienda e il dipartimento è parte di una azienda.



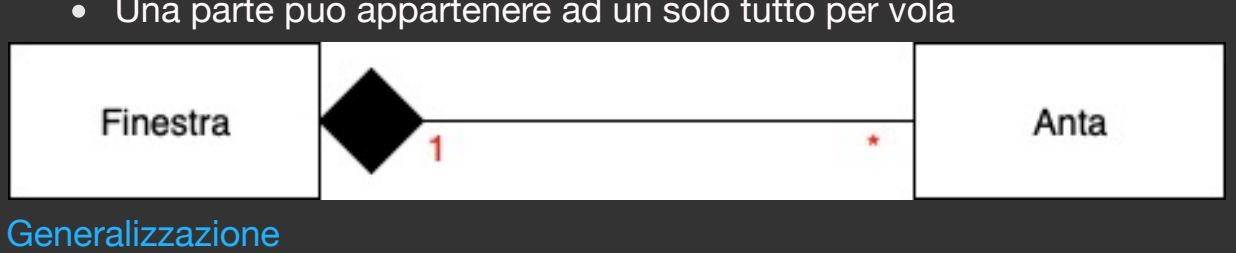
Proprietà

- 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.

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 vola



Generalizzazione

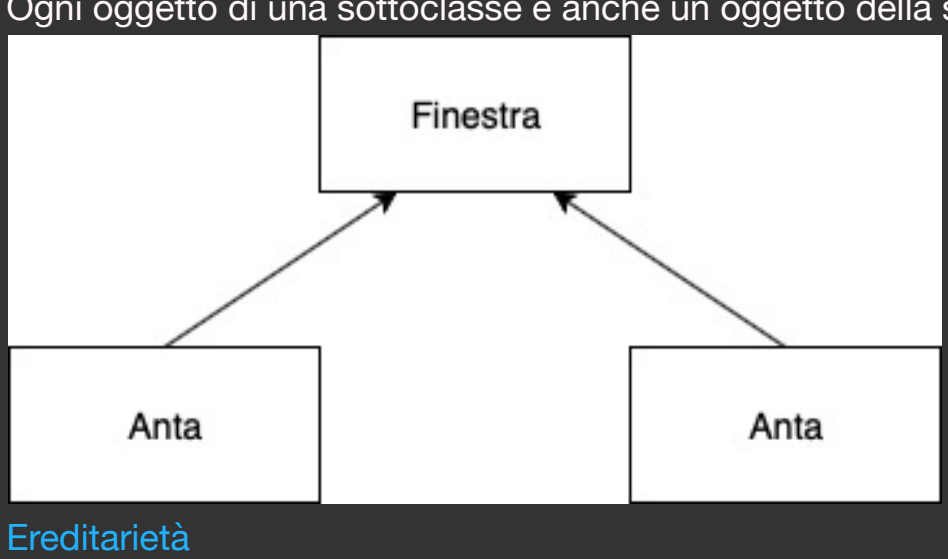
La reazione di generalizzazione rappresenta una tassonomia delle classi

- la classe generale è detta superclasse
- ogni classe specializzata è detta sottoclasse

Può essere letta come

- è un tipo di (verso di generalizzazione)
- può essere un (verso di specializzazione)

Ogni oggetto di una sottoclasse è anche un oggetto della sua superclasse



Ereditarietà

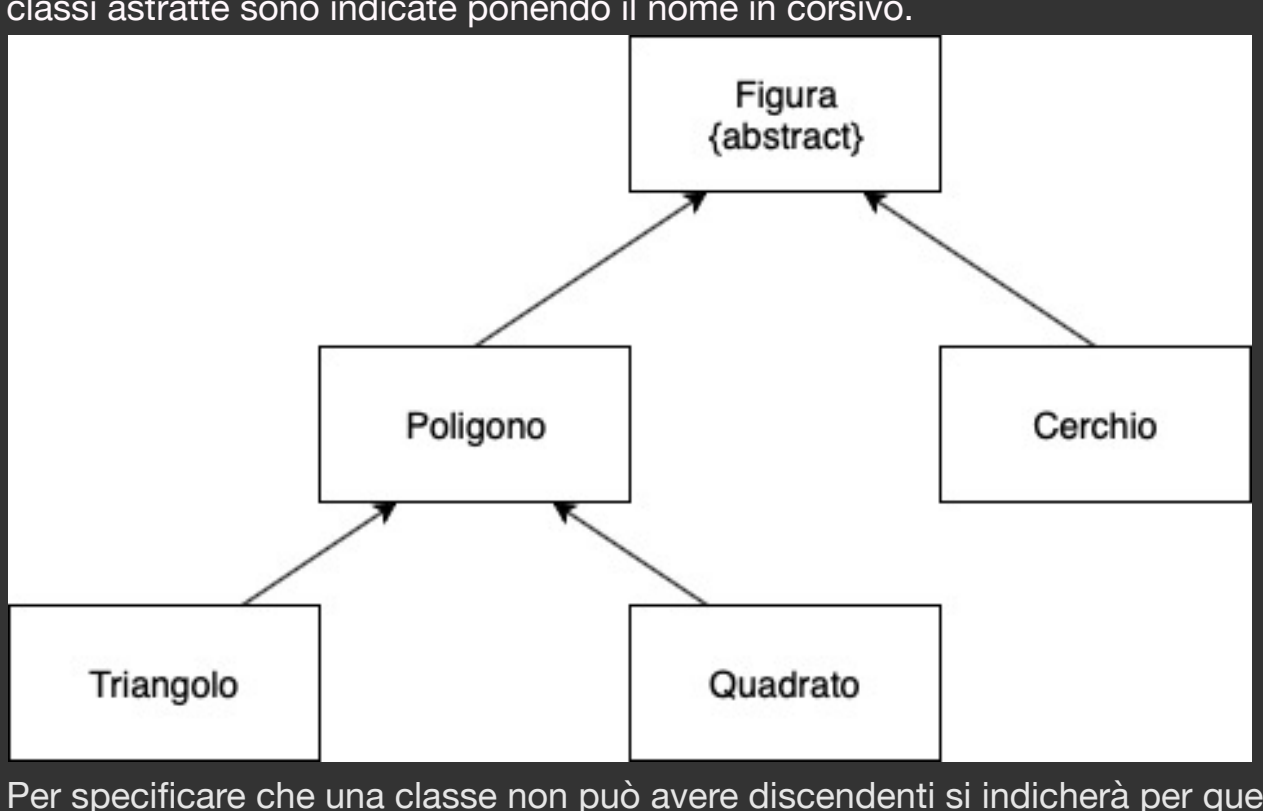
L'ereditarietà è un meccanismo di condivisione delle proprietà degli oggetti in una gerarchia di generalizzazione. Tutte le proprietà di una superclasse possono essere applicati alle sottoclassi.

- Generalizzazione ed ereditarietà godono della proprietà transitiva
- E' possibile definire nuove proprietà per le sottoclassi
- E' possibile ridefinire le proprietà ereditate (overriding)

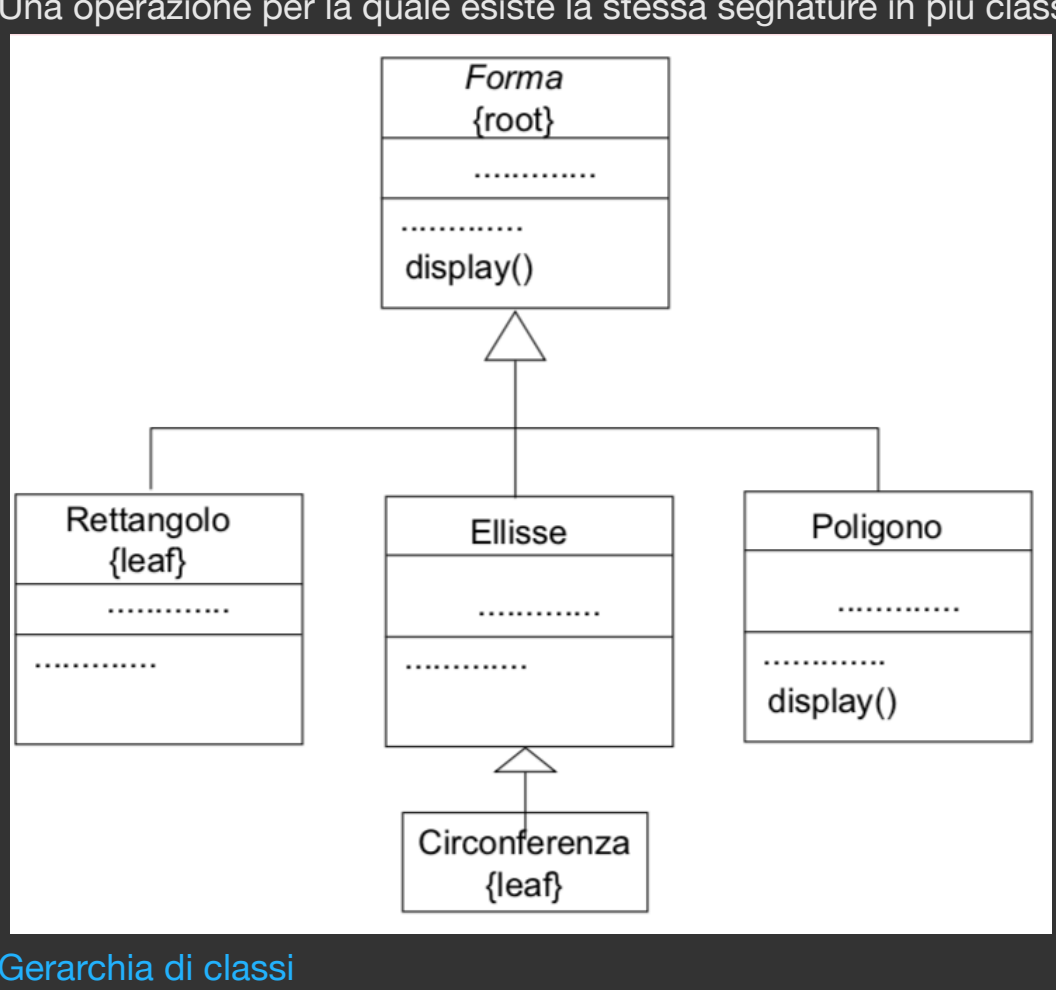
Anche le relazioni di una superclasse valgono per le sottoclassi.

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.



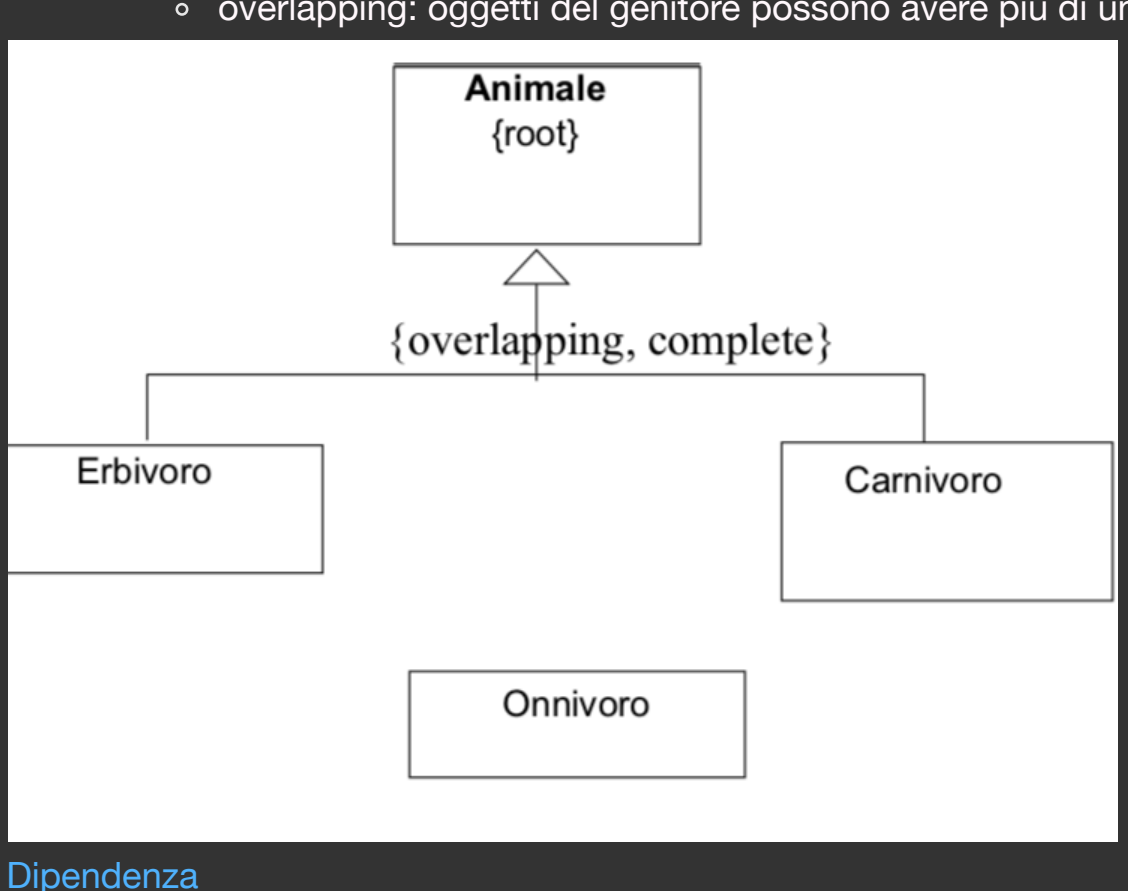
Per specificare che una classe non può avere discendenti si indicherà per questa la proprietà leaf sotto il nome della classe
Per specificare che una classe non può avere antenati si indicherà che per questa la proprietà root sotto il nome della classe
Per indicare che una operazione è astratta il suo nome sarà scritto in corsivo.
Una operazione per la quale esiste la stessa segnature in più classi di una gerarchia è "polimorfico".



Gerarchia di classi

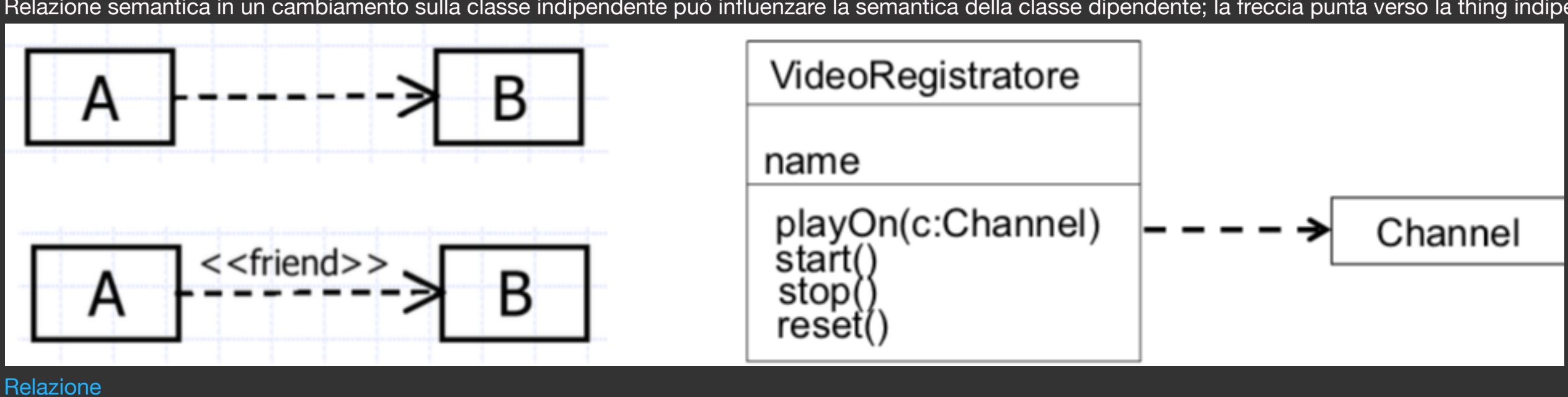
UML definisce

- 1 stereotipo
  - "implementation: la sottoclasse eredita l'implementazione della superclasse ma non rende pubblica né supporta la sua interfaccia.
- 4 contrasti
  - 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



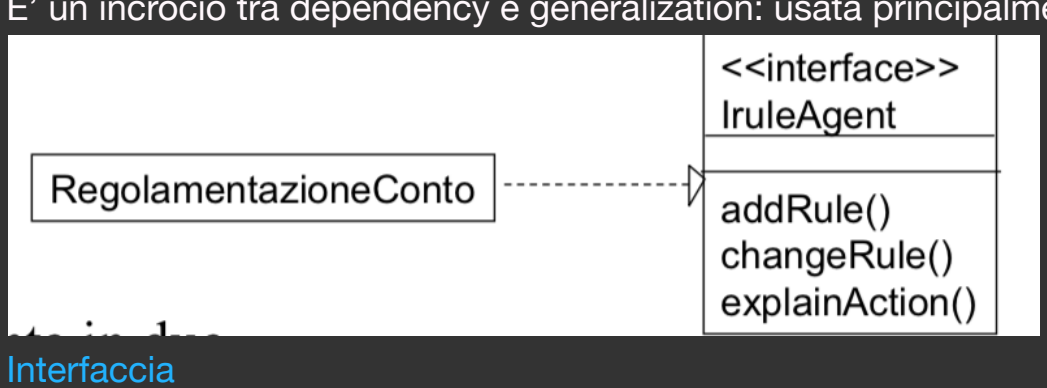
Dipendenza

Relazione semantica in un cambiamento sulla classe indipendente può influenzare la semantica della classe dipendente; la freccia punta verso la thing indipendente.



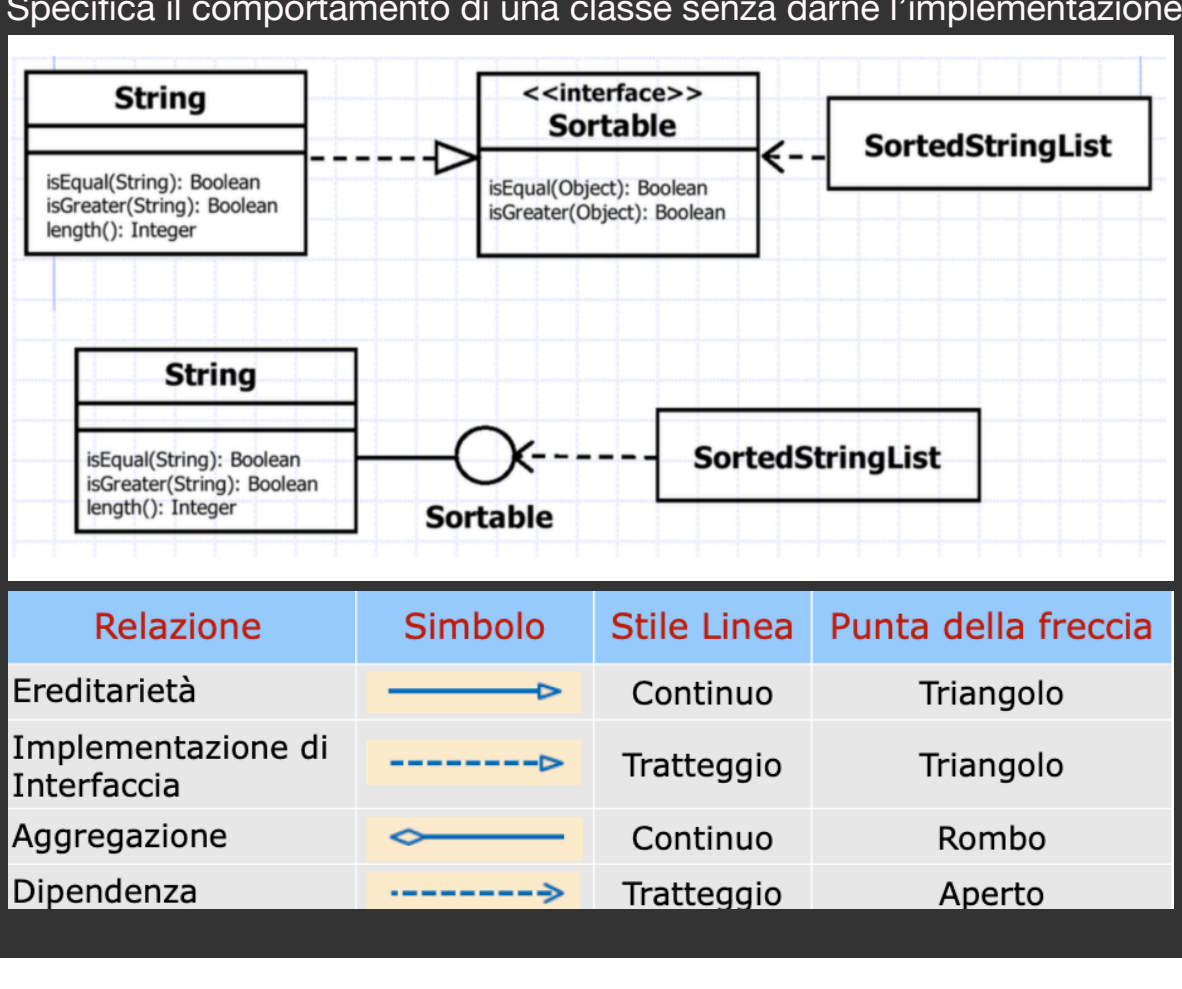
Relazione

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 interfaccia e delle collaborazioni.



Interfaccia

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



Relazione	Simbolo	Stile Linea	Punta della freccia
Ereditarietà	——▷	Continuo	Triangolo
Implementazione di Interfaccia	- - - -▷	Tratteggio	Triangolo
Aggregazione	◇——	Continuo	Rombo
Dipendenza	- - - ->	Tratteggio	Aperto