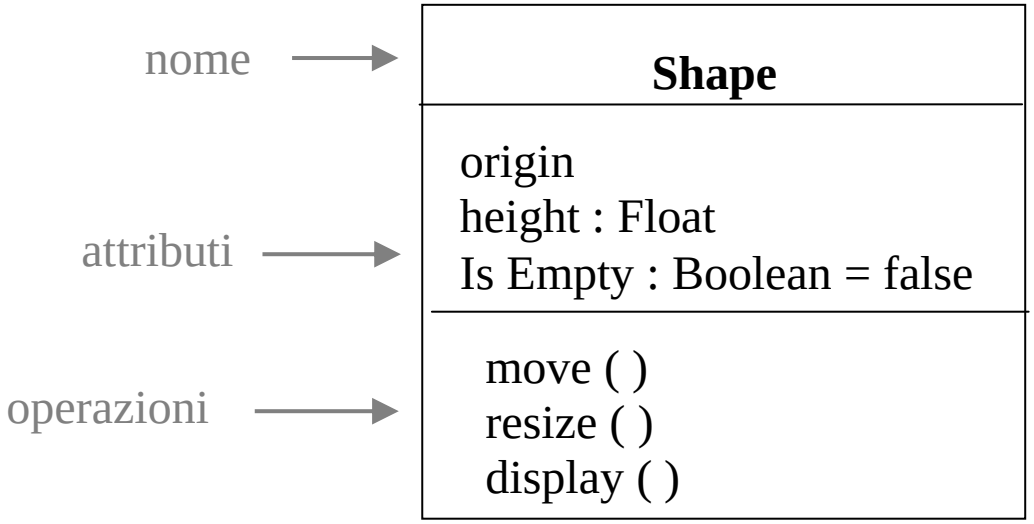
Diagrammi delle classi

- Tecnica centrale di modellazione OO
- Descrizione strutturale statica degli oggetti che compongono il sistema (comprensiva di attributi e operazioni) e delle loro relazioni (restrizioni incluse)
- Descrizione dei vincoli
- Sono simili ai modelli per i dati $\rightarrow$ è facile basare erroneamente il loro sviluppo sui dati piuttosto che sulle responsabilità

Concetti essenziali

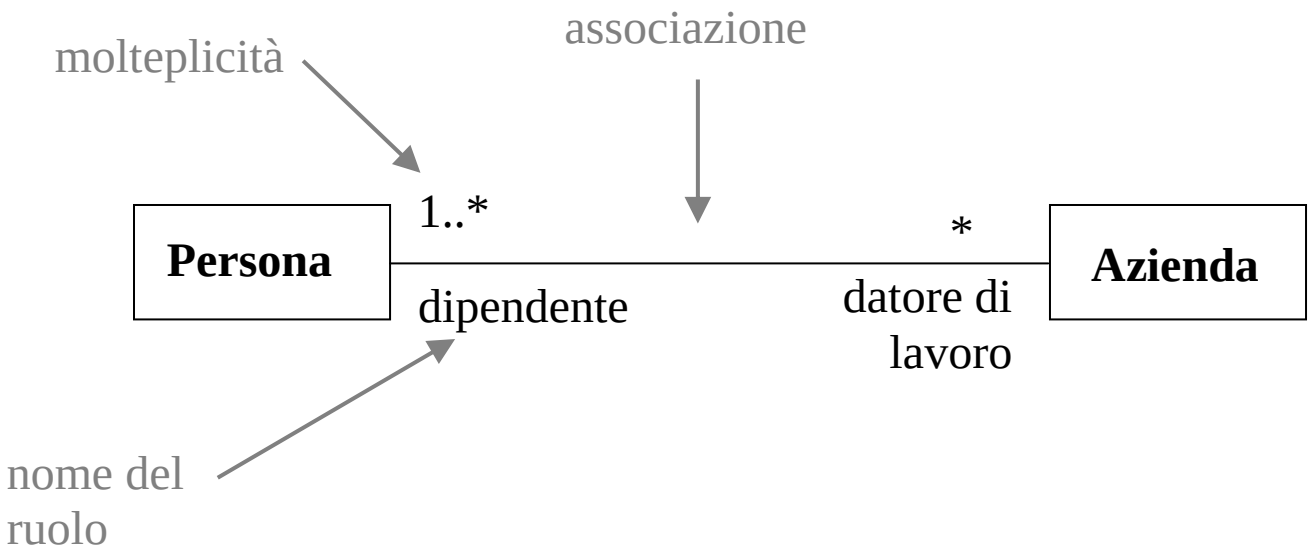
Elementi	Sintassi	Semantica
Classe	Scatola suddivisa in tre parti orizzontali, contenenti rispettivamente *nome*, *attributi* (lo stato) e *operazioni* (il comportamento); *nome* è una stringa in grassetto che identifica univocamente la classe e può essere preceduta dal nome del package che contiene la classe (es. java::awt::Rectangle); la sezione del *nome* è l'unica obbligatoria	Insieme di oggetti che condividono gli stessi attributi, operazioni, relazioni e semantica



Concetti essenziali (cont.)

Elementi	Sintassi	Semantica
Attributo	*visibilità nome: tipo [molteplicità] = valore-di-default {stringa-elenco-proprietà-aggiuntive}* dove solo *nome*, che è una stringa, è obbligatorio `Es. – titolo: String [1] = "UML distilled" {readOnly}`	*default* = valore in un oggetto appena creato, se non specificato diversamente durante la creazione Se la stringa delle proprietà aggiuntive non è presente, generalmente l'attributo è modificabile
Visibilità (di attributi e operazioni)	+ (pubblica) - (privata) # (protected) ~ (package)	• Gli elementi pubblici possono essere usati da un'altra classe, quelli privati invece sono riservati all'uso interno • Per la semantica precisa degli identificatori bisogna fare riferimento alle regole del linguaggio di programmazione adottato

Associazione

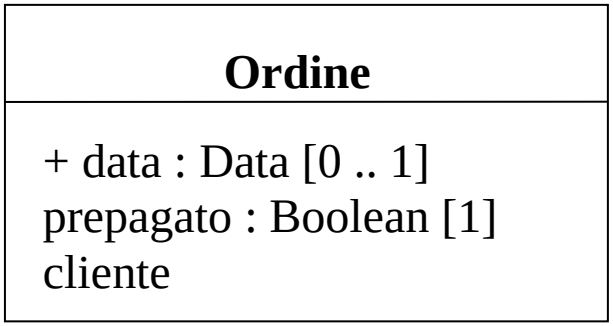


Concetti essenziali (cont.)

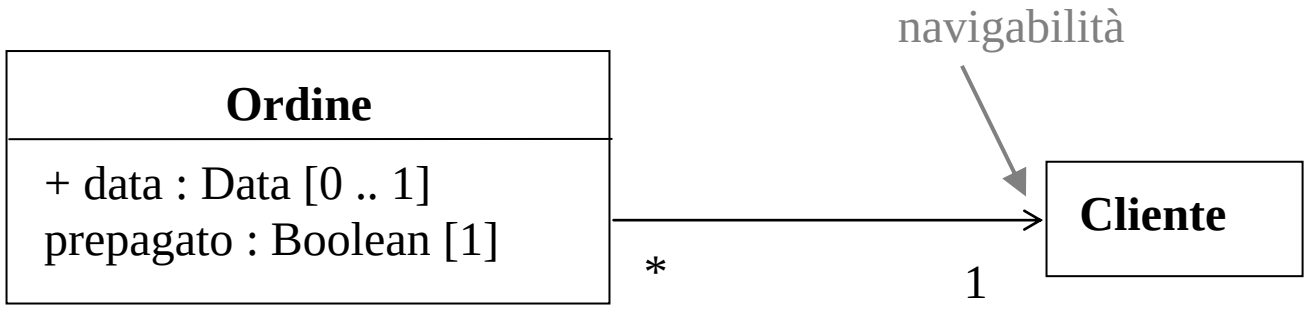
Elementi	Sintassi	Semantica
Associazione (o ruolo)	Linea continua che unisce due classi; opzionalmente ciascun capo della linea può terminare con una punta biforcuta, che indica la *navigabilità*, e può essere etichettato con: • *molteplicità* (es. 1, *, 0..1) • *nome del ruolo* (stringa da specificare solo quando fa aumentare la comprensibilità, altrimenti il nome implicito del ruolo è quello della classe attaccata al capo)	• Relazione statica fra le istanze di due classi; • la *molteplicità* è il numero di oggetti che prendono parte a tale relazione; • la *navigabilità* è il verso di percorribilità del collegamento (e non è rilevante nel modello concettuale); • l'assenza dell'indicazione di navigabilità si interpreta o come mancanza di info circa la stessa o come associazione bidirezionale, che è difficile da implementare perché richiede che le due proprietà siano sempre "sincronizzate"

Proprietà di una classe

Proprietà espressa come attributo



Proprietà espressa come associazione



Molteplicità delle proprietà

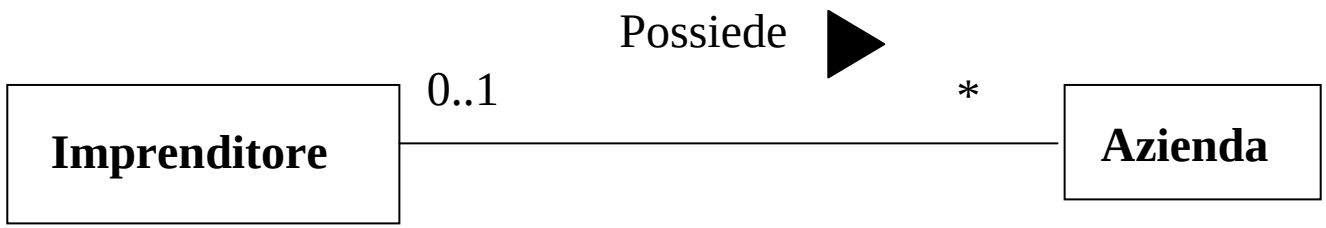
- Si indicano gli estremi inferiore e superiore di un intervallo, come 2..4, dove * rappresenta un valore illimitato
- 1 equivale a 1..1
- * è un'abbreviazione per 0..*
- Se una proprietà ha più valori, è preferibile indicare il suo nome in forma plurale
- Gli elementi di una molteplicità a più valori formano un insieme; se ciò non è accettabile, è necessario dare un'indicazione diversa come `{ordered}`, `{nonunique}` o `{bag}`

Molteplicità delle proprietà

Elementi	Sintassi	Semantica

Vincoli circa una molteplicità a più valori	`{ordered}`
	`{nonunique}`
	`{bag}`
	`{hierarchy}`
	`{dag}`
		Possono coesistere anche più vincoli (purché matematicamente consistenti)
		• `{ordered}`: esiste un ordinamento fra gli elementi
		• `{nonunique}`: gli elementi possono comparire più volte
		• `{bag}`: gli elementi formano un multinsieme
		• `{hierarchy}`: gli elementi formano una gerarchia
		• `{dag}`: gli elementi formano un DAG (grafo orientato aciclico)

Verbo come nome di associazione



Diagrammi delle classi: concetti essenziali (cont.)

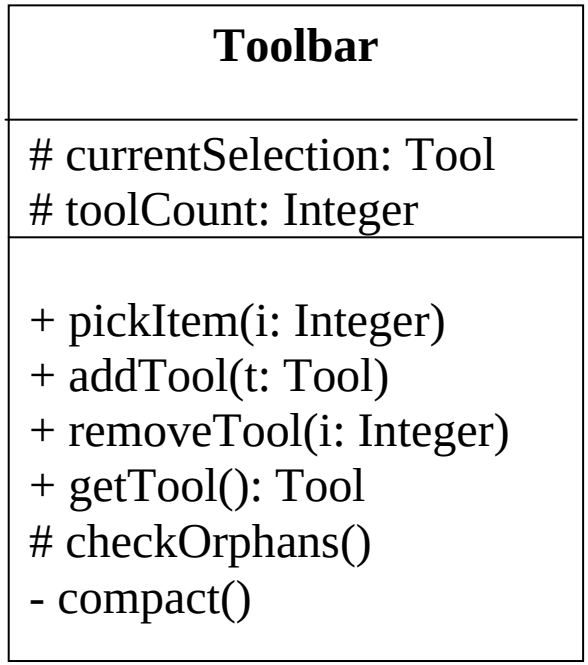
Elementi	Sintassi	Semantica
Operazione	*visibilità nome (lista-parametri): tipo-ritornato {stringa-proprietà}* dove • *nome* è una stringa, • *lista-parametri* è una serie di parametri, ciascuno specificato mediante un elemento *direzione nome: tipo = valore-di-default* e separato dal successivo mediante una virgola (la direzione può assumere i valori *in*, *out*, *inout*, se manca si presume sia *in*), • *stringa-proprietà* indica i valori delle proprietà dell'operazione, cioè *query*, *modificatore*, *get* e *set*	Azione che una classe sa come eseguire; • se un'operazione *i*) non modifica lo stato del sistema e *ii*) ritorna dei valori è una *query*, • se un'operazione *i*) modifica lo stato osservabile del sistema e *ii*) non ritorna valori è un *modificatore*; • un metodo *get* è una particolare *query* che restituisce il valore di un attributo e non fa nient'altro; • un metodo *set* è un particolare *modificatore* che assegna il valore a esso passato a un attributo e non fa nient'altro Una convenzione comune è cercare di scrivere modificatori che non restituiscano mai valori

Operazioni

- Quelle che manipolano le proprietà della classe di solito si possono dedurre, per cui non sono incluse nel diagramma
- Nei modelli concettuali, dovrebbero indicare le principali responsabilità della classe, magari facendo riferimento a una carta CRC
- Corrispondono alla dichiarazione di una procedura e, quindi, si distinguono (sottilmente) dai metodi, che invece corrispondono al corpo della procedura → le due cose sono diverse in presenza di polimorfismo

Responsabilità di una classe = obbligo che la classe si assume nello svolgere un servizio. Fa parte del contratto che essa stipula con le altre classi

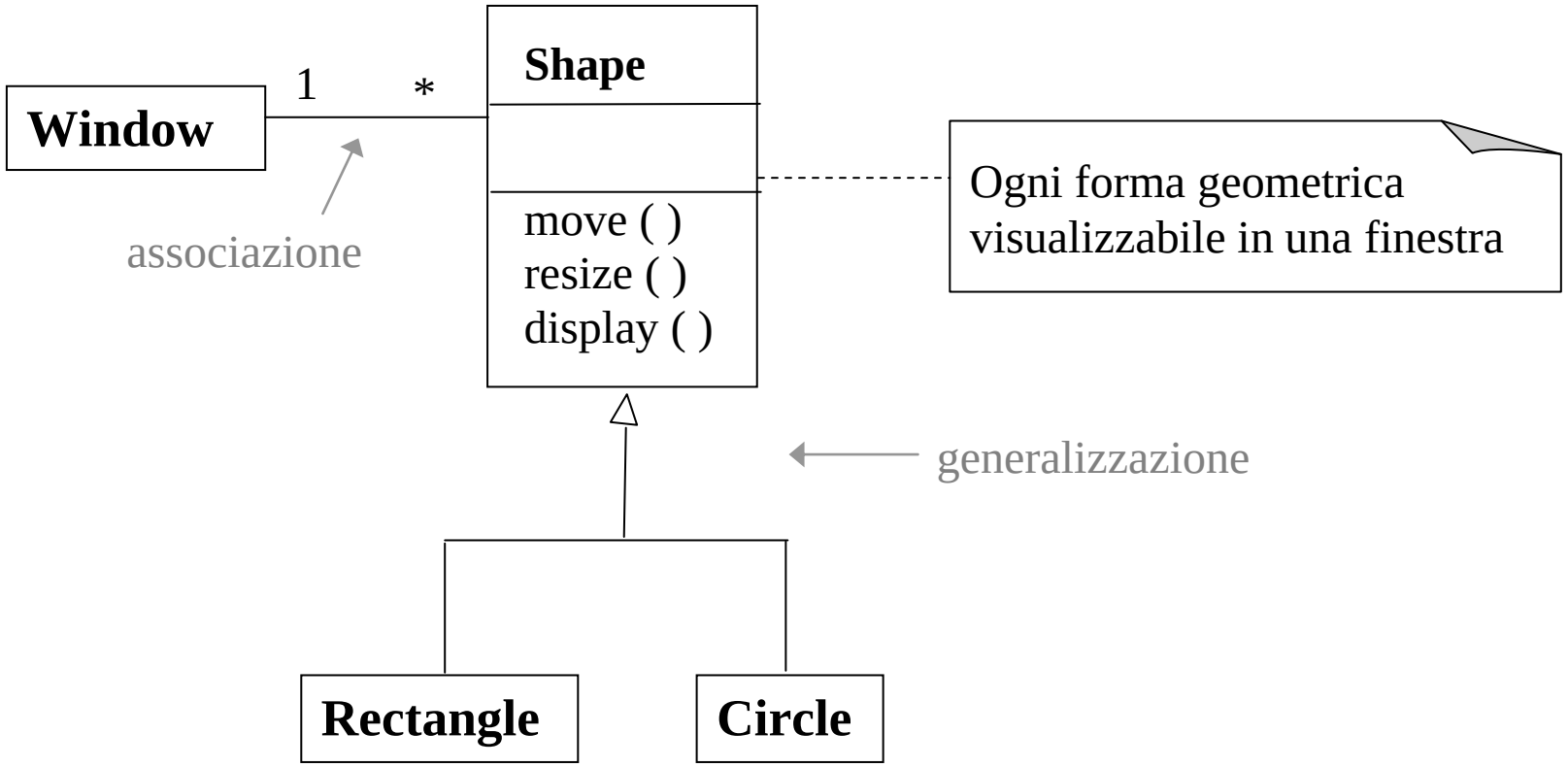
Concetti essenziali (cont.)



Concetti essenziali (cont.)

Elementi	Sintassi	Semantica
Generalizzazione	Freccia con testa vuota e uno o più capi: la testa termina nella superclasse, i capi si dipartono ciascuno da una sottoclasse	Se due o più classi sono diverse ma mostrano anche molte somiglianze, queste si possono raccogliere in una superclasse (relazione IS-A)
Nota	Testo • che segue due trattini −− all'interno di un elemento del diagramma, oppure • contenuto in una scatola con "orecchietta", eventualmente collegata all'elemento a cui si riferisce mediante una linea tratteggiata	Commento aggiuntivo Una o più note possono apparire in qualsiasi tipo di diagramma UML Una convenzione comune prevede di mettere un cerchietto vuoto all'estremità della linea tratteggiata, per meglio collegarla all'elemento a cui si riferisce

Concetti essenziali (cont.)



Prospettive e diagrammi delle classi

Prospettiva	Interpretazione
Concettuale (da adottarsi per realizzare il modello di dominio nella fase di elaborazione del RUP)	classe = concetto nella mente del cliente ($\rightarrow$ il diagramma diventa il linguaggio dell'interlocutore) associazione = relazione fra concetti (la navigabilità non è utile) attributo = proprietà degli oggetti della classe, notazione alternativa rispetto all'associazione operazione = una delle responsabilità principali della classe generalizzazione: qualsiasi cosa si possa dire del supertipo (in termini di associazioni, attributi e operazioni), si può dire anche dei sottotipi

N.B. Le responsabilità di una classe possono essere descritte ciascuna mediante una nota inserita entro il riquadro della classe

Prospettive e diagrammi delle classi (cont.)

Prospettiva	Interpretazione
Specifica	classe = tipo (un tipo è l'interfaccia – o parte di essa - di una o più classi "fisiche", una classe "fisica" può implementare più tipi) associazione = responsabilità (indipendenti dalla struttura dati) ovvero, per ogni relazione R fra due classi C1 e C2 con navigabilità C1→C2, esistenza di un'interfaccia (implicita, che, a livello implementativo può essere realizzata in vario modo) che • dato un elemento di C1, determina tutti gli elementi di C2 che partecipano a R • effettua l'aggiornamento della relazione

Prospettive e diagrammi delle classi (cont.)

Prospettiva	Interpretazione
Specifica (cont.)	attributo = responsabilità: esiste un'interfaccia (implicita) che • consente di impostare il valore dell'attributo di un oggetto • restituisce il valore dell'attributo operazione = metodo pubblico generalizzazione = creazione di sottotipi (o ereditarietà di interfacce): l'interfaccia del sottotipo deve includere tutti gli elementi dell'interfaccia del supertipo (si dice che deve essere conforme all'interfaccia del supertipo); deve essere possibile sostituire un'istanza della sottoclasse all'interno di qualsiasi pezzo di codice che preveda l'esistenza di un'istanza della superclasse e tutto deve continuare a funzionare N.B. la creazione di sottoclassi "fisiche" è solo uno dei modi per implementare la creazione di sottotipi, un altro è la delega

Prospettive e diagrammi delle classi (cont.)

Prospettiva	Interpretazione
Implementazione	classe = classe “fisica” associazione R (C1→C2)= dato un qualsiasi oggetto di C1 esistono specifici meccanismi di collegamento verso gli oggetti corrispondenti di C2 la navigabilità può essere diversa da quella di specifica attributo = campo generalizzazione: la sottoclasse eredita tutti i metodi e i campi della superclasse e può ridefinirne le funzioni (eredità di implementazione)

Dipendenza

Un elemento (detto *client*) di un diagramma (di qualsiasi tipo) dipende da un altro (detto *supplier*, cioè fornitore) se la modifica della definizione del secondo può causare un cambiamento del primo

Una classe C1 dipende da un classe C2 se

- invoca operazioni di C2 (anche solo un costruttore), e/o
- un suo campo è di tipo C2, e/o
- un parametro di una sua operazione è di tipo C2, e/o
- accede ai campi di C2

Man mano che il sistema cresce, il problema del controllo delle dipendenze aumenta

Specificare le dipendenze rende espliciti gli effetti a catena del cambiamento di un elemento $\rightarrow$ il sistema è più facilmente modificabile

Concetti essenziali (cont.)

Elementi	Sintassi	Semantica
Dipendenza	Freccia con linea tratteggiata e punta biforcuta	La modifica della classe destinazione (fornitore) può causare un cambiamento della classe sorgente (client) Può legare pure una classe e un interface (o una classe astratta), dove se l'interface (o la classe astratta) dovesse cambiare, anche la classe potrebbe cambiare
Vincolo	Descrizione in linguaggio naturale racchiusa fra parentesi graffe, oppure descrizione in OCL	OCL (Object Constraint Language) fa parte di UML ed è basato sul calcolo dei predicati

Dipendenza (cont.)

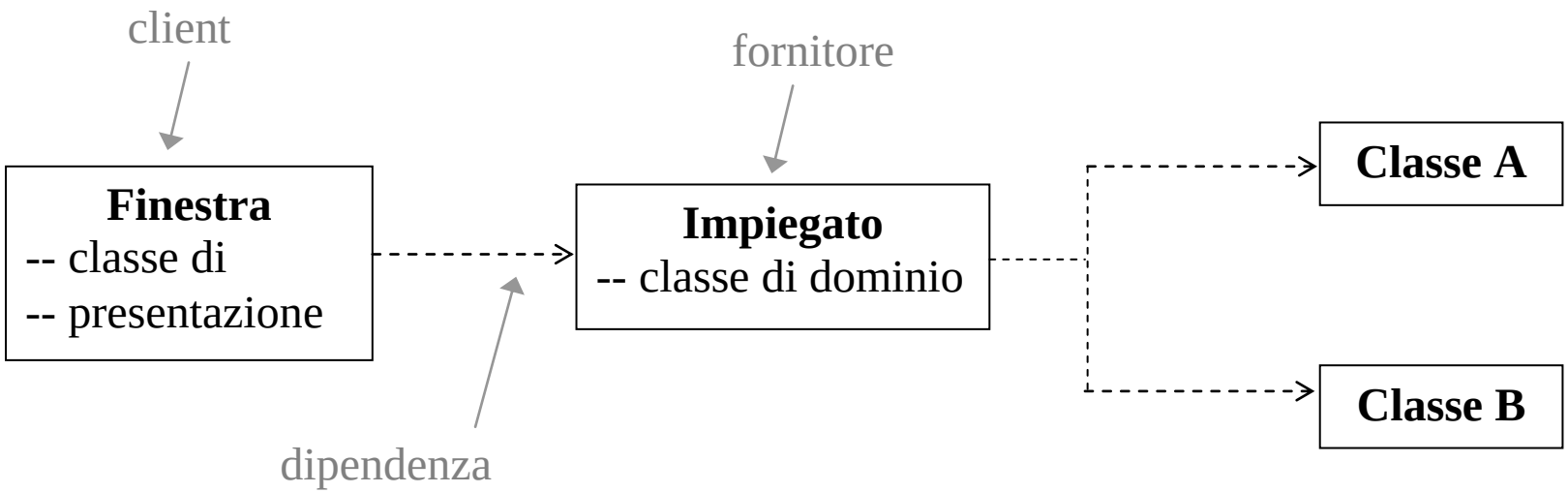
In generale non è una relazione transitiva (ad es. se C1 dipende da C2 che dipende da C3, la relazione non è transitiva se il cambiamento di C2 – conseguente a un qualsiasi cambiamento di C3 – riguarda solo il corpo delle operazioni di C2)

Molte relazioni UML implicano una dipendenza, ad es.

- in un'associazione navigabile C1→C2, C1 dipende da C2
- una sottoclasse dipende dalla superclasse (ma non viceversa)

Dato il codice di un sistema, le dipendenze in esso contenute sono rilevabili automaticamente da uno strumento CASE

Dipendenza e progettazione



Regole di progettazione

Separare presentazione (cioè le classi relative all'interfaccia) dalla logica del dominio (cioè le classi che incarnano il comportamento fondamentale del sistema), con la prima dipendente dalla seconda ma non viceversa

Minimizzare le dipendenze

Evitare i cicli di dipendenze, possibilmente fra classi dello stesso package e assolutamente fra classi di package diversi

Concetti avanzati

Elementi	Sintassi	Semantica
Parola chiave	Parola racchiusa fra virgolette uncinate Es. «`interface`»	Rappresenta un costrutto creato dall'utente perché non compreso in UML ma è simile a un altro che vi è compreso

Parola chiave ≠ stereotipo

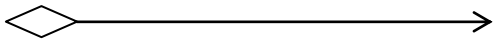
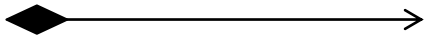
Stereotipo = meccanismo di estensione di UML che agisce sul metamodello

Profilo = estensione di una parte di UML ottenuta per mezzo di un gruppo coerente di stereotipi, adattando così il linguaggio a un particolare dominio (ad es. la modellazione di business)

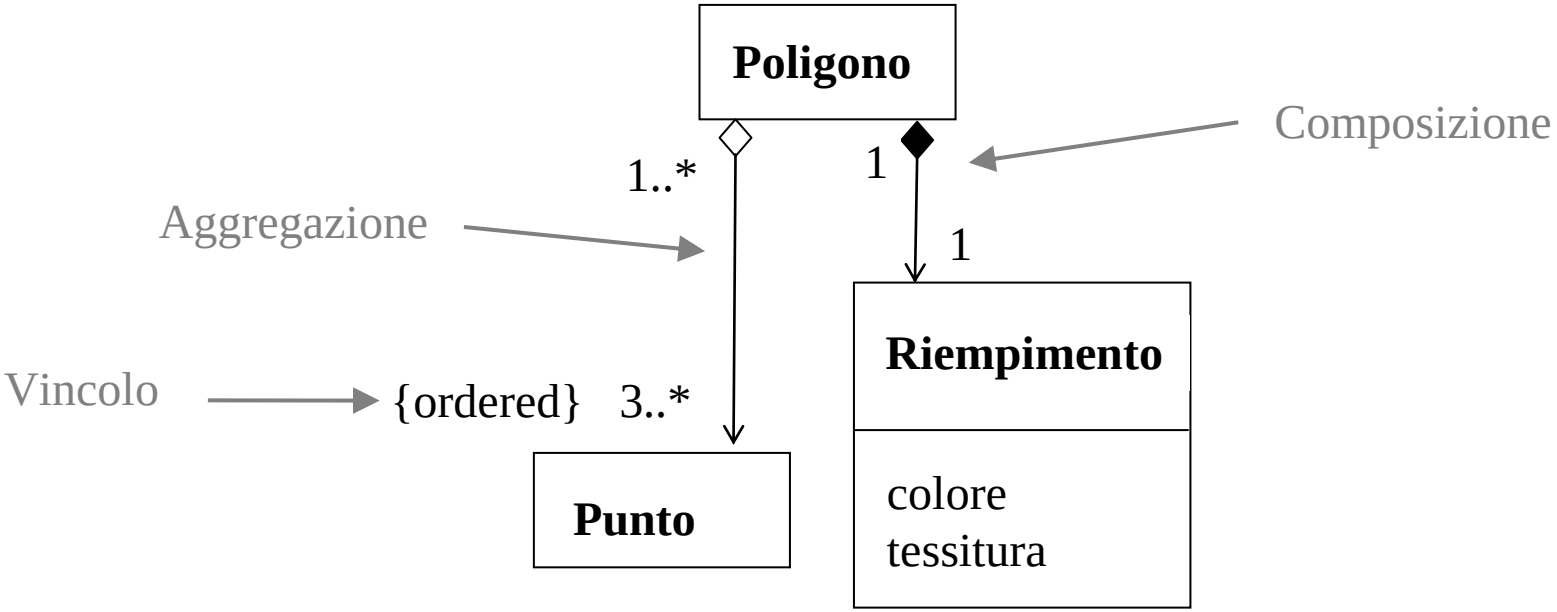
Concetti avanzati (cont.)

Elementi	Sintassi	Semantica
Operazione e attributo statici	Come operazione e attributo d'istanza ma sottolineati	Metodi e attributi con campo d'azione di classe (anziché d'istanza)

Concetti avanzati (cont.)

Elementi	Sintassi	Semantica
Aggregazione	Freccia che termina sulla classe "parte" con una punta biforcuta; all'altra estremità si diparte dalla classe "tutto" con un rombo vuoto; su entrambe le estremità è obbligatoria la molteplicità 	Generica relazione "parte-di" (o HAS-A), difficilmente distinguibile dall'associazione Inclusa in UML senza definirne la semantica → persone diverse la usano con significati diversi
Composizione	Come nell'aggregazione ma il rombo è pieno  La molteplicità sul lato della classe composta è necessariamente 0..1 oppure 1	• Varietà più forte dell'aggregazione: l'oggetto componente può appartenere a un solo oggetto composto (regola di non condivisione) • la cancellazione dell'oggetto composto si propaga a tutti gli oggetti componenti

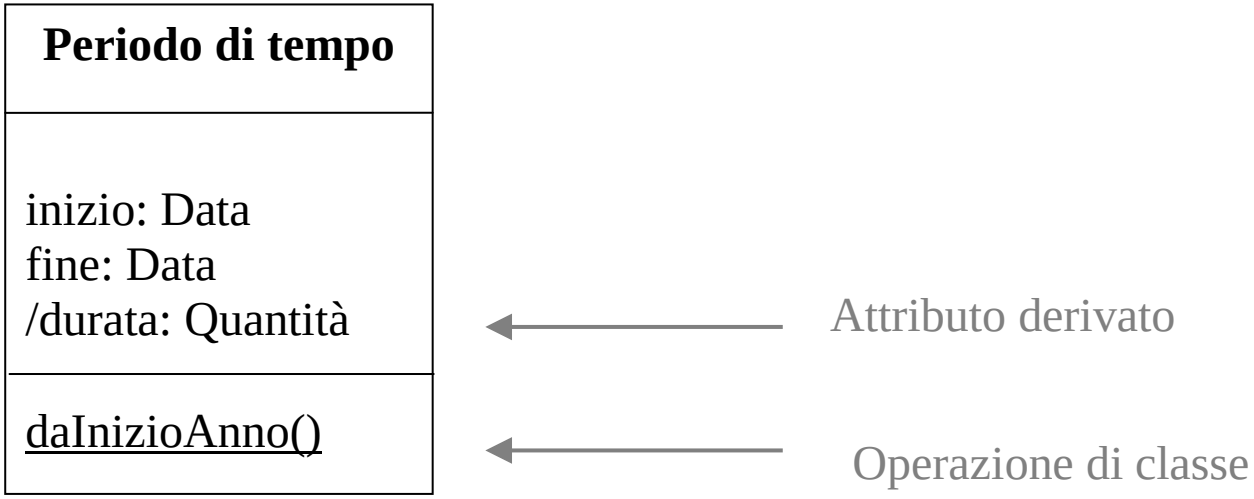
Aggregazione e composizione



Concetti avanzati (cont.)

Elementi	Sintassi	Semantica
Proprietà derivata	Il nome dell'associazione o dell'attributo è preceduto da una barra (/)	• È un attributo/associazione rappresentato esplicitamente anche se può essere derivato a partire da altri attributi/associazioni • dal punto di vista concettuale, indica un vincolo tra valori e può essere usato per ricordarsi di chiedere conferma agli esperti circa una presunta derivabilità • dal punto di vista della specifica, indica un vincolo tra valori (non cosa viene direttamente memorizzato e cosa viene derivato) • dal punto di vista implementativo indica i campi usati come cache per motivi di efficienza

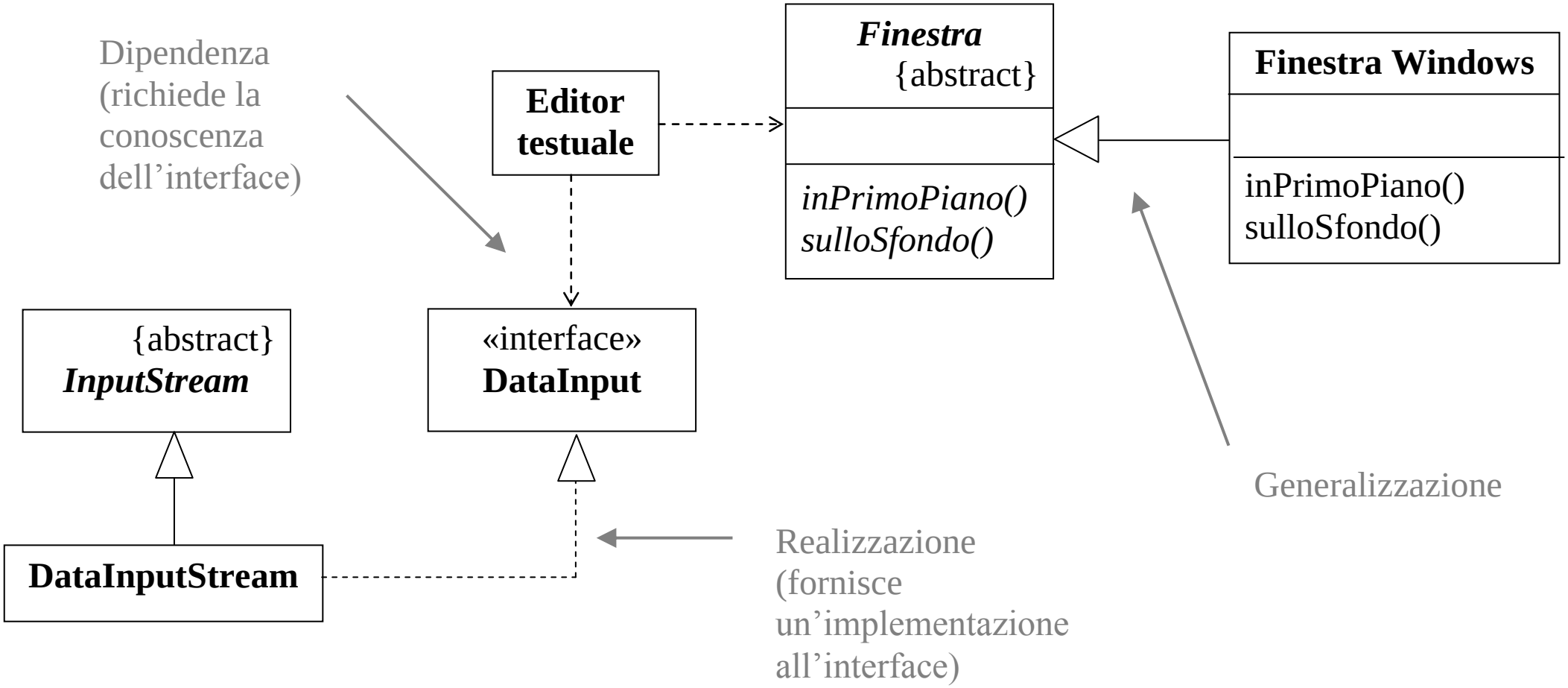
Concetti avanzati (cont.)



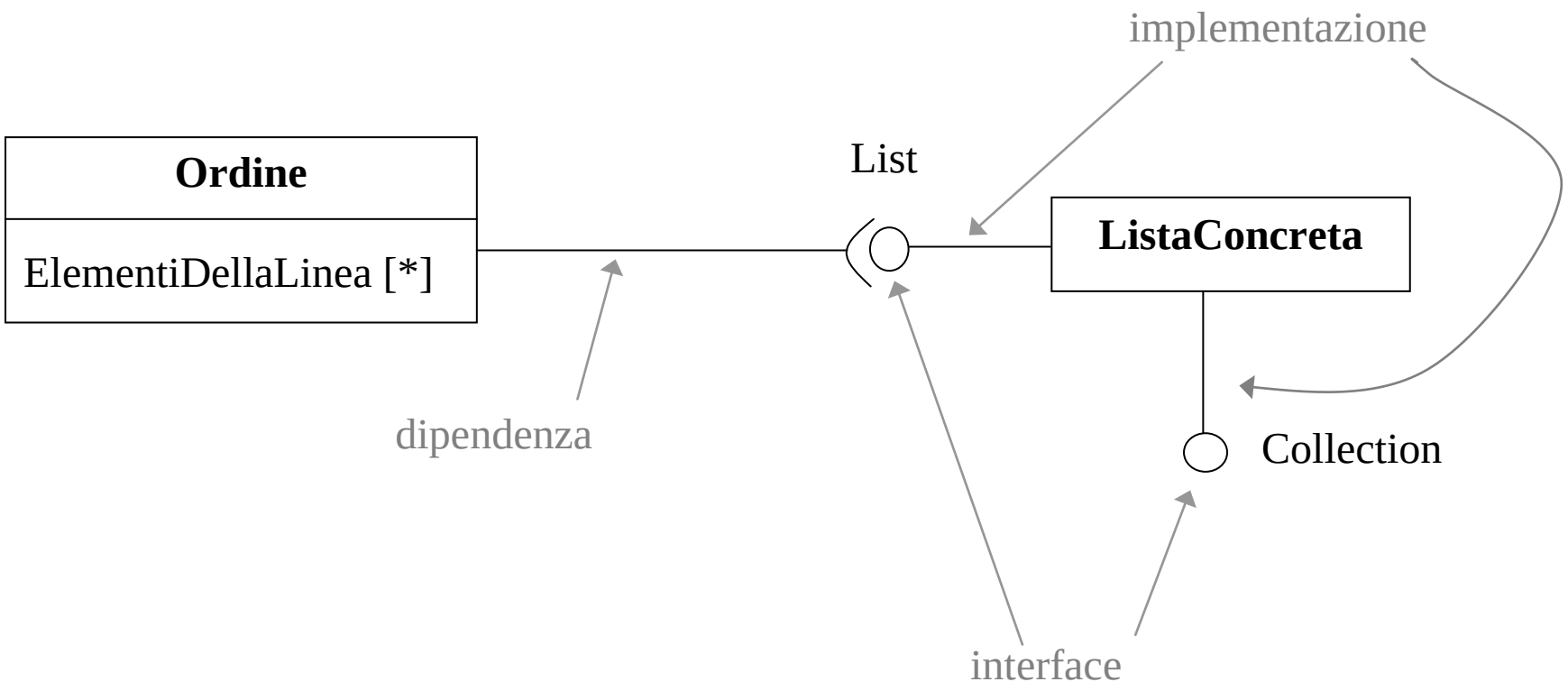
Concetti avanzati (cont.)

Elementi	Sintassi	Semantica
Classe e operazione astratti	Nome in corsivo + eventuale *vincolo* `{abstract}`	Operazione astratta = operazione priva di implementazione Classe astratta = classe non istanziabile direttamente
Interface	Uso della parola chiave «`interface`» entro la scatola della classe	Classe priva di implementazione
Realizzazione (o implementazione)	Freccia con linea tratteggiata e punta vuota diretta dalla classe all'interface	• Relazione fra una classe (concreta o astratta) che implementa un'interface e l'interface stessa • a livello di specifica, realizzazione e subtyping sono indistinguibili

Classi astratte e interface



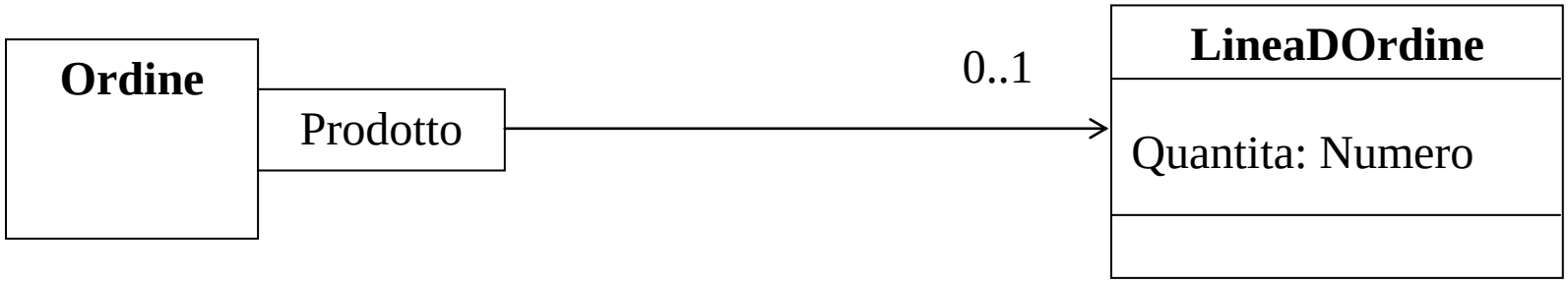
Interface: rappresentazione compatta



Concetti avanzati (cont.)

Elementi	Sintassi	Semantica
Associazione qualificata	Dal riquadro della classe sorgente fuoriesce un riquadro più piccolo, contenente il nome del qualificatore, da cui si diparte la linea continua che rappresenta l'associazione; tale linea termina con una punta che indica la navigabilità	A ogni istanza della classe sorgente corrispondono gli elementi di un array associativo/tabella hash/mappa/dizionario indicizzati dal valore di una chiave (il qualificatore) La molteplicità dell'associazione qualificata va considerata nel contesto del qualificatore (ovvero indica quante istanze della classe destinazione corrispondono a un singolo valore/istanza del qualificatore) Nella modellazione concettuale rappresenta un vincolo

Associazione qualificata



Classificazione

Classificazione = relazione tra un oggetto e il suo tipo; può essere

- singola: un oggetto appartiene a un solo tipo
- multipla: un oggetto può essere descritto da più tipi

Classificazione multipla ≠ ereditarietà multipla

Concetti avanzati (cont.)

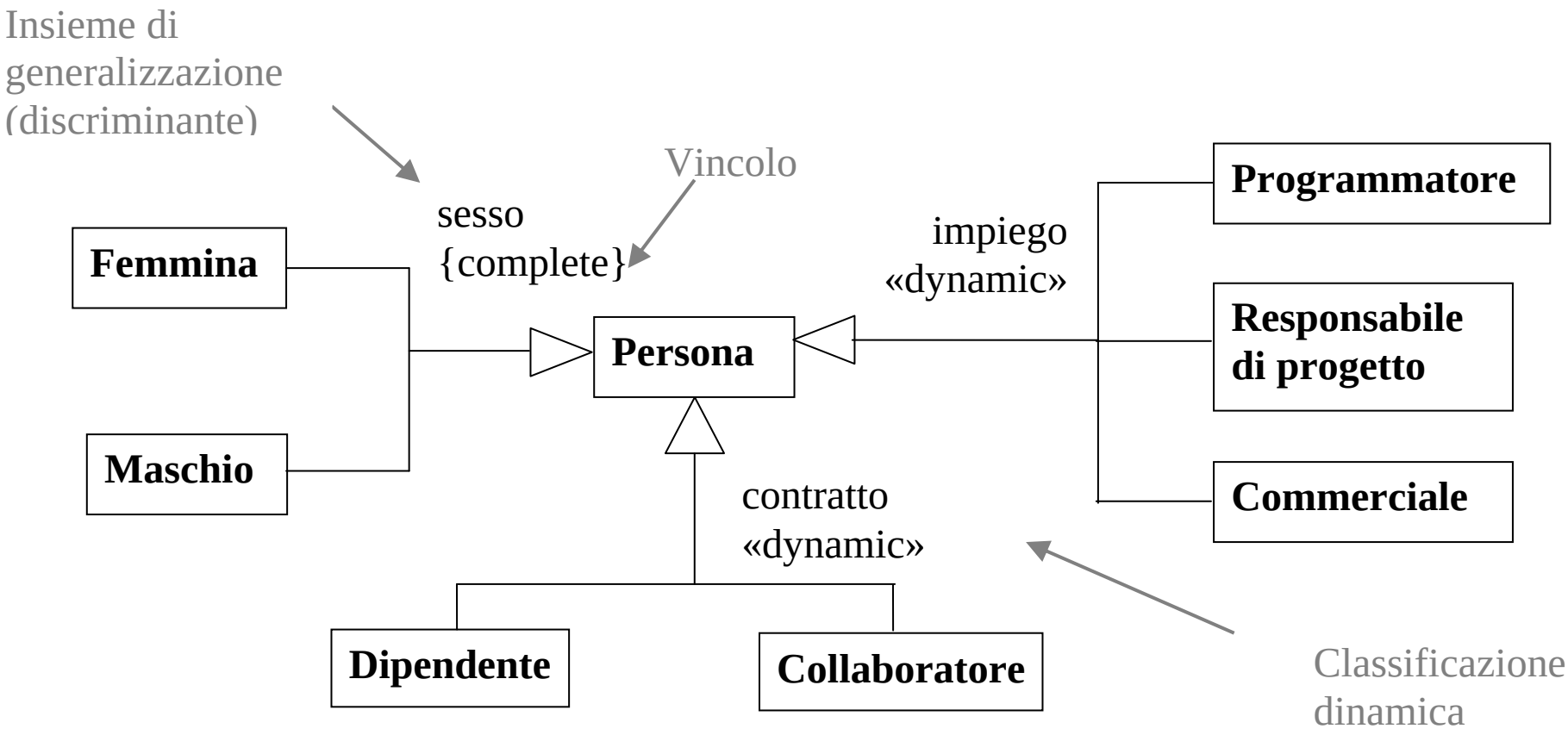
Elementi	Sintassi	Semantica
Classificazione multipla (subtyping)	• Come la generalizzazione, ma etichettata con il nome di un *insieme di generalizzazione* (discriminante), che indica la base della classificazione**;** • il discriminante è accompagnato dal *vincolo* `{complete}` se ogni istanza della superclasse deve essere anche un'istanza di uno dei sottotipi del gruppo così contraddistinto; • ogni combinazione fra sottotipi aventi discriminante distinto deve essere legale; • tutti i sottotipi aventi lo stesso discriminante (gruppo di sottotipi) devono essere disgiunti	• Associa a una classe (la superclasse) più sottotipi ortogonali (le sottoclassi) • utile a livello di modellazione concettuale

Concetti avanzati (cont.)

Elementi	Sintassi	Semantica
Classificazione dinamica	Come la classificazione multipla ma l'insieme di generalizzazione è accompagnato dalla parola chiave «`dynamic`»	• Consente agli oggetti di cambiare tipo all'interno di una struttura di sottotipi • utile a livello di modellazione concettuale

Suggerimento: usare sempre una classificazione singola e statica (che corrisponde all'uso di un singolo anonimo insieme di generalizzazione)

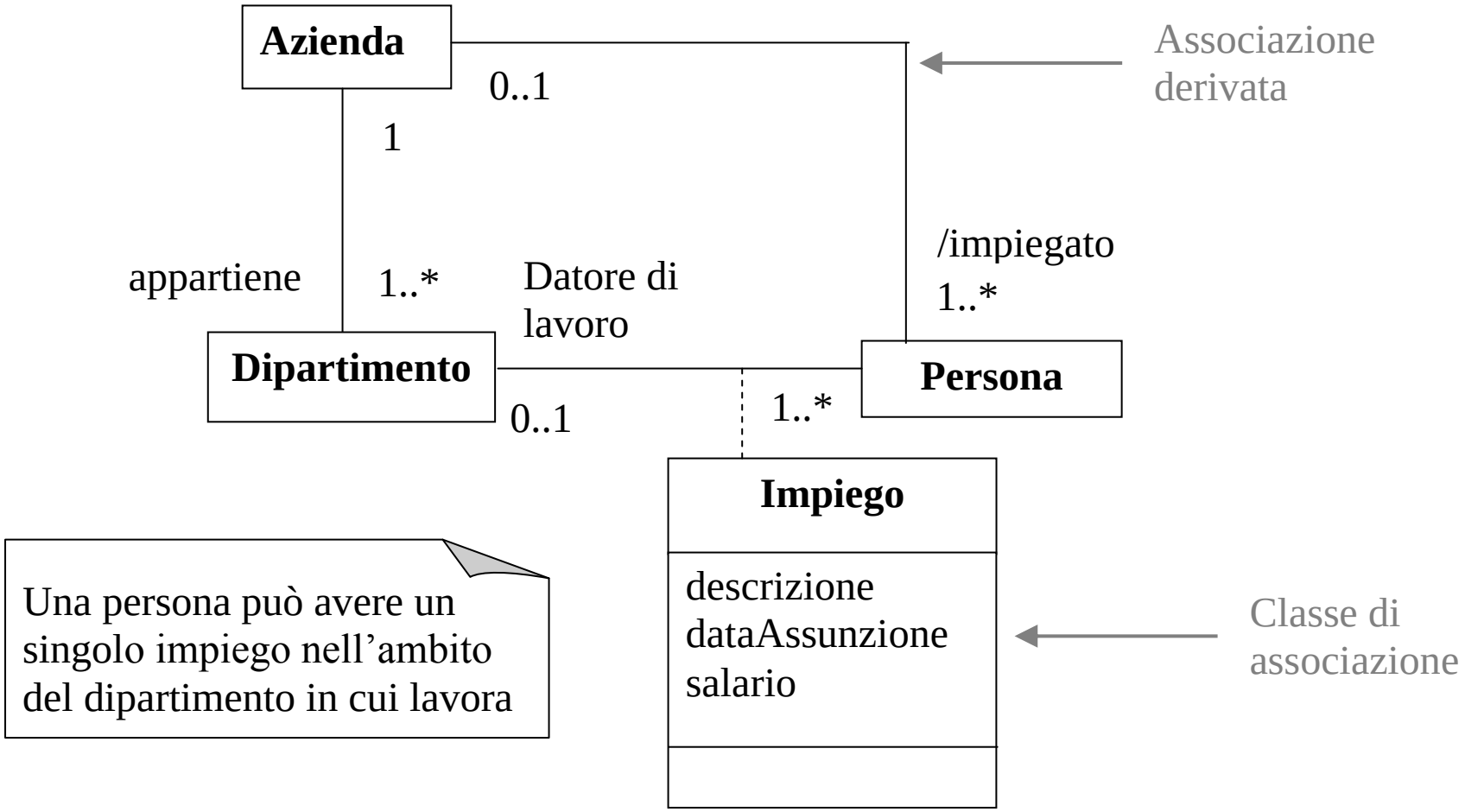
Classificazione



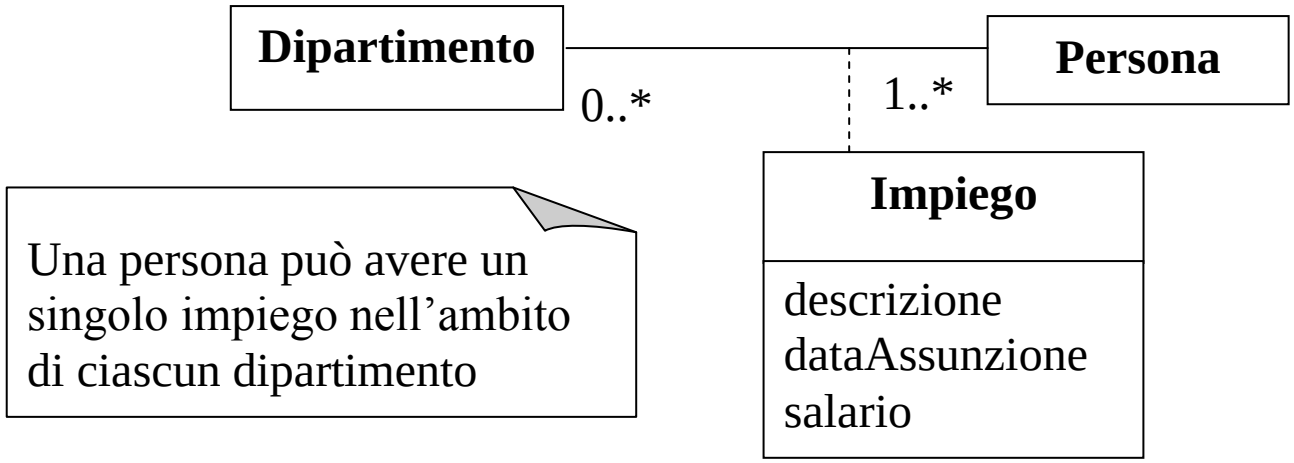
Concetti avanzati (cont.)

Elementi	Sintassi	Semantica
Classe di associazione	Classe collegata a una linea di associazione mediante una linea tratteggiata	Aggiunge un vincolo extra: ci può essere solo un'istanza della classe di associazione tra ogni coppia di oggetti associati
Classe parametrica (o template)	Classe con un riquadro tratteggiato sull'angolo superiore destro, contenente i parametri (tipi di dati) della classe, che possono essere più d'uno	• Usata per lo più per definire collezioni • Raramente utile in fase di modellazione concettuale; da usare in fase di specifica e implementazione solo se effettivamente supportata dal linguaggio di programmazione (è inclusa con il nome di *classe generica* in Java e C#)

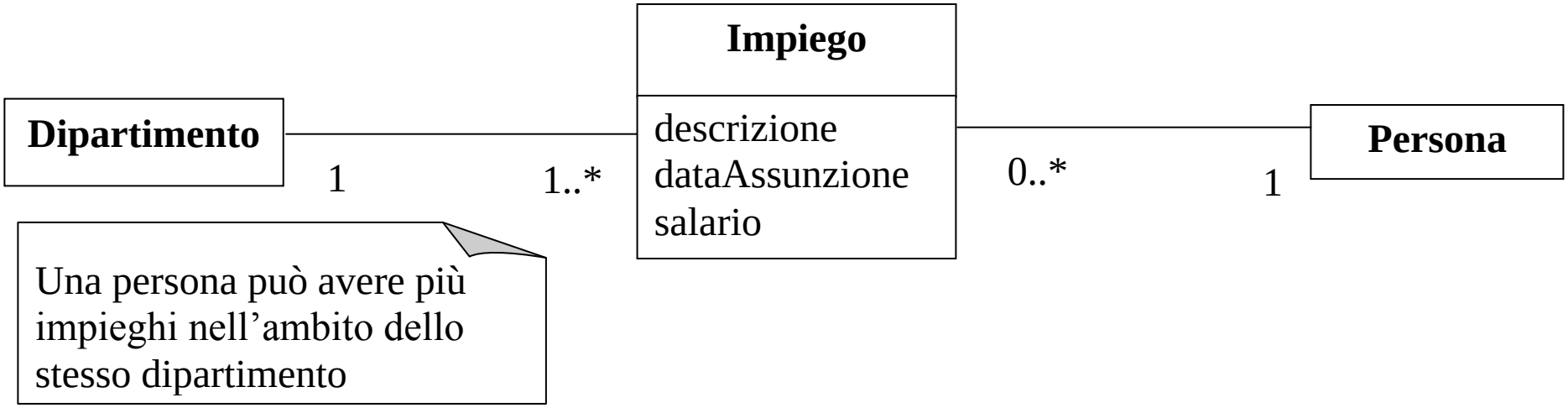
Classe di associazione



Promozione di una classe di associazione



Ma il significato è diverso rispetto a quello del diagramma sottostante, usato soprattutto per rappresentare relazioni temporali (storiche)



Concetti avanzati (cont.)

Elementi	Sintassi	Semantica
Derivazione	• Dalla classe derivata si diparte una freccia di generalizzazione diretta alla classe template; la freccia è affiancata dalla parola chiave «`bind`» seguita dall'elenco dei nomi dei tipi usati come parametri fra parentesi angolari (<>), ciascuno preceduto da *nome_parametro ::* Es. `<T::Dipendente>`, oppure • La classe derivata contiene il nome della classe template, la parola chiave «`bind`» seguita dall'elenco dei nomi dei tipi usati, nello stesso formato di cui al punto precedente	Una classe derivata rappresenta un uso particolare della classe parametrica, corrispondente a una specifica configurazione dei valori dei parametri La classe derivata non è una sottoclasse: infatti non è lecito aggiungerle caratteristiche rispetto a quelle del template

Classi parametriche (template)

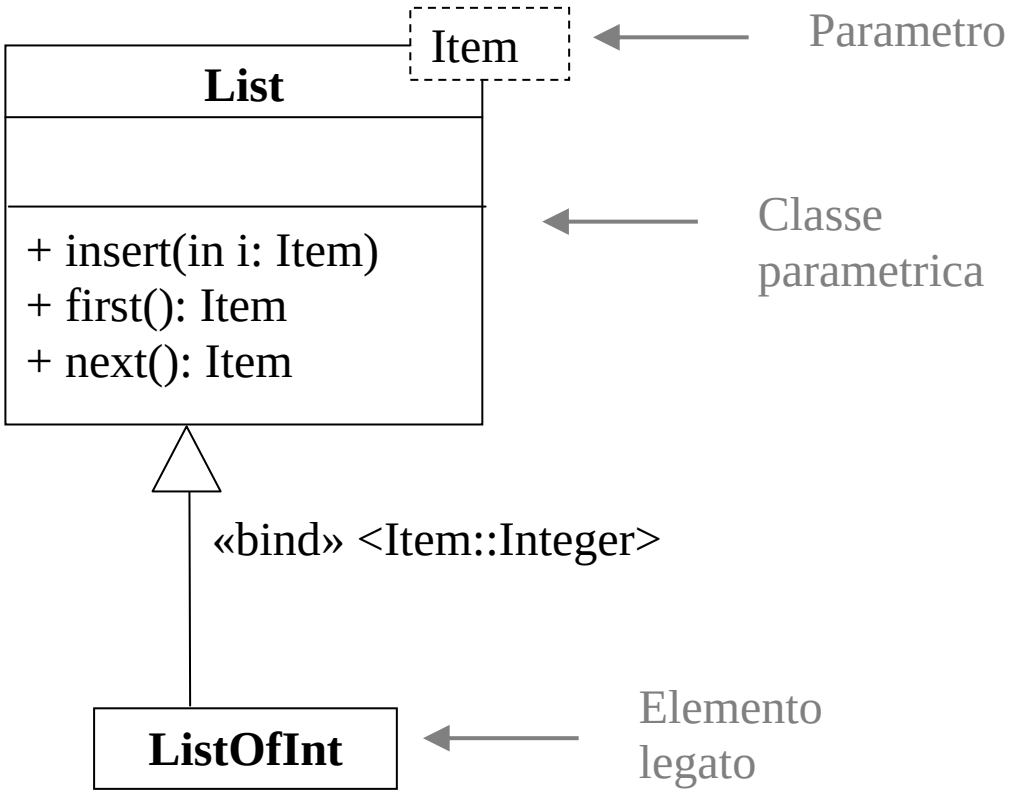


Diagramma degli oggetti (o delle istanze)

- È la fotografia degli oggetti che compongono un sistema sw in un dato momento
- È utile quando le connessioni fra oggetti sono complicate

Elementi	Sintassi	Semantica
Oggetto	Scatola suddivisa in due parti, entrambe dal contenuto opzionale, la prima contenente *nome-istanza : nome-classe* (può comparire o solo *nome-istanza* o solo *: nome-classe*), la seconda zero, uno o più *nome-attributo = "valore"* (l'elenco degli attributi non deve essere necessariamente esaustivo)	Istanza di una classe
Collegamento	Linea continua fra due oggetti, eventualmente etichettata con *nome-associazione*	Istanza di relazione fra due classi

Diagramma degli oggetti (cont.)