

Fondamenti di Informatica T2

Nozioni Base di UML2

Corso di Laurea in Ingegneria Informatica

Anno accademico 2019/2020

Prof. AMBRA MOLESINI

Dipartimento di Informatica – Scienza e Ingegneria (DISI)

Interfacce

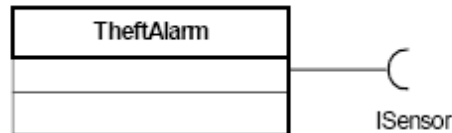
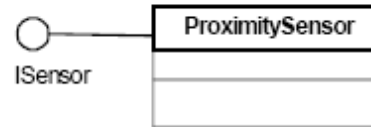


Interfaccia

- Le interfacce forniscono un modo per partizionare e caratterizzare gruppi di proprietà
- Un'interfaccia non deve specificare come possa essere implementata, ma semplicemente quello che è necessario per poterla realizzare
- Le entità che realizzano l'interfaccia dovranno fornire una “**vista pubblica**” (attributi, operazioni, comportamento osservabile all'esterno) conforme all'interfaccia stessa
- Se un'interfaccia dichiara un attributo, non significa necessariamente che l'elemento che realizza l'interfaccia deve avere quell'attributo nella sua implementazione, ma solamente che esso apparirà così ad un osservatore esterno

Interfaccia: notazione

Interfaccia fornita



Interfaccia richiesta

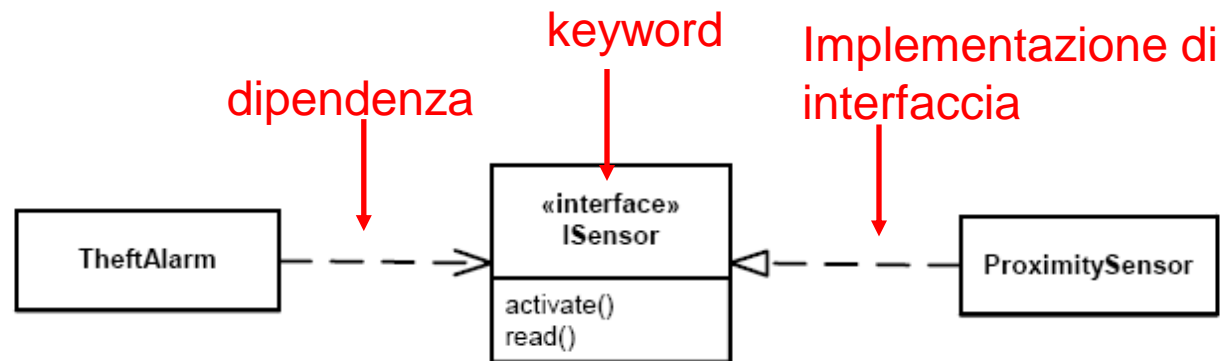


Diagramma delle Classi

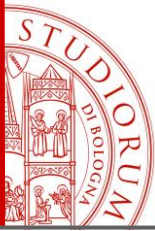
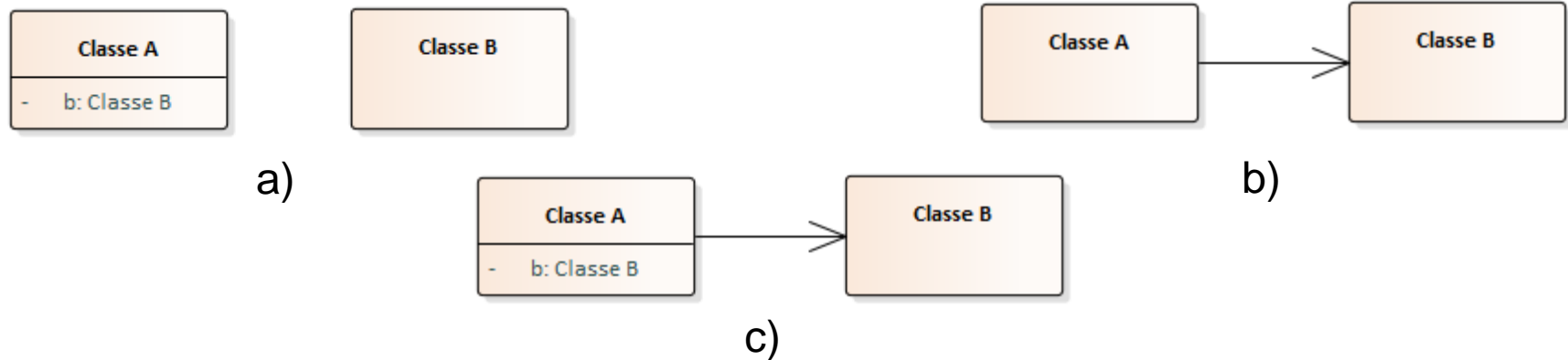


Diagramma delle Classi

- Un **diagramma delle classi** descrive il tipo degli oggetti facenti parte di un sistema e le varie tipologie di relazioni statiche tra di essi
- I diagrammi delle classi mostrano anche le *proprietà* e le *operazioni* di una classe e i *vincoli* che si applicano alla classe e alle relazioni tra classi
- Le *proprietà* rappresentano le caratteristiche strutturali di una classe:
 - sono un unico concetto, rappresentato però con due notazioni molto diverse: *attributi* e *associazioni*
 - benché il loro aspetto grafico sia molto differente, concettualmente sono la stessa cosa

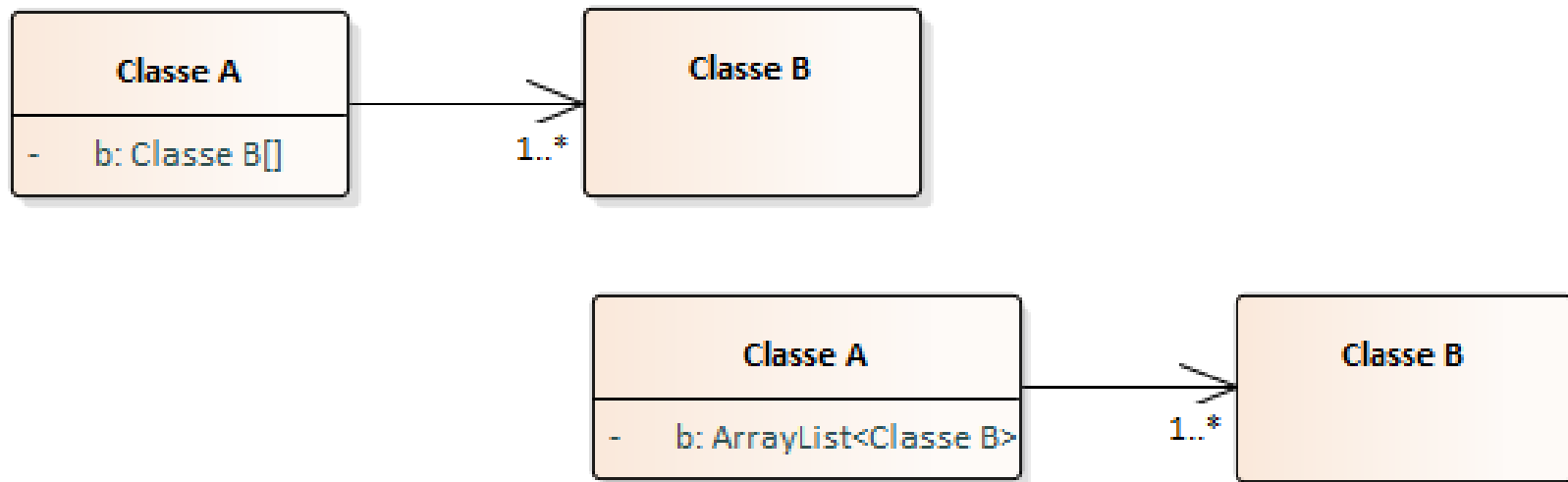
Attributi e Associazioni



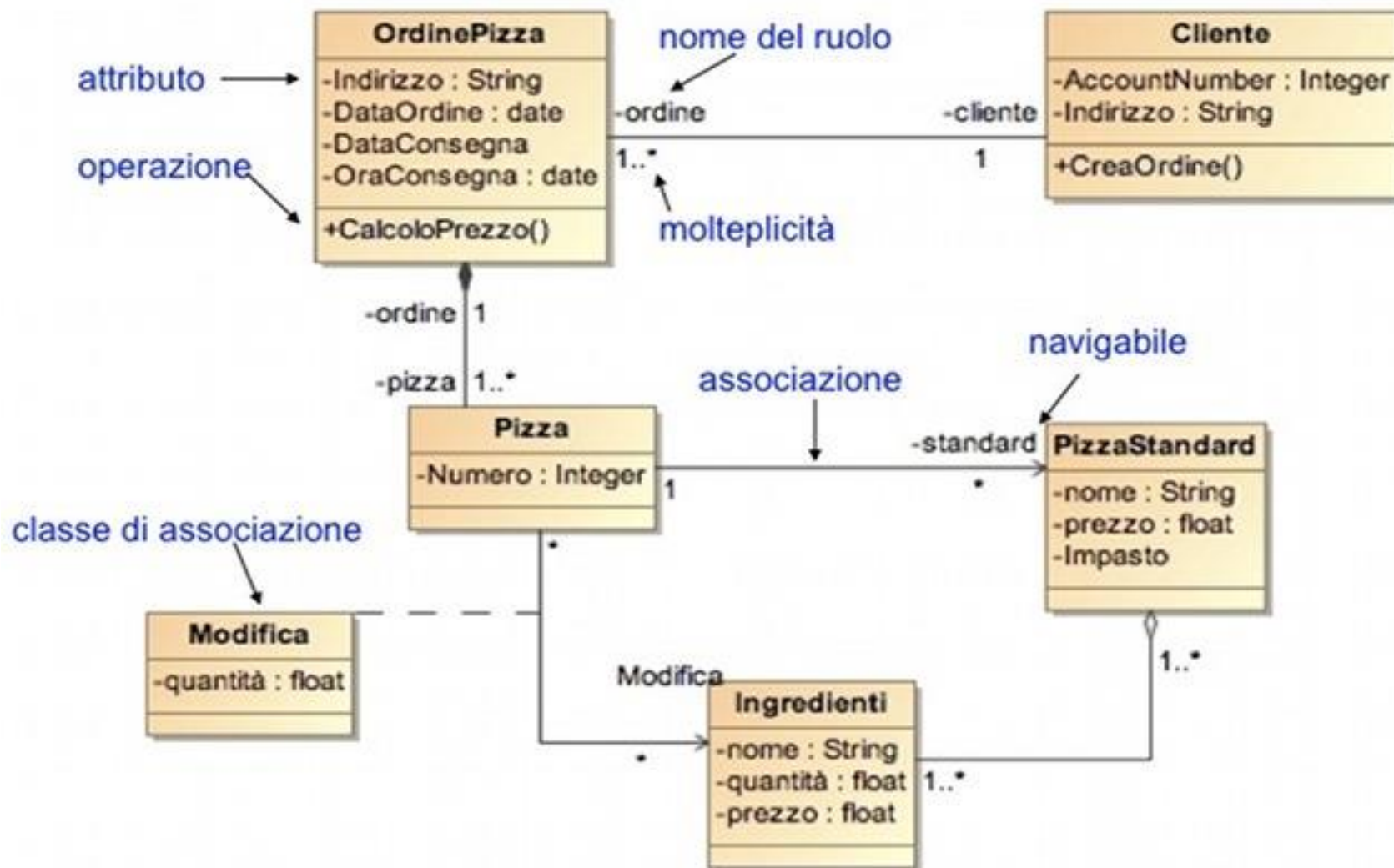
- I tre diagrammi esprimono la relazione tra Classe A e Classe B ed intendono mostrare che attributi e associazioni sono la stessa cosa
- Solo a) e b) sono perfettamente aderenti alla specifica UML
- Il diagramma c) presenta informazioni ridondanti, esso non è sbagliato, ma ci dice due volte la stessa cosa e sarebbe da evitare

Attributi e Associazioni

- Il diagramma c), anche se ridondante,
 - è spesso molto usato nei tool
 - può essere adottato anche nei casi in cui abbiamo associazioni con molteplicità diversa da 1 e vogliamo specificare quale tipo di struttura dati vogliamo adottare

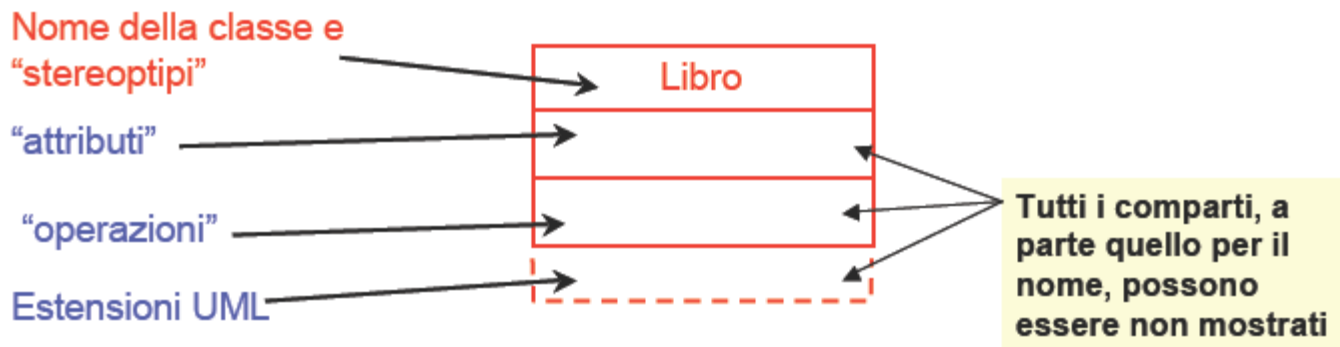


Esempio



Classe

- Una **classe** modella un insieme di entità (le istanze della classe) aventi tutti lo stesso tipo di caratteristiche (attributi, associazioni, operazioni...).
- Ogni classe è descritta da:
 - un **nome**
 - un insieme di **features**: attributi, operazioni, ...





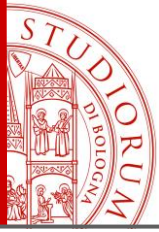
Attributi

- La notazione degli **attributi** descrive una proprietà con una riga di testo all'interno del box della classe. La forma completa è:
 - **visibilità nome:tipo molteplicità =default {stringa di proprietà}**
- Un esempio di attributo è:
 - stringa: String [10] = “Pippo” {readOnly}
- L'unico elemento necessario è il nome.
 - Visibilità: attributo pubblico (+) o privato(-) o protected (#)
 - Nome: corrisponde al nome dell'attributo
 - Tipo: vincolo sugli oggetti che possono rappresentare l'attributo
 - Default: valore dell'attributo in un oggetto appena creato;
 - Stringa di proprietà: caratteristiche aggiuntive (readOnly);
 - Molteplicità: ...



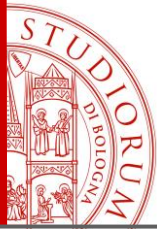
Molteplicità

- È l'indicazione di quanti oggetti possono entrare a far parte di una proprietà. Le molteplicità più comuni sono:
 - **1, 0..1, ***
- In modo più generale, le molteplicità si possono definire indicando gli estremi inferiore e superiore di un intervallo (per esempio **2..4**).
- Molti termini si riferiscono alla molteplicità degli attributi:
 - **Opzionale:** indica un limite inferiore di 0
 - **Obbligatorio:** implica un limite inferiore di 1 o più
 - **A un solo valore:** implica un limite superiore di 1
 - **A più valori:** implica che il limite superiore sia maggiore di 1, solitamente *



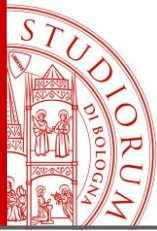
Visibilità

- È possibile etichettare ogni operazione o attributo con un identificatore di visibilità.
- UML fornisce comunque quattro abbreviazioni per indicare la visibilità:
 - + (public)
 - - (private)
 - ~ (package)
 - # (protected)



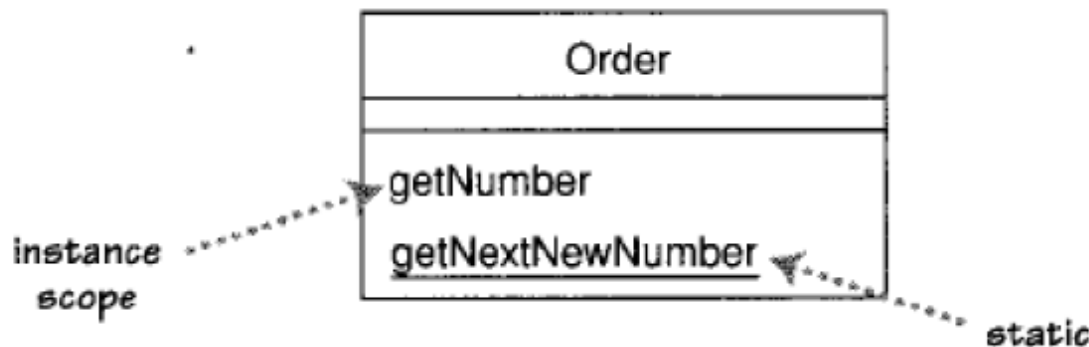
Operazioni

- Le operazioni sono le azioni che la classe sa eseguire, e in genere si fanno corrispondere direttamente ai metodi della corrispondente classe a livello implementativo
- Le operazioni che manipolano le proprietà della classe di solito si possono dedurre, per cui non sono incluse nel diagramma
- La sintassi UML completa delle operazioni è
visibilità nome (lista parametri) : tipo ritorno {stringa di propr}
 - Visibilità : operazione pubblica (+) o privata (-)
 - Nome : stringa
 - Lista parametri: lista parametri dell'operazione
 - Tipo di ritorno: tipo di valore restituito dall'operazione, se esiste
 - Stringa di proprietà: caratteristiche aggiuntive che si applicano all'operazione



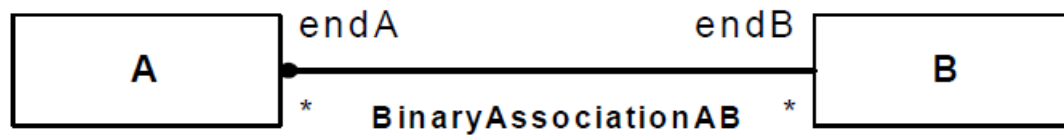
Operazioni e Attributi Statici

- UML chiama **static** un'operazione o un attributo che si applicano a una classe anziché alle sue istanze
- Questa definizione equivale a quella dei membri statici nei linguaggi come per esempio java e C#
- Le caratteristiche statiche vanno sottolineate sul diagramma



Associazioni

- Le associazioni sono un altro modo di rappresentare le proprietà
- Gran parte dell'informazione che può essere associata a un attributo si applica anche alle associazioni
- Un'associazione è una linea continua che collega due classi, orientata dalla classe sorgente a quella destinazione
- Il nome e la molteplicità vanno indicati vicino all'estremità finale dell'associazione:
 - la classe destinazione corrisponde al tipo della proprietà



Associazioni

- Assegnare dei nomi ai “**ruoli**” svolti da ciascun elemento di un associazione
- Anche nei casi in cui non è strettamente necessario, il ruolo può essere utile per aumentare la leggibilità del diagramma



Associazioni Bidirezionali

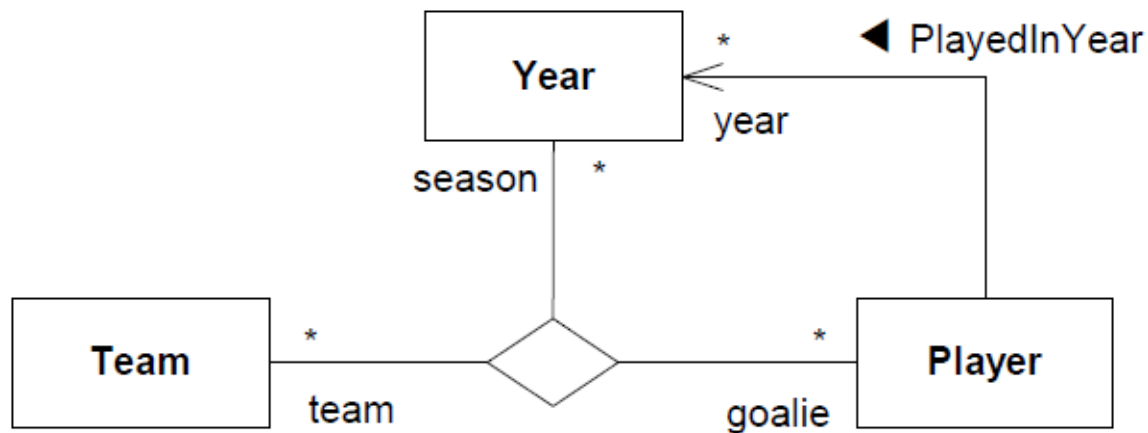
- Una tipologia di associazione è quella **bidirezionale (o binaria)**, costituita da una coppia di proprietà collegate, delle quali una è l'inversa dell'altra
- Il collegamento inverso implica che, se seguite il valore di una proprietà e poi il valore della proprietà collegata, dovreste ritornare all'interno di un insieme che contiene il vostro punto di partenza



la natura bidirezionale dell'associazione è palesata dalle **frecce di navigabilità** aggiunte a entrambi i capi della linea

Associazioni Ternarie

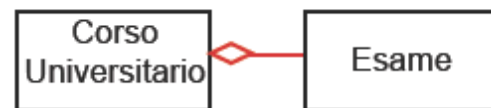
- Quando si hanno associazioni ternarie (o che coinvolgono più classi) si introduce il simbolo “**diamante**”



Aggregazione e Composizione

- **Aggregazione:**

- è un'associazione che corrisponde ad una relazione intuitiva Intero-Parte (“**part-of**”)
- è rappresentata da un **diamante vuoto** sull'associazione, posto vicino alla classe le cui istanze sono gli “interi”



- **Composizione:**

- è un'aggregazione che rispetta due vincoli ulteriori:
 - una parte può essere inclusa in al massimo un intero in ogni istante
 - solo l'oggetto intero può creare e distruggere le sue parti
- è rappresentata da un **diamante pieno** vicino alla classe che rappresenta gli “interi”



Aggregazione e Composizione

- **Aggregazione:**

- è una relazione **binaria**
- può essere **ricorsiva**



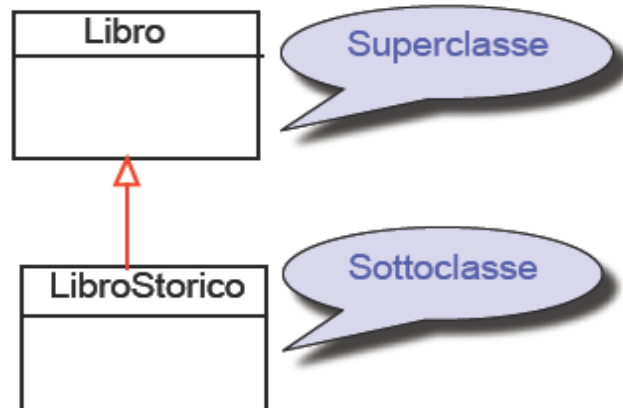
- **Composizione:**

- se l'oggetto che compone viene distrutto, anche i figli vengono distrutti, ...
- ... anche se i figli possono essere creati/distrutti in momenti diversi dalla creazione/distruzione dell'oggetto che compone
- può essere **ricorsiva**



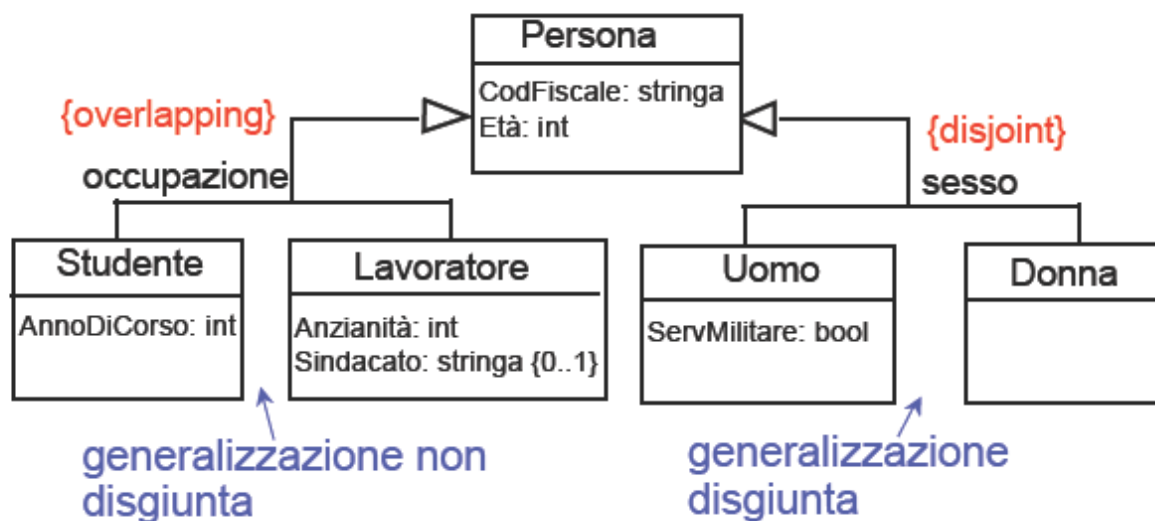
Generalizzazione

- La generalizzazione è indicata con una freccia vuota fra due classi dette **sottoclasse** e **superclasse**
- Il significato della generalizzazione è il seguente: ogni istanza della sottoclasse è anche istanza della superclasse



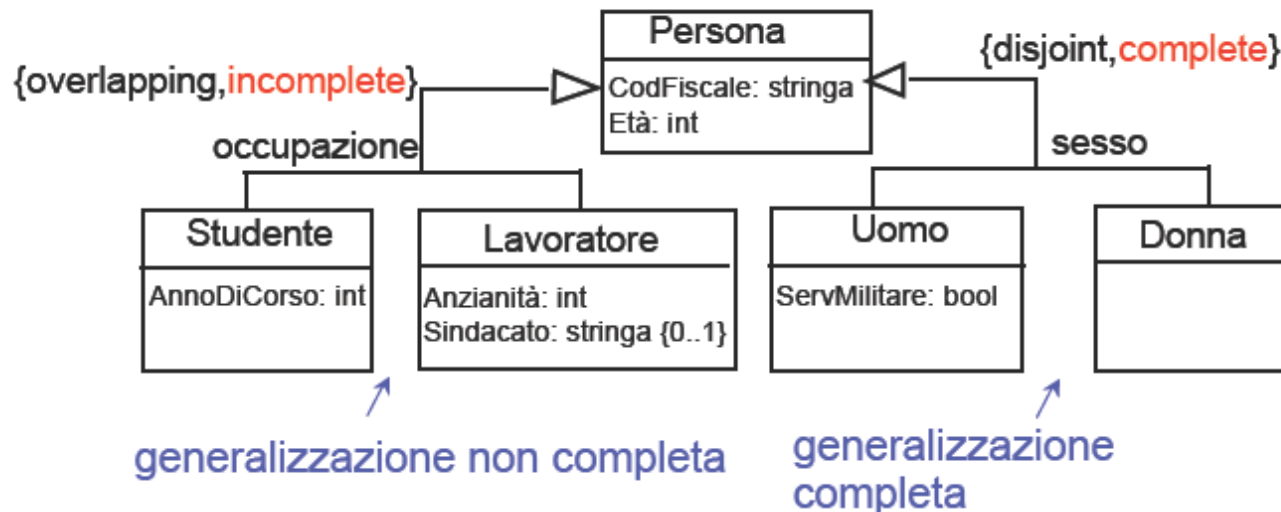
Generalizzazione

- La stessa superclasse può partecipare a diverse generalizzazioni
- Una generalizzazione può essere **disgiunta**, cioè le sottoclassi sono disgiunte (non hanno istanze in comune), o no



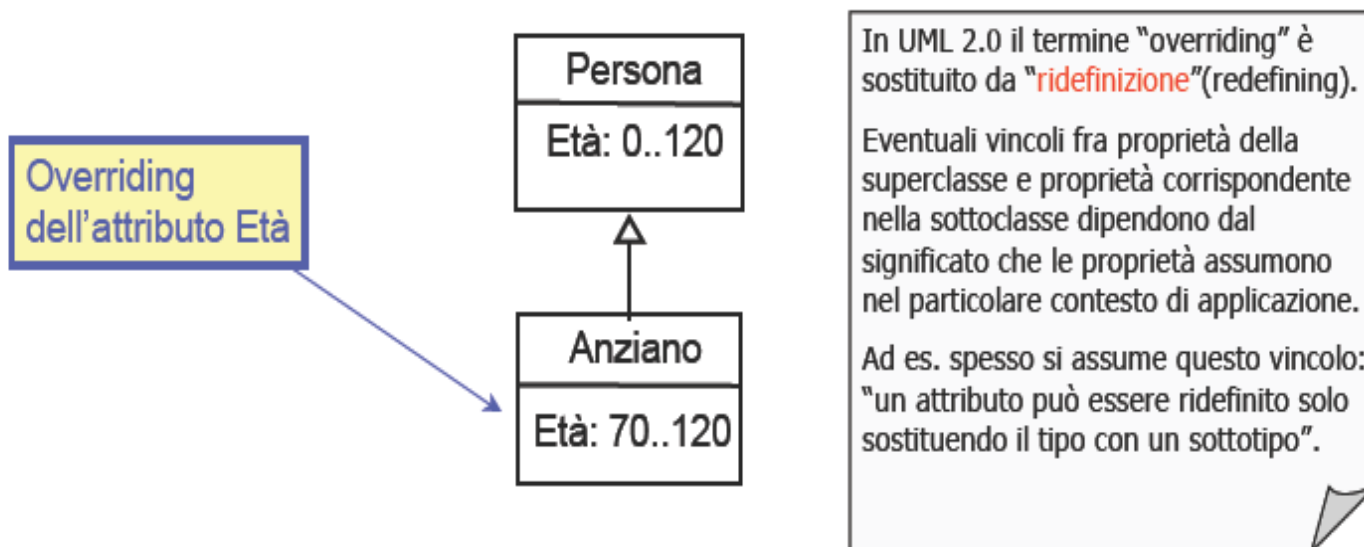
Generalizzazione

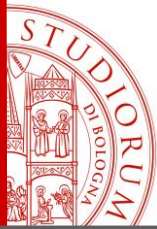
- Una generalizzazione può essere **completa** (l'unione delle istanze delle sottoclassi è uguale all'insieme delle istanze della superclasse) o no
- Attenzione: I valori di default** sono {incomplete, disjoint}



Generalizzazione

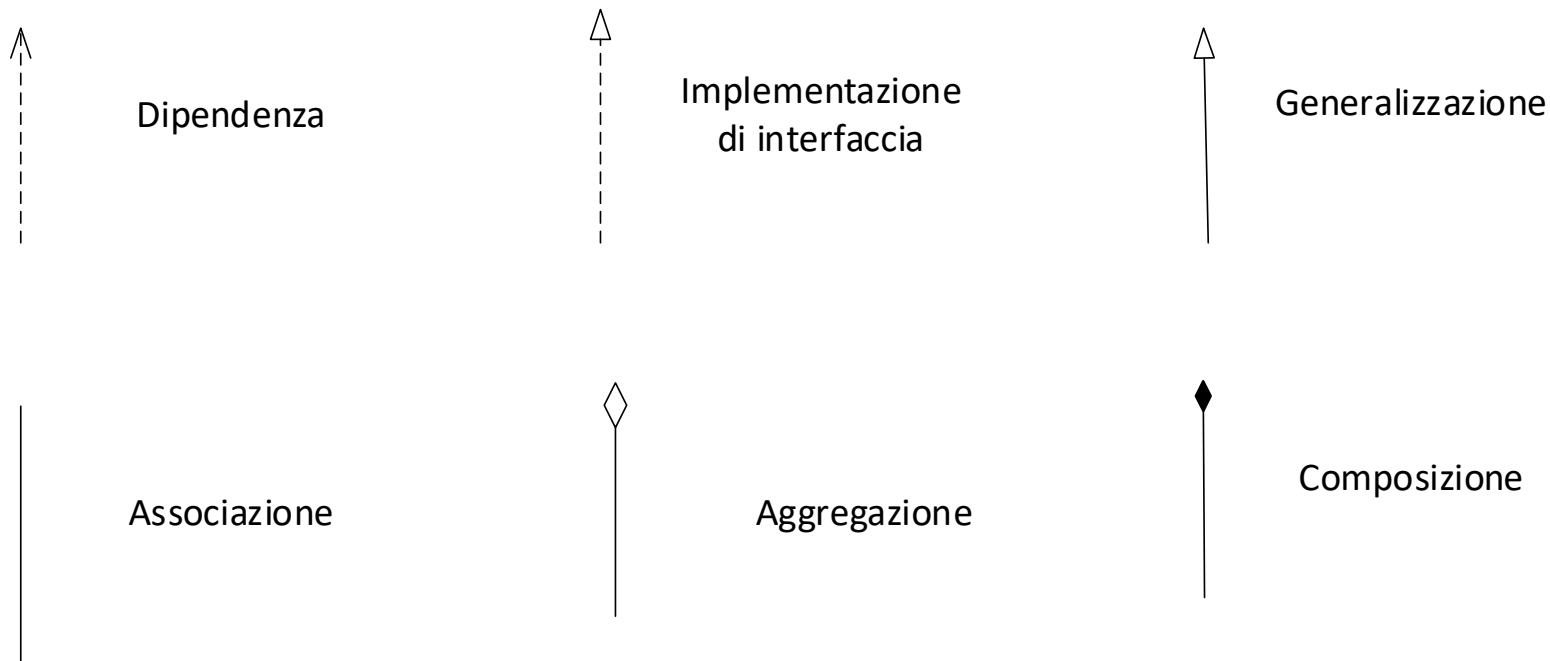
- In una generalizzazione la sottoclasse non solo può avere caratteristiche aggiuntive rispetto alla superclasse, ma può anche **sovrascrivere** (**overriding**) le proprietà ereditate dalla superclasse





Relazioni tra classi: sintassi

- Attenzione all'uso corretto delle frecce
- UML è un linguaggio (anche se grafico) e scambiare un freccia per un'altra è un errore non da poco





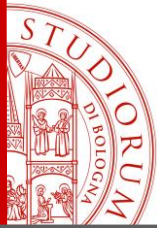
Classi Astratte

- Una **classe astratta** è una classe che non può essere direttamente istanziata: per farlo bisogna prima crearne una sottoclasse concreta
- Tipicamente, una classe astratta ha una o più operazioni astratte
- Un'operazione astratta non ha implementazione: è costituita dalla sola dichiarazione, resa pubblica affinché le classi client possano usufruirne
- Il modo più diffuso di indicare una classe o un'operazione astratta in UML è **scrivere il nome in corsivo**
- Si possono anche rendere astratte le proprietà
- Indicandole direttamente come tali o rendendo astratti i metodi d'accesso



Classi Astratte

- A cosa serve?
 - serve come superclasse comune per un insieme di sottoclassi concrete
 - queste sottoclassi, in virtù del subtyping, sono in qualche misura compatibili e intercambiabili fra di loro
 - infatti sono tutte sostituibili con la superclasse: sulle istanze di ognuna di esse possiamo invocare i metodi ereditati dalla classe astratta



Enumerazioni

- Le enumerazioni sono usate per mostrare un insieme di valori prefissati che non hanno altre proprietà oltre al loro valore simbolico
- Sono rappresentate con una classe marcata dalla parola chiave «enumeration»

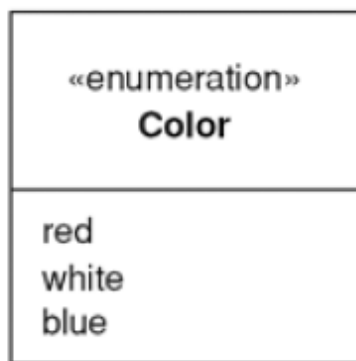


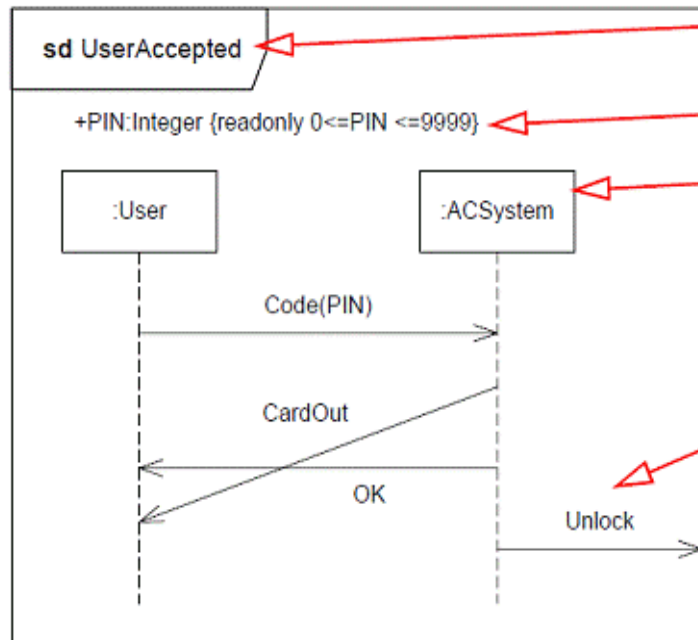
Diagramma di Sequenza



Diagramma di Sequenza

- Diagramma che illustra le interazioni tra le classi / entità disponendole lungo una sequenza temporale
- In particolare mostra i soggetti (chiamati tecnicamente **Lifeline**) che partecipano all'interazione e la sequenza dei messaggi scambiati
- In ascissa troviamo i diversi soggetti (anche non in ordine di esecuzione), mentre in ordinata abbiamo la scala dei tempi sviluppata verso il basso

Diagramma di Sequenza



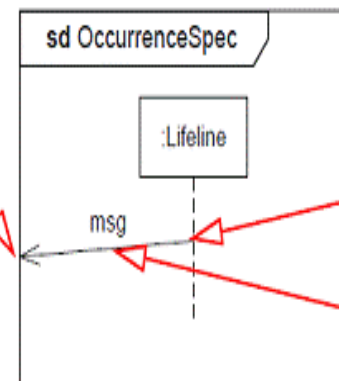
Name of Interaction

Local Attribute

Lifeline

Message

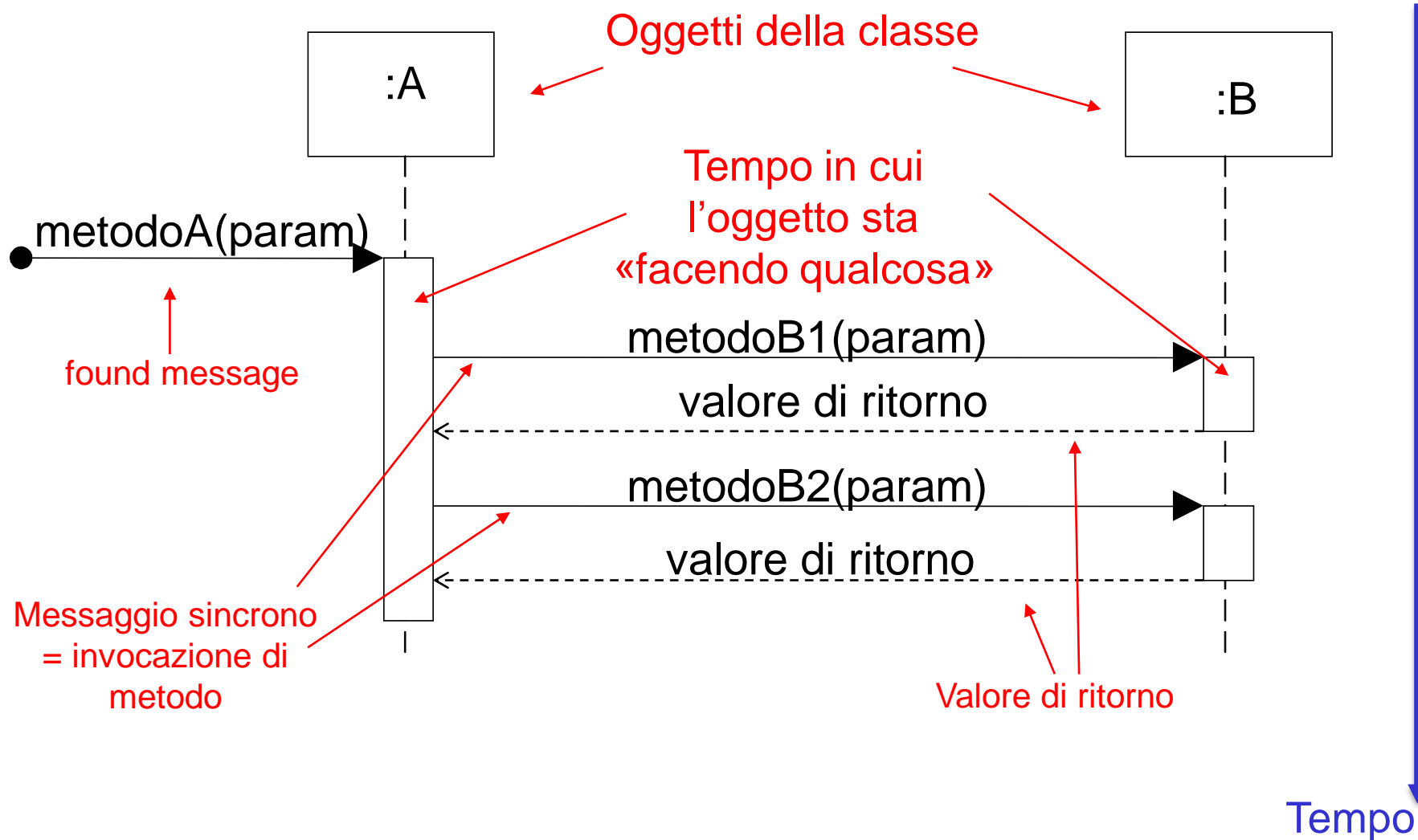
*(formal)
Gate*

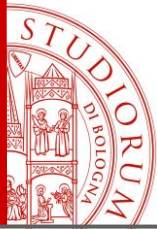


OccurrenceSpecification

Message

Diagramma di Sequenza



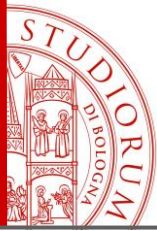


Lifeline

- In un diagramma di sequenza, i partecipanti solitamente sono istanze di classi UML caratterizzate da un nome
- La loro vita è rappresentata da una *Lifeline*, cioè una **linea tratteggiata verticale ed etichettata**, in modo che sia possibile comprendere a quale componente del sistema si riferisce
- In alcuni casi il partecipante non è un'entità semplice, ma composta
 - è possibile modellare la comunicazione fra più sottosistemi, assegnando una Lifeline ad ognuno di essi

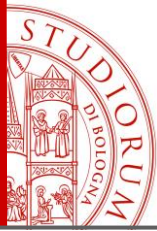
Lifeline

- L'ordine in cui le OccurrenceSpecification (cioè l'invio e la ricezione di eventi) avvengono lungo la Lifeline rappresenta **esattamente l'ordine in cui tali eventi si devono verificare**
- La distanza (in termini grafici) tra due eventi non ha rilevanza dal punto di vista semantico
- Dal punto di vista notazionale, una Lifeline è rappresentata da un rettangolo che costituisce la "testa" seguito da una linea verticale che rappresenta il tempo di vita del partecipante
- E' interessante notare che nella sezione della notazione, viene indicato espressamente che il "rettangolino" che viene apposto sulla Lifeline rappresenta l'attivazione di un metodo



Riferimento ad altri Diagrammi

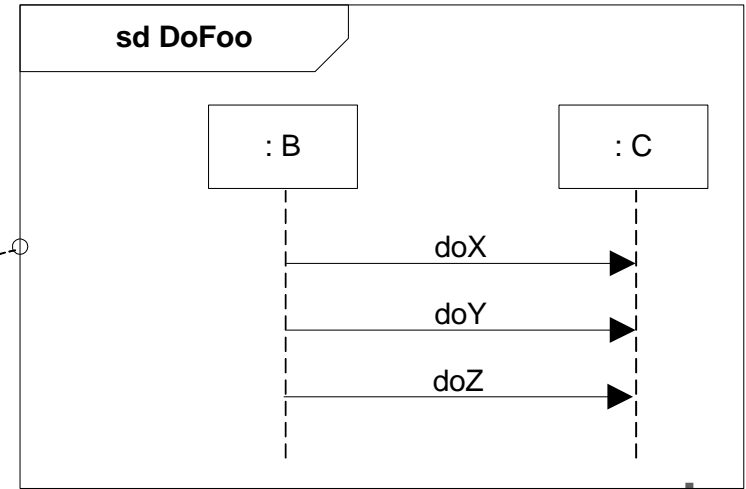
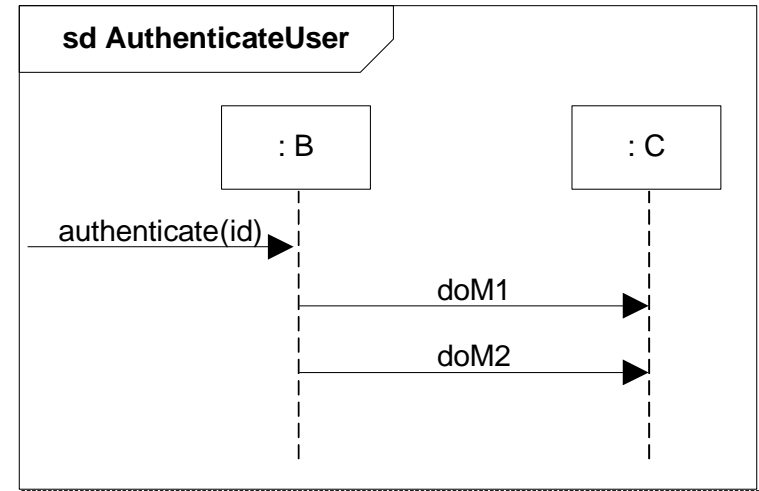
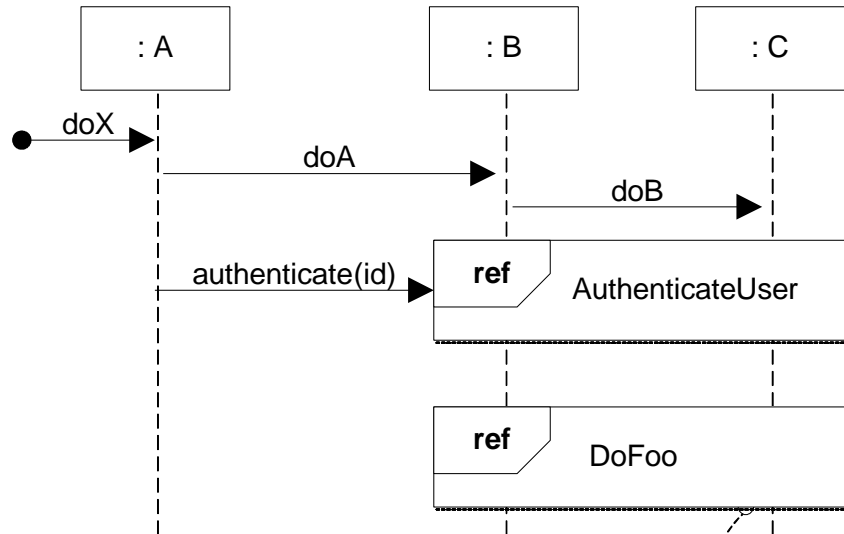
- Spesso i diagrammi di sequenza possono assumere una certa complessità
 - necessità di poter definire comportamenti più articolati come composizione di nuclei di interazione più semplici
- Oppure, se una sequenza di eventi ricorre spesso, potrebbe essere utile definirla una volta e richiamarla dove necessario
- Per questa ragione, UML permette di inserire **riferimenti ad altri diagrammi** e passare loro degli argomenti



Riferimento ad altri Diagrammi

- Ovviamente ha senso sfruttare quest'ultima opzione solo se il diagramma accetta dei parametri sui quali calibrare l'evoluzione del sistema
- Questi riferimenti prendono il nome di *InteractionUse*
- I punti di connessione tra i due diagrammi prendono il nome di *Gate*
- Un Gate rappresenta un **punto di interconnessione** che mette in relazione un messaggio al di fuori del frammento di interazione con uno all'interno del frammento

Riferimento ad altri Diagrammi



interaction occurrence
note it covers a set of lifelines
note that the sd frame it relates to
has the same lifelines: B and C



Messaggio

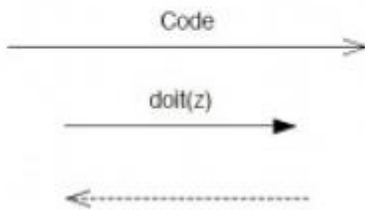
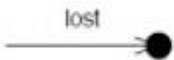


- Un messaggio rappresenta un'interazione realizzata come comunicazione fra Lifeline
- Questa interazione può consistere nella creazione o distruzione di un'istanza, nell'invocazione di un'operazione, o nella emissione di un segnale
- UML permette di rappresentare tipi differenti di messaggi



Tipi di Messaggio

- Se sono specificati mittente e destinatario allora è un **complete message**; *la semantica è rappresentata* quindi dall'occorrenza della coppia di eventi <sendEvent, receiveEvent>
- Se il destinatario non è stato specificato allora è un **lost message**; *in questo caso è noto* solo l'evento di invio del messaggio
- Se il mittente non è stato specificato allora è un **found message**; *in questo caso è noto* solo l'evento di ricezione del messaggio
- Nel caso non sia noto né il destinatario né il mittente allora è un **unknown message**

Message

<i>NODE TYPE</i>	<i>NOTATION</i>	<i>REFERENCE</i>
Message		<p>Messages come in different variants depending on what kind of Message they convey. Here we show an asynchronous message, a call and a reply. These are all <i>complete</i> messages.</p>
Lost Message		<p>Lost messages are messages with known sender, but the reception of the message does not happen.</p>
Found Message		<p>Found messages are messages with known receiver, but the sending of the message is not described within the specification.</p>
GeneralOrdering		



Tipi di Messaggio




- Attenzione alle frecce che usate nei messaggi → hanno significati diversi:
 - **riga continua freccia piena**: indica un messaggio (**call**) **sincrono** in cui il mittente **aspetta** il completamento dell'esecuzione del destinatario prima di continuare la sua esecuzione. **Necessita di un valore di ritorno** per sbloccare l'esecuzione del mittente 
 - **riga continua freccia vuota**: indica un messaggio **asincrono** in cui il mittente **non aspetta** il completamento dell'esecuzione del destinatario ma continua la sua esecuzione. Il valore di ritorno potrebbe o meno essere necessario, dipende dalla semantica 
 - **riga tratteggiata freccia vuota**: indica il ritorno di un messaggio 

Diagramma di Sequenza

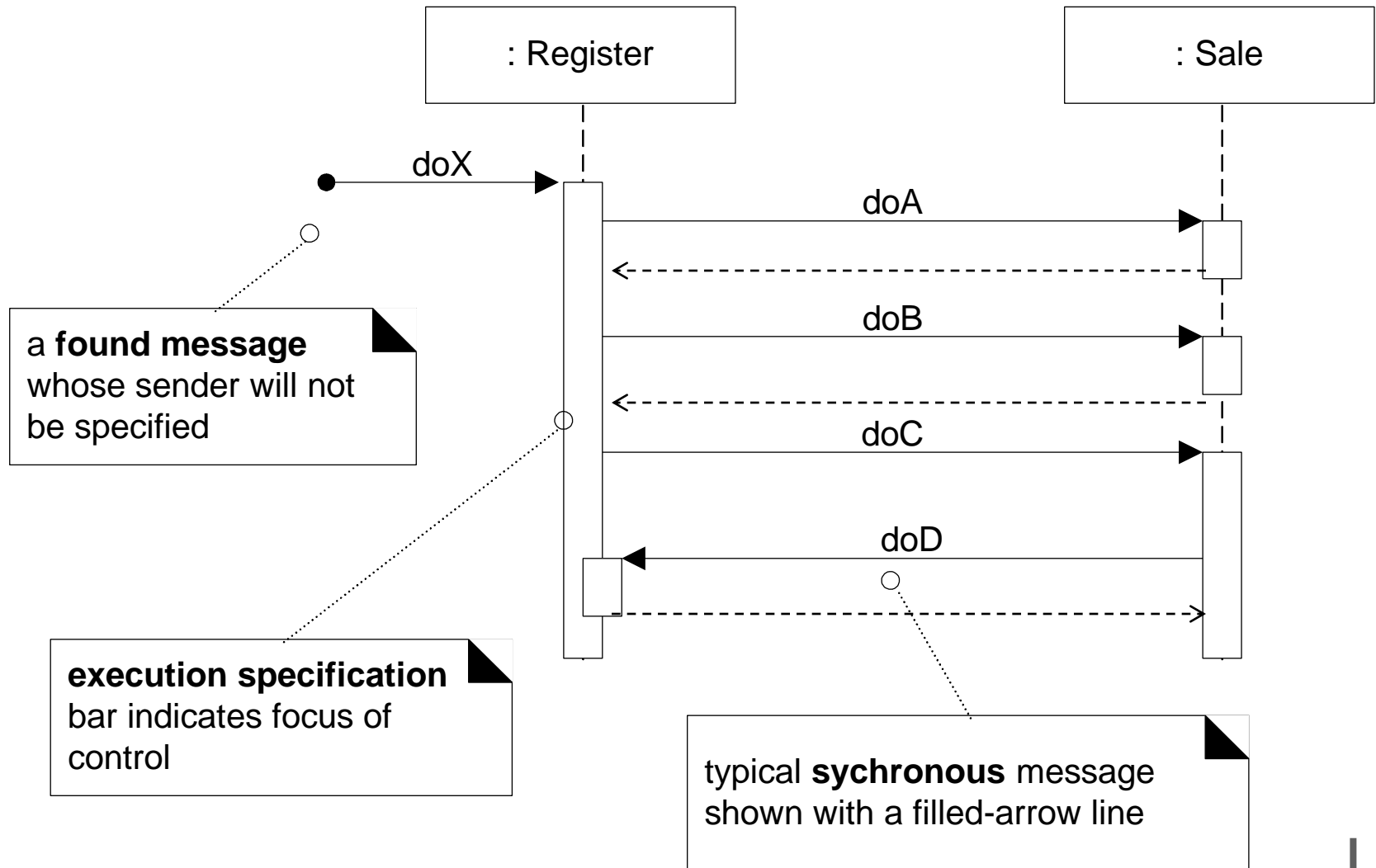
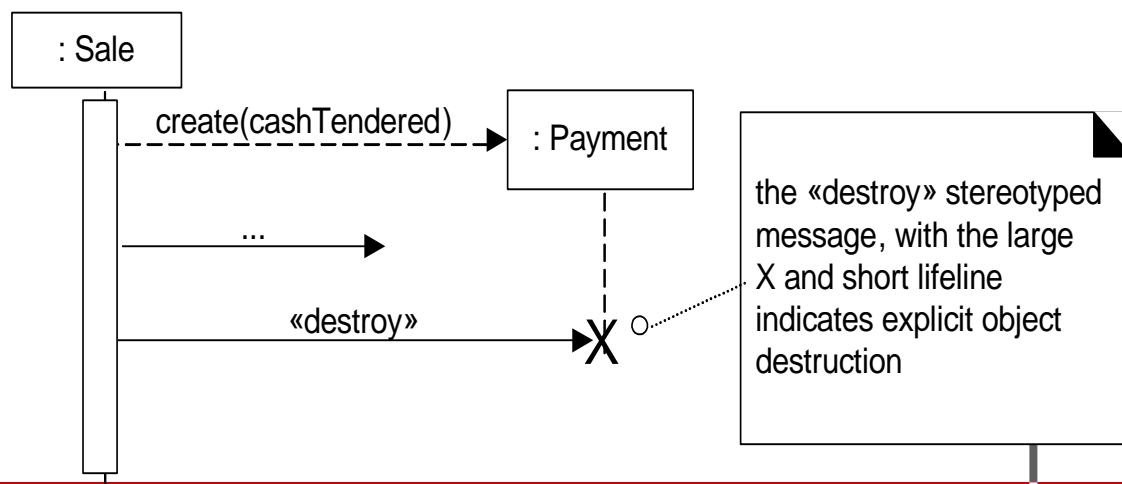
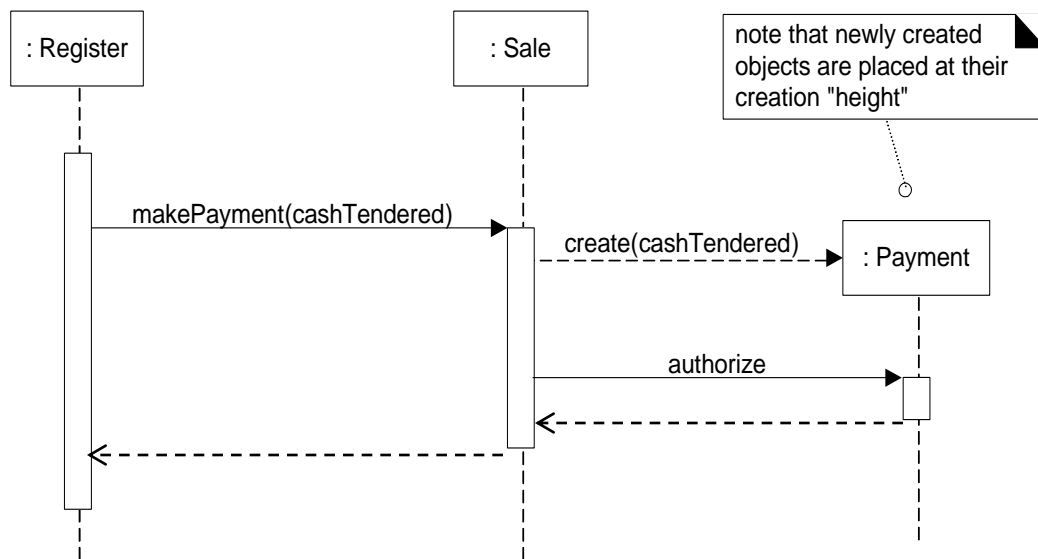
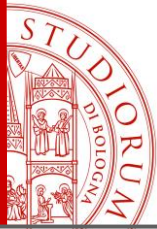


Diagramma di Sequenza





CombinedFragment

- La specifica di UML permette di esprimere comportamenti più complessi rispetto al singolo scambio di messaggi
- È ora possibile rappresentare l'esecuzione atomica di una serie di interazioni, oppure che un messaggio deve essere inviato solo in determinate condizioni
- A tale scopo UML mette a disposizione i *Combined Fragment*, cioè contenitori atti a delimitare un'area d'interesse nel diagramma
- Servono per spiegare che una certa catena di eventi, racchiusa in uno o più operandi, si verificherà in base alla semantica dell'operatore associato
- Ogni fragment ha un operatore ed una eventuale una guardia



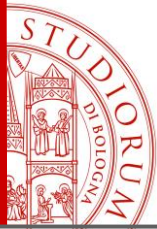
CombinedFragment

- *Loop*: specifica che quello che è racchiuso nell'operando sarà eseguito ciclicamente finché la guardia sarà verificata
- *Alternatives* (*alt*): indica che sarà eseguito il contenuto di uno solo degli operandi, quello la cui guardia risulta verificata
- *Optional* (*opt*): indica che l'esecuzione del contenuto dell'operando sarà eseguita solo se la guardia è verificata
- *Break* (*break*): ha la stessa semantica di *opt*, con la differenza che in seguito l'interazione sarà terminata
- *Critical*: specifica un blocco di esecuzione atomico (non interrompibile)
- *Parallel* (*par*): specifica che il contenuto del primo operando può essere eseguito in parallelo a quello del secondo



CombinedFragment

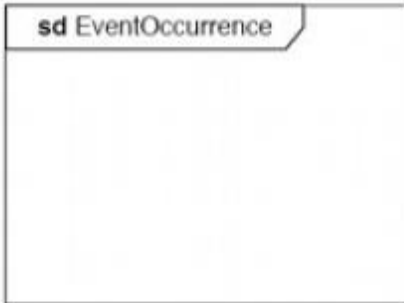
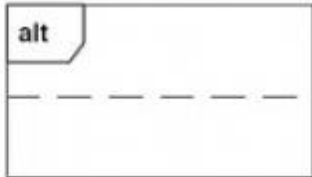
- *Weak Sequencing* (*seq*): specifica che il risultato complessivo può essere una qualsiasi combinazione delle interazioni contenute negli operandi, purché: (1) l'ordinamento stabilito in ciascun operando sia mantenuto nel complesso; (2) eventi che riguardano gli stessi destinatari devono rispettare anche l'ordine degli operandi, cioè i messaggi del primo operando hanno precedenza su quelli del secondo; (3) eventi che riguardano destinatari differenti non hanno vincoli di precedenza vicendevole
- *Strict Sequencing* (*strict*): indica che il contenuto deve essere eseguito nell'ordine in cui è specificato, anche rispetto agli operandi



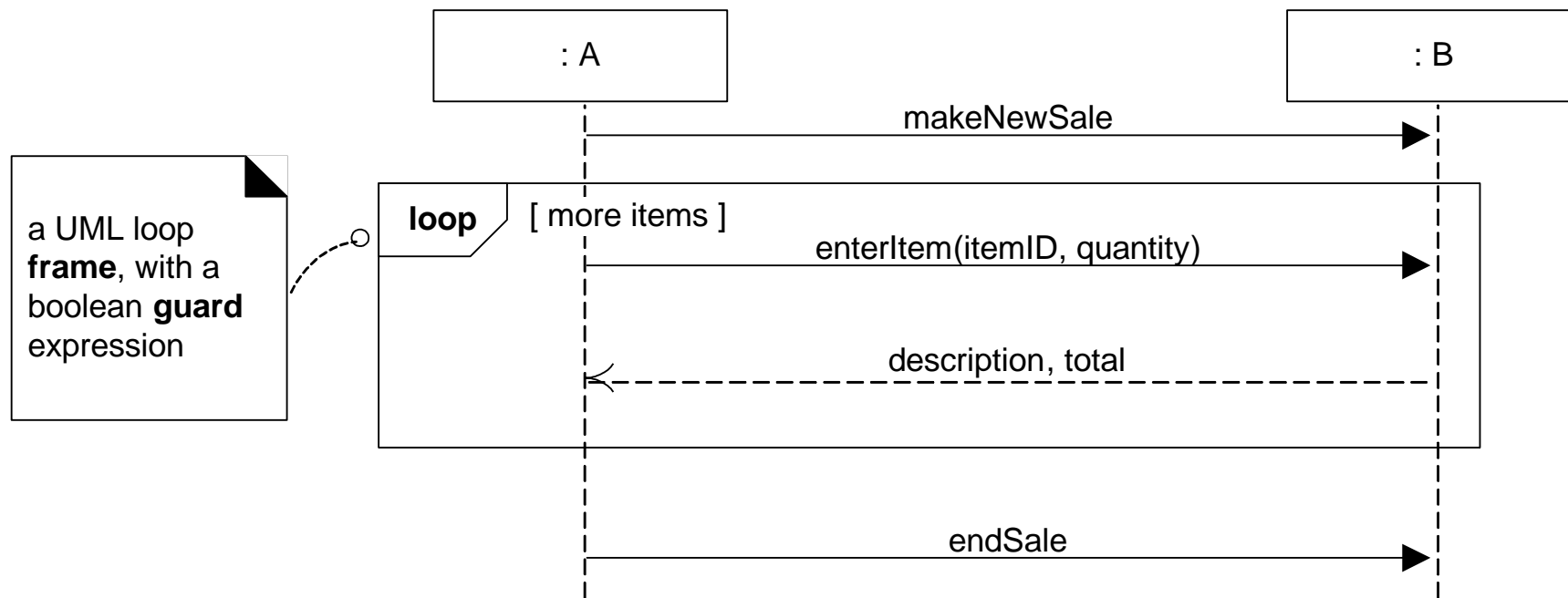
CombinedFragment

- *Ignore*: indica che alcuni messaggi, importanti ai fini del funzionamento del sistema, non sono stati rappresentati, perché non utili ai fini della comprensione dell'interazione
- *Consider*: è complementare ad ignore
- *Negative* (*neg*): racchiude una sequenza di eventi che non deve mai verificarsi
- *Assertion* (*assert*): racchiude quella che è considerata l'unica sequenza di eventi valida. Di solito è associata all'utilizzo di uno State Invariant come rinforzo

CombinedFragment

<i>NODE TYPE</i>	<i>NOTATION</i>	<i>REFERENCE</i>
Frame		The notation shows a rectangular frame around the diagram with a name in a compartment in the upper left corner.
CombinedFragment		

CombinedFragment



CombinedFragment

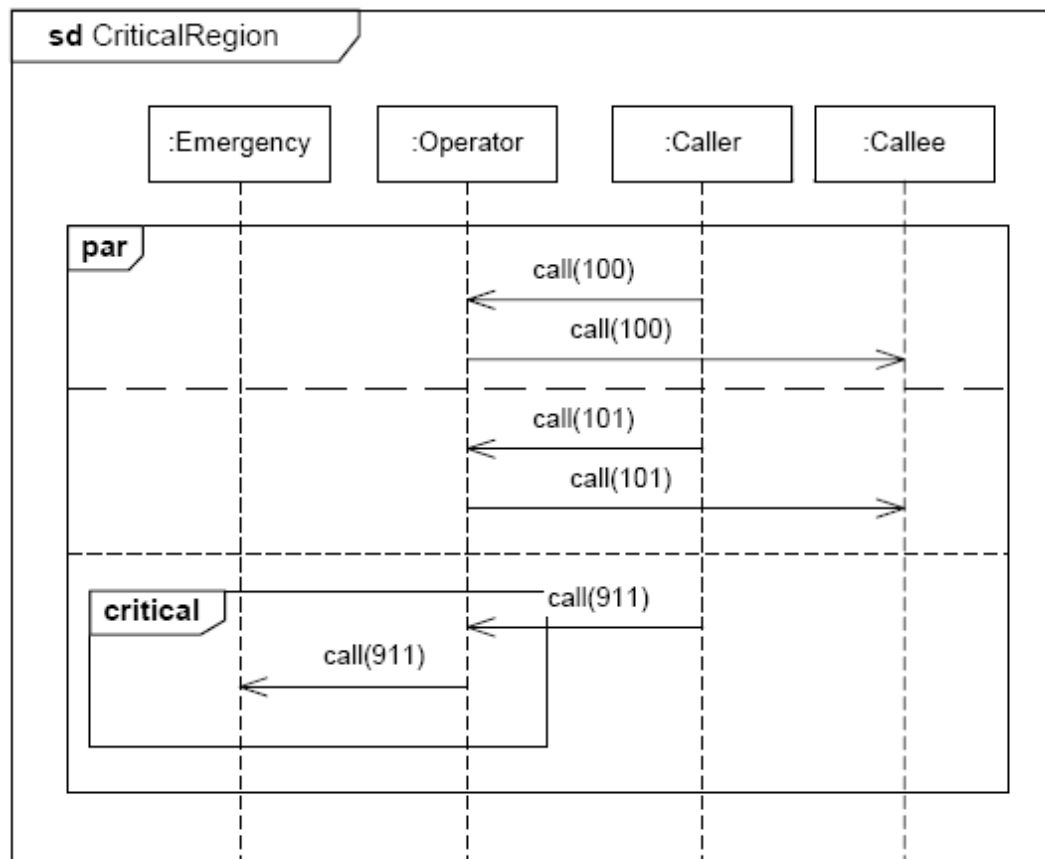


Diagramma delle Attività

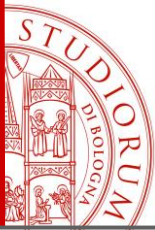
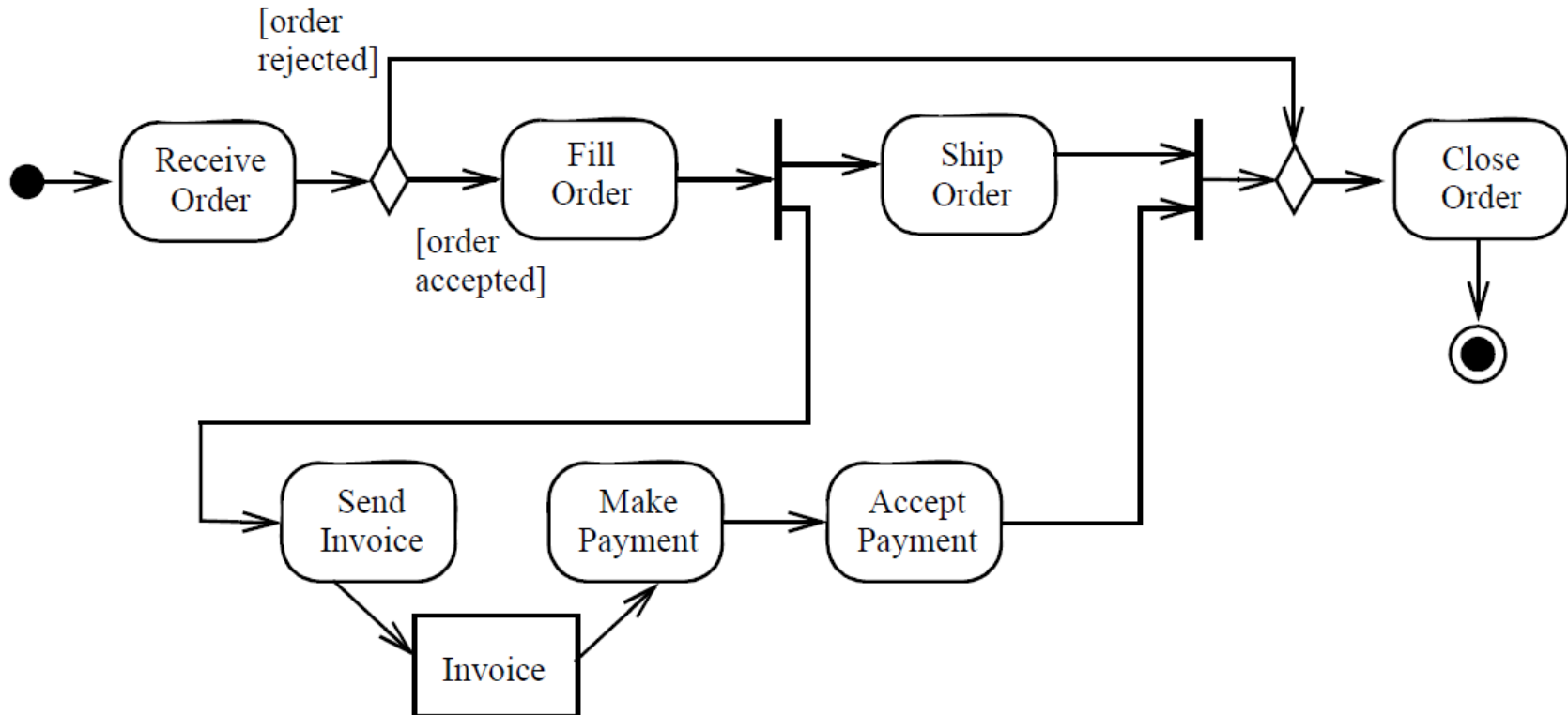


Diagramma delle Attività

- I diagrammi delle attività descrivono il modo in cui diverse attività sono coordinate e possono essere usati per mostrare l'implementazione di una operazione
- Un diagramma delle attività mostra le attività di un sistema in generale e delle sotto-parti, specialmente quando un sistema ha diversi obiettivi e si desidera modellare le dipendenze tra essi prima di decidere l'ordine in cui svolgere le azioni
- I diagrammi delle attività sono **utili anche per descrivere lo svolgimento dei singoli casi d'uso e la loro eventuale dipendenza da altri casi**

Diagramma delle Attività



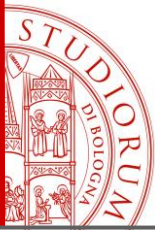
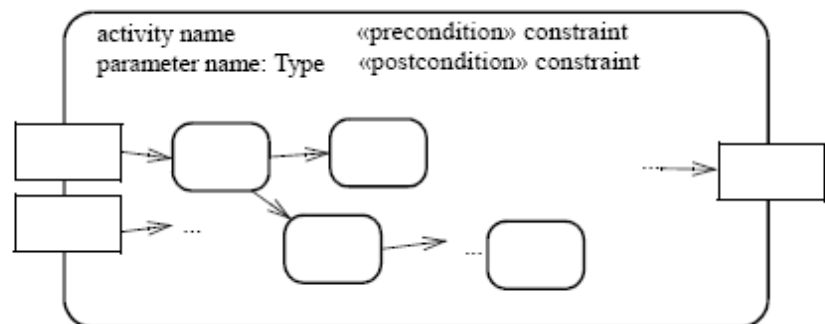


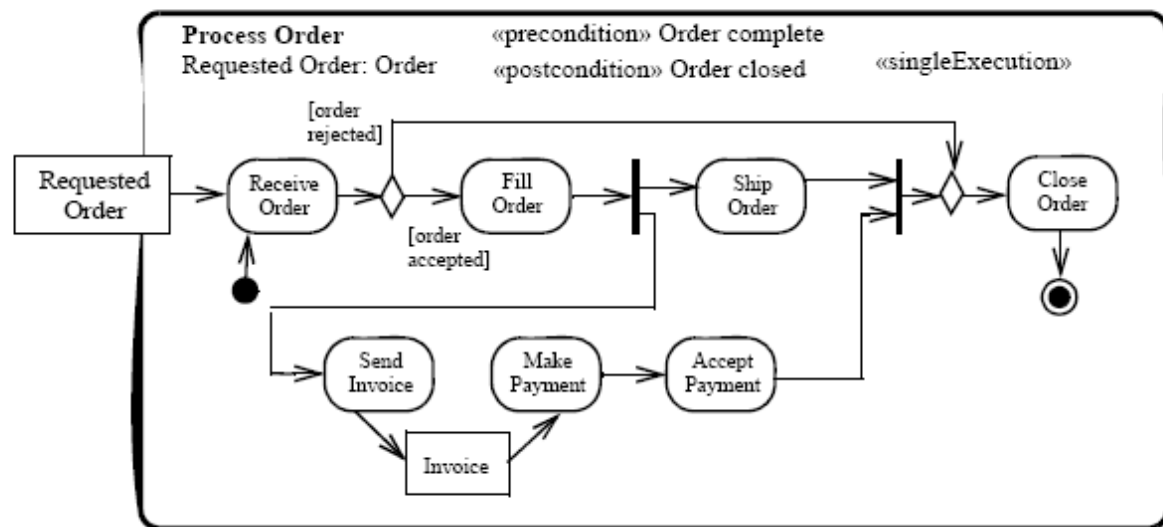
Diagramma delle Attività

- **Attività**: indica un lavoro che deve essere svolto. Da ogni attività possono uscire uno o più **archi**, che indicano il percorso da una attività ad un'altra
- **Arco**: è rappresentato come una freccia proprio come una transizione, ma a differenza di una transizione, un arco non può essere etichettato con eventi o azioni. Un arco può essere etichettato con una condizione di guardia, se l'attività successiva la richiede
- **Sottoattività**: “nasconde” un diagramma delle attività interno ad una attività
- **Start e End Point**: punti di inizio e fine del diagramma. Gli End Point possono anche non essere presenti, oppure essere più di uno

Notazione



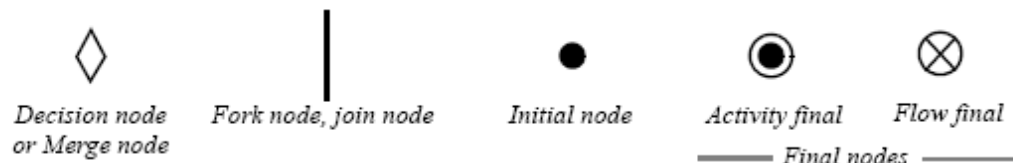
Attività



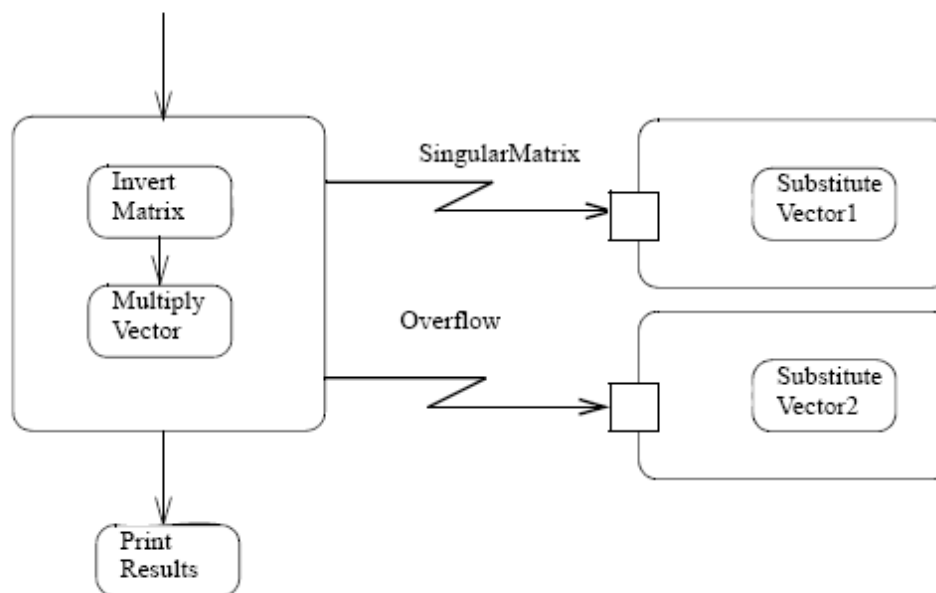
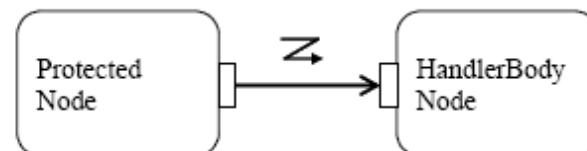
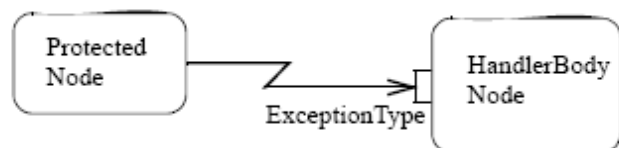
Attività con parametro di ingresso

Diagramma delle Attività

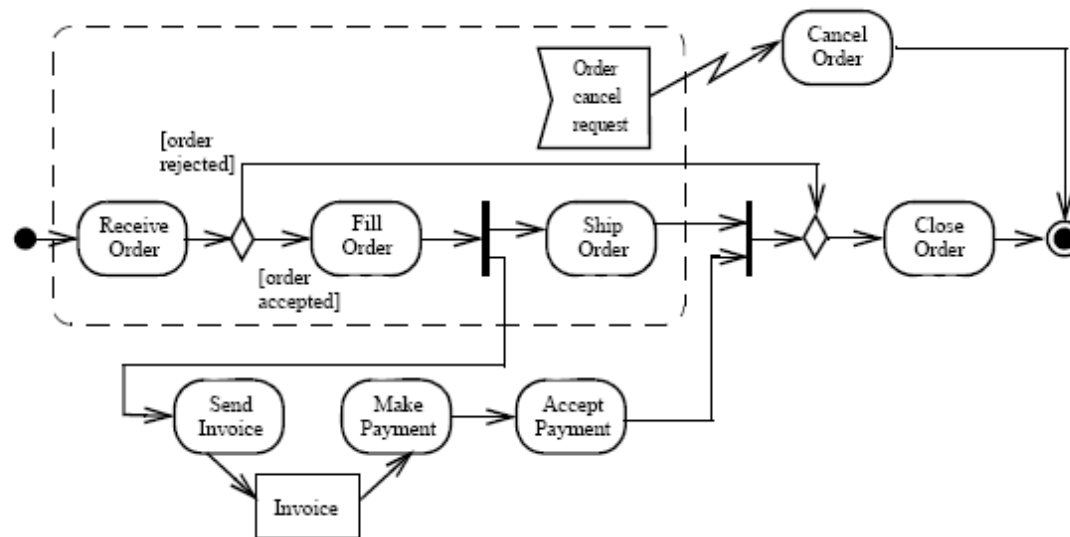
- *Decision e merge* : determinano il comportamento condizionale:
 - *decision* ha una singola transizione entrante e più transizioni uscenti in cui solo una di queste sarà prescelta
 - *merge* ha più transizioni entranti e una sola uscente e serve a terminare il blocco condizionale cominciato con un decision
- *Fork e join*: determinano il comportamento parallelo:
 - quando scatta la transazione entrante, si eseguono in parallelo tutte le transazioni che escono dal fork. Con il parallelismo non è specificata la sequenza. Le fork possono avere delle guardie
 - per la sincronizzazione delle attività parallele è presente il costrutto di join che ha più transazioni entranti ed una sola transazione uscente



Exception Handler

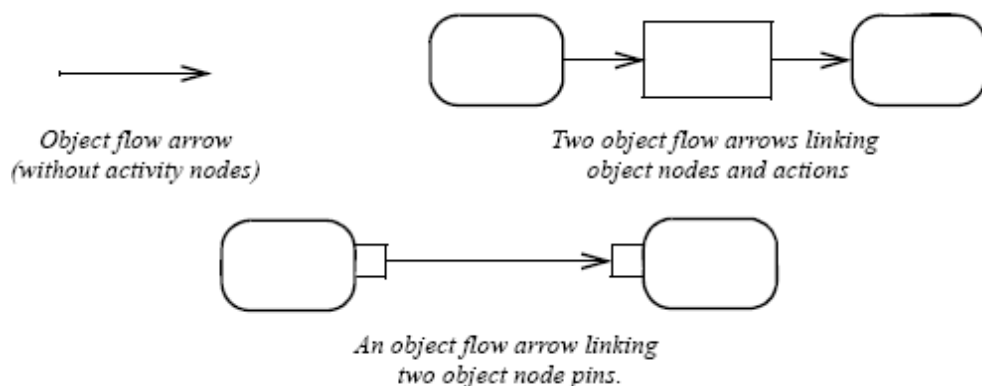


InterruptibleActivityRegion

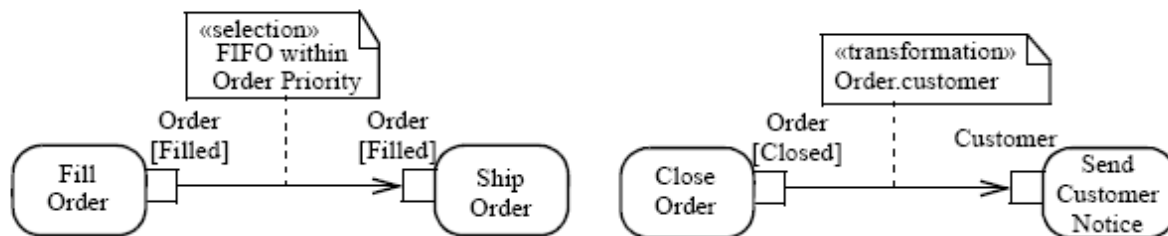


Object Flow

- È un arco che ha oggetti o dati che fluiscono su di esso

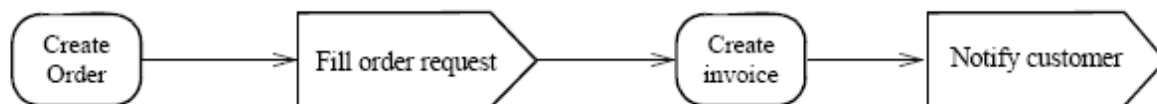


Notazione

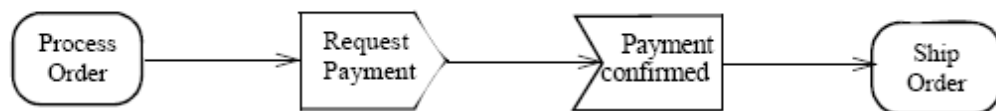


Esempio

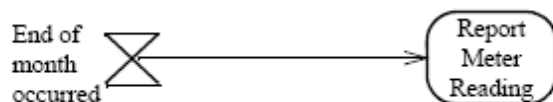
Segnali ed Eventi



SendSignal Action



AcceptSignal Action



Evento ripetuto nel tempo