



Università
Ca' Foscari
Venezia

UML e progettazione sw



Università
Ca' Foscari
Venezia

Diagrammi UML

livello “logico”:

dei casi d'uso

Use Case Diagram

delle classi

Class Diagram

di sequenza

Sequence Diagram

di collaborazione

Collaboration Diagram

di transizione di stato

Statechart Diagram

delle attività

Activity Diagram

livello “fisico”:

dei componenti

Component Diagram

di distribuzione dei componenti

Deployment Diagram



Università
Ca' Foscari
Venezia

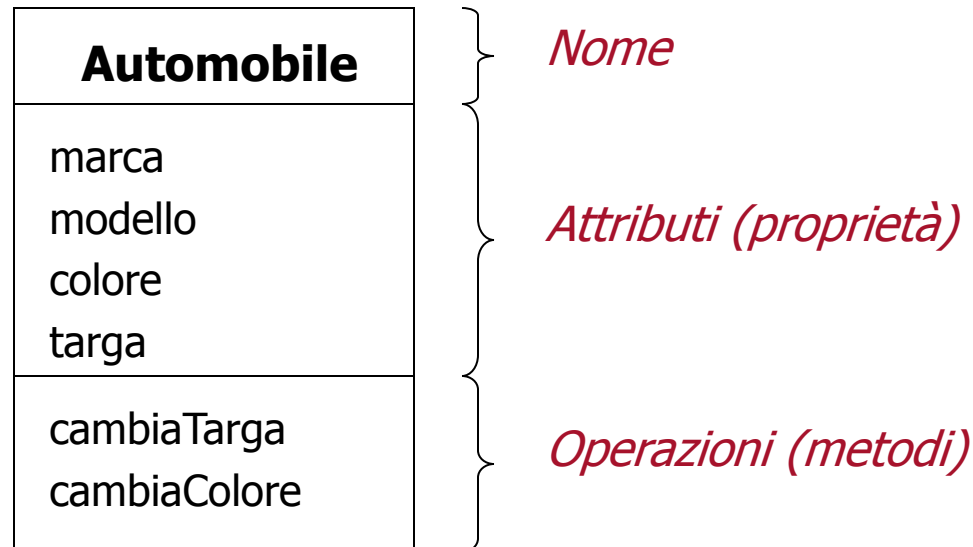
Diagramma delle classi

- rappresenta le classi di oggetti del sistema con i loro attributi e operazioni
- mostra le relazioni tra le classi (associazioni, aggregazioni e gerarchie di specializzazione/generalizzazione)
- può essere utilizzato a diversi livelli di dettaglio (in analisi e in progettazione)

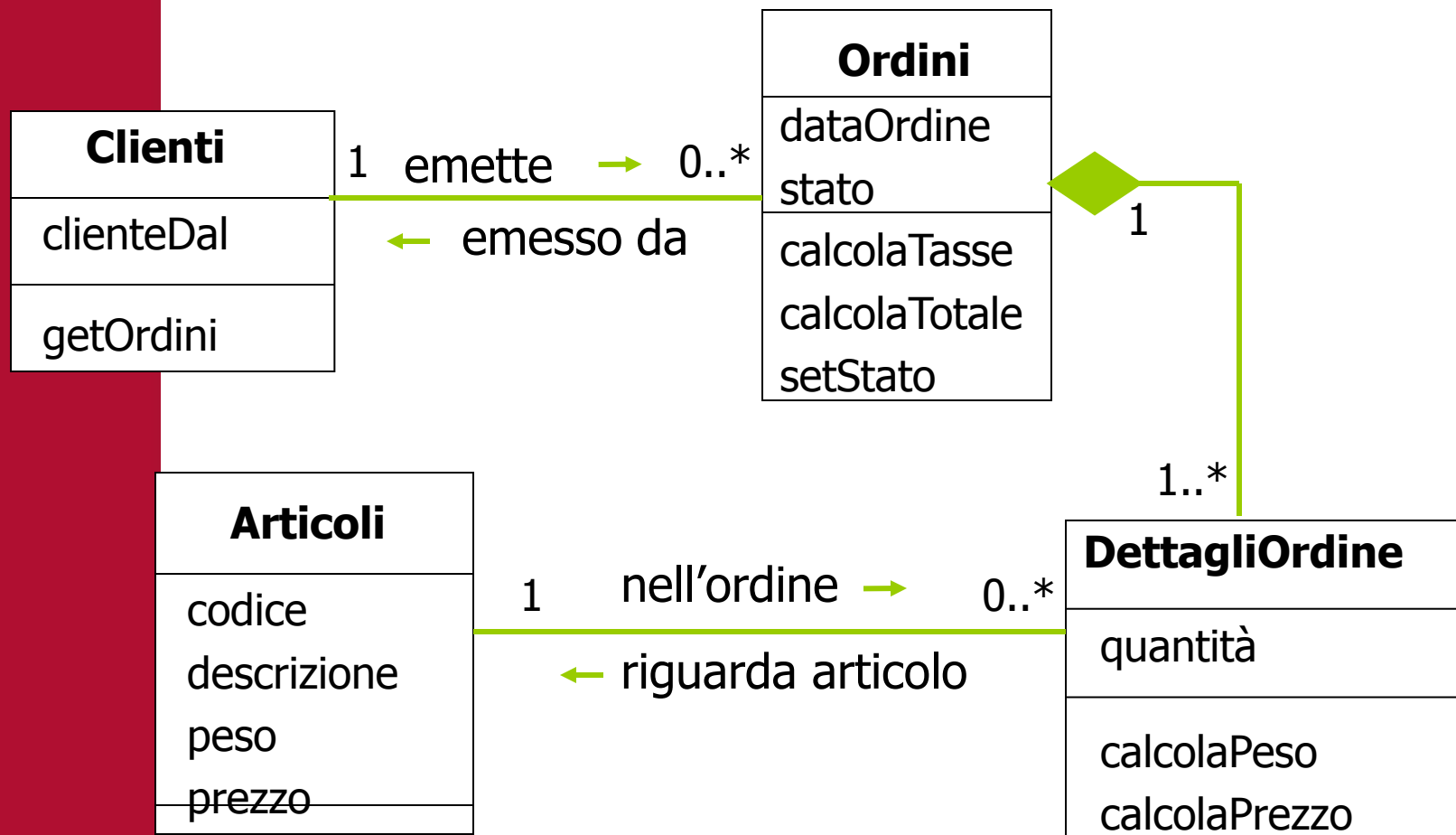


Class diagram: le classi

- Una classe è una tipologia di oggetti, con propri attributi e operazioni
- Rappresentazione di una classe in UML:



Class diagram: associazioni





Class diagram: associazioni

- **Associazione**: correlazione fra classi; nel diagramma è una linea continua fra due classi, con esplicita semantica nei due sensi
- **Molteplicità**: numero di oggetti che partecipano all'associazione. Esempi di molteplicità sono:

1	Esattamente una istanza
0..*	Nessun limite al numero di istanze
1..*	Almeno una istanza
n..m	Da n a m istanze



Class diagram: aggregazione

- Aggregazione: tipo particolare di associazione; esprime concetto “è *parte di*” (*part of*), che si ha quando un insieme è relazionato con le sue parti.
- Si rappresenta con un diamante dalla parte della classe che e' il contenitore



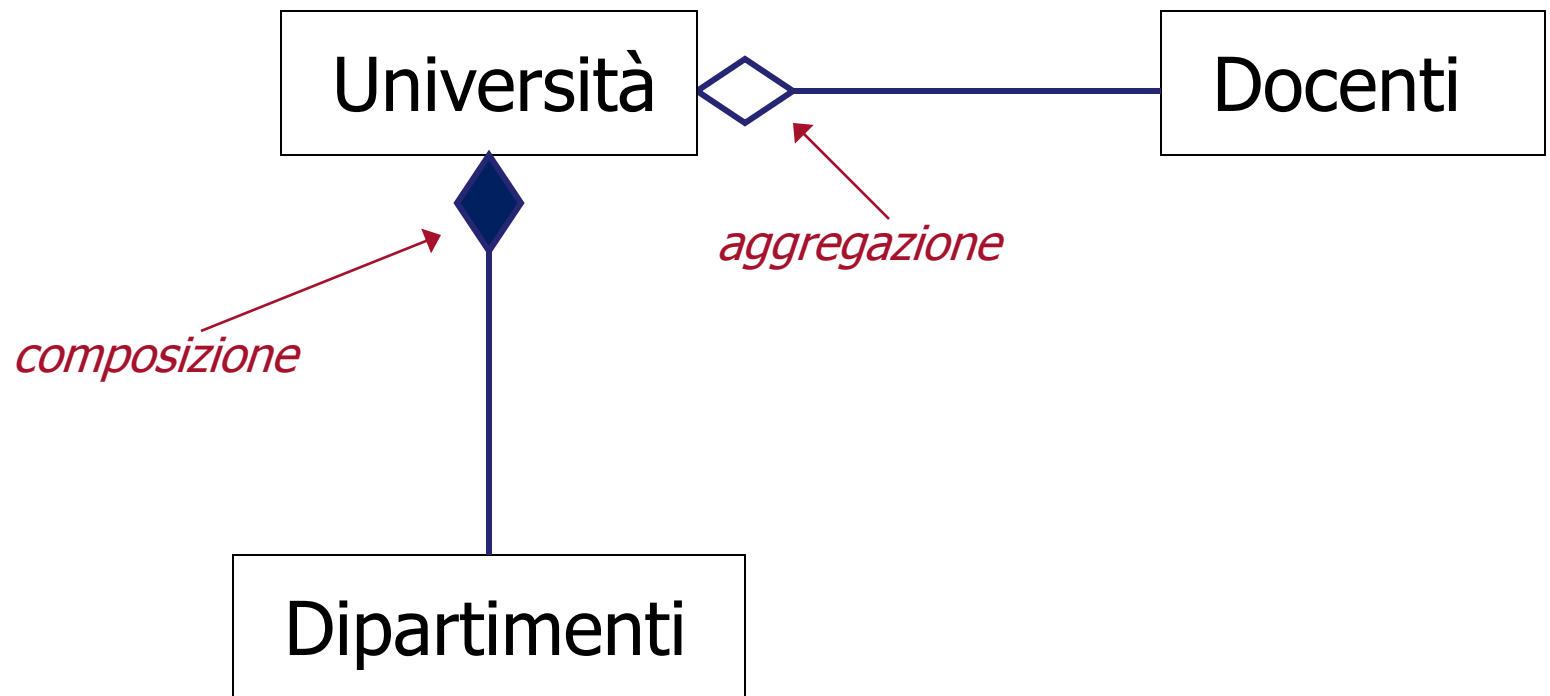
Class diagram: composizione

- E' un caso particolare di aggregazione in cui:
 1. la parte (*componente*) non può esistere da sola, cioè senza la classe *composto*
 2. una componente appartiene ad un solo composto
- Il diamante si disegna pieno



Università
Ca' Foscari
Venezia

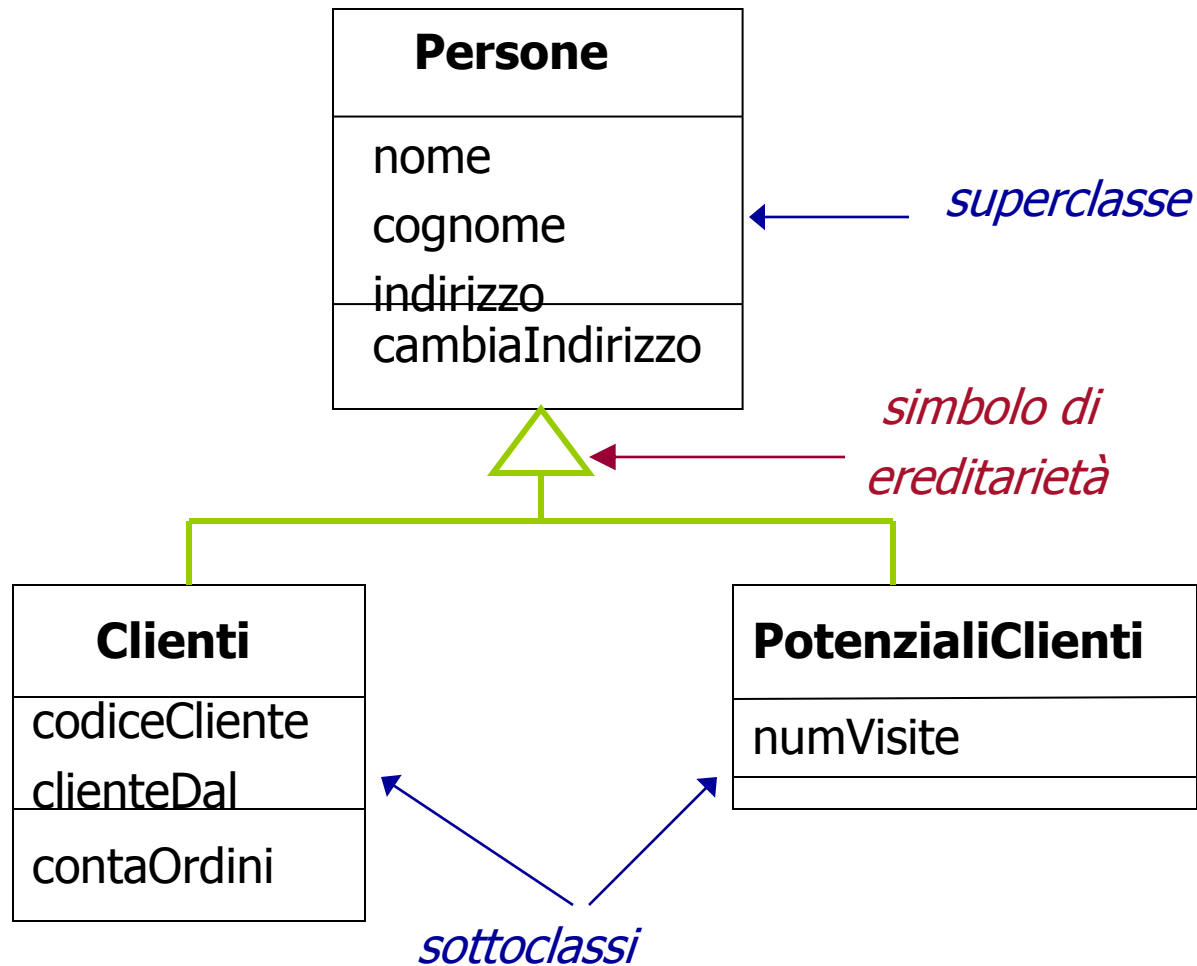
Class diagram: aggregazione/composizione





Università
Ca' Foscari
Venezia

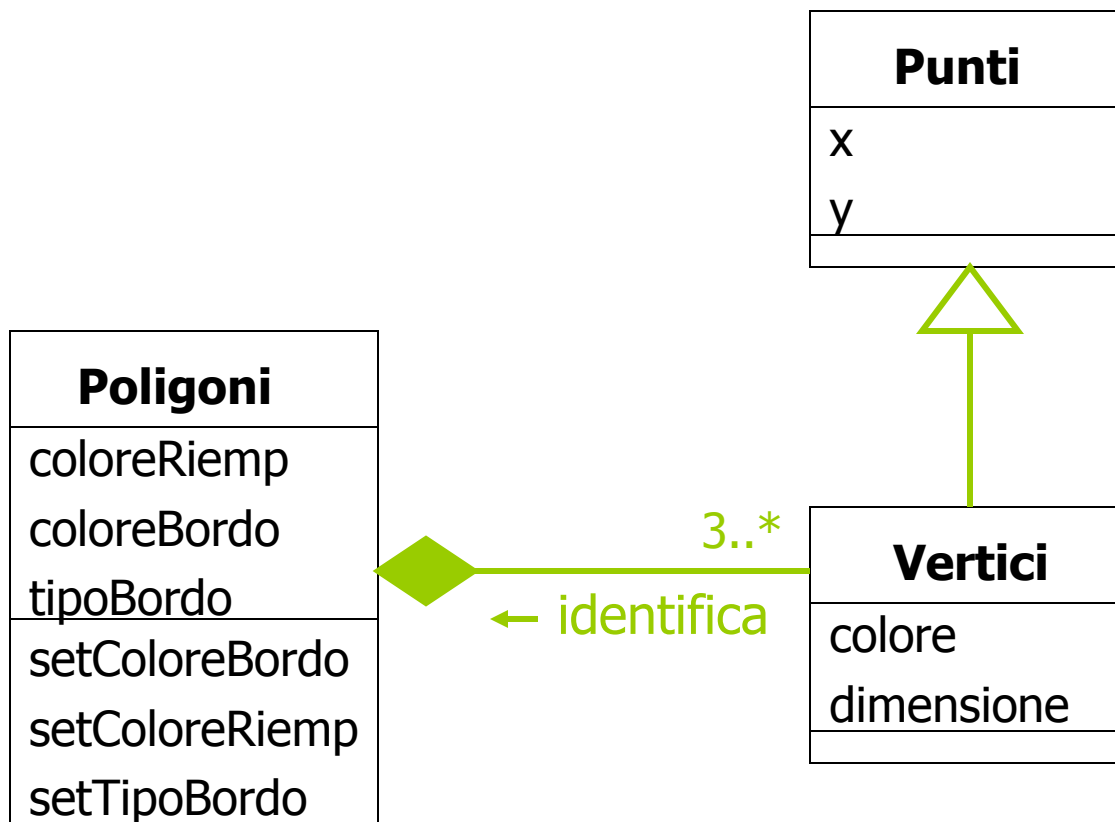
Class diagram: ereditarietà



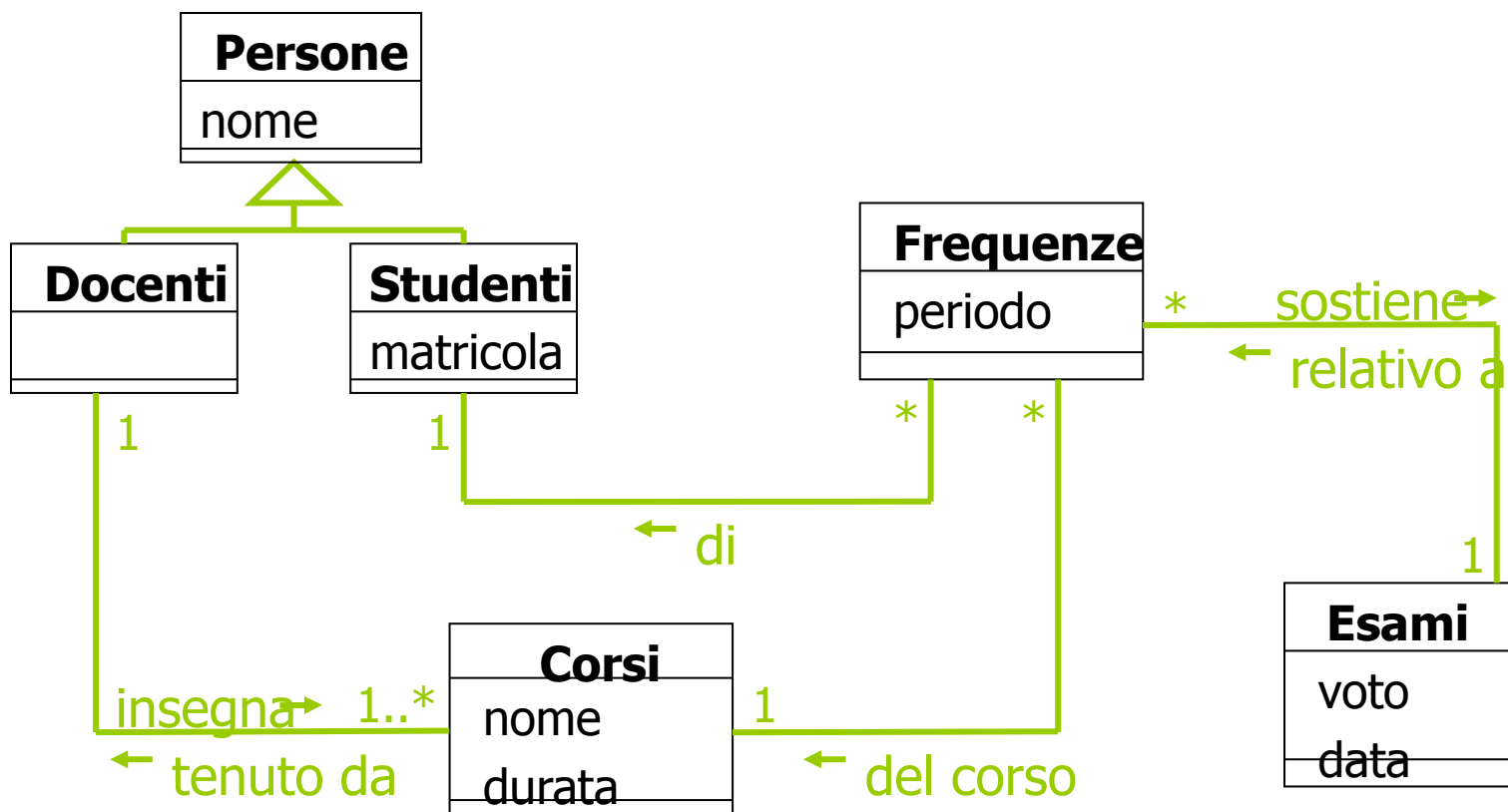


Università
Ca' Foscari
Venezia

Class diagram: esempio



Class diagram: esempio





Università
Ca' Foscari
Venezia

Diagramma di sequenza

- è utilizzato per definire la logica di uno scenario (specifica sequenza di eventi) di un caso d'uso (in analisi e poi ad un maggior livello di dettaglio nella progettazione)
- è uno dei principali input per l'implementazione dello scenario
- mostra gli oggetti coinvolti specificando la sequenza temporale dei messaggi che gli oggetti si scambiano
- è un diagramma che evidenzia come un caso d'uso è realizzato tramite la collaborazione di un insieme di oggetti



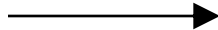
Università
Ca' Foscari
Venezia

Alcune definizioni

- Un diagramma di sequenza è un diagramma che descrive **interazioni** tra oggetti che **collaborano** per svolgere un compito
- Gli oggetti collaborano scambiandosi **messaggi**
- Lo scambio di un messaggio in programmazione ad oggetti equivale all'invocazione di un metodo



Scambio Messaggi Sincroni (1/2)

- Si disegna con una **freccia chiusa**  da chiamante a chiamato. La freccia è etichettata col nome del metodo invocato, e opzionalmente i suoi parametri e il suo valore di ritorno
- Il chiamante attende la terminazione del metodo del chiamato prima di proseguire
- Il **life-time** (durata, vita) di un metodo è rappresentato da un rettangolino che collega freccia di invocazione e freccia di ritorno

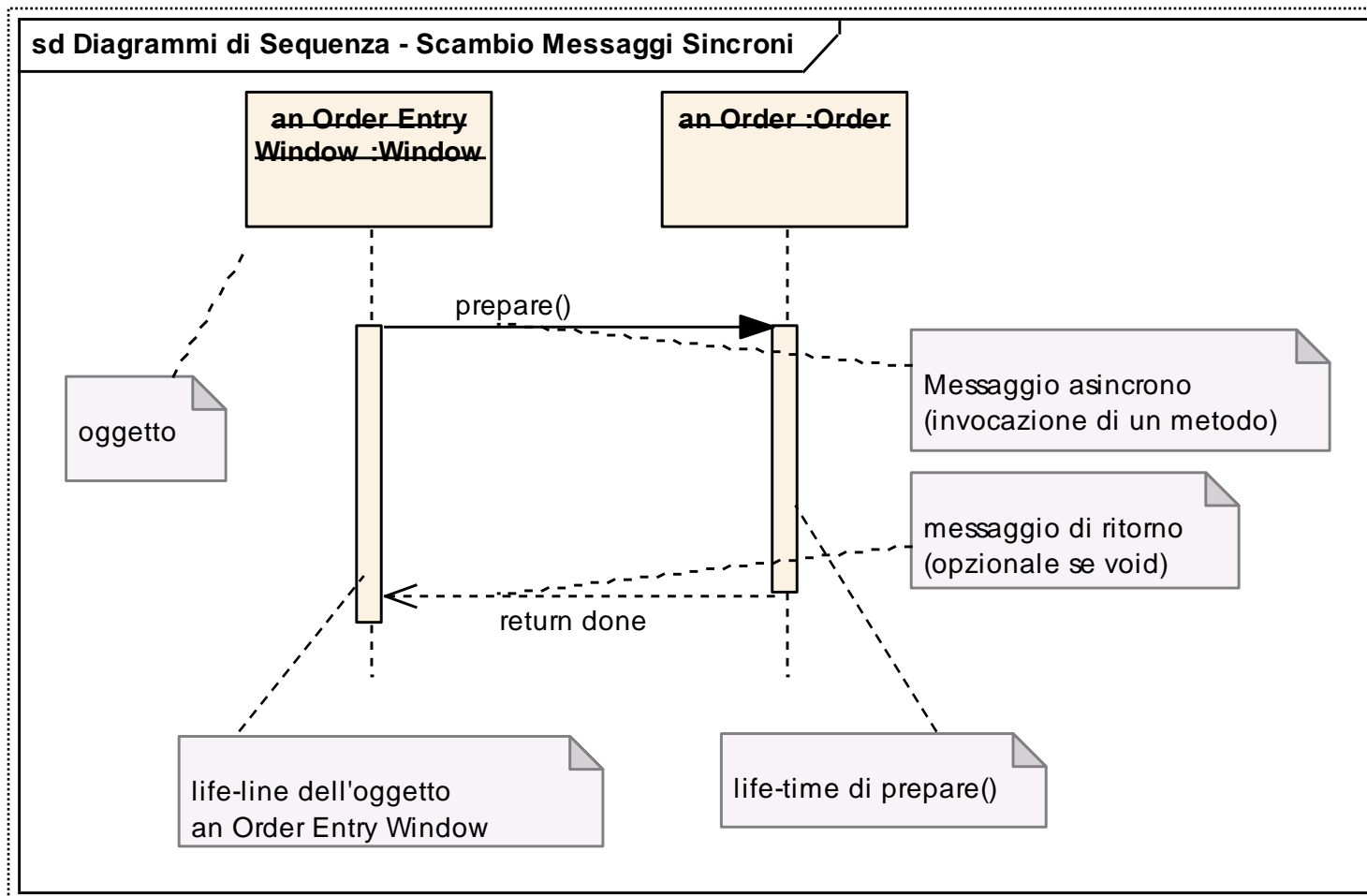


Università
Ca' Foscari
Venezia

Scambio Messaggi Sincroni (2/2)

- Life-time corrisponde ad avere un record di attivazione di quel metodo sullo stack di attivazione
- Il ritorno è rappresentato con una freccia tratteggiata
- Il ritorno è sempre opzionale. Se si omette, la fine del metodo è decretata dalla fine del life-time

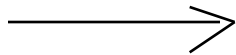
Scambio Messaggi Sincroni



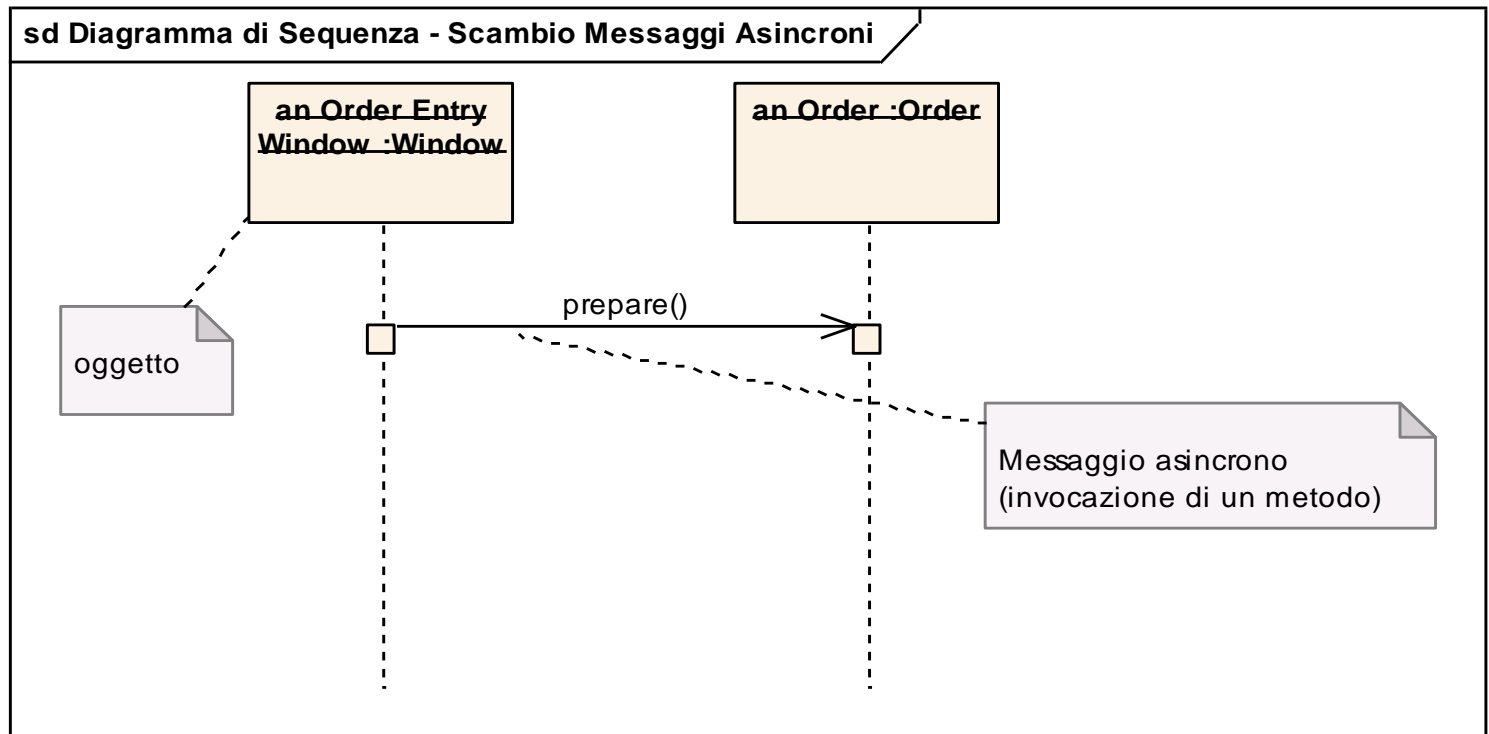


Università
Ca' Foscari
Venezia

Scambio Messaggi Asincroni

- Si usano per descrivere interazioni concorrenti
- Si disegna con una **freccia aperta**  da chiamante a chiamato. La freccia è etichettata col nome del metodo invocato, e opzionalmente i suoi parametri e il suo valore di ritorno
- Il chiamante non attende la terminazione del metodo del chiamato, ma prosegue subito dopo l'invocazione
- Il ritorno non segue quasi mai la chiamata

Scambio Messaggi Asincroni





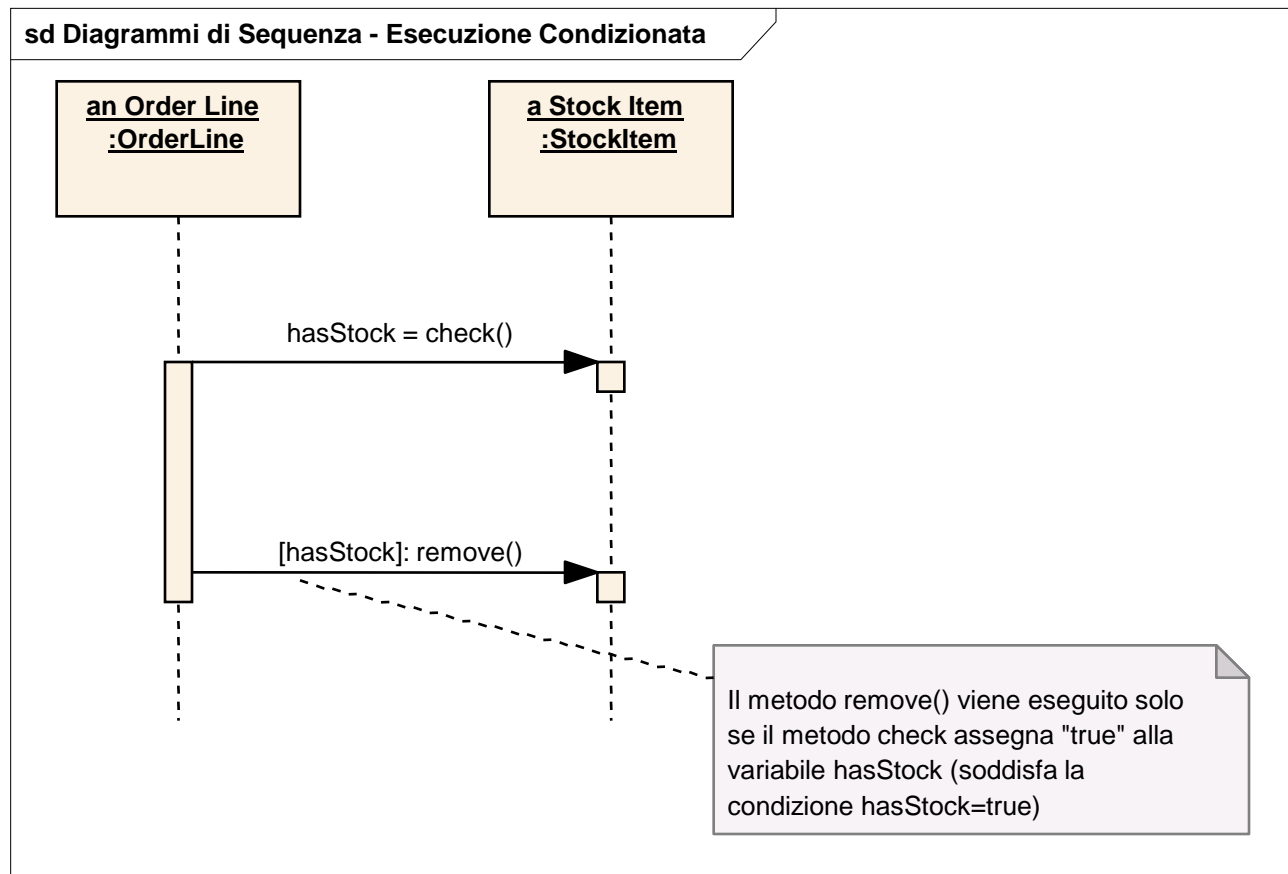
Università
Ca' Foscari
Venezia

Esecuzione condizionale di un messaggio

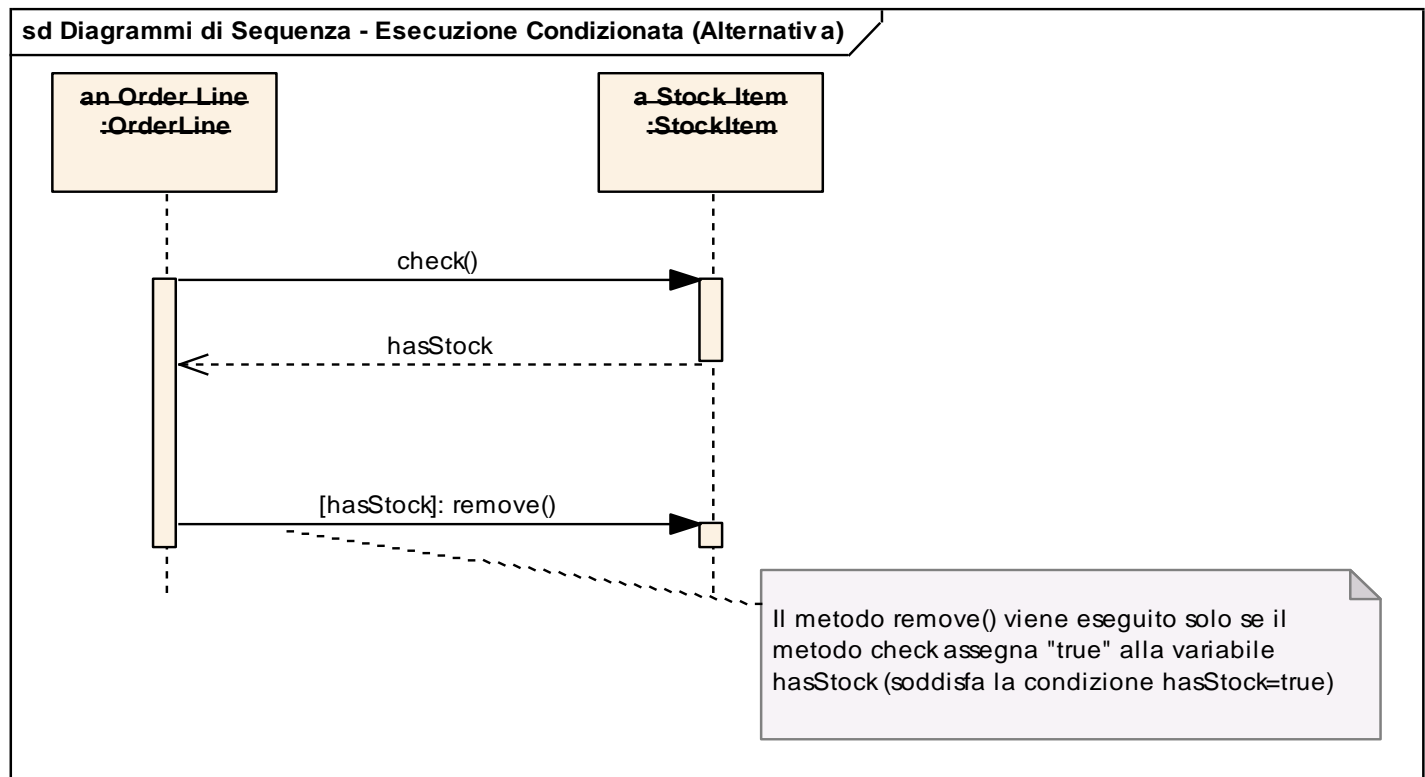
- L'esecuzione di un metodo può essere assoggettata ad una **condizione**. Il metodo viene invocato solo se la condizione risulta verificata a run-time
- Si disegna aggiungendo la condizione, racchiusa tra parentesi quadre, che definisce quando viene eseguito il metodo
- Sintassi:
`[cond] : nomeMetodo()`

Esecuzione condizionale di un messaggio

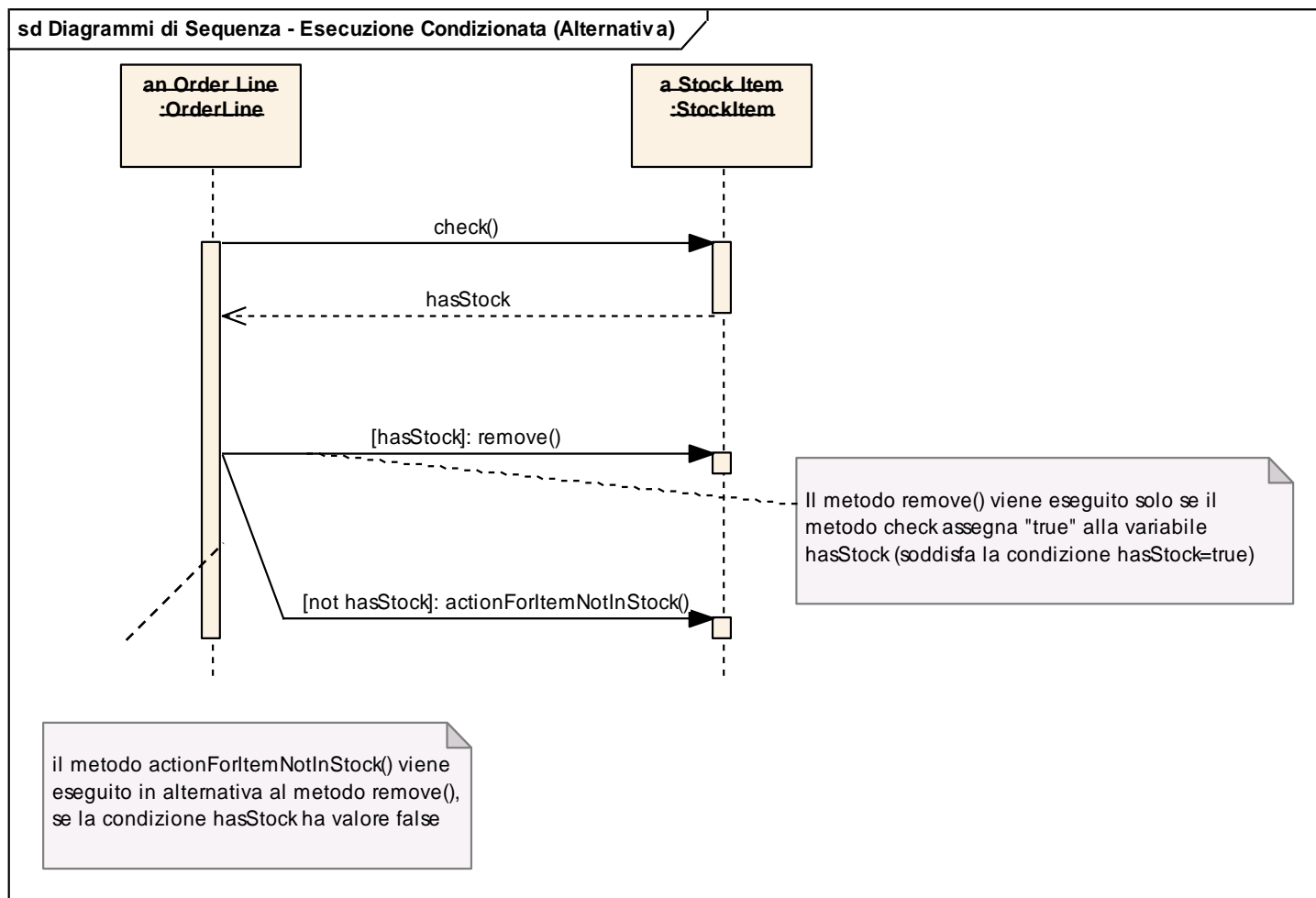
Esempio



Esecuzione condizionata di un messaggio – esempio (un'alternativa)



Esecuzione condizionata di un messaggio (un'altra alternativa)





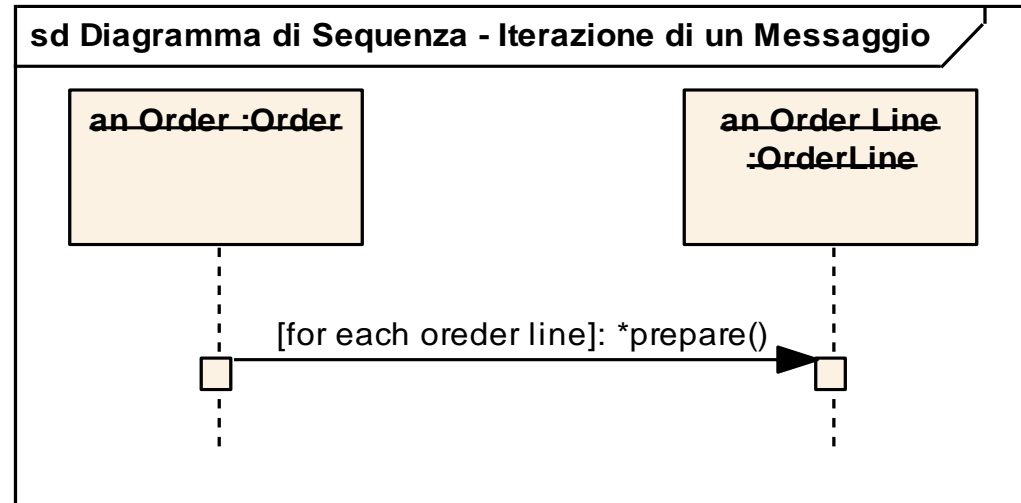
Università
Ca' Foscari
Venezia

Iterazione di un messaggio

- Rappresenta l'**esecuzione ciclica** di messaggi
 - Si disegna aggiungendo un * (asterisco) prima del metodo su cui si vuole iterare
 - Si può aggiungere la condizione che definisce l'iterazione
 - La condizione si rappresenta tra parentesi quadre.
- Sintassi completa:

[cond] : * nomeMetodo()

Iterazione di un messaggio Esempio



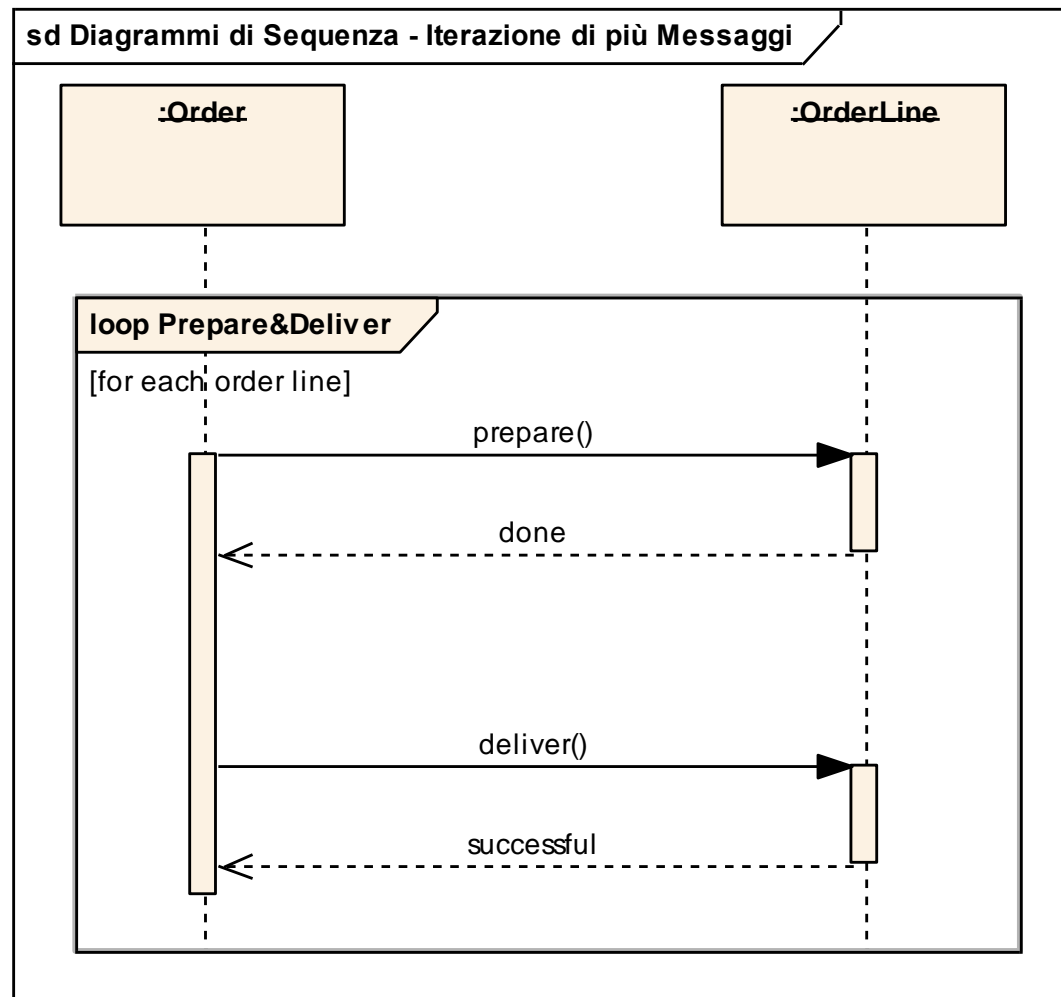


Università
Ca' Foscari
Venezia

Iterazione di un blocco di messaggi

- Rappresenta l'esecuzione ciclica di più messaggi
- Si disegna raggruppando con un **blocco** (riquadro, box) i messaggi (metodi) su cui si vuole iterare
- Si può aggiungere la condizione che definisce l'iterazione sull'angolo in alto a sinistra del blocco
- La condizione si rappresenta al solito tra parentesi quadre

Iterazione di un blocco di messaggi

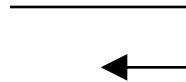




Università
Ca' Foscari
Venezia

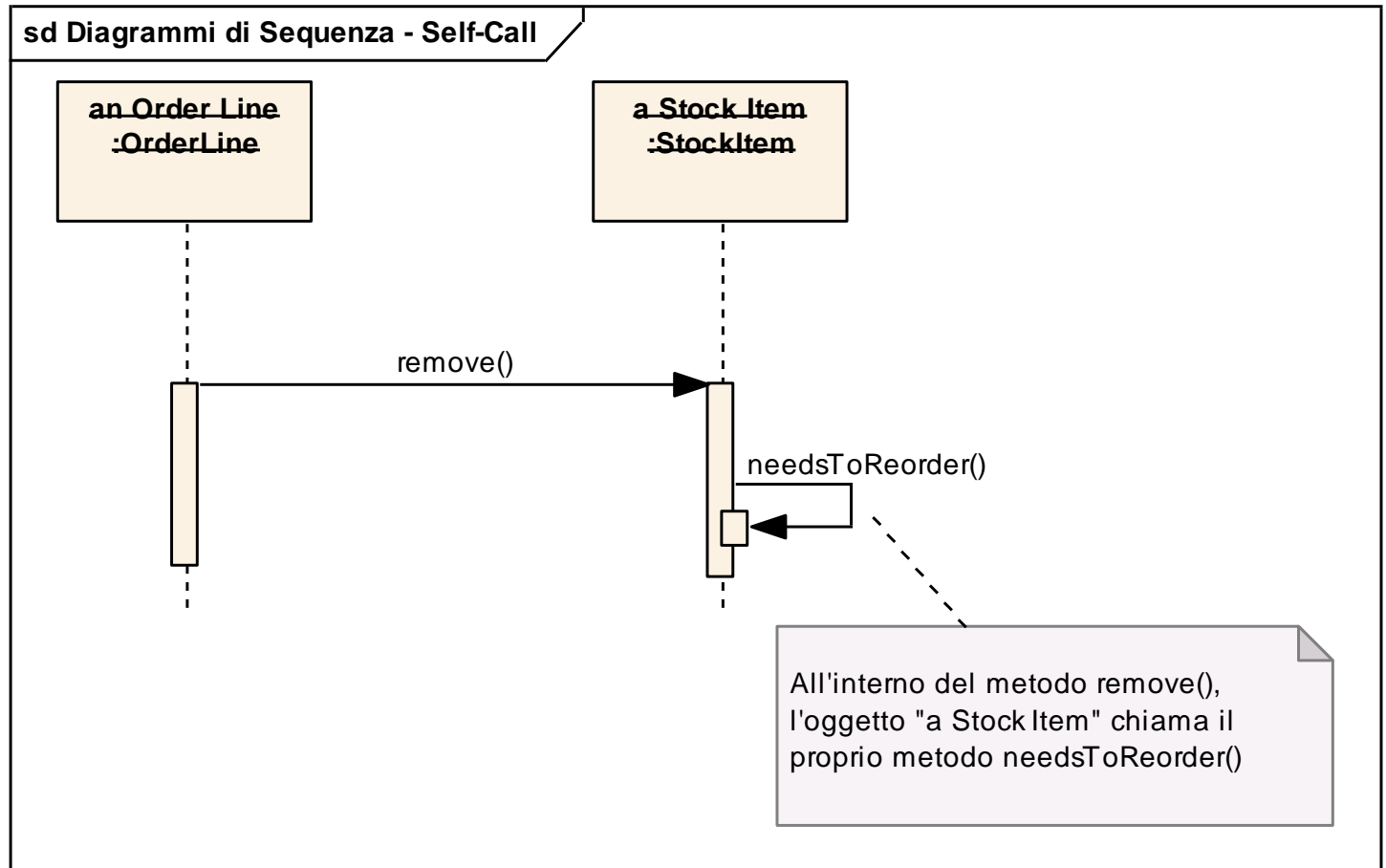
“Auto-Chiamata” (Self-Call)

- Descrive un oggetto che invoca un suo metodo (chiamante e chiamato coincidono)
- Si rappresenta con una “freccia circolare”
che rimane all’interno del life time di uno stesso metodo





Auto-Chiamata" (Self-Call)



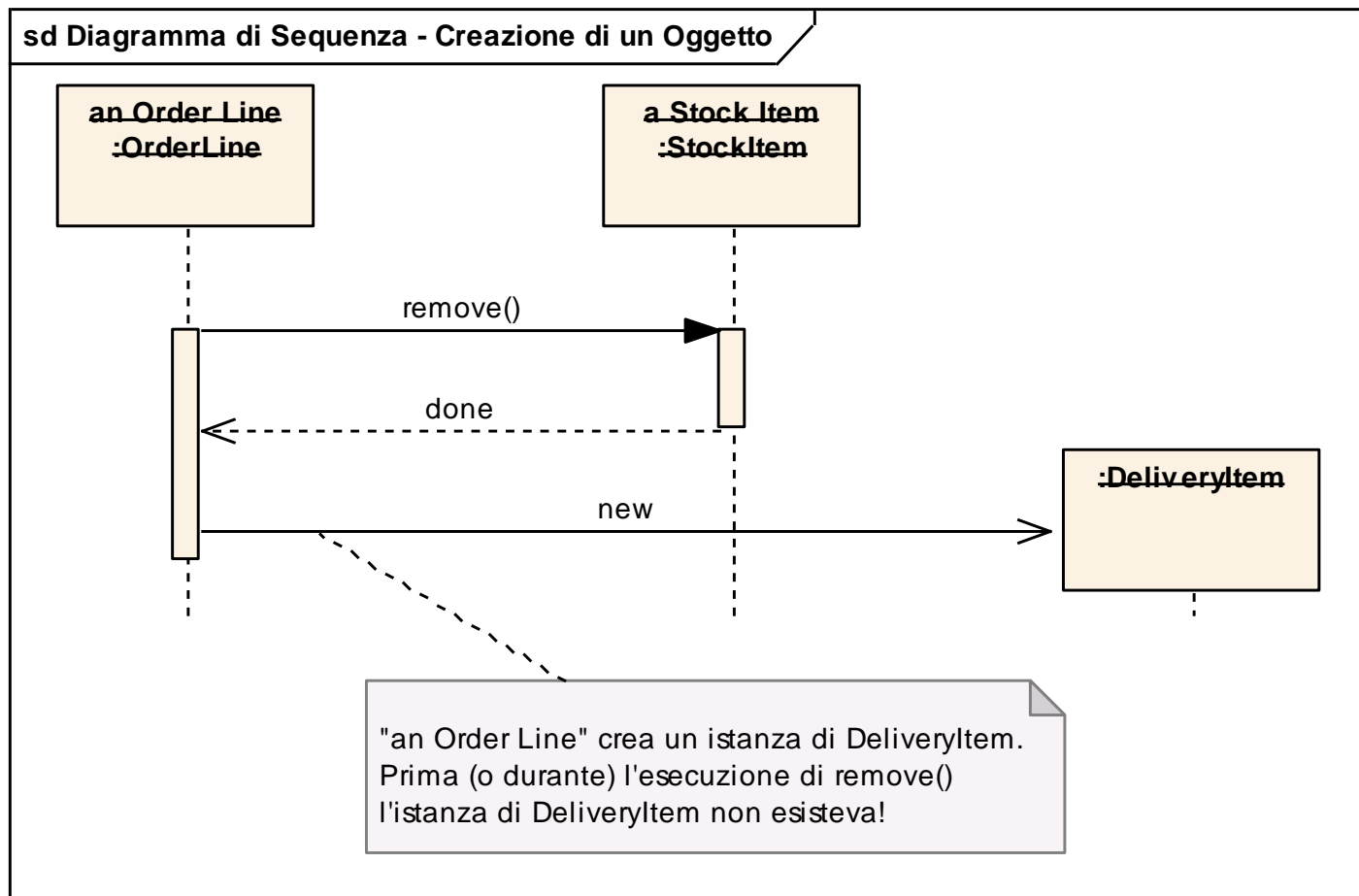


Università
Ca' Foscari
Venezia

Costruzione di un oggetto

- Rappresenta la costruzione di un nuovo oggetto non presente nel sistema fino a quel momento
- Messaggio etichettato new, create,...
- L'oggetto viene collocato nell'asse temporale in corrispondenza dell'invocazione nel metodo new (o create...)

Costruzione di un oggetto



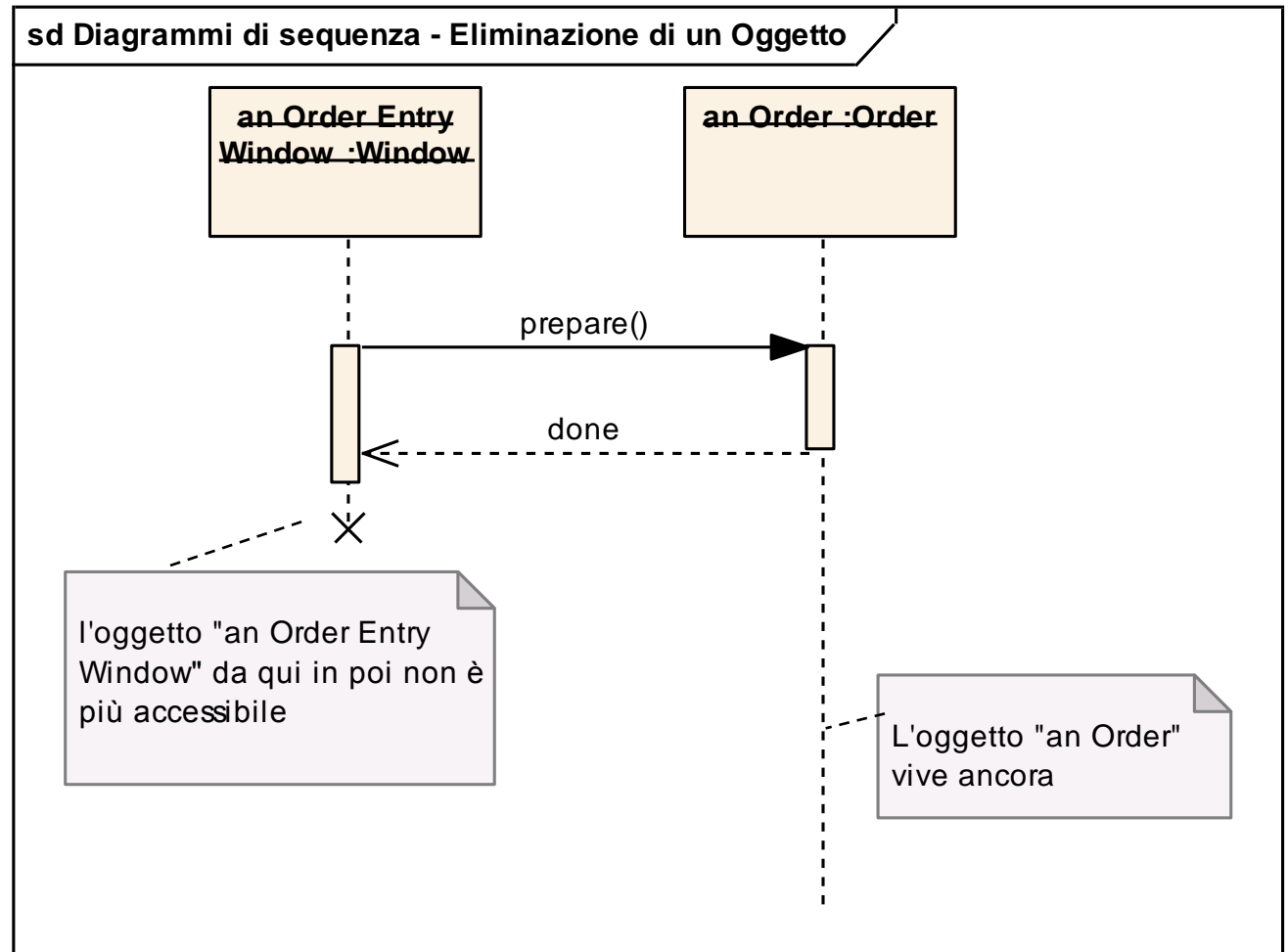


Università
Ca' Foscari
Venezia

Eliminazione di un oggetto

- Rappresenta la distruzione di un oggetto presente nel sistema fino a quel momento
- Si rappresenta con una X posta in corrispondenza della life-line dell'oggetto
- Da quel momento in poi non è legale invocare alcun metodo dell'oggetto distrutto

Eliminazione di un oggetto



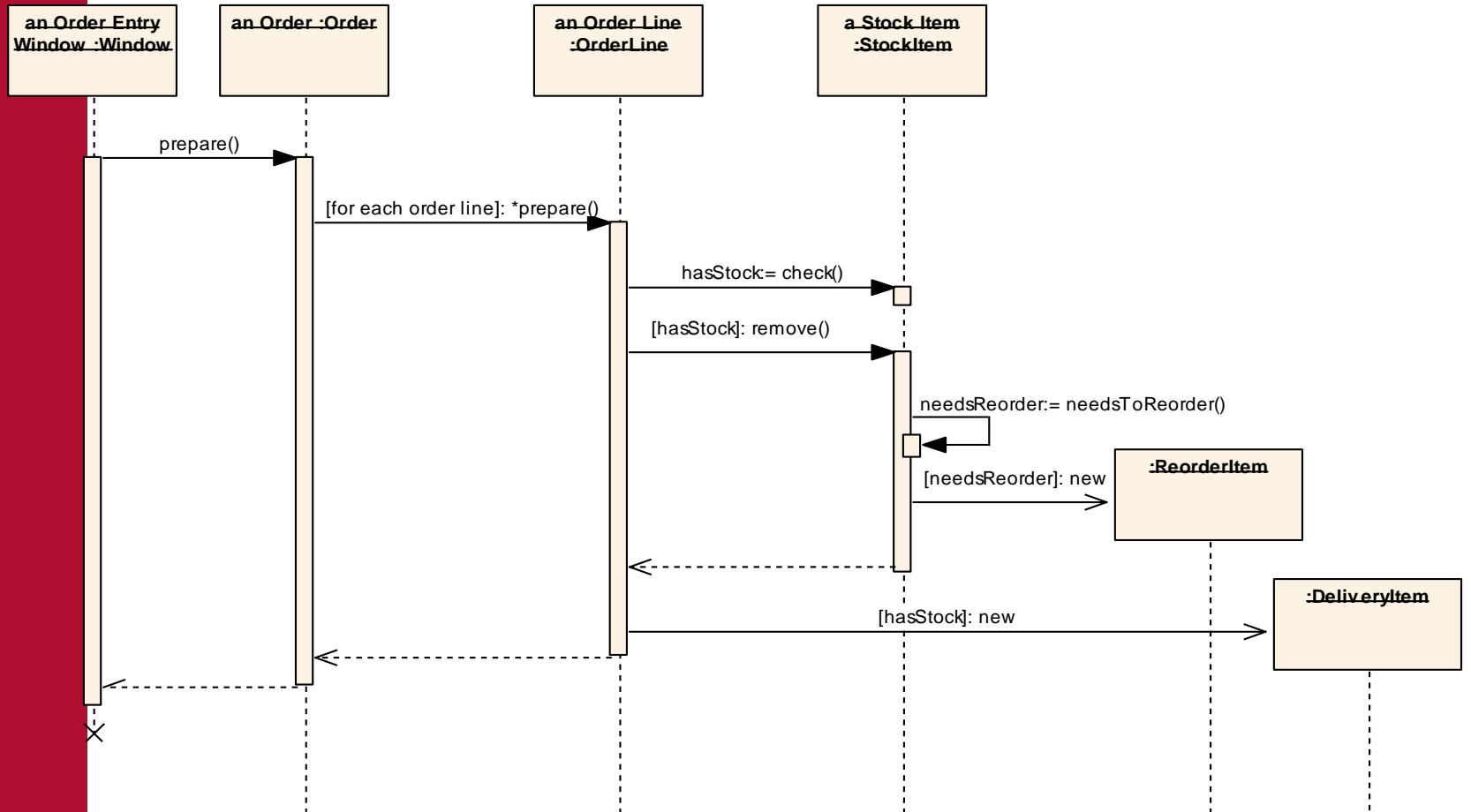


Mettiamo tutto insieme...

- Costruiamo un diagramma di sequenza per il seguente use case
 - Una finestra di tipo Order Entry invia il messaggio “prepare” ad un Ordine (Order)
 - L'ordine invia il messaggio “prepare” ad ogni sua linea (Order Line)
 - Ogni linea verifica gli elementi in stock (Stock Item)
 - Se il controllo ha esito positivo, la linea rimuove l'appropriata quantità di elementi in stock e crea un'unità di delivery (DeliveryItem)
 - Se gli elementi in stock rimanenti scendono al di sotto di una soglia di riordino, viene richiesto un riordino (ReorderItem)

Mettiamo tutto insieme...

sd Diagrammi di Sequenza - La Sintassi



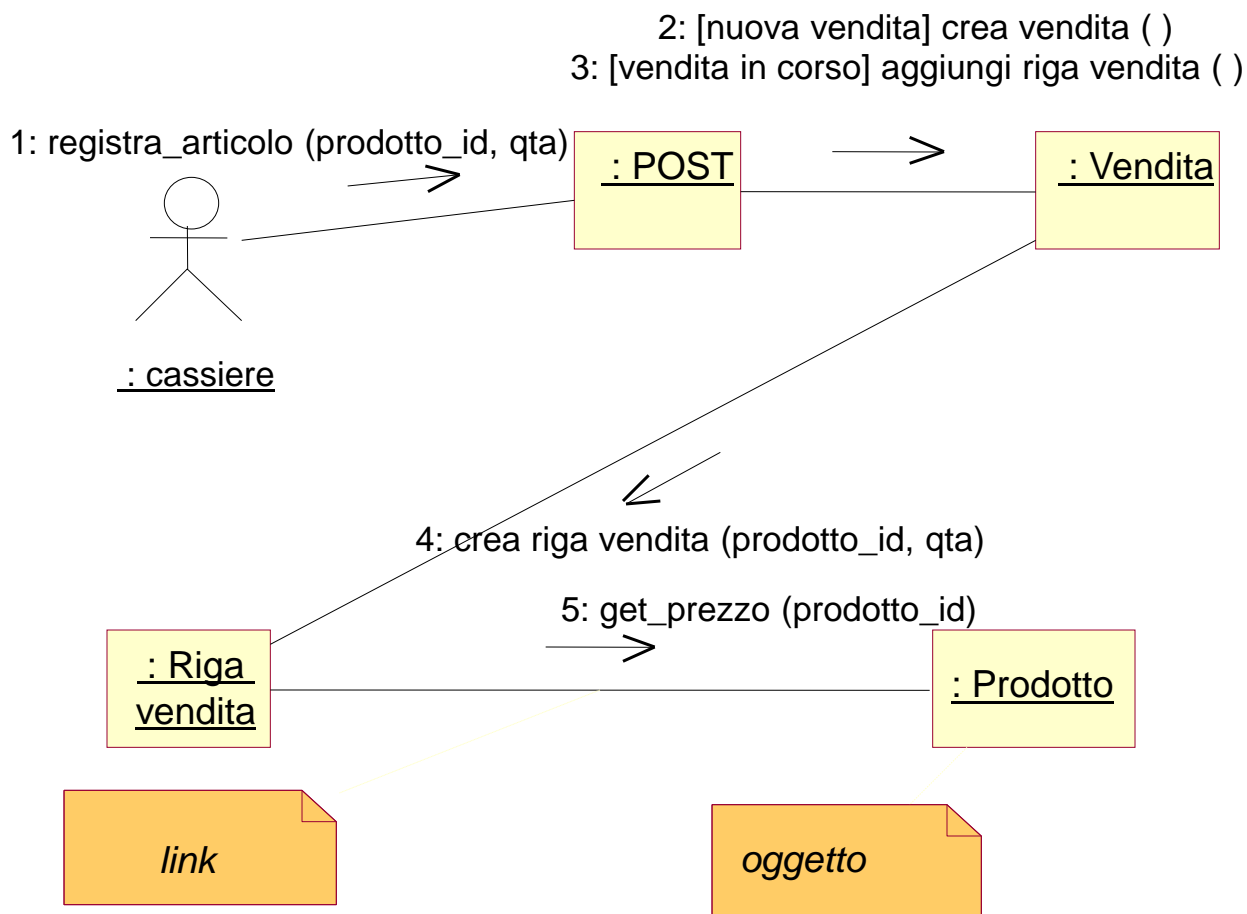


Università
Ca' Foscari
Venezia

Diagramma di collaborazione

- è un diagramma che rappresenta un insieme di oggetti che collaborano per realizzare il comportamento di uno scenario di un caso d'uso
- a differenza del diagramma di sequenza, mostra i link (legami) tra gli oggetti che si scambiano messaggi, mentre la sequenza di tali messaggi è meno evidente
- può essere utilizzato in fasi diverse (analisi, disegno di dettaglio)

Diagramma di collaborazione





Statecharts

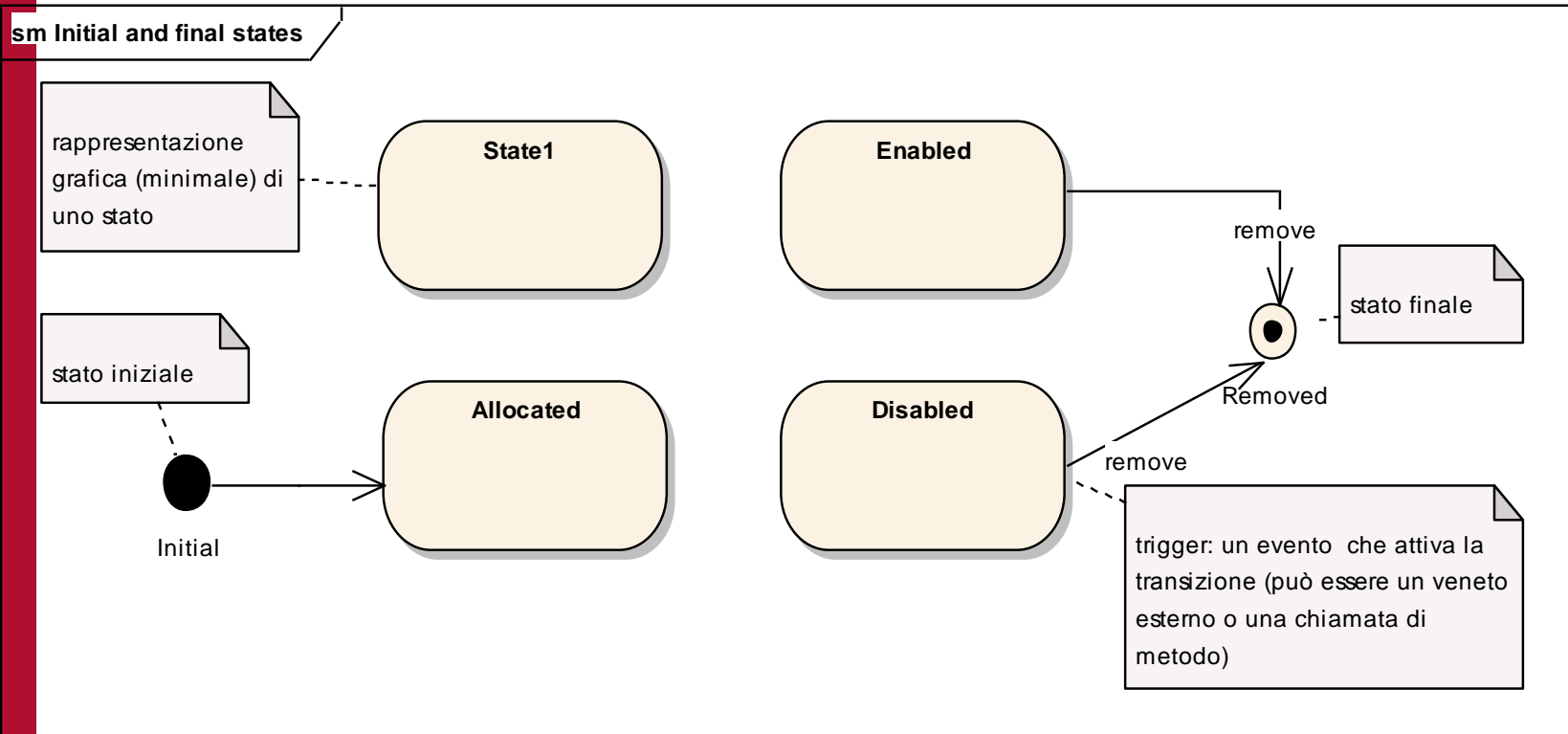
- Un diagramma di transizione di stati (statechart) è normalmente utilizzato per modellare il ciclo di vita degli oggetti di una singola classe
- mostra gli eventi che causano la transizione da uno stato all'altro, le azioni eseguite a fronte di un determinato evento
- quando un oggetto si trova in un certo stato può essere interessato da determinati eventi (e non da altri)
- è opportuno utilizzarlo solo per le classi che presentano un ciclo di vita complesso e segnato da una successione ben definita di eventi



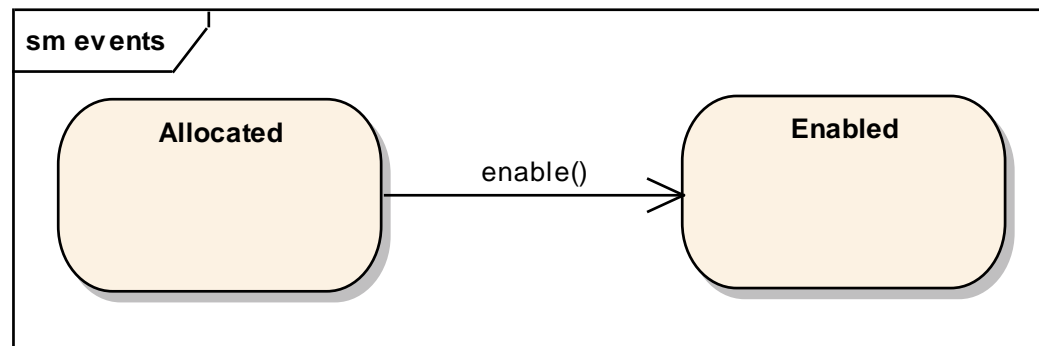
Diagrammi di stato / statecharts)

- ❑ Un diagramma di stato rappresenta **il ciclo di vita degli oggetti di una classe**
- ❑ Il ciclo di vita è descritto in termini di
 - **Eventi**
 - **Stati**
 - **Transizioni di stato**
- ❑ Gli eventi possono attivare delle transizioni di stato
- ❑ Un evento in uno statechart corrisponde ad un messaggio in un sequence diagram
- ❑ Uno stato è costituito da un insieme di “valori significativi” assunti dagli attributi dell’oggetto che ne influenzano il comportamento

- ❑ Esistono due stati “speciali”, detti *pseudostati*:
 - Lo stato iniziale
 - Lo stato finale
- ❑ Un oggetto può non avere uno stato finale (se non viene mai distrutto)



- ❑ Un evento può essere:
 - **L'invocazione sincrona** di un metodo (una “call”)
 - La ricezione di una **chiamata asincrona** (“signal”) – ad esempio la notifica di un'eccezione lanciata
 - **Una condizione predefinita** che diventa vera (si parla in questo caso di “change event”)
 - La fine di un **“periodo di tempo”** come quello impostato da un timer (“elapsed-time event”)
- ❑ Un evento si può rappresentare graficamente con una freccia (transizione) etichettata con il nome del metodo o della condizione associata all'evento stesso





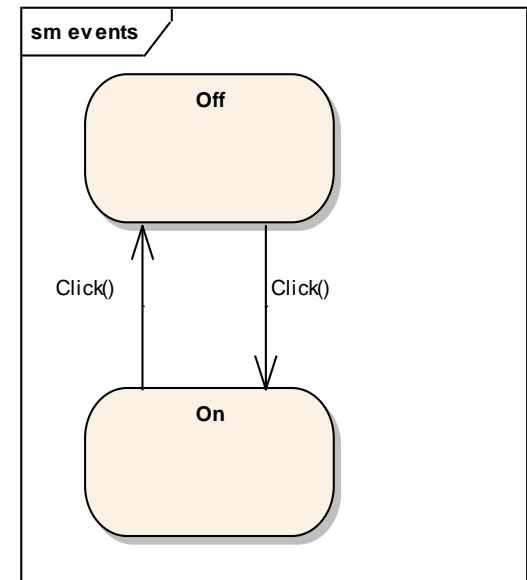
- ❑ Un evento può essere rappresentato anche mediante un'espressione testuale avente la seguente sintassi:
 - `event-name` `(' [comma-separated-parameter-list] ')`
`[['guard-condition']] / [action-expression]`

dove:

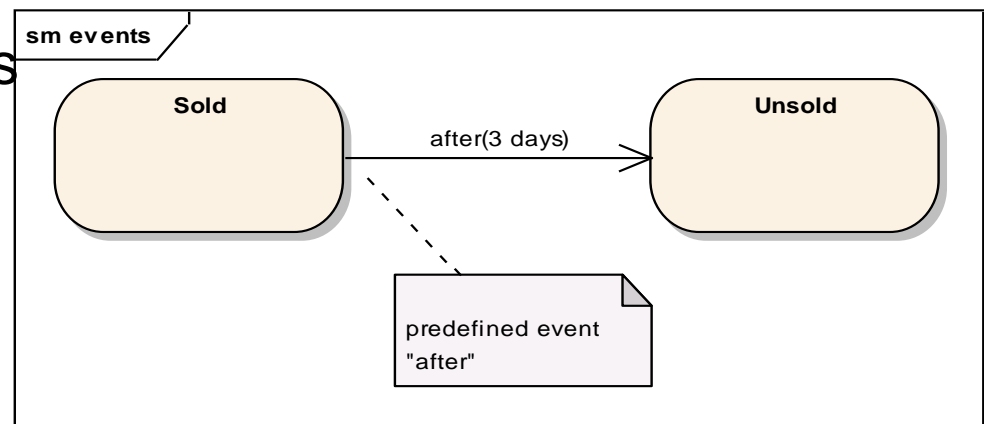
- **event-name** identifica l'evento
- **parameter-list** definisce i valori dei dati che possono essere passati come parametro con l'evento
- **guard-condition** determina se l'oggetto che riceve l'evento deve rispondere ad esso (ossia eseguire il metodo associato)
- **action-expression** definisce come l'oggetto risponde all'evento

❑ Event + state = response

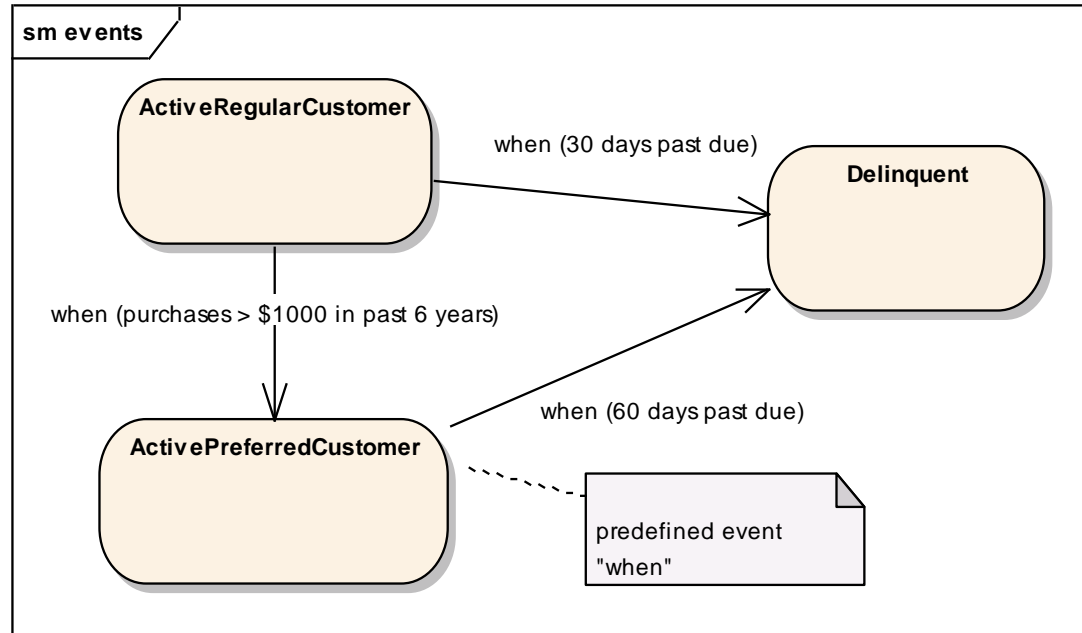
Lo stesso evento causa diversi comportamenti in base allo stato in cui l'oggetto che riceve l'evento si trova



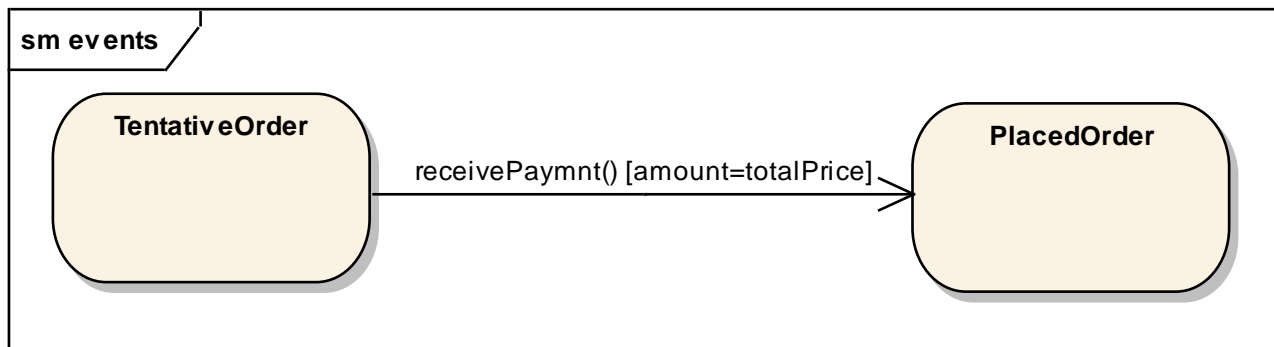
Elapsed-time Events



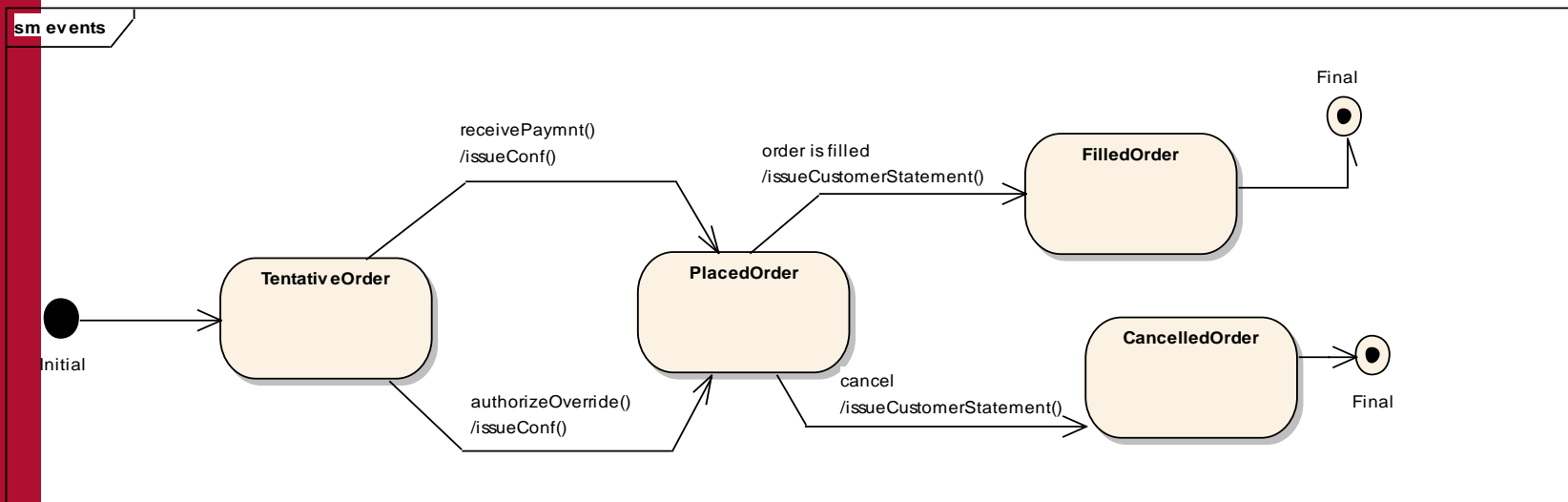
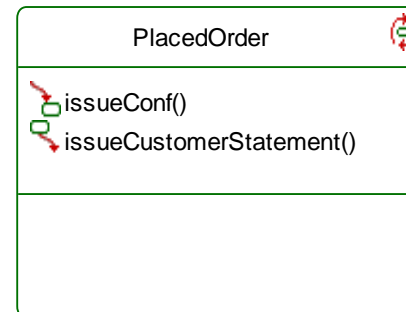
□ Change event



□ Guarded event



- ❑ **entry action**: azione che viene eseguita in una transizione entrante nello stato
- ❑ **exit action**: azione che viene eseguita in una transizione uscente dallo stato





Modellare le attività

- ❑ All'interno degli stati posso essere eseguite delle attività
- ❑ Negli statechart distinguiamo tra
 - ❑ *Azion*: operazioni atomiche
 - ❑ *Attività*: operazioni generalmente non atomiche
- ❑ Le azioni provocano un cambiamento di stato (entry/exit) e quindi non possono essere interrotte
- ❑ Le attività non alterano lo stato dell'oggetto



- ❑ Quando si verifica un evento associato ad una transizione, l'ordine di esecuzione è il seguente:
 1. Se è in esecuzione un'attività, questa viene interrotta
 2. Si esegue l'exit action
 3. Si esegue l'azione associata all'evento
 4. Si esegue l'entry action del nuovo stato
 5. Si inizia l'esecuzione delle eventuali attività del nuovo stato

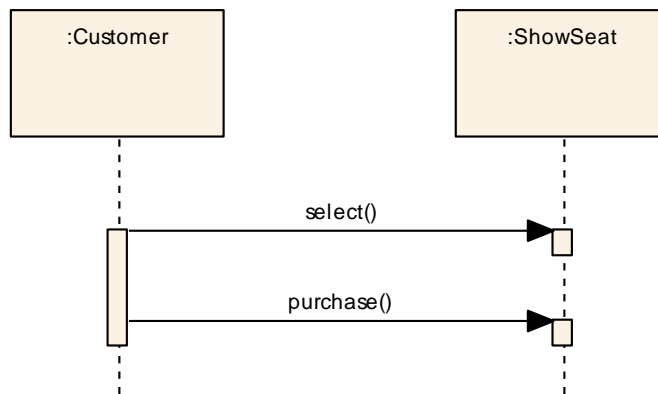
Diagrammi di stato e di sequenza

- ❑ Due scenari (sequence diagram): successo e fallimento di una transazione

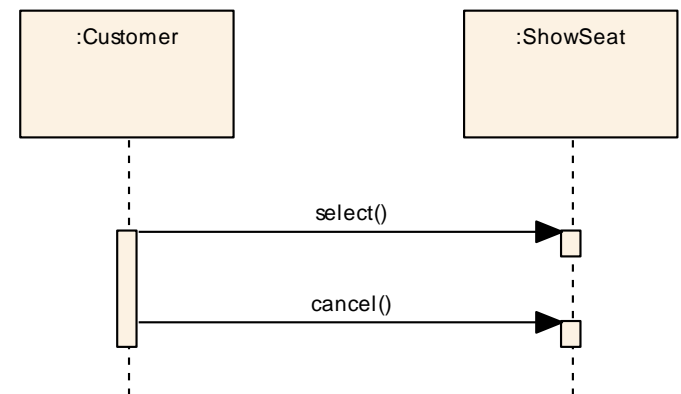
(Successo)

(Fallimento)

sd Dynamic model 3 - transaction success

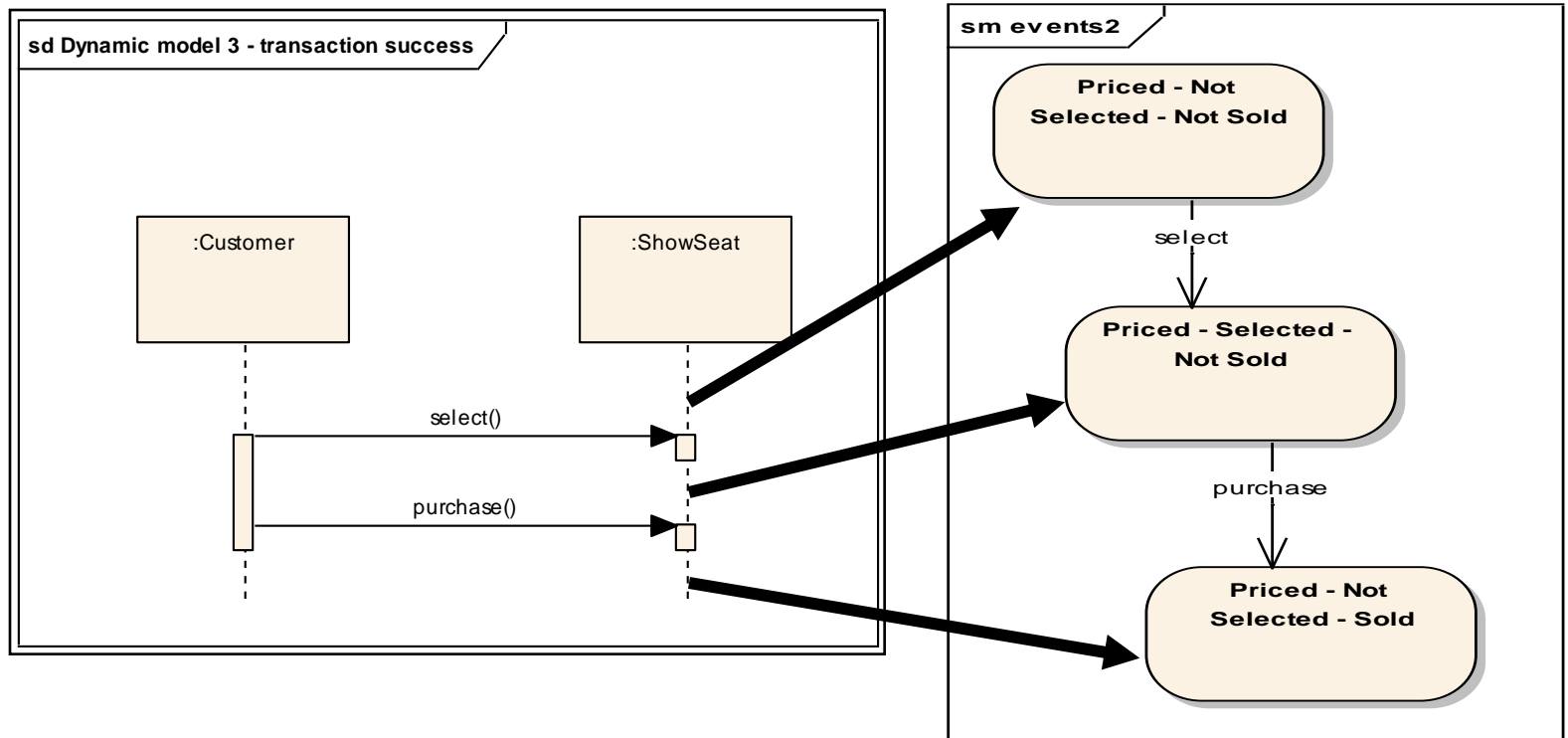


sd Dynamic model 3 - transaction failure



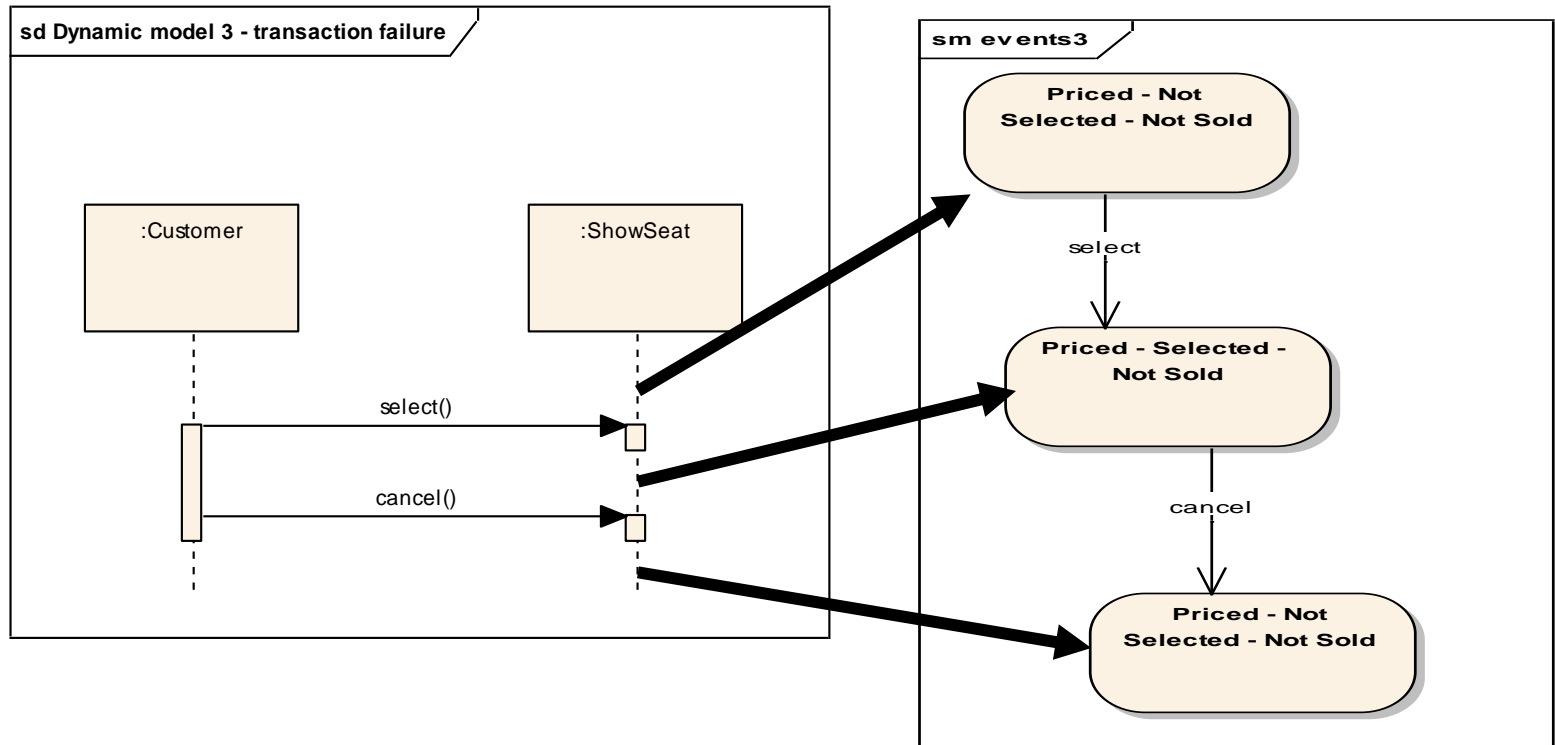
Diagrammi di stato e di sequenza

- Scenario di successo e relativo (parziale) diagramma a stati
(Successo) (diagramma a stati parziale)

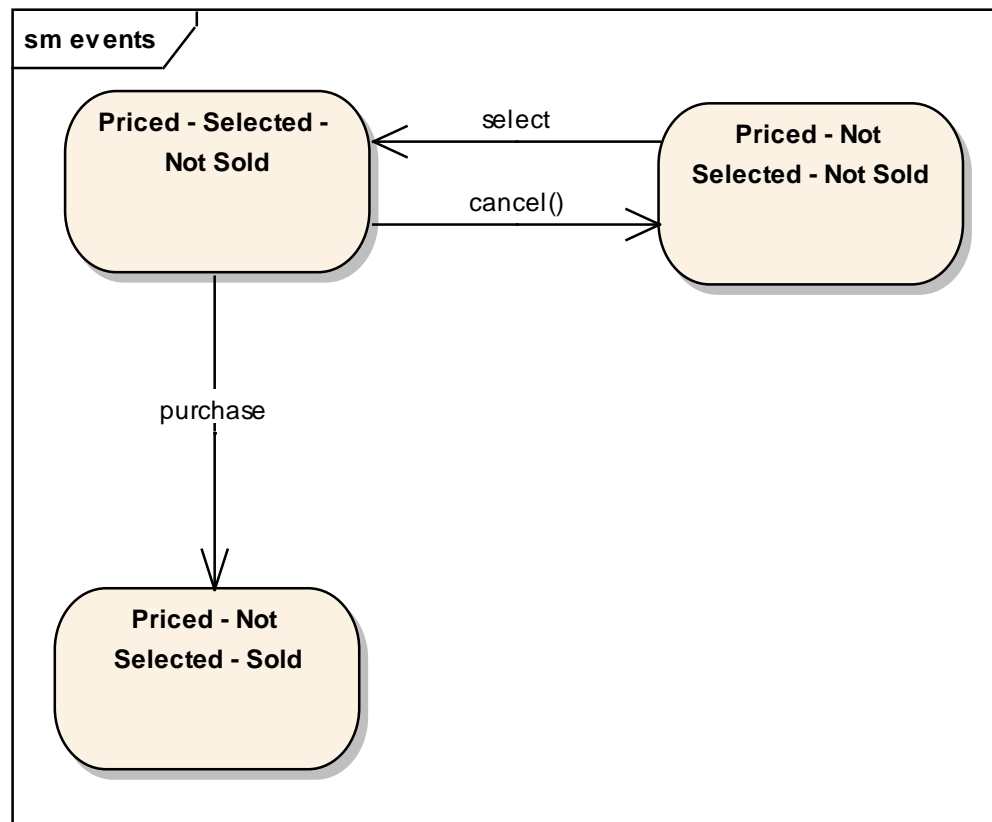


Diagrammi di stato e di sequenza

- Scenario di fallimento e relativo (parziale) diagramma a stati
(Fallimento) (diagramma a stati parziale)

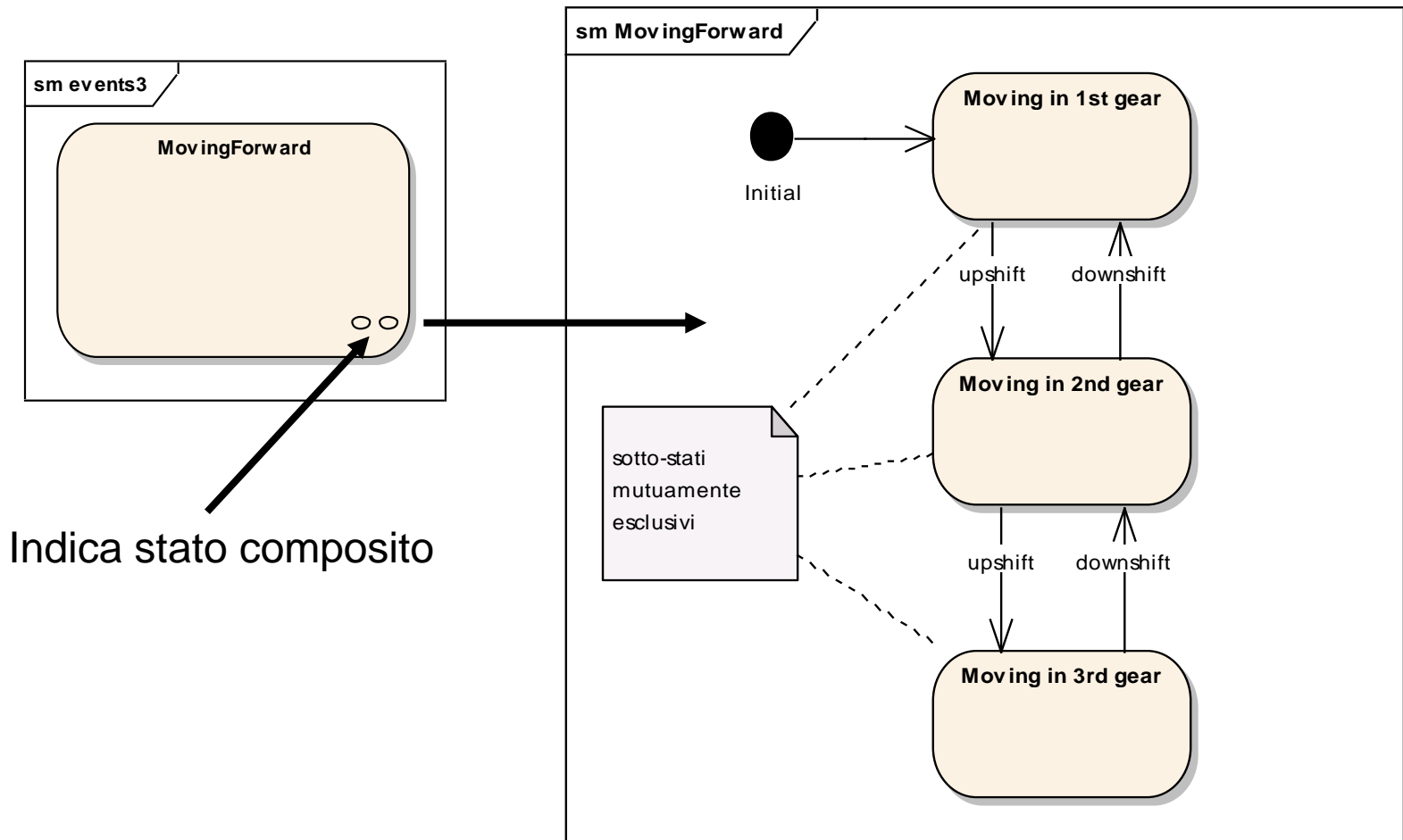


- Il diagramma a stati completo (relativo ai due scenari discussi)



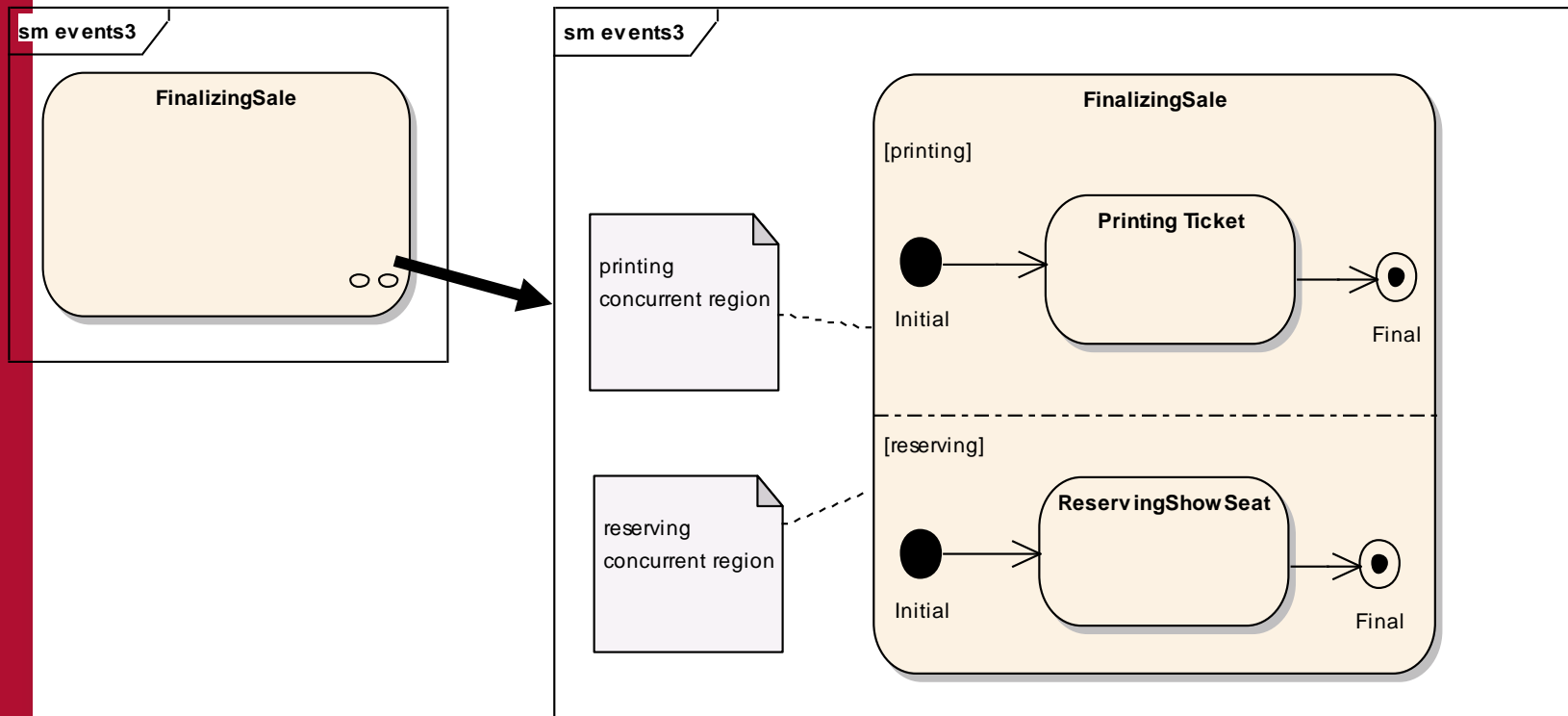
Diagrammi di stato composti

- Uno stato può contenere al suo interno più **sottostati mutuamente esclusivi**



Diagrammi di stato composti

- ❑ Uno stato può contenere al suo interno **sottostati concorrenti**



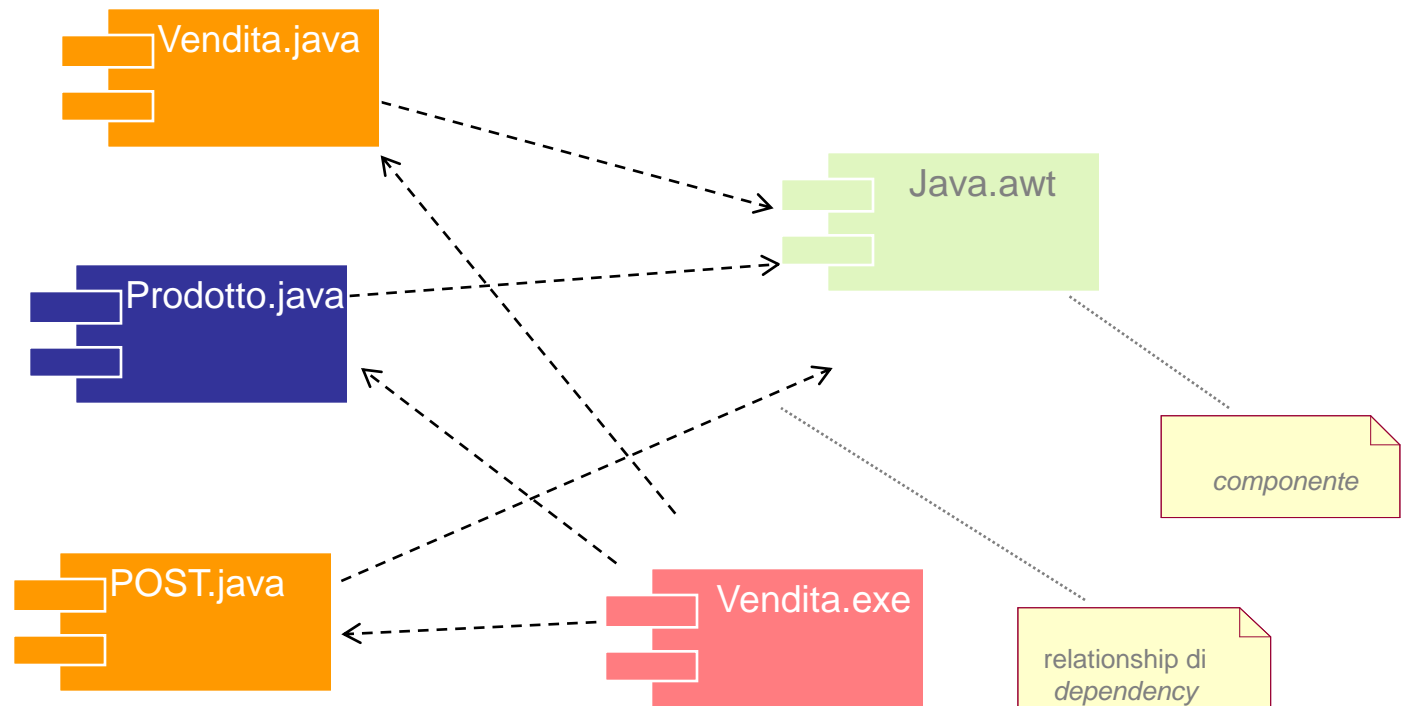


Università
Ca' Foscari
Venezia

Diagramma dei componenti

- evidenzia l'organizzazione e le dipendenze tra i componenti software
- i componenti (come a livello logico i casi d'uso o le classi) possono essere raggruppati in package
 - un **componente** è una qualunque porzione fisica riutilizzabile con un'identità e un'interfaccia (dichiarazione di servizi offerti) ben definite
 - un componente può essere costituito dall'aggregazione di altri componenti

Diagramma dei componenti



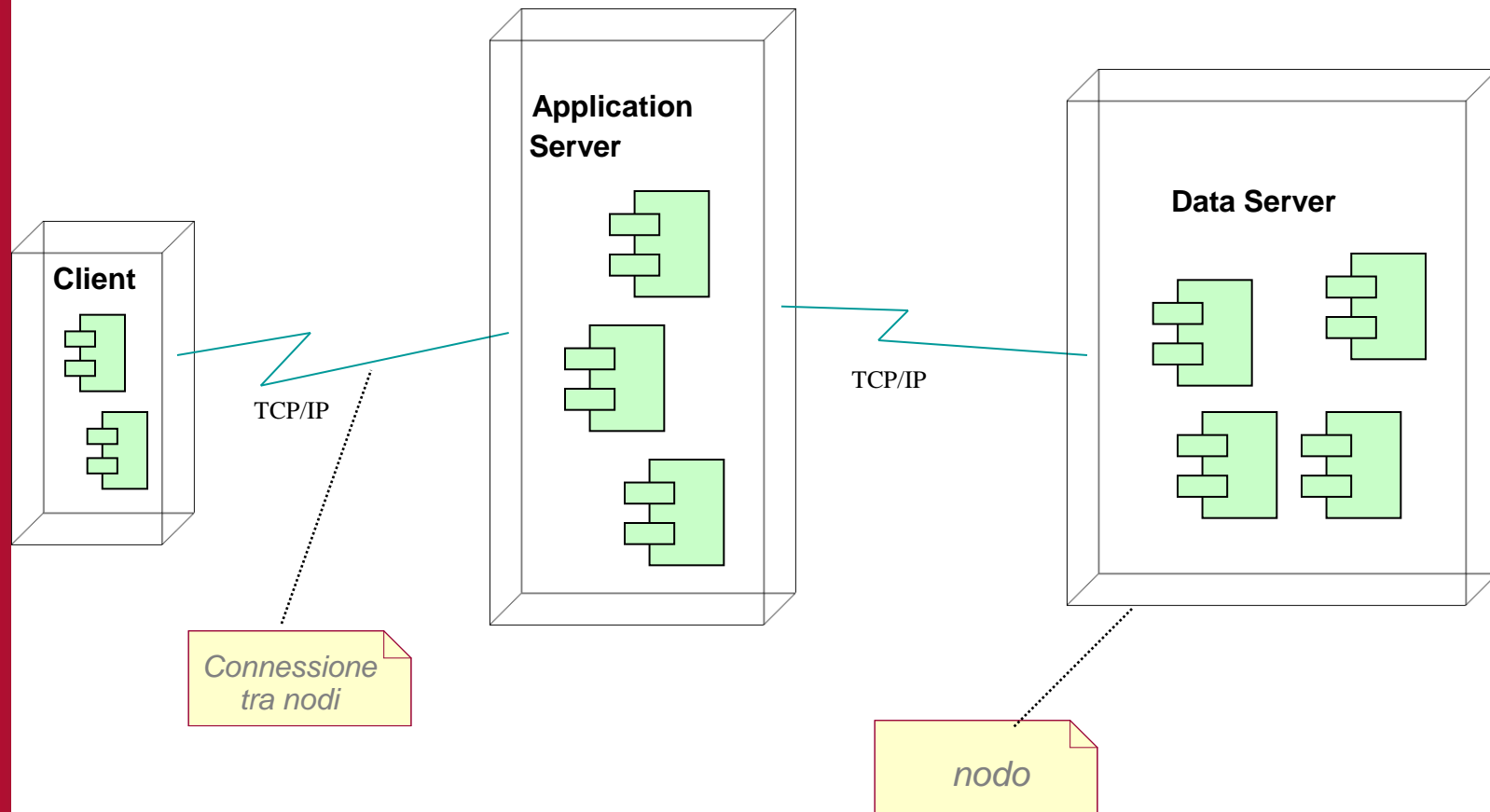


Università
Ca' Foscari
Venezia

Diagramma di distribuzione

- è utilizzato per mostrare come sono configurate e allocate le unità hardware e software per un'applicazione
- evidenzia la configurazione dei nodi elaborativi in ambiente di esecuzione (run-time), e dei componenti, processi ed oggetti allocati su questi nodi

Diagramma di distribuzione





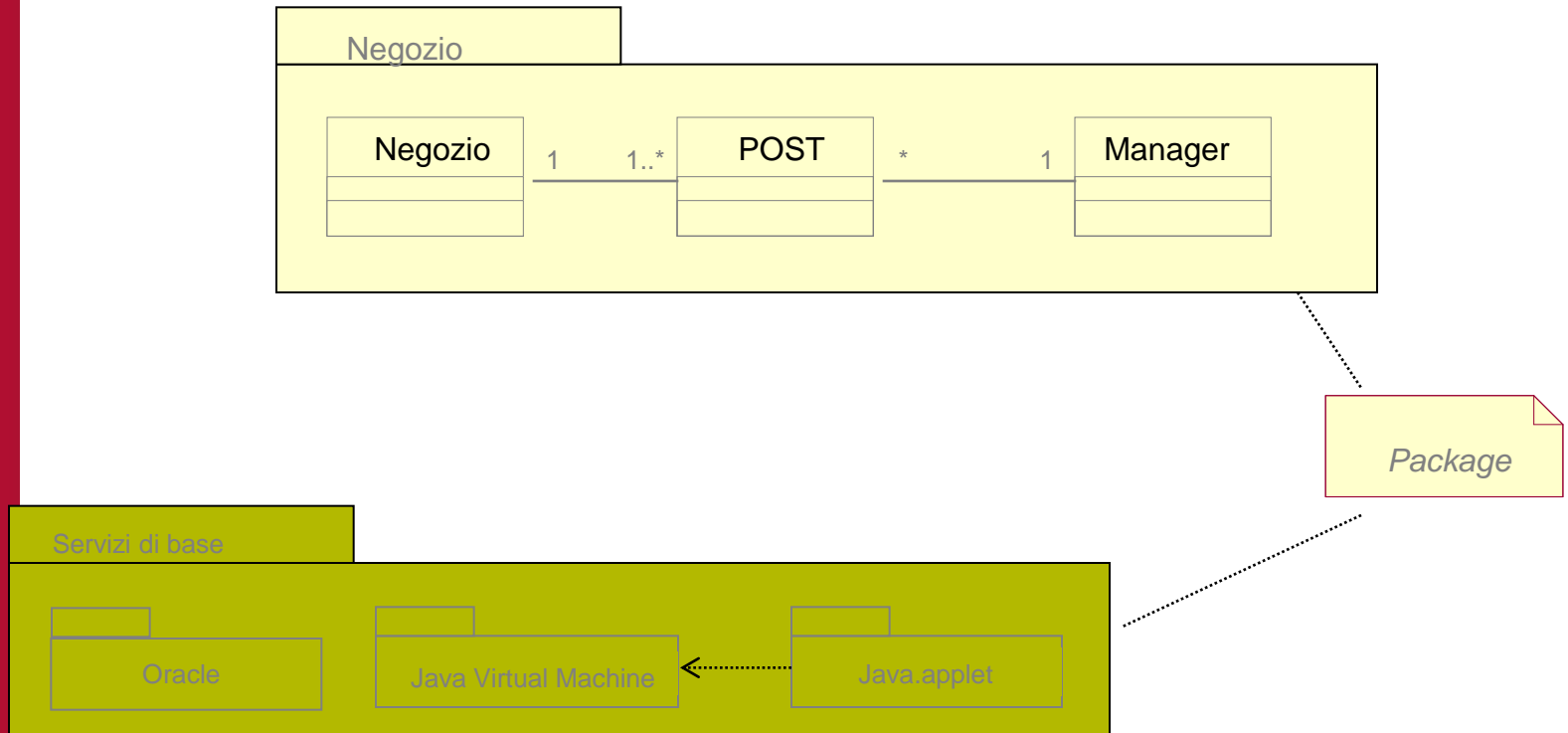
Package

- consente di partizionare il sistema in sottosistemi costituiti da elementi omogenei di:
 - natura **logica** (classi, casi d'uso, ...)
 - natura **fisica** (moduli, tabelle, ...)
 - altra natura (processori, risorse di rete, ...)
- ogni elemento appartiene ad un solo package
- un package può referenziare elementi appartenenti ad altri package



Università
Ca' Foscari
Venezia

Package





Università
Ca' Foscari
Venezia

UML - considerazioni finali

UML è sufficientemente complesso per rispondere a tutte le necessità di modellazione, ma è opportuno “ritagliarlo” in base alle specifiche esigenze dei progettisti e dei progetti, utilizzando solo ciò che serve nello specifico contesto

“keep the process as simple as possible!”