



# Modellare il comportamento



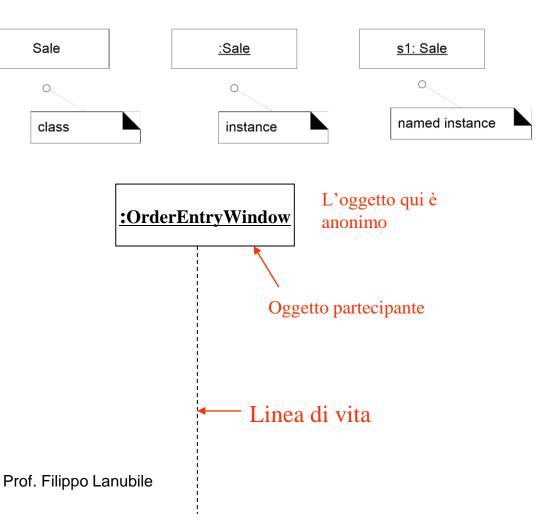
# Diagramma di sequenza

- Mostra l'interazione tra oggetti con enfasi sulla sequenza dei messaggi
- Documenta, dal punto di vista del software, il comportamento di un gruppo di oggetti che realizzano un singolo scenario di utilizzo del sistema

# Partecipanti in un diagramma di sequenza



- Partecipante
  - Oggetto
    rappresentato come
    un rettangolo in cima
    a una linea
    tratteggiata
- Linea di vita
  - Linea tratteggiata
  - Rappresenta lo scorrere del tempo



# Messaggi in un diagramma di

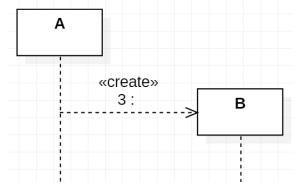


#### sequenza

:ProductOrder

:StockItem

- Messaggi
  - Comunicazione tra oggetti
  - Realizzate come chiamate di metodi
- Tipi speciali di messaggi
  - Chiamata interna (all'oggetto "this")
  - Ritorno da chiamata (si mostra solo se aggiunge informazione)
  - Creazione di un oggetto <<create>>



Distruzione esplicita di un oggetto <<destroy>>

Messaggio check() needsToReorder() Chiamata interna Ritorno (opzionale) instance1: Object1 instance2: Object2 synchronousMessage()

asynchronousMessage()

Prof. Filippo Lanubile

# Barre di attivazione in un diagramma

:Order



:StockItem

check()

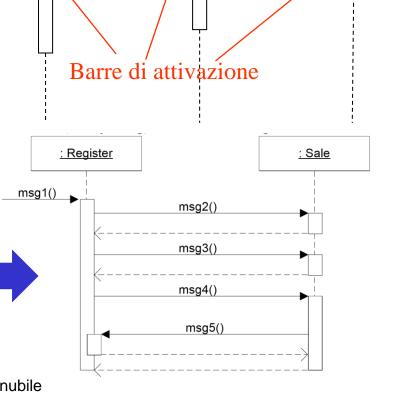
[check="true"]

remove()

### di sequenza

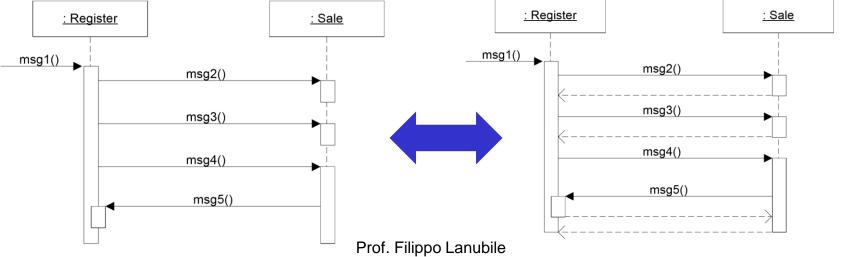
#### Barra di attivazione

- Rettangolo sottile posto sulla linea di vita di un oggetto
- Indica quando il partecipante è attivo nell'interazione
- Rende ridondante l'uso di linee di restituzione del controllo (return line)



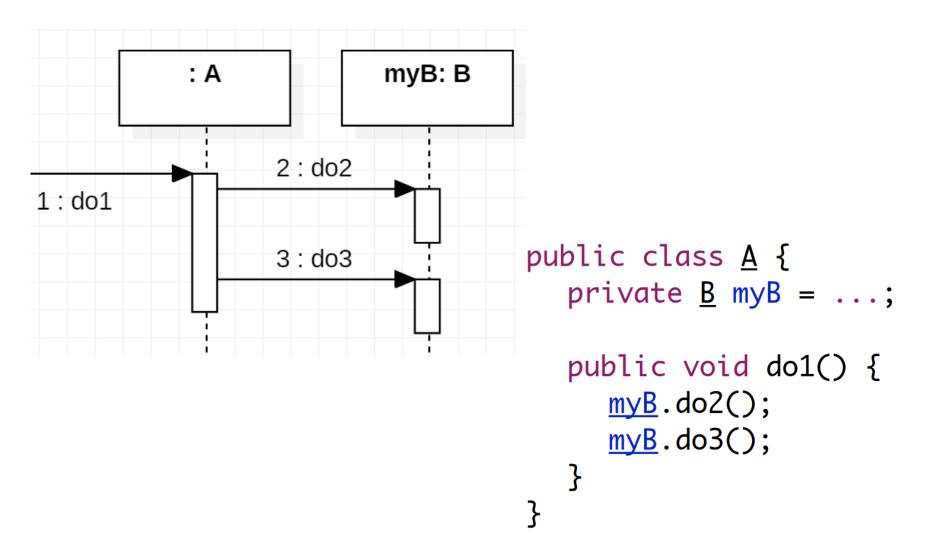
:ProductOrder

\*prepare()



# 





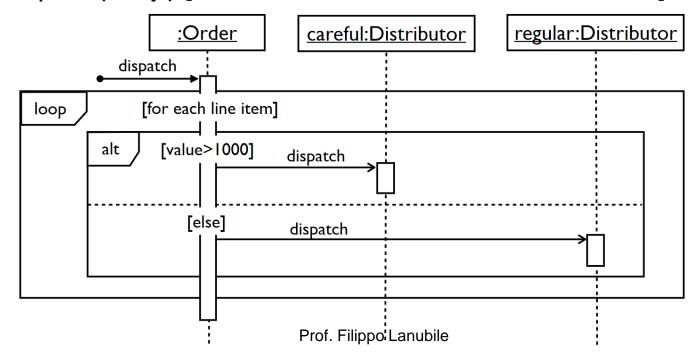
Esempio di diagramma di sequenza che realizza la user story : Schedule : Student : Course Catalog RegisterForCoursesForm RegistrationController CourseCatalogSystem : Student 1. // create schedule() 1.1. // get course offerings() Student wishes to 1.1.1. // get course offerings(forSemester) create a new schedule 1.1.1.1. // get course offerings() 1.2. // display course offerings() A list of the available course offerings for this semester are displayed A blank schedule 1.3. // display blank schedule() is displayed for the students to select offerings 2. // select 4 primary and 2 alternate offerings() 2.1. W create schedule with offerings() 2.1.1. // create with offerings() 2.1.2. // add schedule(Schedule) At this, point the Submit Schedule subflow is executed. Prof. Filippo Lanubile

# Strutture di controllo in un diagramma di sequenza: **UML 2**



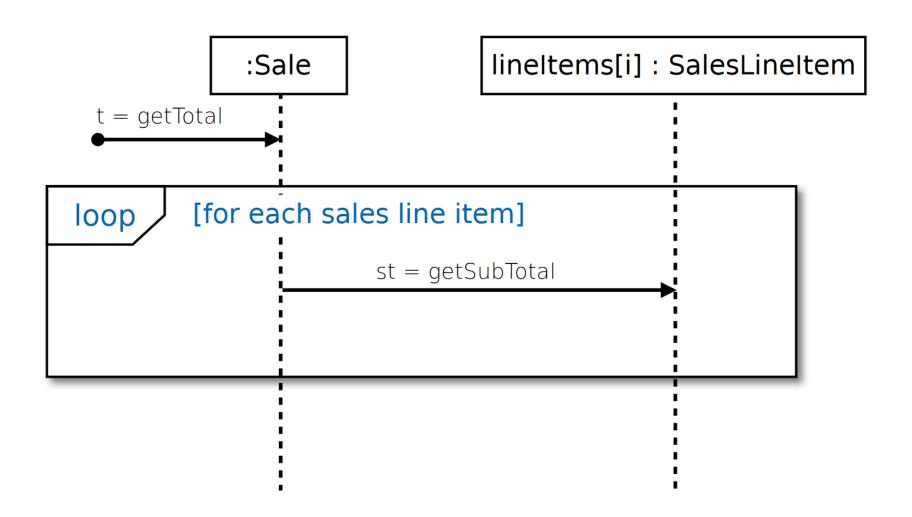
Frame di interazione: una cornice intorno a una parte del diagramma di sequenza

- if  $\rightarrow$  (**opt**) [condition]
- if/else → (alt) [condition], separati da una linea tratteggiata orizzontale
- loop → (loop) [condizione o elementi su cui iterare]



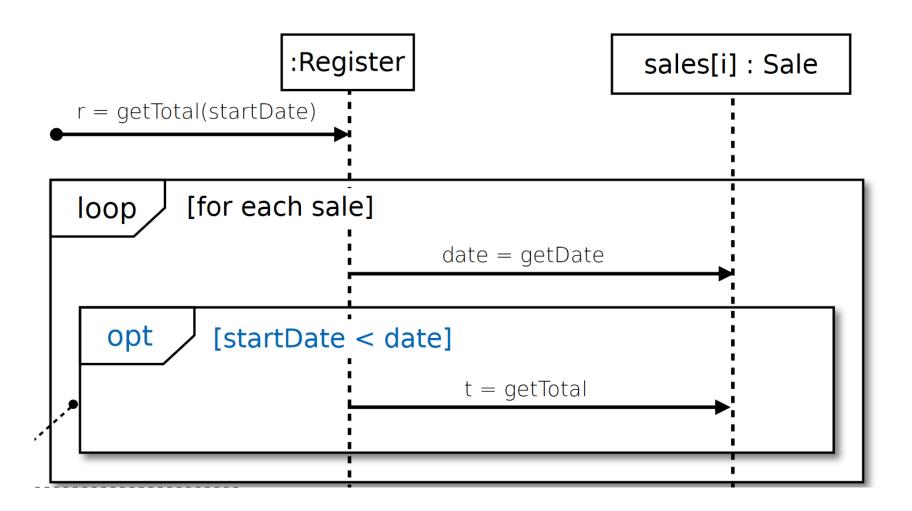


### Esempio loop



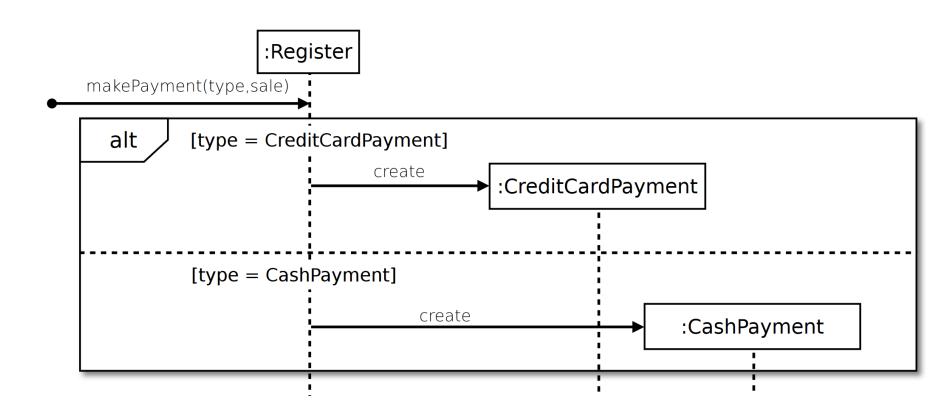


## Esempio opt





# Esempio alt







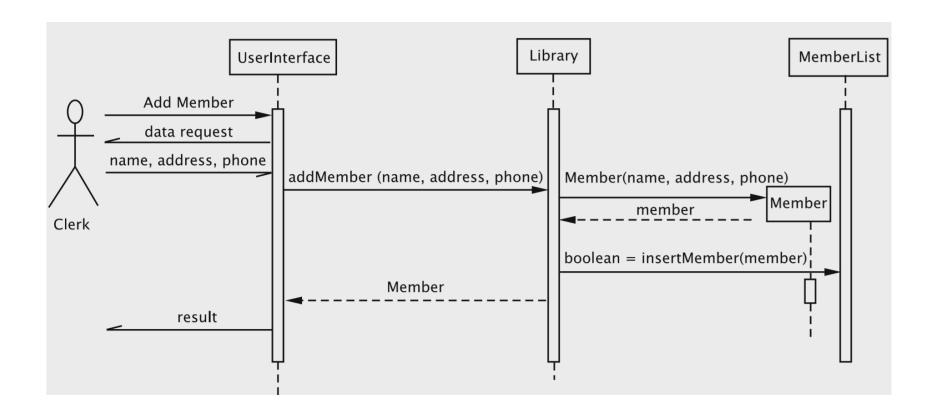
#### User stories

- Adding a new member
- Adding books
- Issuing books
- Returning books
- Removing books
- Placing a hold

Tratto da *Object-Oriented Analysis, Design and Implementation*, chapter 7 Design and Implementation

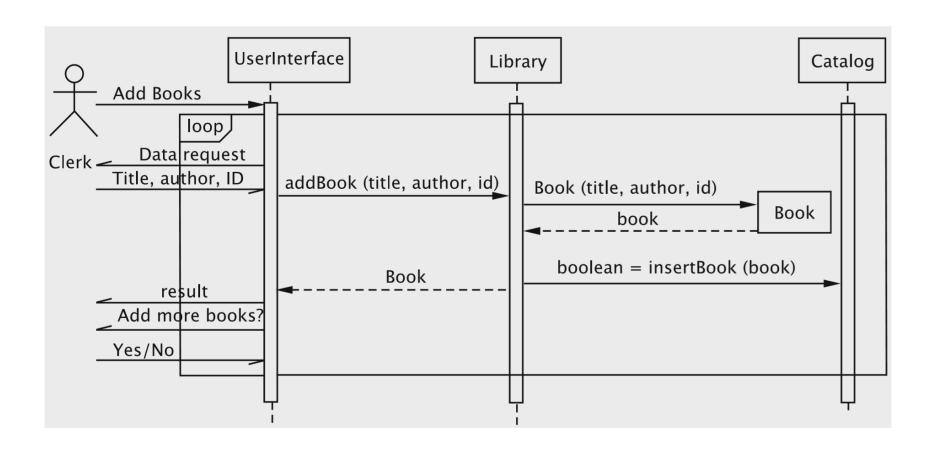


# Adding a new member



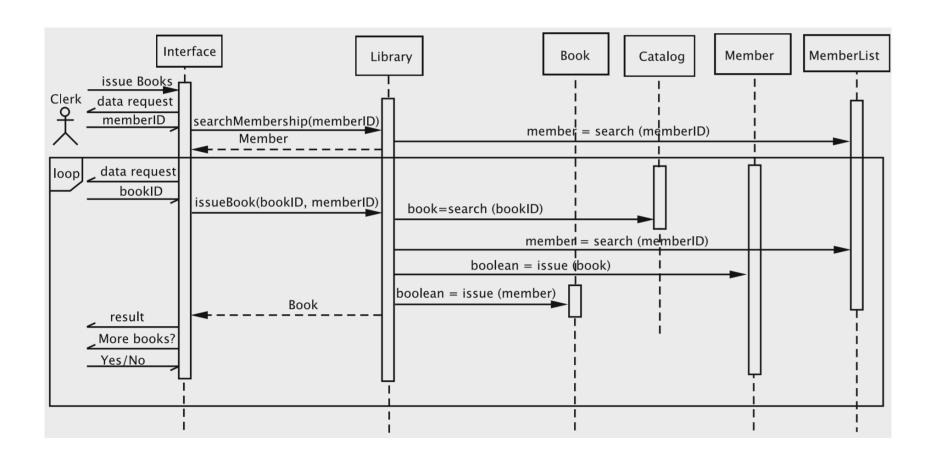


# Adding books



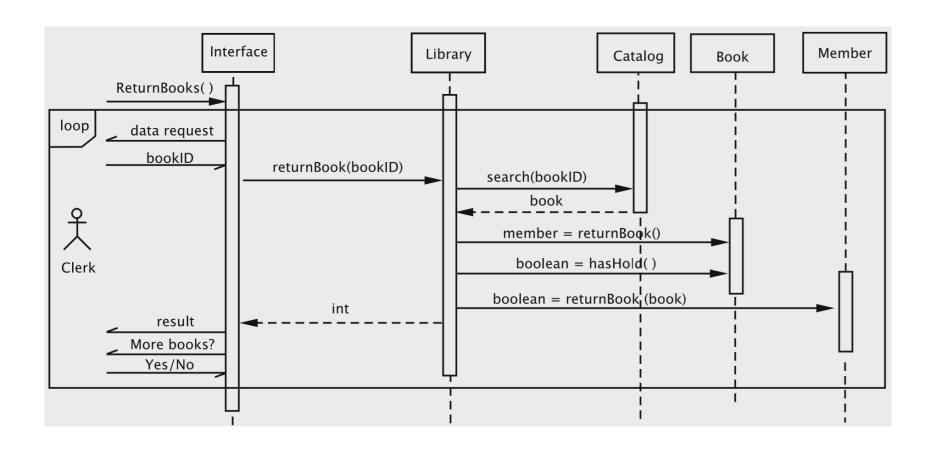


# Issuing books



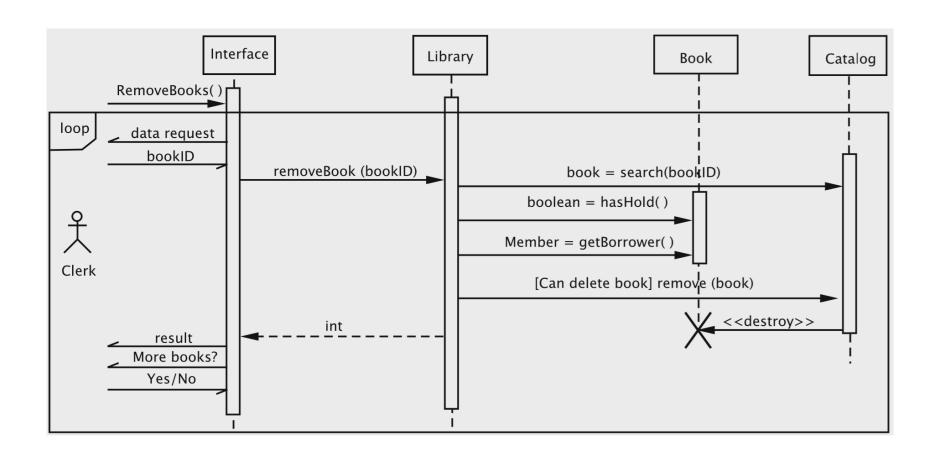


# Returning books





# Removing books





# Placing a hold

