

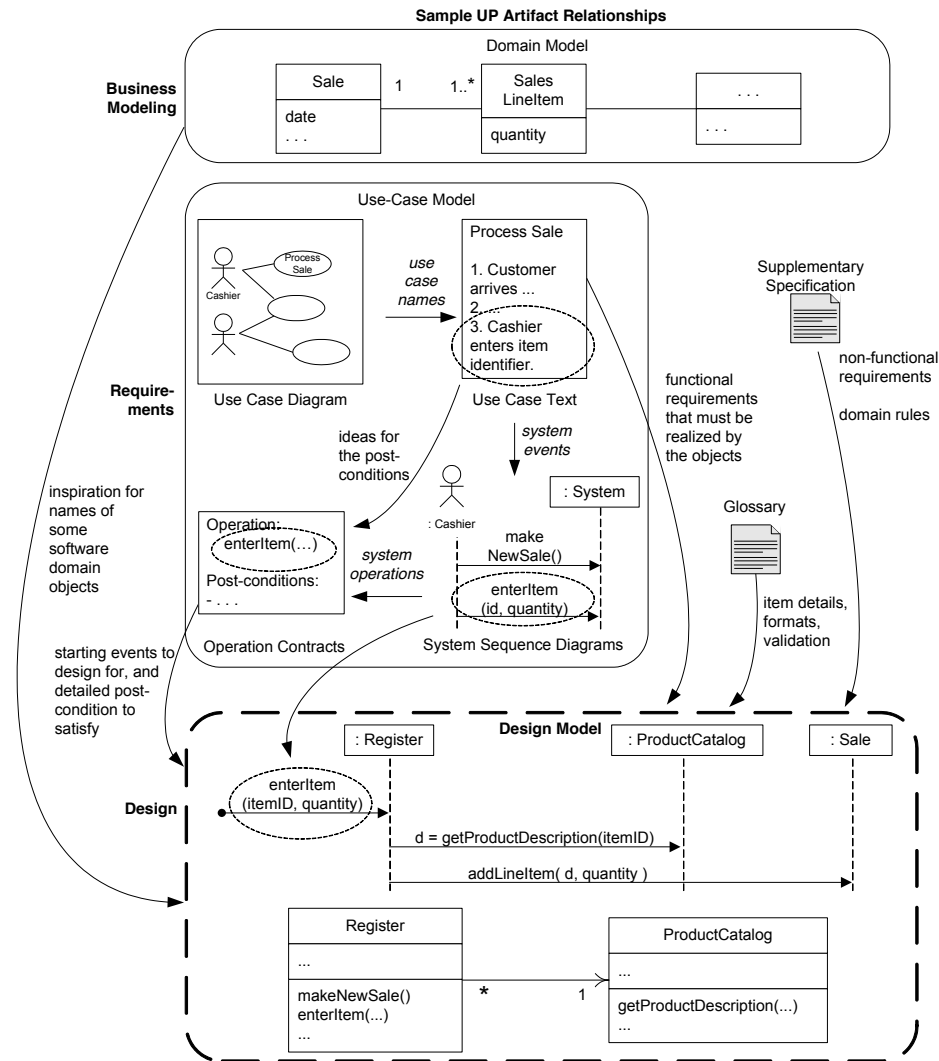
# **Svilippo delle applicazioni software**

V. Bono

# Capitolo 18

Esempi di progettazione a oggetti  
con i pattern GRASP (case study  
POS NextGen)

# Fig. 18.1



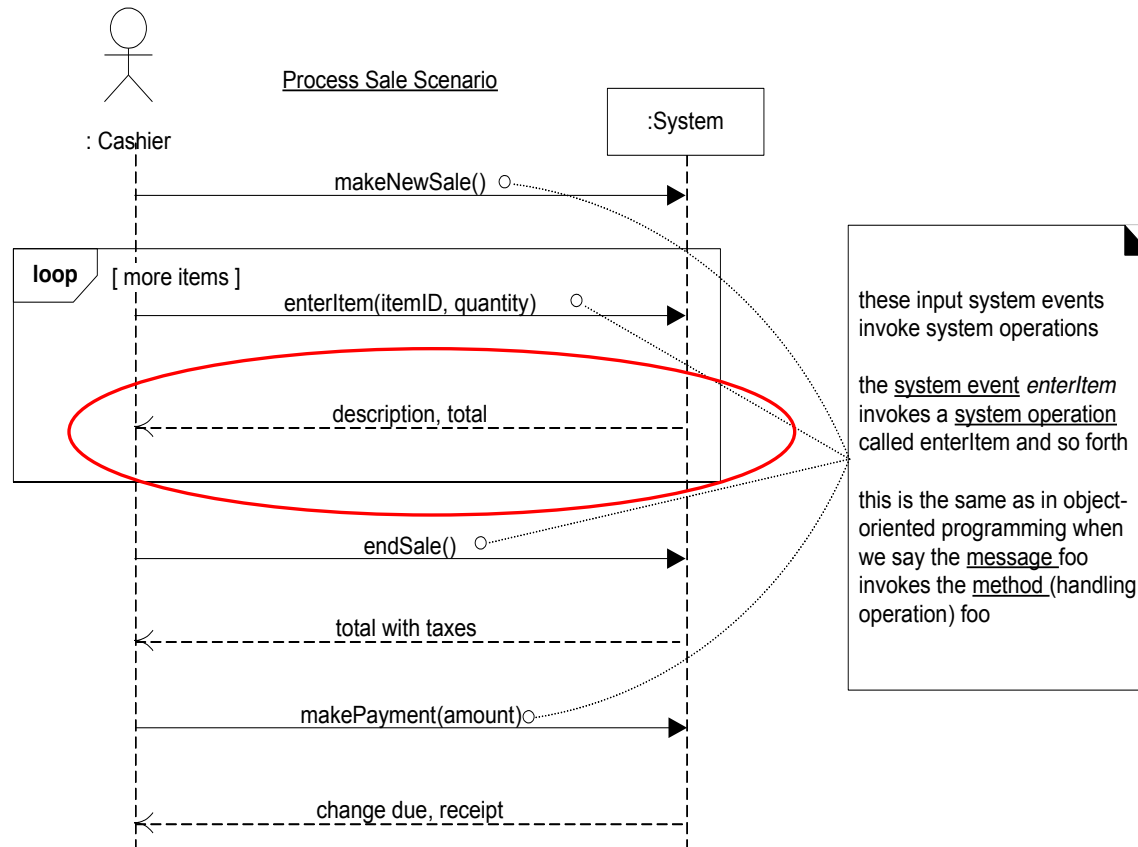
## Realizzazione di caso d'uso

- Si va verso il modello di progetto, partendo dal modello di dominio
- Caso d'uso suggerisce le *operazioni di sistema* date dai System Sequence Diagram
- Le *operazioni di sistema* diventano i messaggi iniziali sui Controller negli interaction diagram per lo strato del dominio (dallo strato della GUI)
- I diagrammi di interazione dello strato di dominio illustrano come gli oggetti interagiscono, per soddisfare i compiti richiesti, ovvero sono la realizzazione dei casi d'uso

## POS NextGen

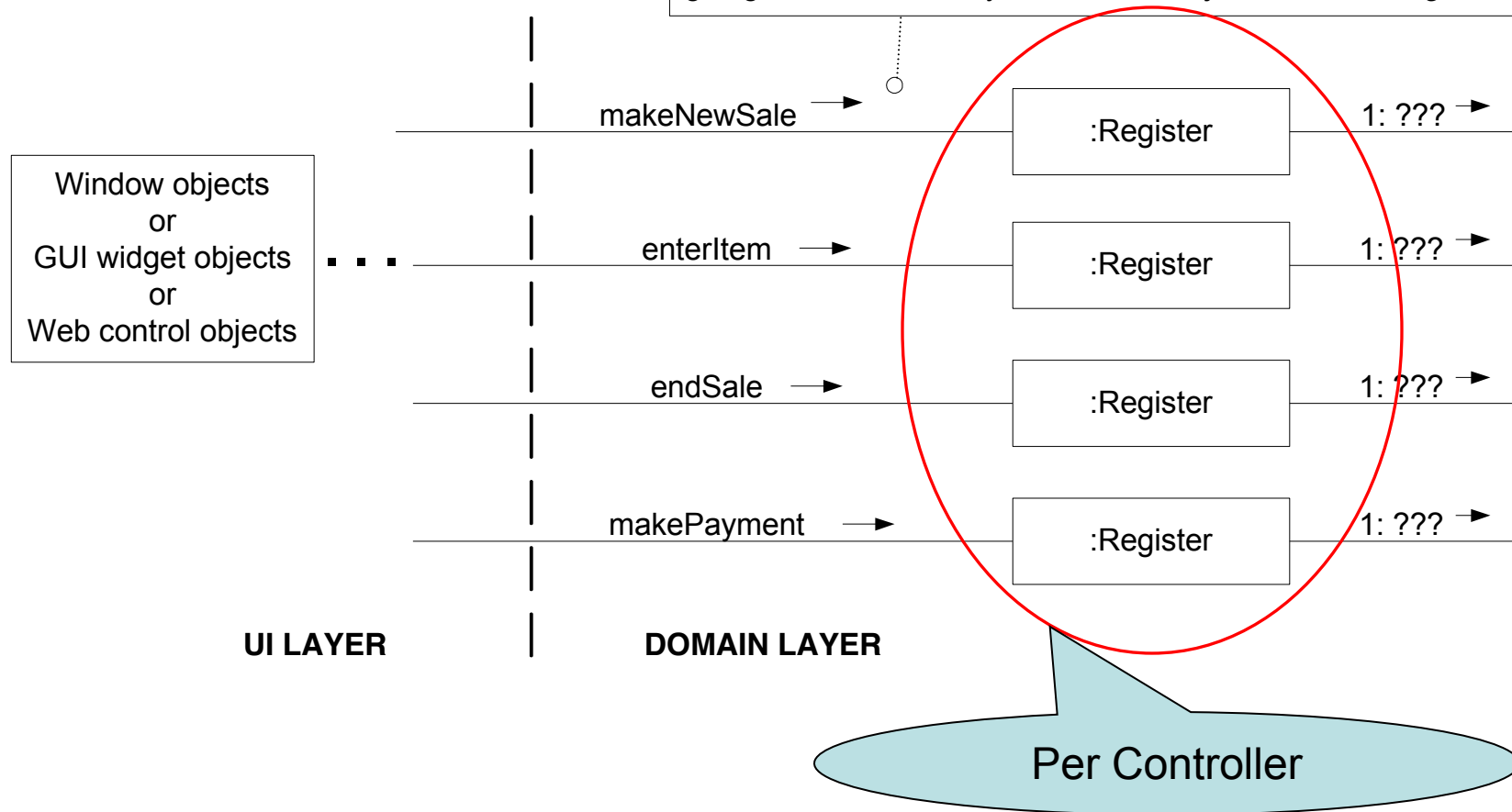
- Nella prima iterazione si prendono in considerazione gli scenari e le operazioni di sistema degli SSD del caso d'uso *Process Sale*
- Le operazioni sono:
  - *makeNewSale*
  - *enterItem*
  - *endSale*
  - *makePayment*

# Fig. 11.2

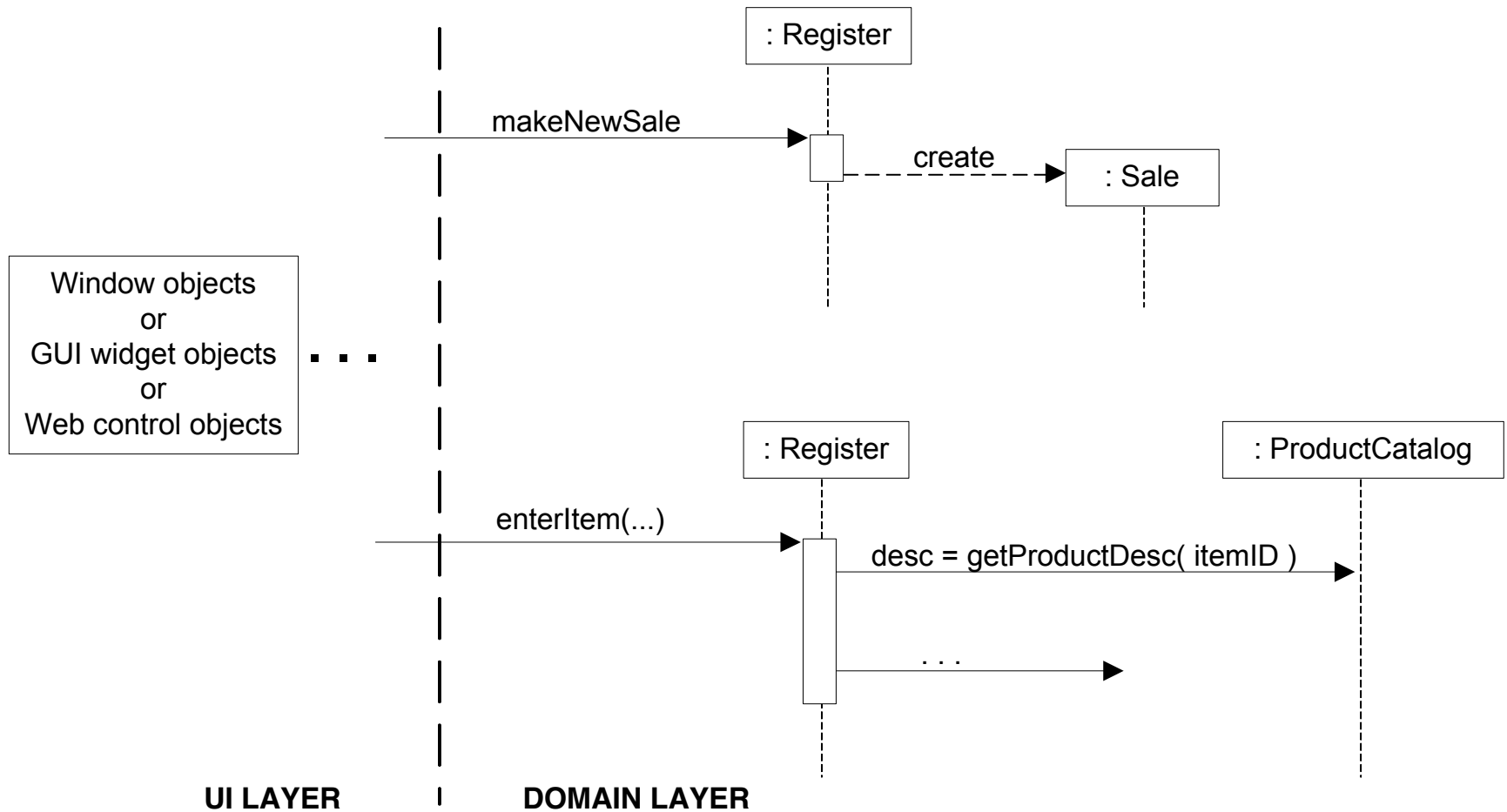


# Fig. 18.2: gestione delle comunicazioni di sistema (communication diagram)

*makeNewSale*, etc., are the system operations from the SSD  
each major interaction diagram starts with a system operation going into a domain layer controller object, such as *Register*



**Fig. 18.3: gestione delle comunicazioni di sistema  
(sequence diagram)**





## Cosa si usa? (I)

- Tutto! Vedi Fig. 18.1
- Importante l'interazione costante con il cliente (principio fondamentale della programmazione *agile*)
- Per operazioni complesse, puo` essere necessario scrivere *contratti* con maggior dettaglio di analisi che quella contenuta nei testi di caso d'uso (spesso basta questa piu' conoscenza personale del dominio)...

## **Esempio di contratto (C2: enterItem)**

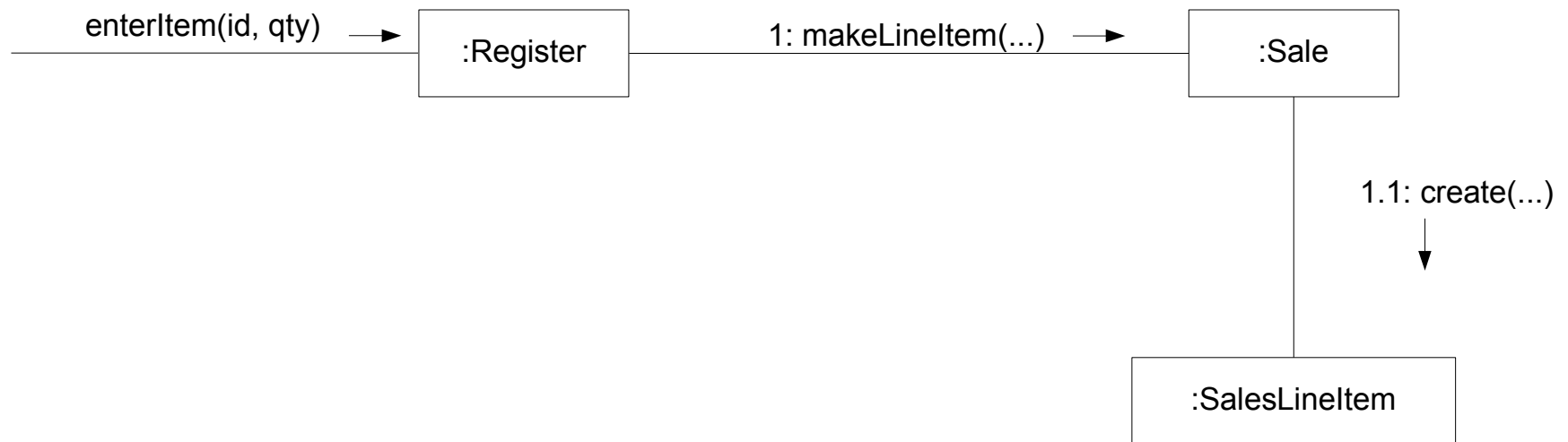
**Operazione:** enterItem(itemID:itemID, quantity:integer)

**Riferimenti:** casi d'uso: Process Sale

**Pre-condizioni:** e` in corso una vendita

**Post-condizioni:** - e` stata creata un'istanza sli di  
SalesLineItem (creazione di istanza)  
- ...

**Fig. 18.4: diagramma parziale per C2 (es. di output)**



## Cosa si usa? (II)

- Il Modello di Dominio ispira alcuni oggetti software del Modello di Progetto
- Ma si possono scoprire nuovi concetti durante il passaggio al progetto...
- ... per cui si possono avere nuove classi concettuali,...
- ... ma anche classi software che *non* fanno parte del Modello di Dominio (classi introdotte dai pattern, ecc.)

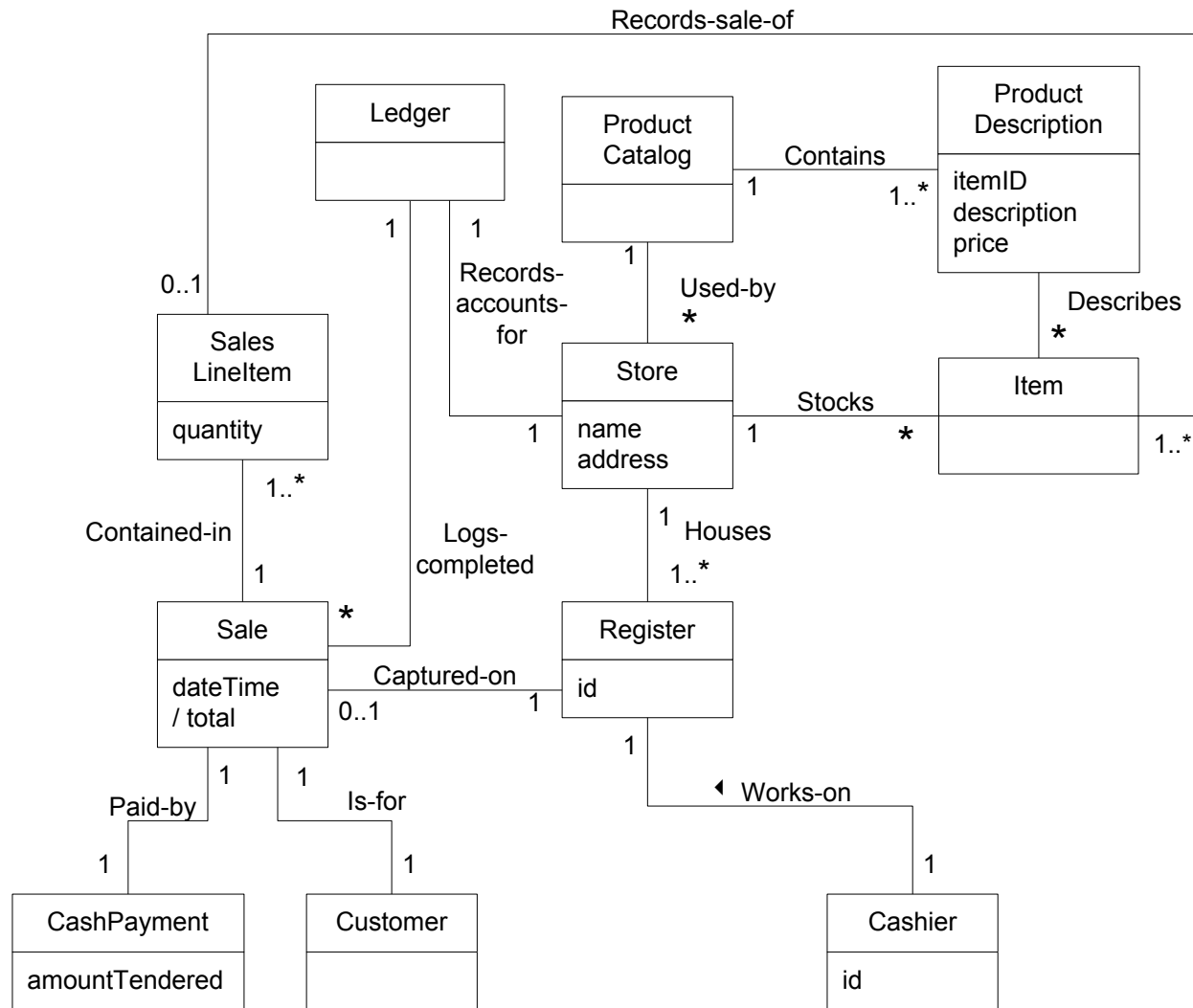
## Inizializzazione e caso d'uso d'avviamento

- *Start Up use case*
- Si creano gli oggetti root e quelli che “vivranno a lungo”
- A **livello di implementazione**, meglio partire con una parte delle inizializzazioni dello Start Up
- A **livello di progetto**, meglio farlo verso la fine, quando si sa cosa serve agli altri casi d'uso (fatto così per POS NextGen)

## POS NextGen

- Adoperiamo sistematicamente:
  - Il modello del dominio
  - La descrizione via contratti
  - I GRASP

**Fig. 9.27: modello di dominio parziale di POS Next Gen**



## CO1: makeNewSale

**Operazione:** makeNewSale()

**Riferimenti:** casi d'uso: Process Sale

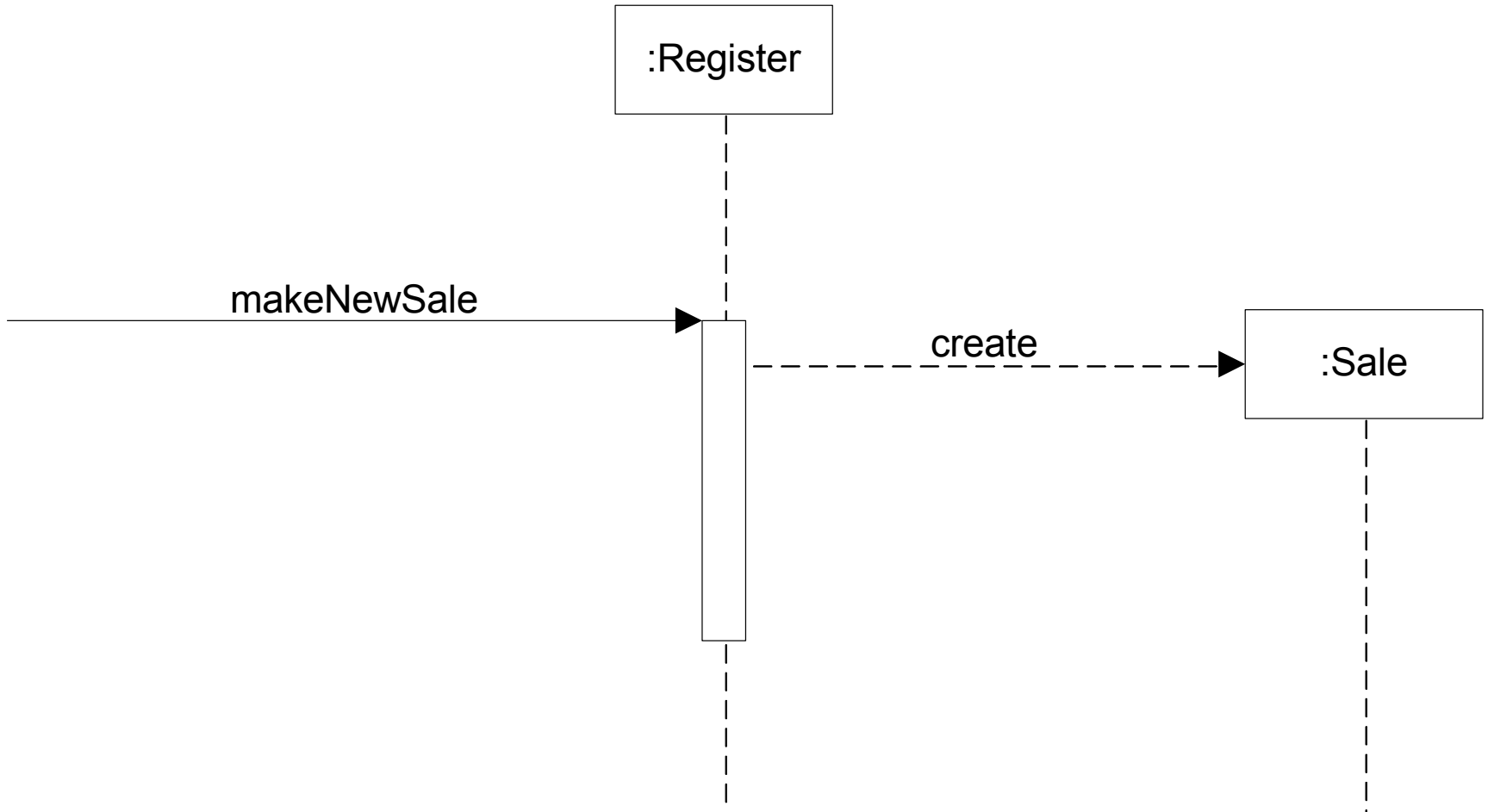
**Pre-condizioni:** ---

**Post-condizioni:**

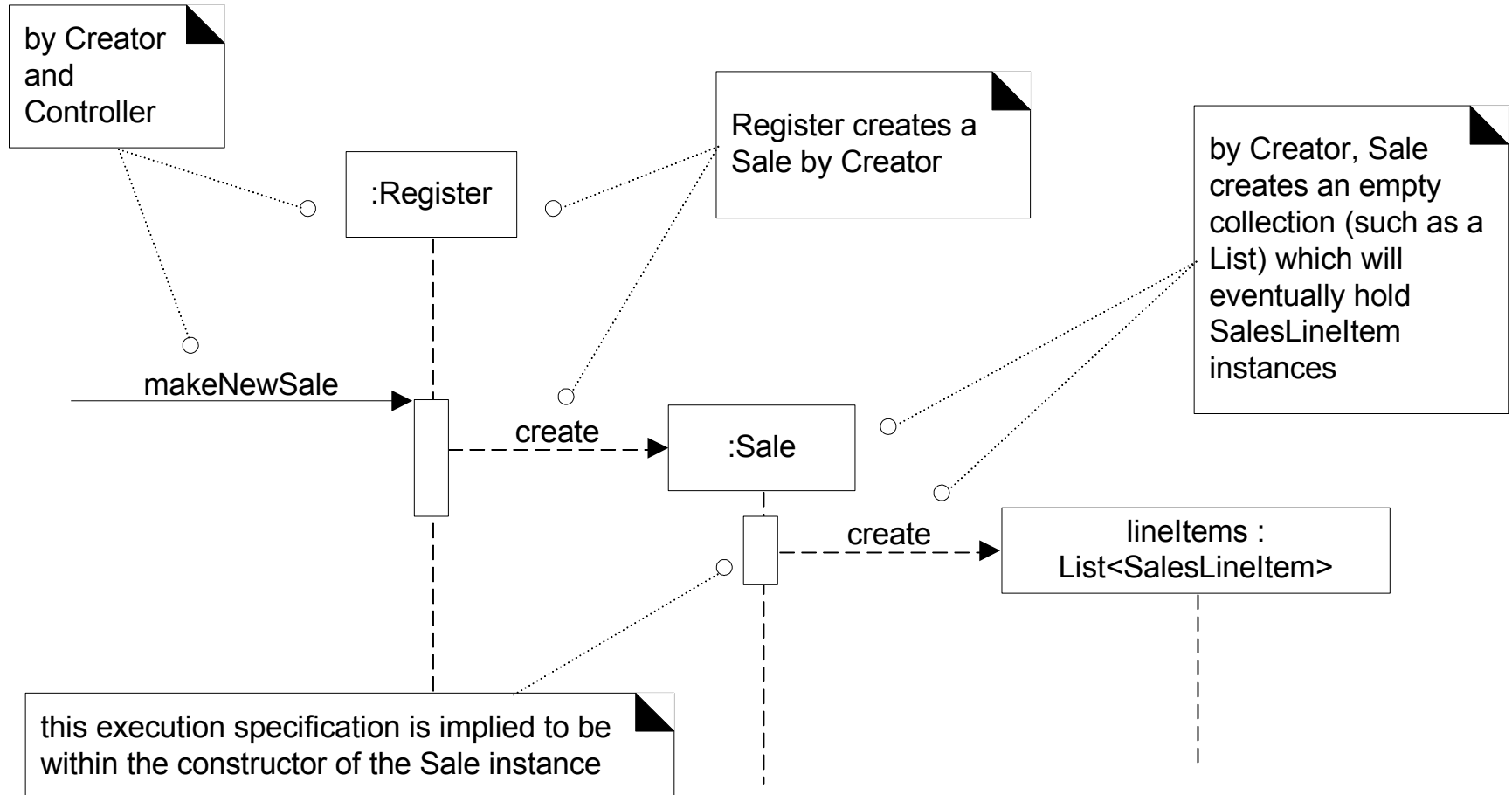
- e' stata creata un'istanza s di Sale (creazione di istanza)
- s e' stata associata con Register (associazione formata)
- gli attributi di s sono inizializzati



**Fig. 18.5: applicazione di GRASP Controller**



# Fig. 18.6: creazione di Sale e della collezione di item



## CO2: enterItem

**Operazione:** enterItem(itemID:itemID, quantity:integer)

**Riferimenti:** casi d'uso: Process Sale

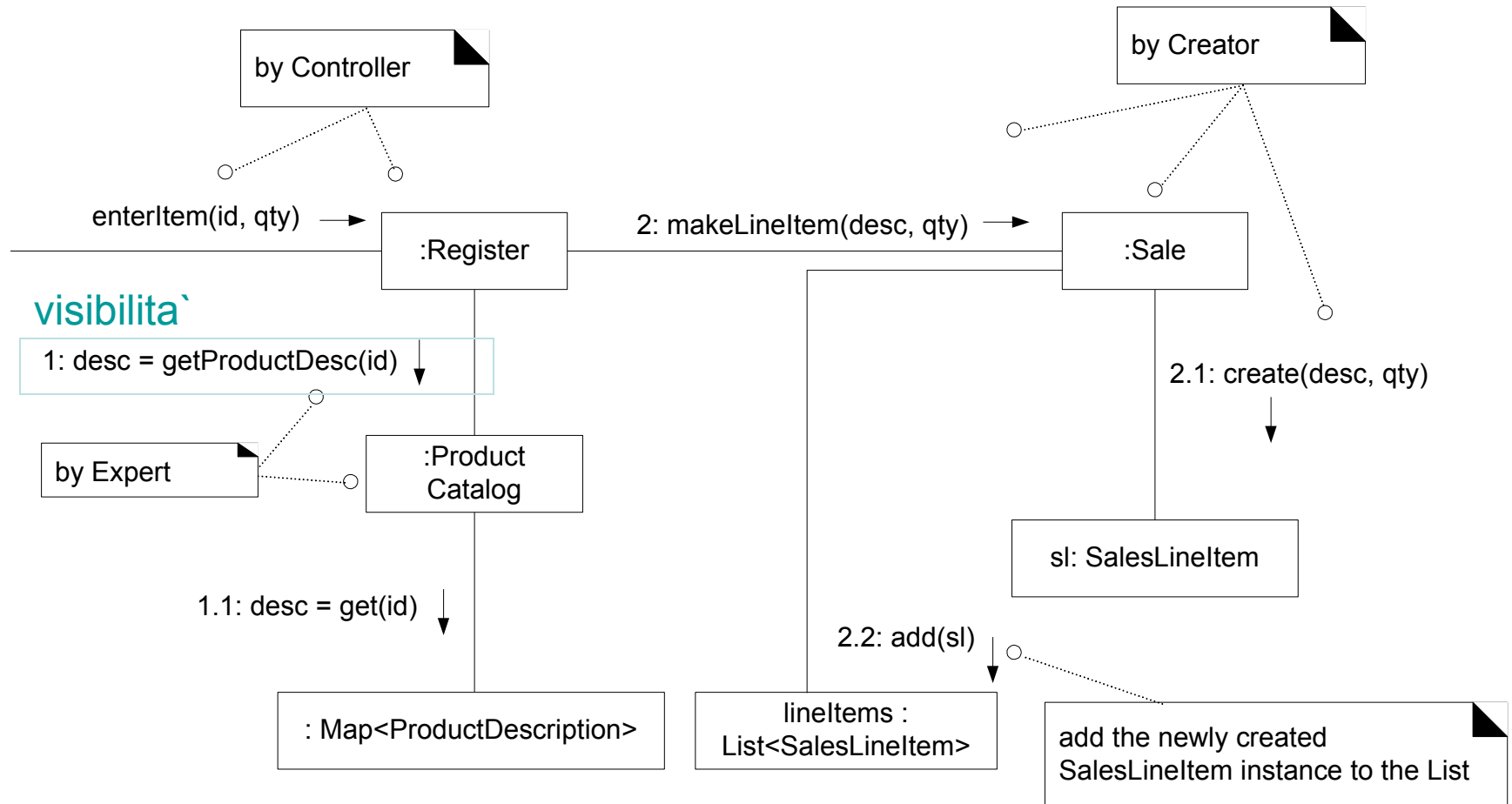
**Pre-condizioni:** e` in corso una vendita

**Post-condizioni:**

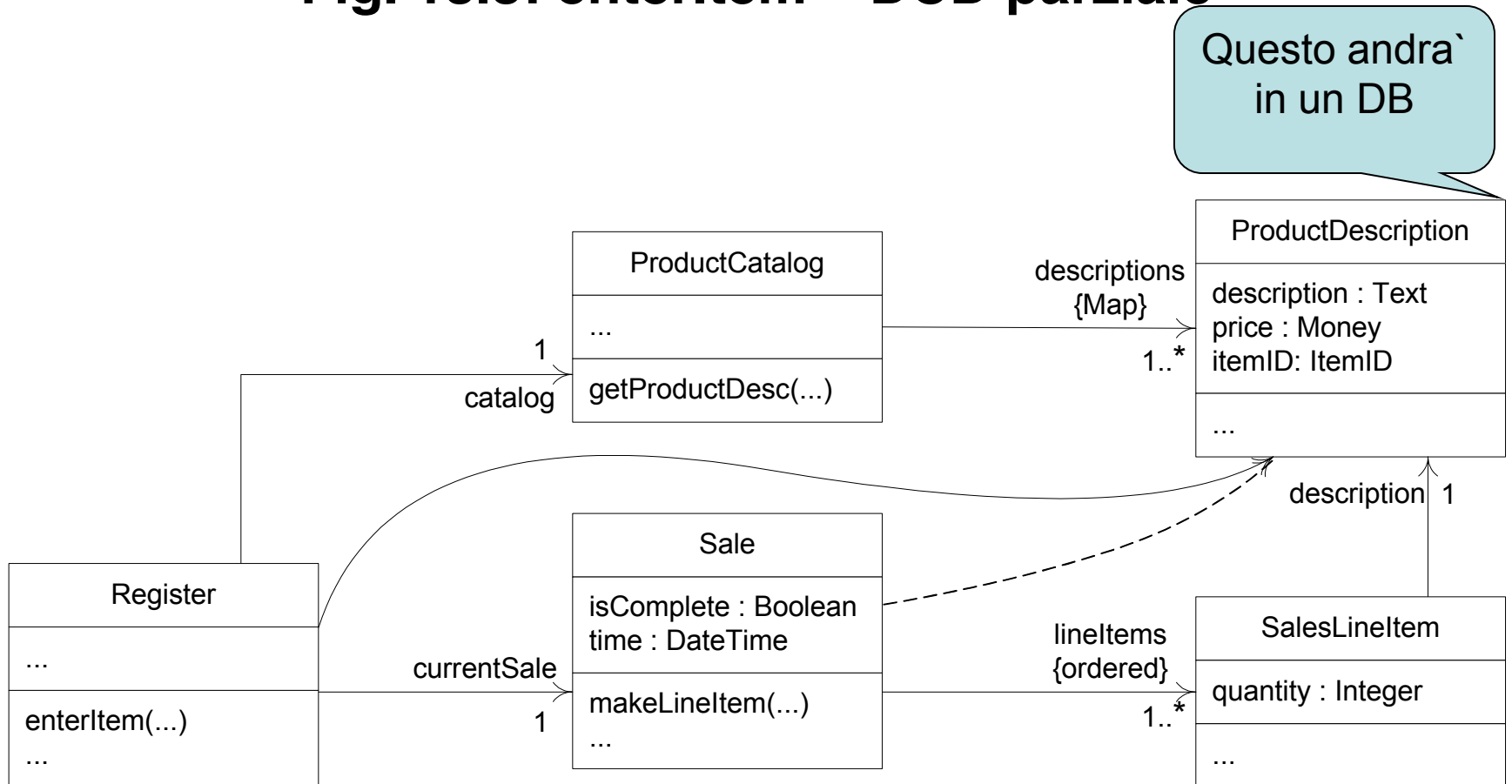
- e` stata creata un'istanza sli di SalesLineItem (creazione di istanza)
- sli e` stata associata con la Sale corrente (associazione formata)
- sli.quantity = quantity (modifica attributo)
- sli e` stata associata con una ProductDescription in base alla corrispondenza con itemID (associazione formata)

[non trattiamo qui la visualizzazione di descrizione e prezzo richiesta dal caso d'uso, ci basta che questi valori siano disponibili → **Principio di Separazione Modello-Vista**]

# Fig. 18.7: enterItem – vista dinamica



**Fig. 18.8: enterItem – DCD parziale**



## CO3: endSale

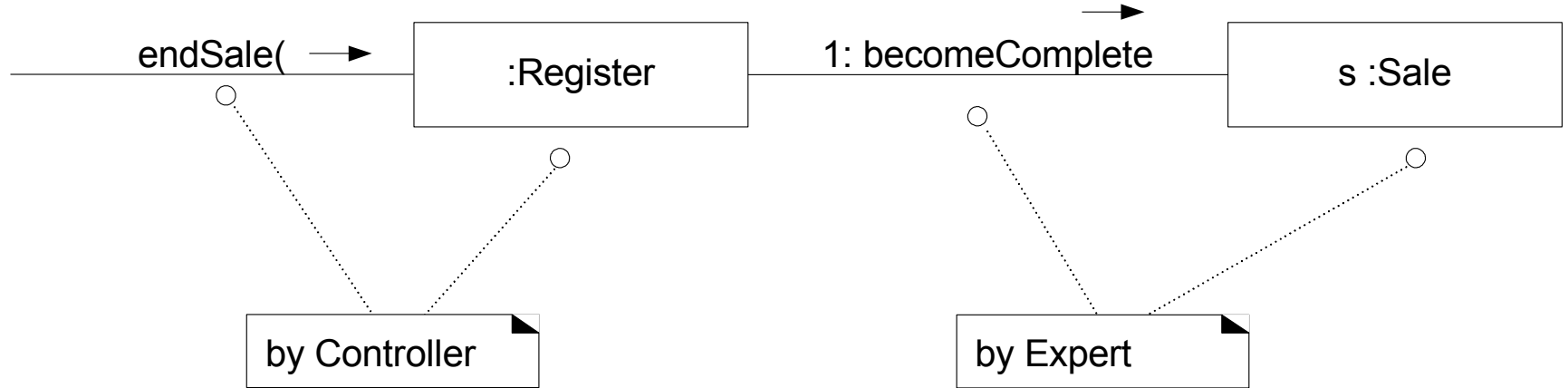
**Operazione:** endSale()

**Riferimenti:** casi d'uso: Process Sale

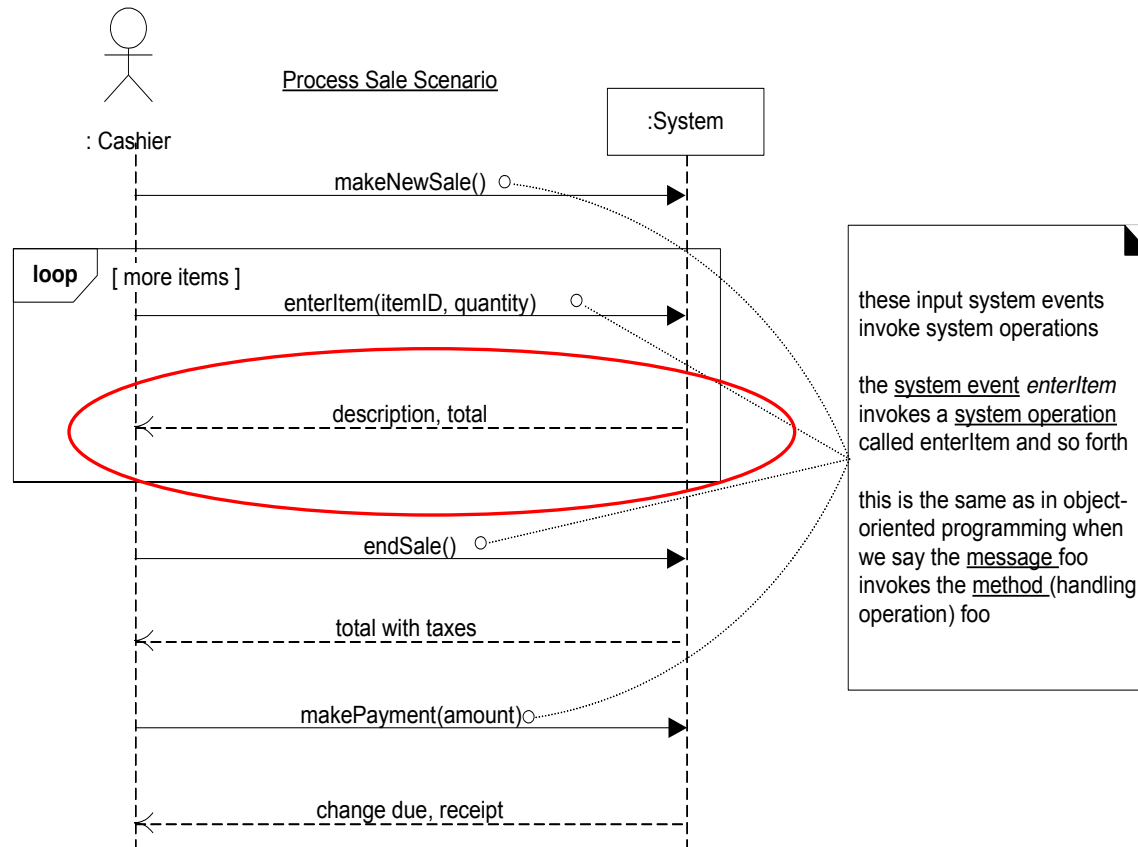
**Pre-condizioni:** e` in corso una vendita

**Post-condizioni:** Sale.isComplete = true (modifica di  
attributo)

**Fig. 18.9: mettere isComplete a true**



# Fig. 11.2





# Contratto CO2: enteritem

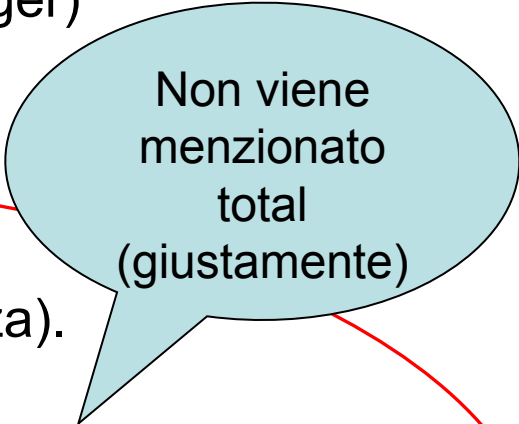
**Operazione:** enteritem(itemID:ItemID, quantity: integer)

**Riferimenti:** casi d'uso: Process Sale

**Pre-condizioni:** è in corso una vendita.

**Post-condizioni:**

- è stata creata un'istanza sli di SalesLineItem (creazione di istanza).
- sli è stata associata con la Sale (vendita) corrente (associazione formata).
- sli.quantity è diventata quantity (modifica di attributo).
- sli è stata associata con una ProductDescription, in base alla corrispondenza con itemID (associazione formata).



Non viene  
menzionato  
total  
(giustamente)

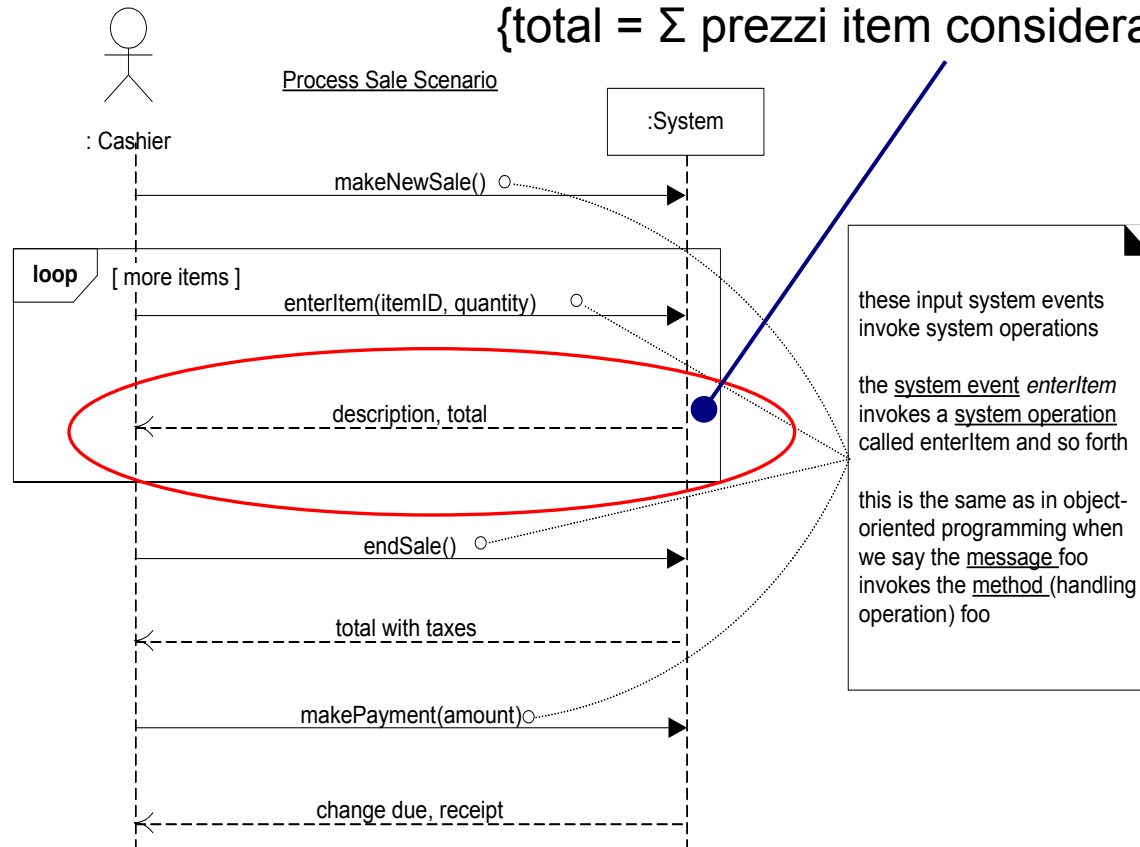
## **Calcolo del totale: scenario principale di successo della Process Sale**

- 1) Il Cliente arriva
- 2) Il Cassiere dice al Sistema di creare una nuova vendita
- 3) Il Cassiere inserisce il codice articolo
- 4) Il Sistema registra la riga di vendita per l'articolo e... il cassiere ripete 3-4 fino alla fine
- 5) Il Sistema mostra il totale

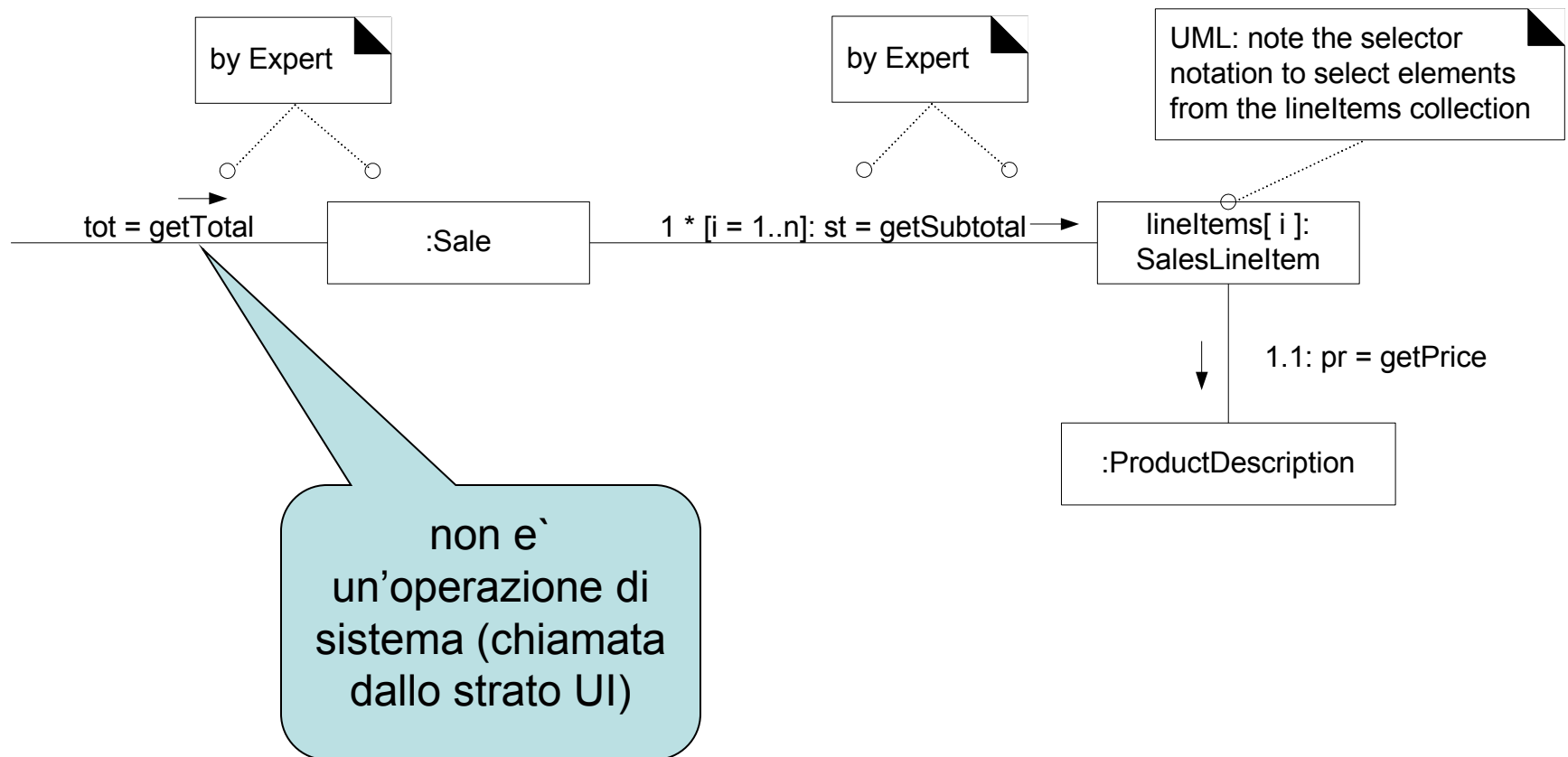
Invariante di  
ciclo

Fig. 11.2

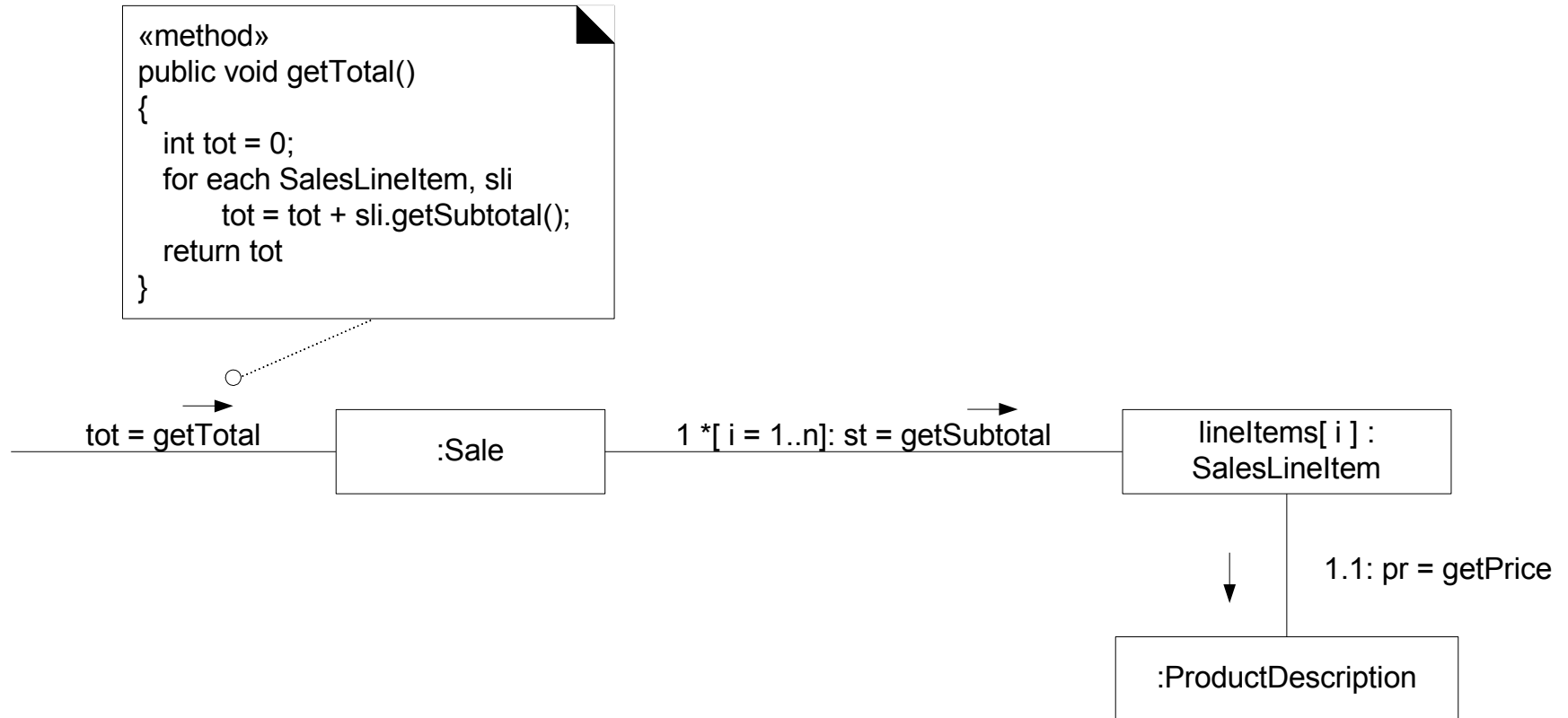
{total =  $\Sigma$  prezzi item considerati fino ad ora}



# Fig. 18.10: calcolare il totale della vendita



**Fig. 18.11: come indicare algoritmi (nota)**



## CO4: makePayment

**Operazione:** makePayment(amount: Money)

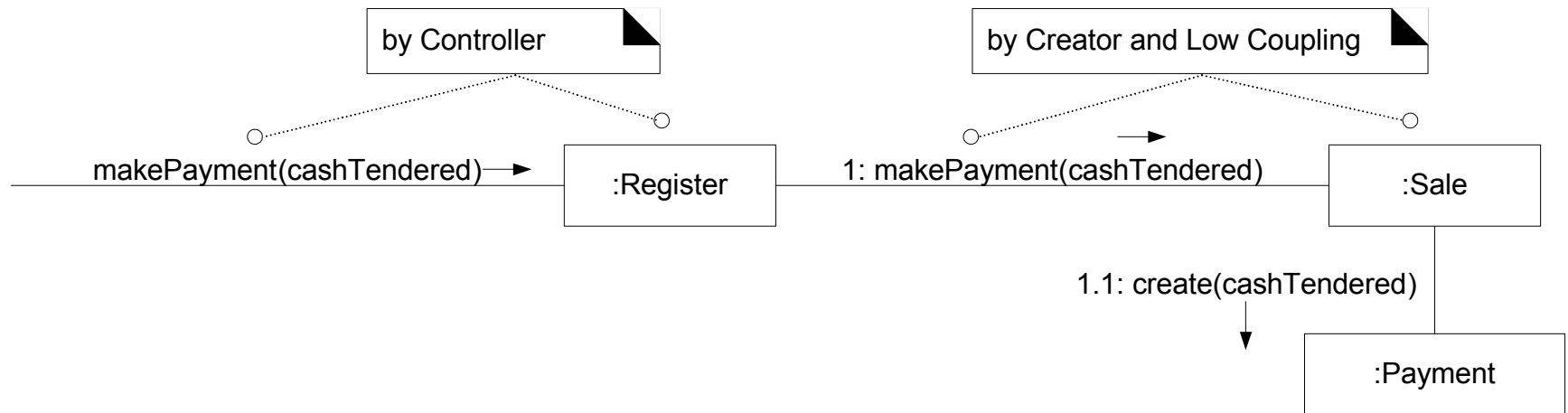
**Riferimenti:** casi d'uso: Process Sale

**Pre-condizioni:** e` in corso una vendita

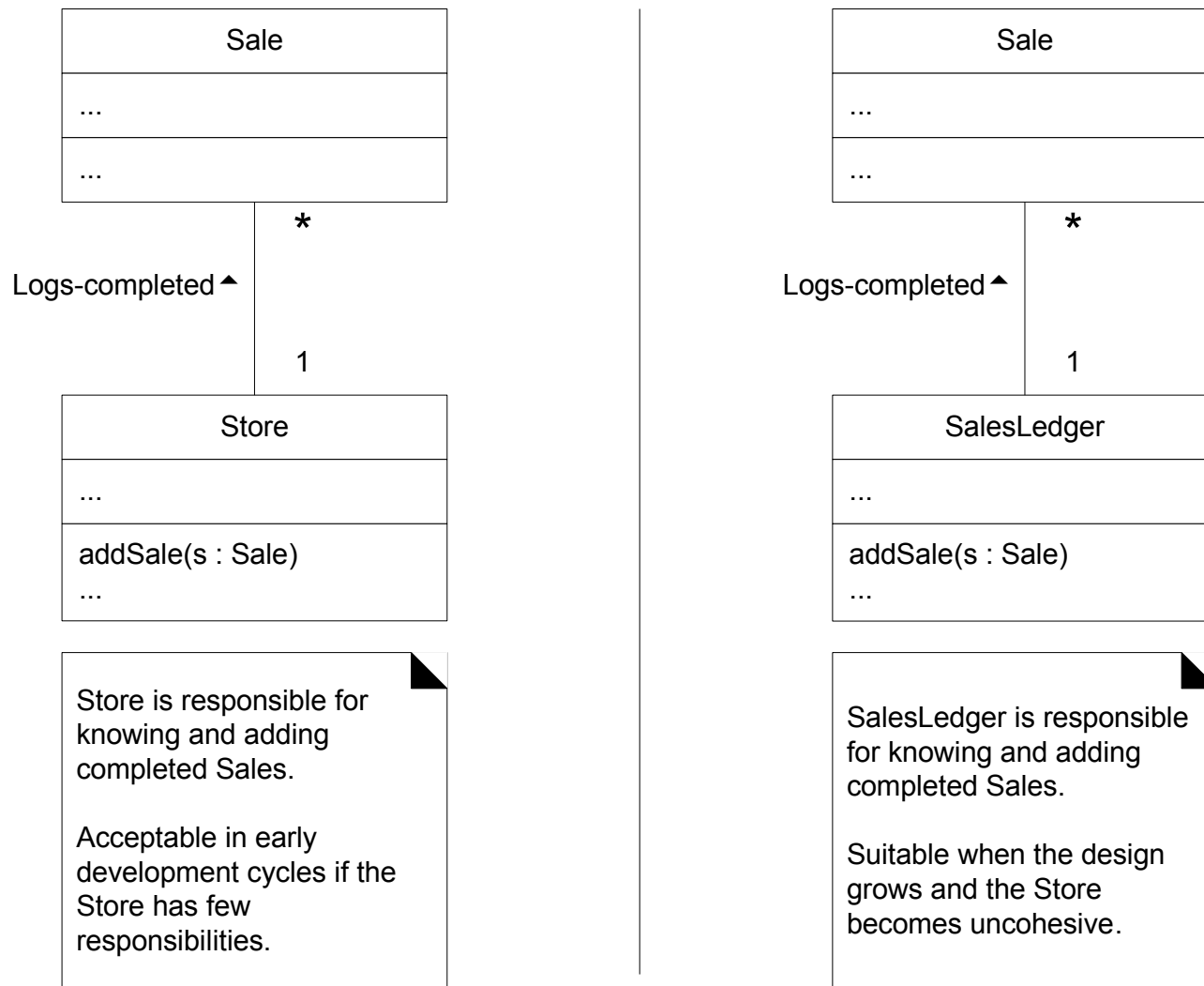
**Post-condizioni:**

- e` stata creata un'istanza p di Payment
- p.amountTendered = amount
- p e` stato associato con la Sale corrente
- la Sale corrente e` stata associata con lo Store (aggiunta al registro storico (*log*) delle vendite completate)

**Fig. 18.12: collaboration diagram per makePayment  
(uso Sale invece di Register: scelto con Low Coupling  
[gia` visto])**

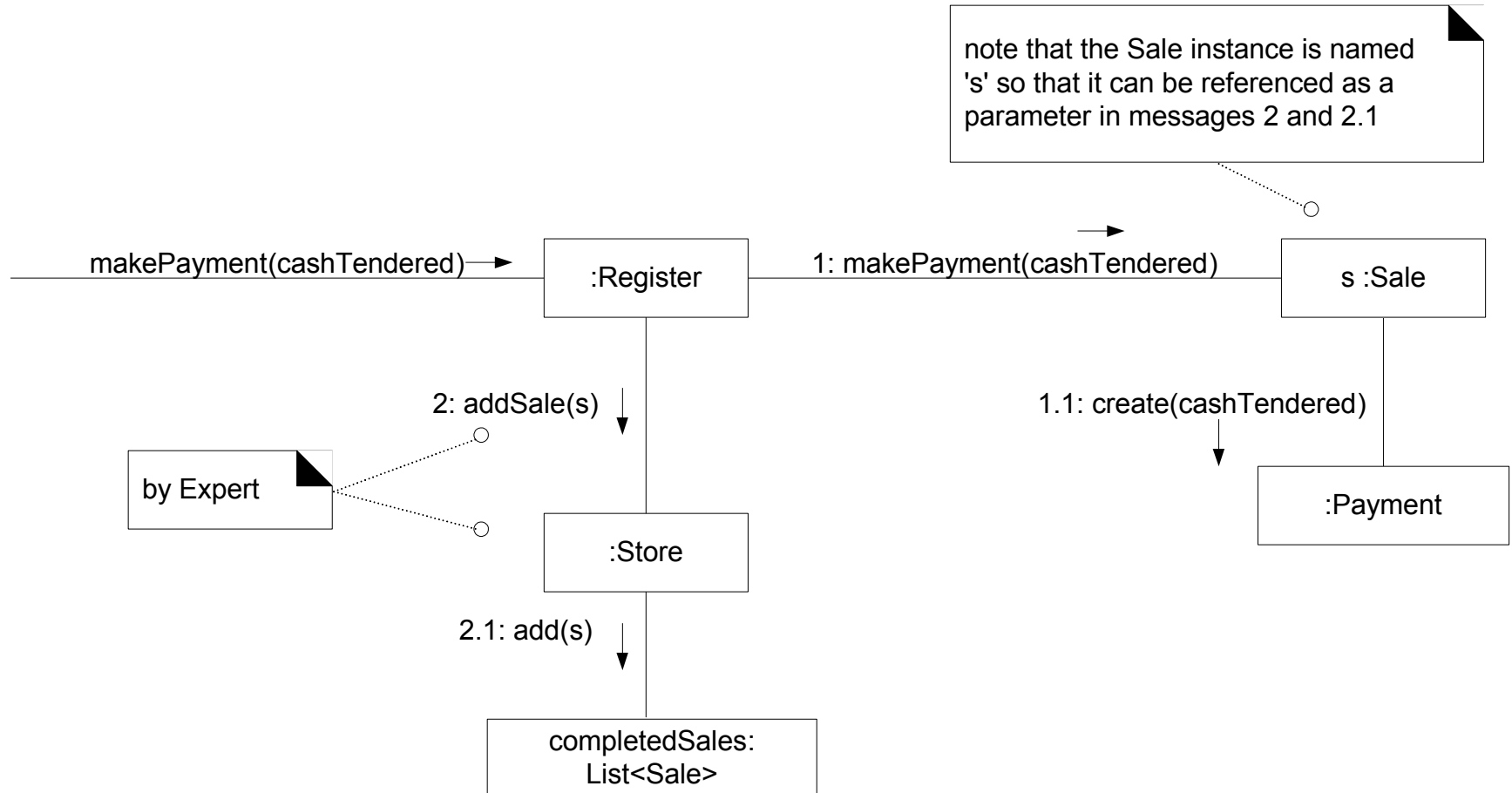


## Fig. 18.13: chi e' responsabile di conoscere le vendite completate?



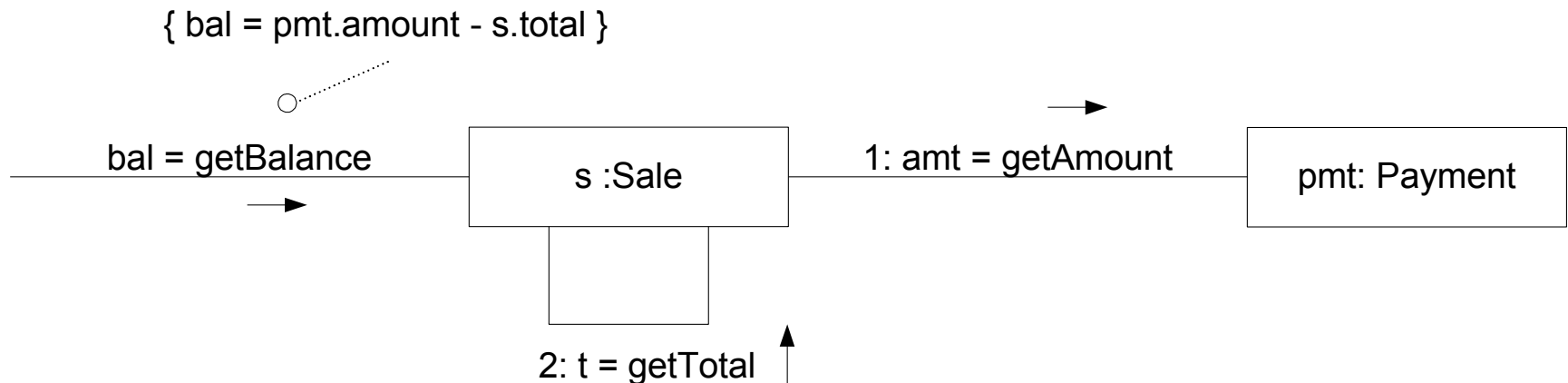


**Fig. 18.14: per adesso, scegliamo Store (anche se SalesLedger e' piu' adatto. NB: p. 357)**

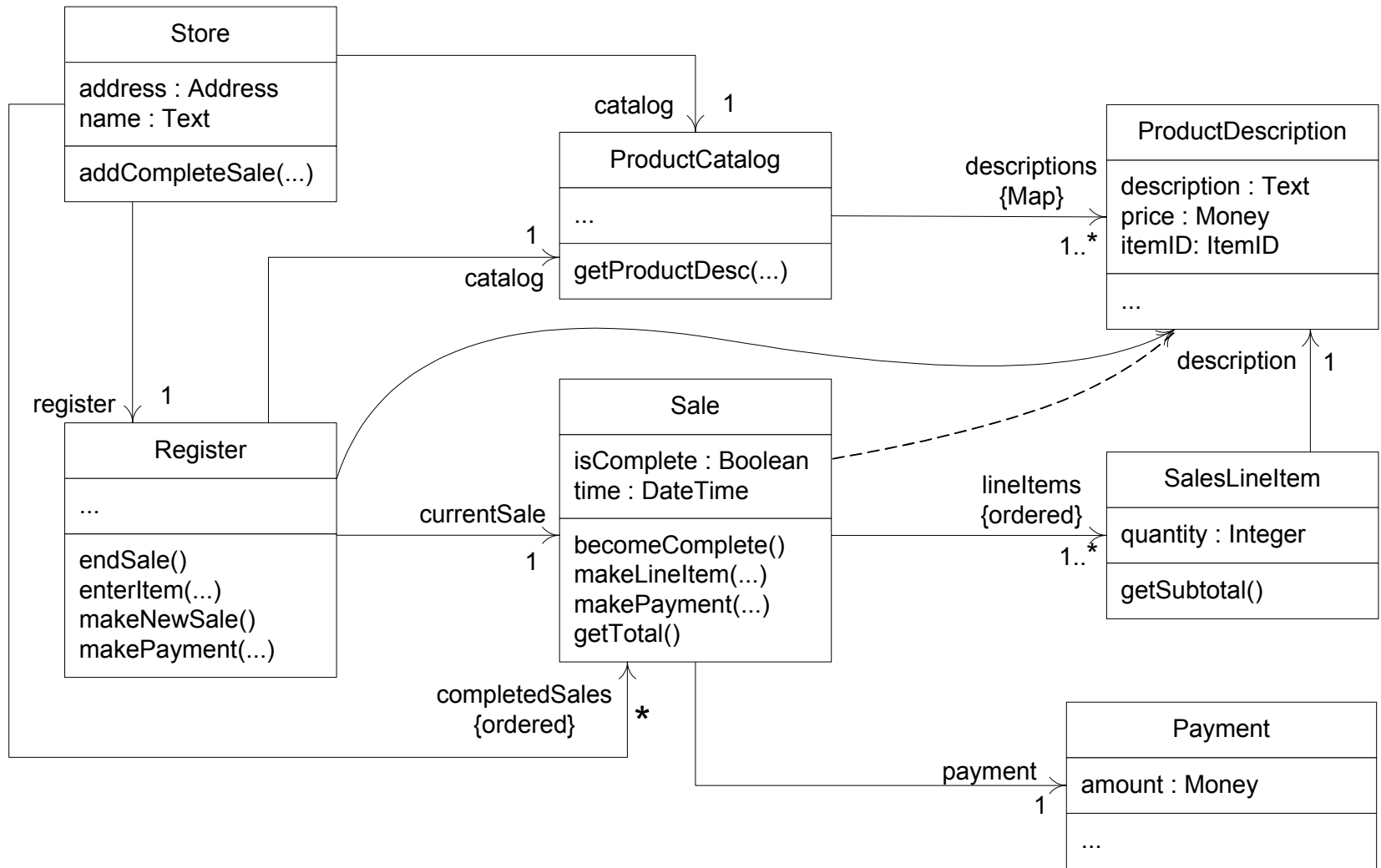


**Fig. 18.15: calcolo del resto (Payment e Sale hanno la conoscenza parziale per il resto, pero' meglio dare piu' responsabilita' a Sale che conosce gia' Payment dato che lo crea, e non e' vero il viceversa)**

**NB: calcolo del resto fa parte dello scenario principale di successo della Process Sale**



**Fig. 18.16: dove siamo arrivati per Process Sale (it. 1)**

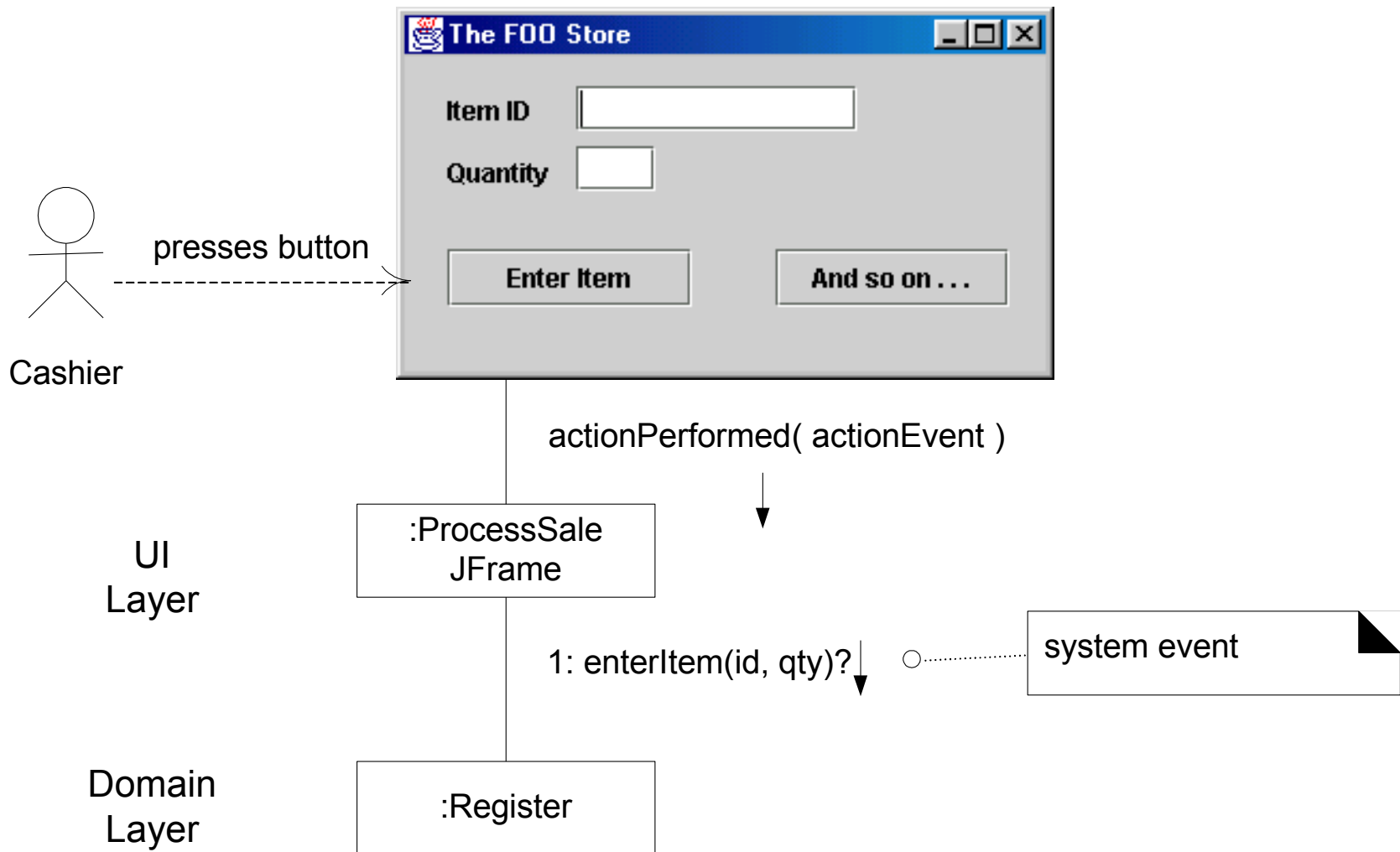


## Come collegare strato dell'UI a strato del dominio

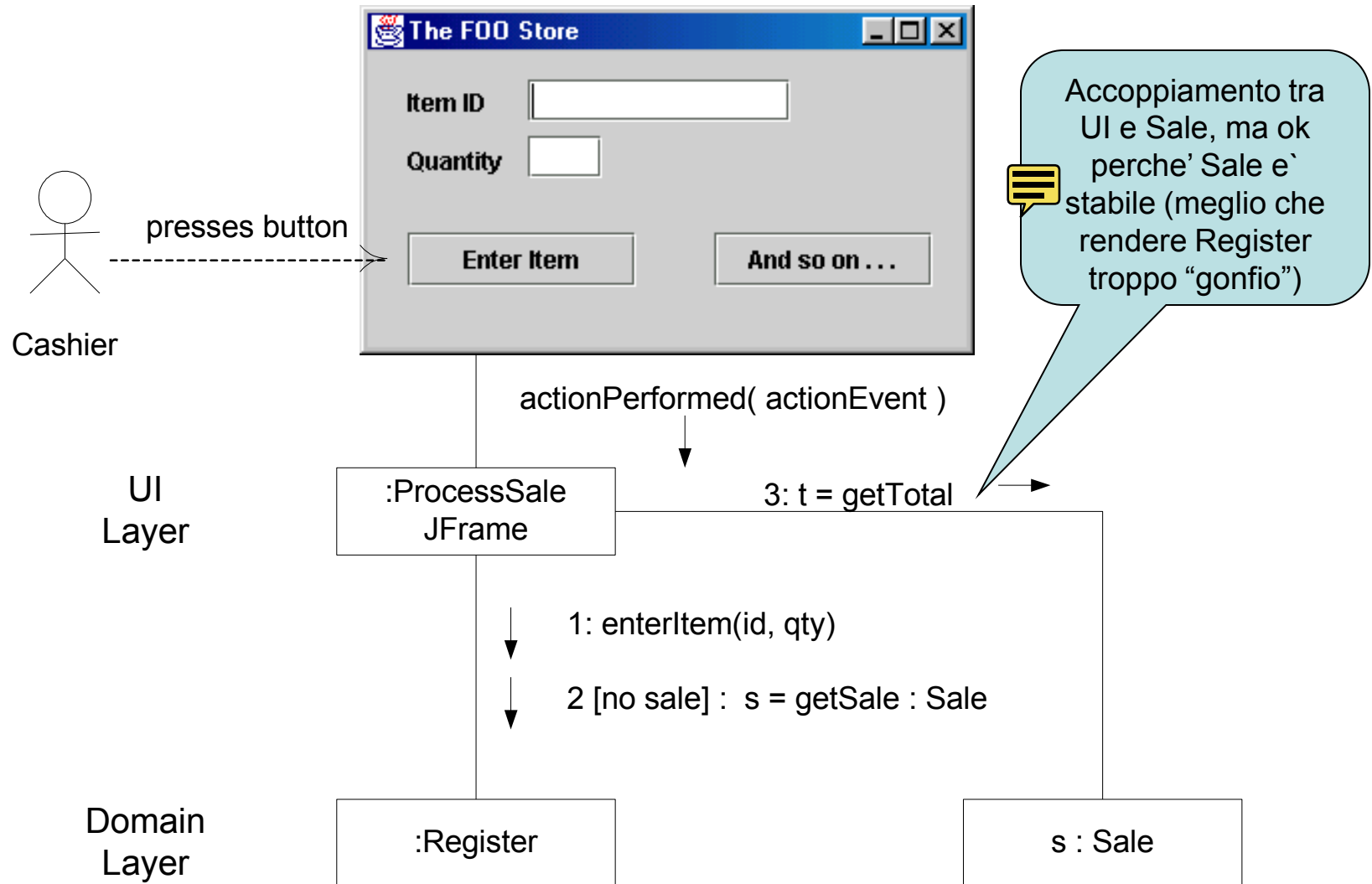
- Un oggetto iniziatore usato dal metodo iniziale dell'applicazione (es. il *main* di Java), che crea sia oggetto UI che oggetto di dominio e passa l'oggetto di dominio all'oggetto UI
- Un oggetto UI che recupera l'oggetto di dominio da sorgente nota, es. oggetto Factory responsabile della creazione di oggetti di dominio

In entrambi i casi, quando oggetto UI ha connessione con istanza di Register (il nostro facade controller), puo` mandargli i messaggi per eventi di sistema (es. enterItem e endSale)

**Fig. 18.17: connessione tra UI e dominio**




**Fig. 18.18: connessione UI e dominio (per mostrare totale parziale dopo ogni inserimento, o resto)**



## Possibile codice di start-up (in Java)

```
public class Main{  
    public static void main(String[] args){  
        //Store e` l'oggetto di dominio iniziale.  
        //Esso crea degli altri ogg. di dominio.  
        Store store = new Store();  
        Register register = store.getRegister();  
        ProcessSaleJFrame =  
            new ProcessSaleJFrame(register);  
        ...  
    }
```



collegamento  
tra UI e dominio

**Fig. 18.19: creazione dell'oggetto di dominio iniziale e degli oggetti successivi (*Start Up*)**

