Sviluppo delle applicazioni software

V. Bono

Capitolo 19

Progettare per la visibilita`

Visibilita`

Visibilita` = capacita` di un oggetto di vedere / di avere un riferimento a un altro oggetto (scope, campo d'azione)

Fig. 19.1: Register deve vedere ProductCatalog

```
class Register
                         private ProductCatalog catalog;
                                                         : ProductCatalog
                  : Register
   enterItem
(itemID, quantity)
                           desc = getProductDesc( itemID )
public void enterItem( itemID, qty )
 desc = catalog.getProductDesc(itemID)
```

4 modi

- 1) Visibilita` per attributo
- 2) Visibilita` per parametro (di metodo)
- 3) Visibilita` locale
- 4) Visibilita` globale

 Oggetto A puo` inviare messaggio a oggetto B se B e` visibile ad A

Commenti sui 4 modi

- 1) Visibilita` che dura quanto durano A e B
- Visibilita` che dura durante l'attivazione e nello scope del metodo
- 3) Come per (2)
- 4) Come per (1) (NB In C++ possibile via variabile globale, in Java no, si puo' pero' utilizzare GoF pattern *Singleton*)

Fig. 19.2 (visibilita` per attributo catalog)

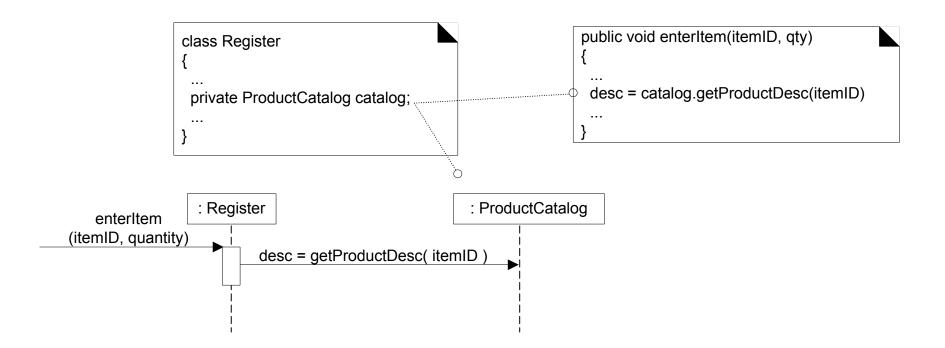


Fig. 19.3 (visibilita` per parametro desc)

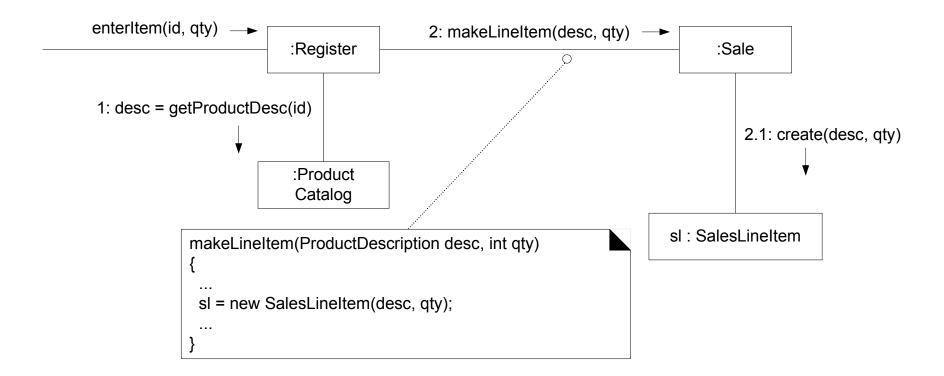


Fig. 19.4 (da v. parametro ad v. attributo)

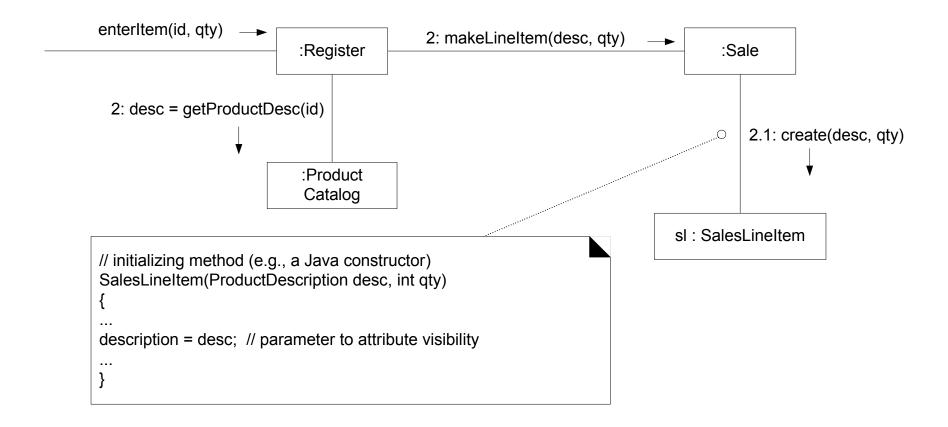


Fig. 19.5 (visibilita` locale: desc)

```
enterItem(id, qty)
// local visibility via assignment of returning object
ProductDescription desc = catalog.getProductDes(id);
                  : Register
                                                           : ProductCatalog
   enterItem
(itemID, quantity)
                           desc = getProductDesc( itemID )
```

Visibilita` locale implicita



```
// visibilita` locale implicita all'oggetto foo
// restituito dalla chiamata a getFoo
anObject.getFoo().doBar();
```