Progettazione del Software

Giuseppe De Giacomo & Massimo Mecella
Dipartimento di Informatica e Sistemistica
SAPIENZA Università di Roma

Diagramma degli stati e delle transizioni
Progetto

Progetto di classi con associato diagramma degli stati e delle transizioni

Decisioni preliminari sulla gestione degli eventi

- Per realizzare gli oggetti "reattivi", cioè con associato un diagramma degli stati e delle transizioni, sono possibili varie scelte. In particolare:
  - Gli **eventi sono chiamate a funzioni** che modificano lo stato dell'oggetto reattivo:
    - È uno schema realizzativo idoneo soprattutto quando l'interazione con gli oggetti è pilotata da un cliente esterno al sistema.
    - È stato utilizzato in vecchie edizioni di questo corso (Vecchio Prog. SW 1)
  - Gli **eventi sono messaggi** che oggetti reattivi si scambiano:
    - È uno schema realizzativo che permette una forte interazione tra i vari oggetti del sistema.
    - Porta a realizzare parti del programma **orientate agli eventi** (**event-based programming** – molto usato per interfacce grafiche, video-giochi, realizzazione di dispositivi reattivi)

Noi focalizziamo su questo!

Eventi come messaggi

- In questa edizione del corso noi ci **focalizzeremo su eventi come messaggi**, mettendo in piedi un "framework" opportuno per la gestione degli eventi.

- Tale gestione degli eventi verrà **inizialmente** proposta considerando un **unico flusso di controllo** (programmazione basata su eventi, realizzata con programmi sequenziali).

- Successivamente renderemo i **flussi di controllo** dei singoli oggetti indipendenti e **concorrenti** (programmazione basata su eventi, realizzato con programmi multi-thread).

Eventi come messaggi

- Assumeremo che ogni **evento** o messaggio abbia un **mittente** ed un **destinatario** esplicito, realizzando *connessioni point-to-point*.

- Inoltre permetteremo di mandare **messaggi in broadcasting** a tutti gli oggetti reattivi del sistema, cioè *connessioni broadcasting*

- Per semplicità **non affronteremo** il caso in cui i messaggi abbiano **più di un destinatario**, ma non siano in broadcasting, cioè *connessioni multicasting*. Va comunque osservato che quanto proposto può facilmente essere esteso a questo caso.

- Ammetteremo che un mittente possa non dichiararsi quando manda un messaggio (perché conoscere il mittente è irrilevante, o per altri motivi)

Supporto per lo scambio degli eventi:

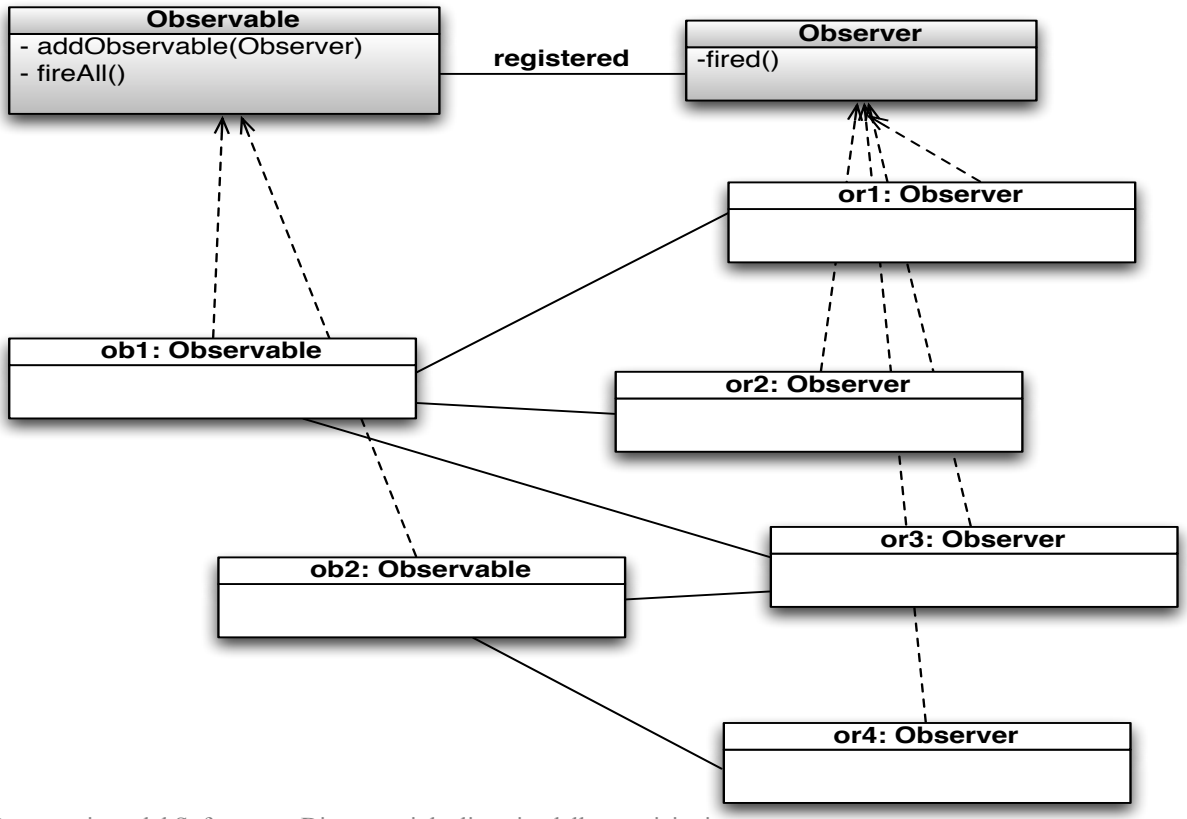
- Lo scambio degli eventi segue sostanzialmente il **pattern Observable-Observer**.

Pattern Observable-Observer

- Un **Observable** rappresenta un oggetto osservabile da altri oggetti legati ad esso ("registrati") detti **Observer**.
- Ogni **Observer** implementa una speciale funzione per reaggire alle notifiche dell'**Observable**, qui chiamata **fired()**.
- Un **Observable** registra, attraverso la funzione **addObserver()**, i sui **Observer**.
- Quando l'**Observable** vuole **notificare** qualcosa, attraverso **fireAll()** chiama su ciascun **Observer** registrato il suo metodo **fired()**.

*NB: la comunicazione **Observable-Observer** è unidirezionale **(responsabilità singola di Observable)**:*
*L'**Observable** comunica ai suoi **Observer** l'avvenimento di qualcosa.*

Pattern Observable-Observer



Pattern Observable-Observer

- In Java esistono le classi Observable e Observer ma essendo classi e non intefacce sono di fatto deprecate, perché costringono all'uso della derivazione tra classi per **riuso** invece che per **ISA**.

- Invece il pattern è implementato in molte librerie, eg Java Swing, utlizzando il nome di **Listener** invece di **Observer**

*Anche noi useremo **Listener** invece di **Observer** nelle nostre implementazioni.*

Environment: supporto per lo scambio degli eventi

- Nel nostro caso l'idea generale del pattern **Observable-Observer** va adattata in modo opportuno, visto che tutti gli oggetti reattivi ricevono eventi ma anche lanciano eventi (**comunicazione bidirezionale**).

- Per realizzare tale comunicazione bidirezionale faremo uso di un particolare oggetto **Environment**, che agisce da canale di comunicazione:

  - Tutti gli **oggetti reattivi manderanno all'Environment** i propri eventi

  - L'**Environment si occuperà di inoltrare ciascun evento** al gusto destinatario (che, si ricorda, è scritto sull'evento stesso).