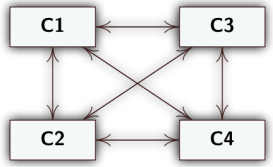
Mediator

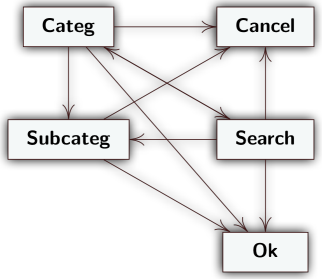
Prof. A. M. Calvagna

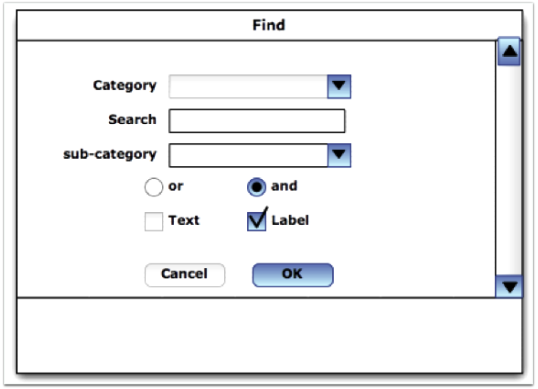
Mediator

- **Intento**
  - Definire una interfaccia per la comunicazione tra un gruppo di oggetti che interagisce.
- **Motivazione**
  - **Riuso**: Se ho molte interazioni (dipendenze) l'intero sistema si comporta come se fosse monolitico.
  - **Estensione**: Diventa difficile cambiare il comportamento anche dell'intero sistema poiché è distribuito fra gli oggetti e dovrei sottoclassarne molti o tutti.



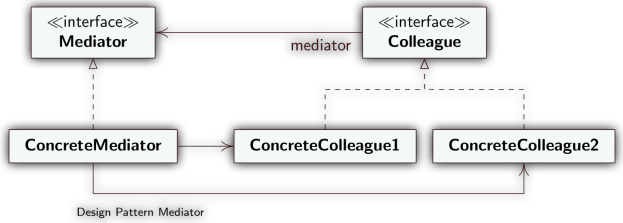
Mediator





Mediator

- **Soluzione**
  - Isolare le comunicazioni (complesse) tra oggetti dipendenti (cooperanti) in un sottosistema, creando una classe dedicata Mediator



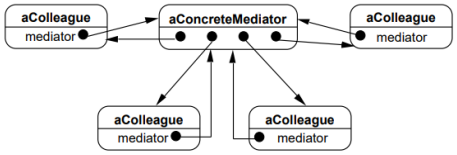
Esempio: COOLING SYSTEM

- a simple cooling system consists of a fan, a power supply, and a button.
- Pressing the button will either turn on or turn off the fan.
- Before we turn the fan on, we need to turn on the power.
- Similarly, we have to turn off the power right after the fan is turned off.

- Tre component interdipendenti: **Fan**, **PowerSupply** e **Button**.

Mediator

- Struttura a run-time



- L'oggetto mediator è responsabile per la gestione (**istanzia e mantiene i riferimenti**) ed il **coordinamento** dell'interazione tra i colleghi: invoca i loro servizi su loro richiesta e/o iniziativa propria
- Non è una semplice facciata (facade) : è un direttore che dirige i suoi musicisti
- Façade non aggiunge funzionalità; mediator ha invece al suo interno **tutta la logica di funzionamento** che prima era distribuita e può incorporarne **anche di nuova**
- I colleghi sono **consapevoli** del mediator, con cui interagiscono; i componenti del sottosistema sono inconsapevoli dell'esistenza del Façade. Per i client non cambia nulla in entrambi i casi

Mediator

- Applicabilità
  - Usare il Mediator quando
    - Un insieme di oggetti comunicano in modo **ben definito ma intricato**. Le interdipendenze che ne risultano sono non strutturate e difficili da comprendere
    - **Riusare un oggetto** è difficile poiché esso comunica con tanti altri oggetti
    - Un comportamento che è distribuito fra tante classi dovrebbe essere **modificabile senza dover ricorrere a sottoclassi**: lo centralizzo nel ConcreteMediator e cambio solo lui.

Varianti

- Il concrete mediator può accedere alle interfacce di tutti i concrete colleagues direttamente
- Ma I colleagues accedono al mediator…
  - Mediator con metodi espliciti: corrispondenti a quelli nei colleague
  - Mediator con metodo unico generico: eventualmente passo il colleague e lo identifico
  - Metodo generico in overloading: uno per ogni tipo di concrete colleague

Metodo generico

```
class DialogDirector {
    private Button button;
    private Fan fan;
    private PowerSupply powerSupply;

    // altro codice…

    public void doWork( Colleague me) {
       if ( me == button ) blah
       else if ( me == fan) more blah
       else if ( me == powerSupply ) even more blah
    }
}
```


Metodo generico in overload

```
class DialogDirector {
    private Button button;
    private Fan fan;
    private PowerSupply powerSupply;
     private Button anotherButton;


    // altro codice

    public void doWork(Button b) {
       if ( b == button ) // Process button task
       else if ( b == otherButton) // Process other button task
    }
    public void doWork(Fan f) {
       // Process fan task
    }
    public void doWork(PowerSupply b) {
       // Process button task
    }

}
```


Considerazioni finali

- La maggior parte della **complessità** che risulta nella gestione delle dipendenze è **spostata** dagli oggetti cooperanti **al Mediator**. Questo rende gli oggetti più facili da implementare e mantenere
- Le classi Colleague sono più riusabili poiché la loro **funzionalità** fondamentale **non è mischiata con** il codice che gestisce le **dipendenze**
- Il codice del **Mediator non è in genere riusabile** poiché la gestione delle dipendenze implementata è solo per una specifica applicazione: può essere direttamente una **classe concreta**
- Mediator è una astrazione proprio per consentire eventualmente di cambiarne implementazione a parità di colleagues