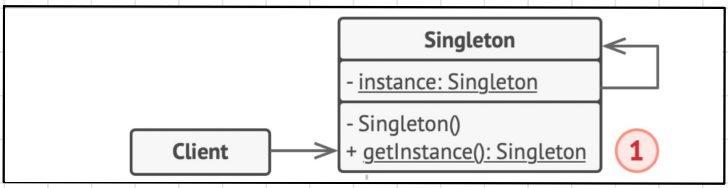
Design pattern

- **Creazionali :**
  1) **Singleton**
  2) **Factory Method**
  3) **Abstract Factory**
  13) **Prototype**

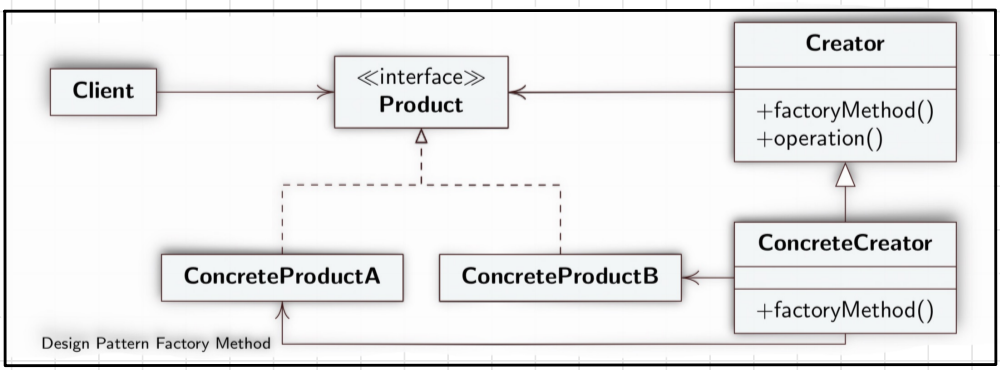
- **Strutturali :**
  4) **Adapter (Obj adapter / Class adapter)**
  5) **Facade**
  8) **Composite**
  9) **Decorator**
  11) **Bridge**

- **Comportamentali :**
  6) **State**
  7) **Observer | Reactive Streams | Model view controller (MVC)**
  10) **Mediator**
  12) **Chain of Responsability**
  14) **Command**

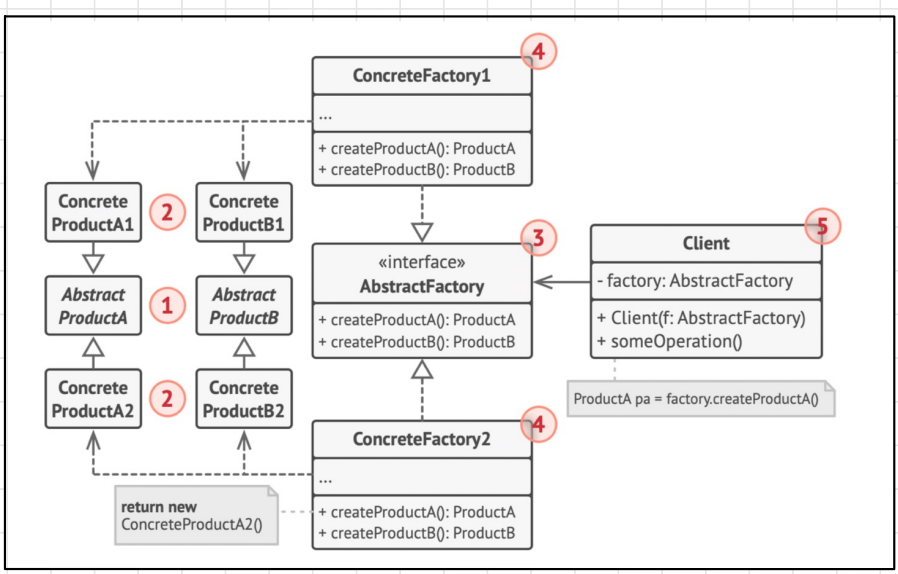
Singleton è un design pattern creazionale che consente di garantire che una classe abbia una sola istanza, fornendo al tempo stesso un punto di accesso globale a questa istanza.



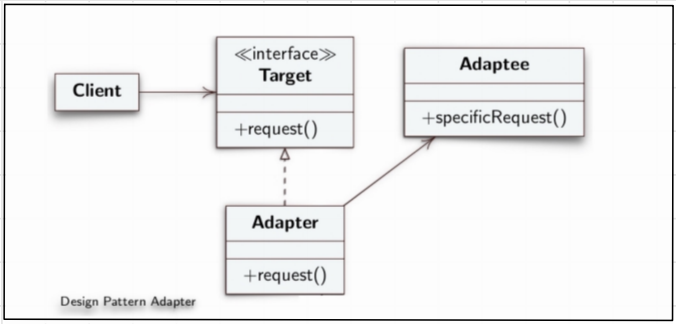
Factory Method è un design pattern creazionale che fornisce un'interfaccia per la creazione di oggetti in una superclasse, ma consente alle sottoclassi di modificare il tipo di oggetti che verranno creati.



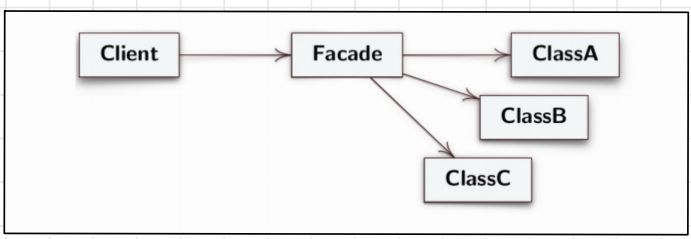
Abstract Factory è un design pattern creazionale che consente di produrre famiglie di oggetti correlati senza specificare le loro classi concrete.



Adapter è un design pattern strutturale che consente la collaborazione tra oggetti con interfacce incompatibili.

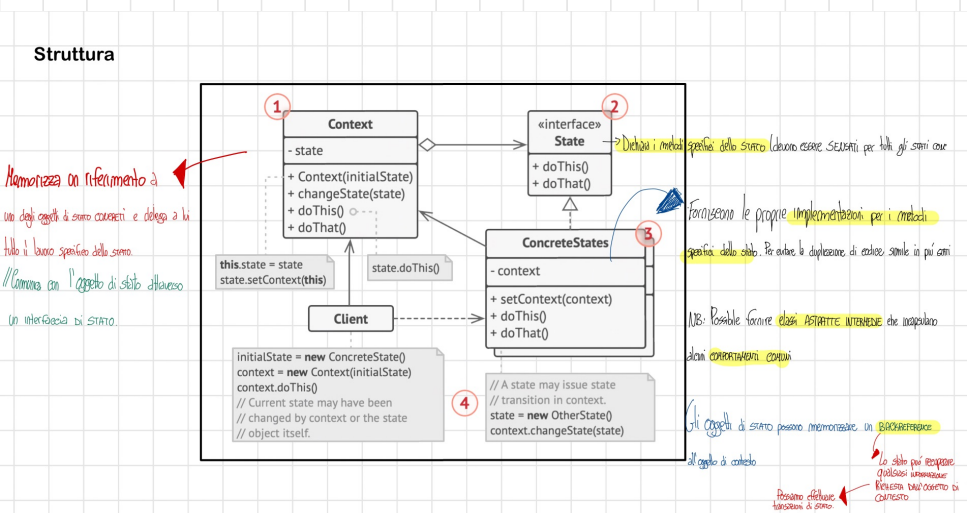


Facade è un design pattern strutturale che fornisce un'interfaccia semplificata a una libreria, un framework o qualsiasi altro insieme complesso di classi.



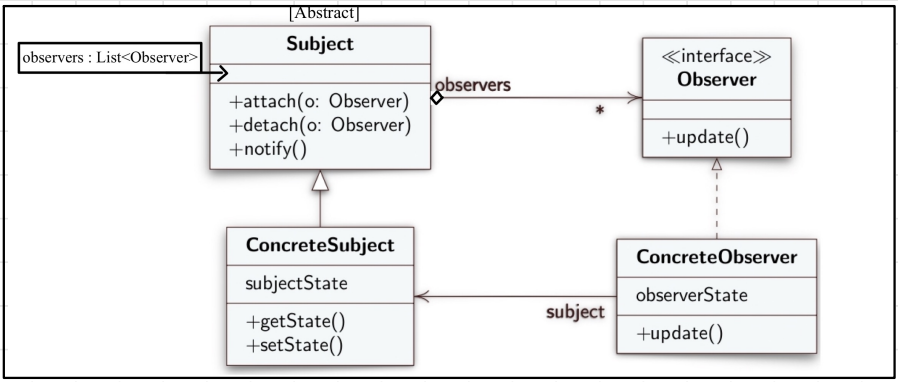
State è un design pattern comportamentale che consente a un oggetto di modificare il proprio comportamento quando il suo stato interno cambia. Sembra che l'oggetto abbia cambiato classe.

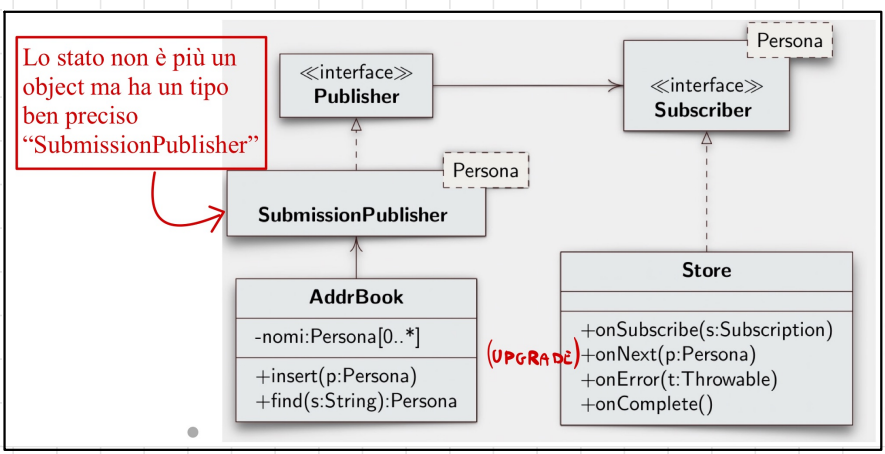
Struttura

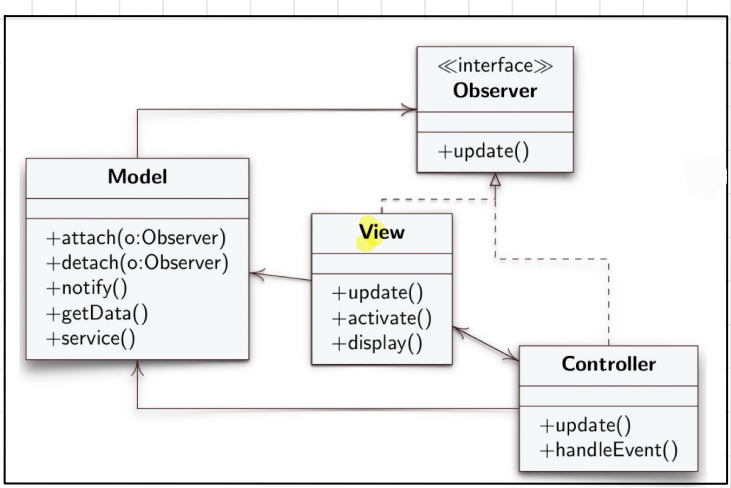


1. Il **Context** memorizza un riferimento a uno degli oggetti di stato concreti e delega ad esso tutto il lavoro specifico dello stato. Comunica con l'oggetto di stato attraverso un'interfaccia di stato.

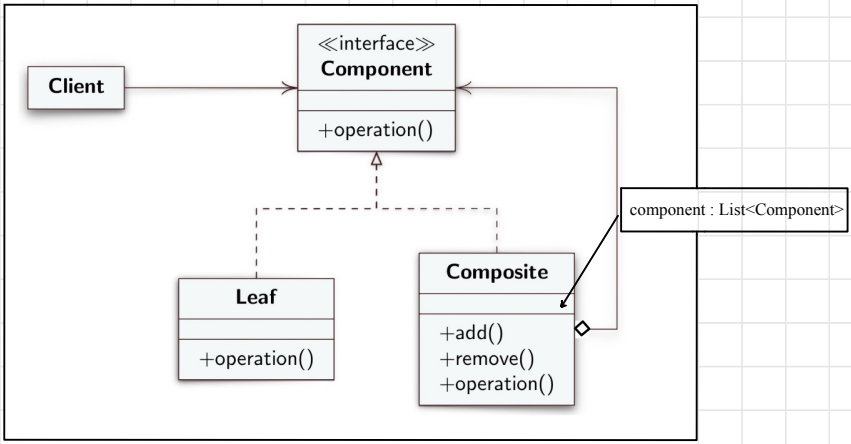
Observer è un design pattern comportamentale che consente di definire un meccanismo di sottoscrizione per notificare a più oggetti gli eventi che si verificano all'oggetto che stanno osservando.



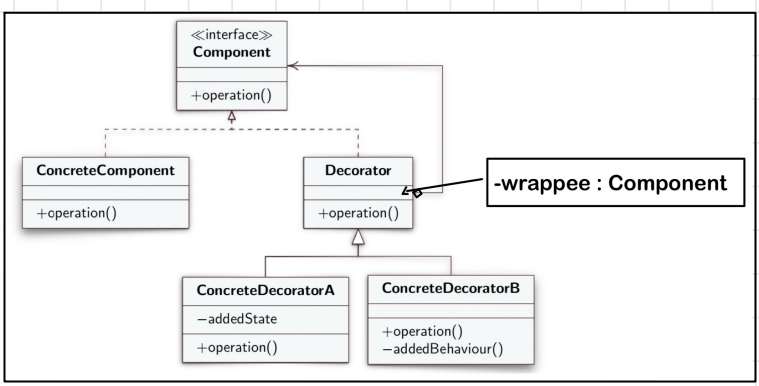




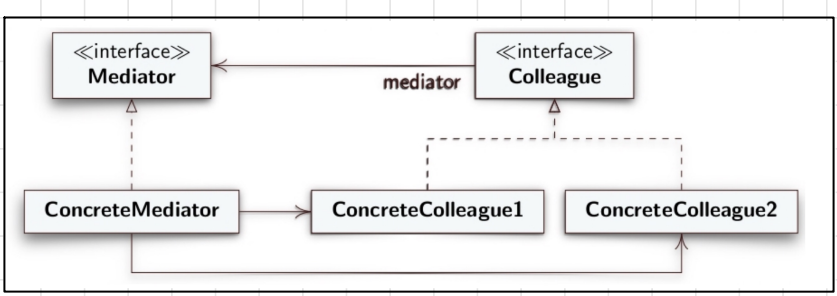
Composite è un design pattern strutturale che consente di comporre oggetti in strutture ad albero e quindi lavorare con queste strutture come se fossero singoli oggetti.



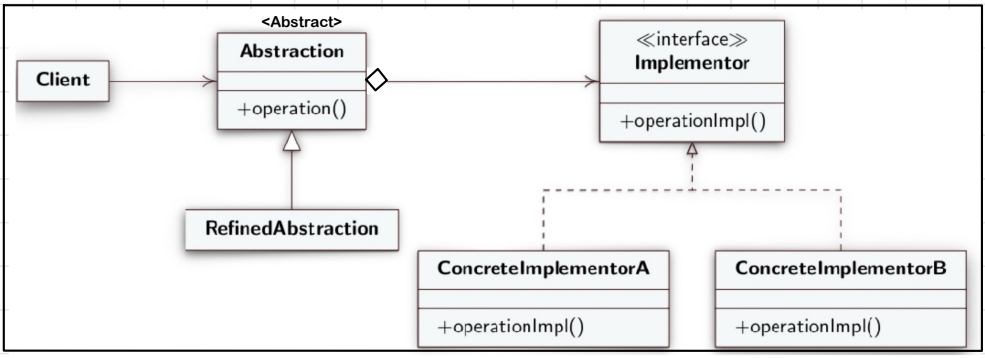
Decorator è un design pattern strutturale che consente di associare nuovi comportamenti agli oggetti inserendo questi oggetti all'interno di oggetti wrapper speciali che contengono i comportamenti.



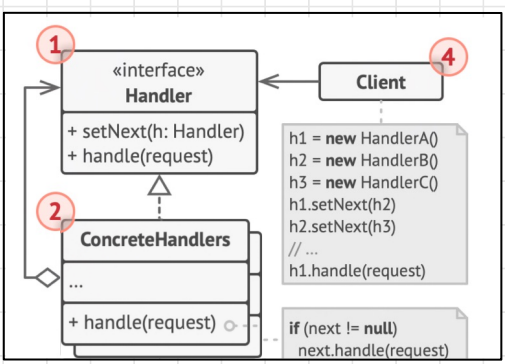
Mediator è un design pattern comportamentale che consente di ridurre le dipendenze caotiche tra gli oggetti. Il pattern limita le comunicazioni dirette tra gli oggetti e li costringe a collaborare solo tramite un oggetto mediatore.



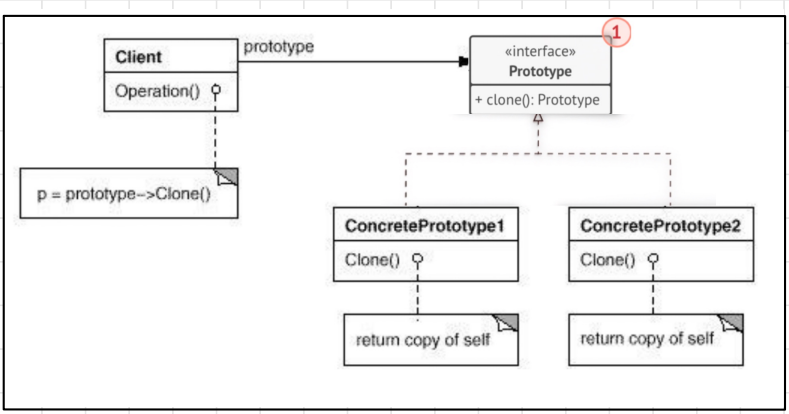
Bridge è un design pattern strutturale che consente di suddividere una classe di grandi dimensioni o un insieme di classi strettamente correlate in due gerarchie separate, astrazione e implementazione, che possono essere sviluppate indipendentemente l'una dall'altra.



Chain of Responsability è un design pattern comportamentale che consente di passare le richieste lungo una catena di gestori. Alla ricezione di una richiesta, ciascun gestore decide di elaborare la richiesta o di passarla al gestore successivo nella catena.



Prototype è un design pattern creazionale che consente di copiare oggetti esistenti senza rendere il codice dipendente dalle loro classi, ma si delega il processo di clonazione agli oggetti effettivi che vengono clonati. Il pattern dichiara un'interfaccia comune per tutti gli oggetti che supportano la clonazione.



Command è un design pattern comportamentale che trasforma una richiesta in un oggetto autonomo che contiene tutte le informazioni sulla richiesta. Questo pattern permette di separare chi richiede un'azione (client) dall'oggetto che effettivamente esegue l'azione (Receiver), creando un'interfaccia standard (Command) tra i due. Il pattern Command consente inoltre di registrare le richieste effettuate e permettendo di annullare o ripetere (undo e il redo) le operazioni.