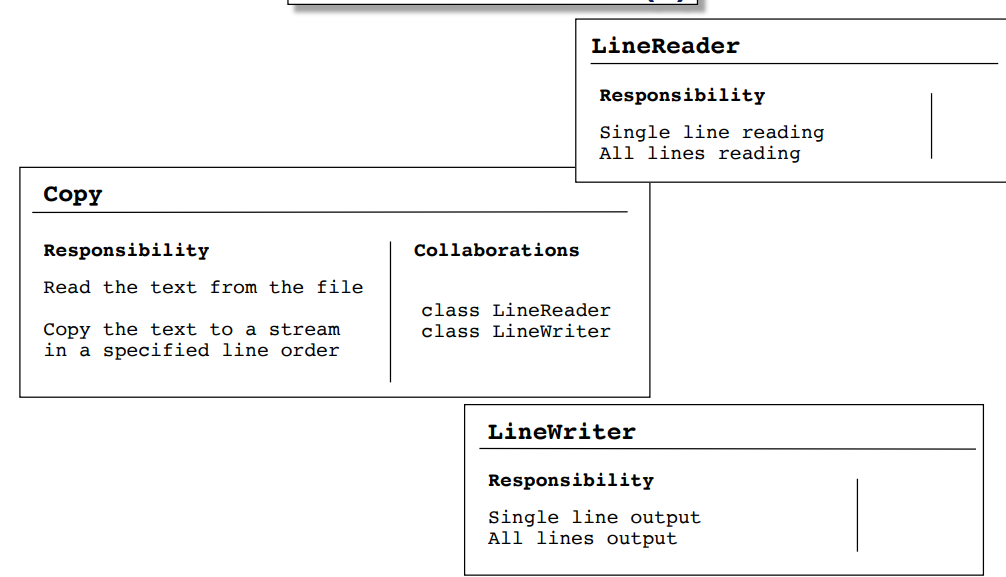
**Patterns**

Lo scopo della generalizzazione è quello di avere una gestione migliore della complessità dei sistemi, migliore modularizzazione, flessibilità, sistema più robusto….

**Class-Responsability-Collaboration (CRC) Cards**

Le classi devono avere poche responsabilità.

Le classi collaborano tra di loro per delegare alcuni task

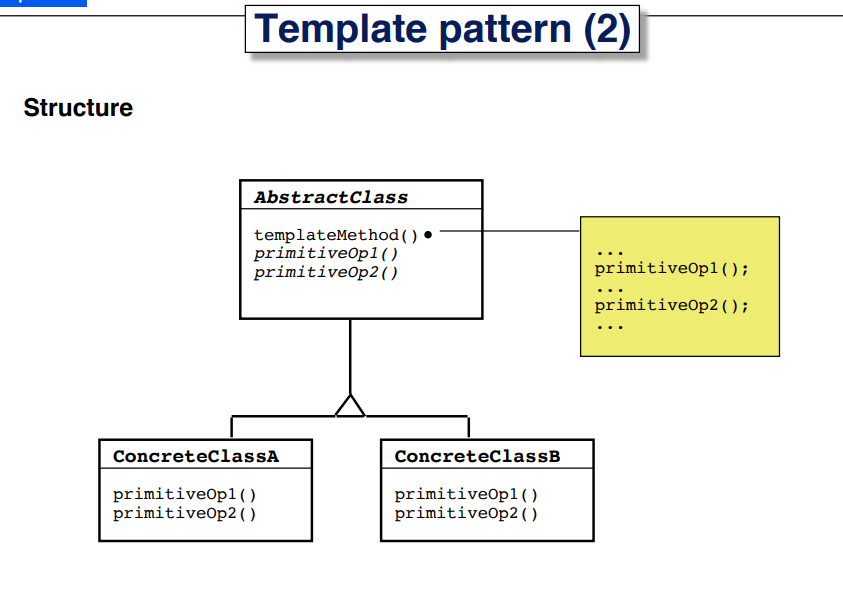


Il pattern strategy si combina bene con le funzioni lambda

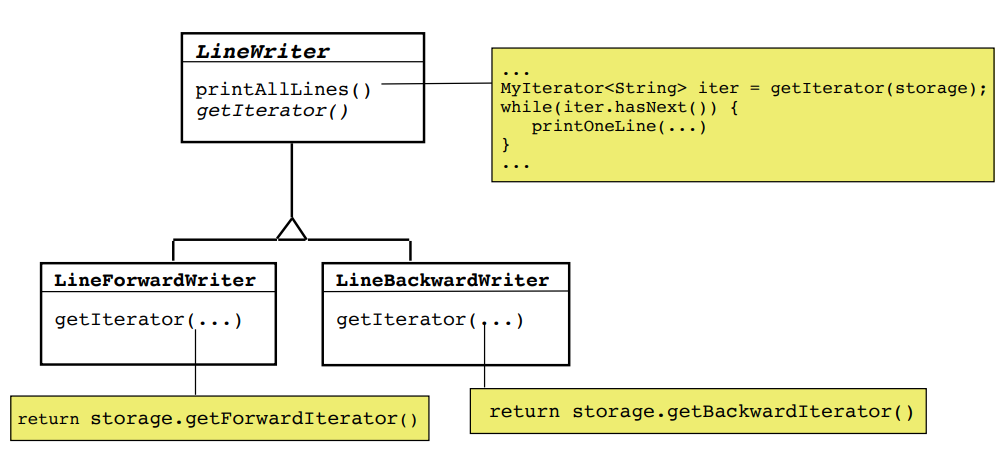
**Template pattern**

Permette di stabilire lo scheletro di un algoritmo delegando alcune funzioni ad altre classi.

Consente di avere più varianti dello stesso algoritmo.



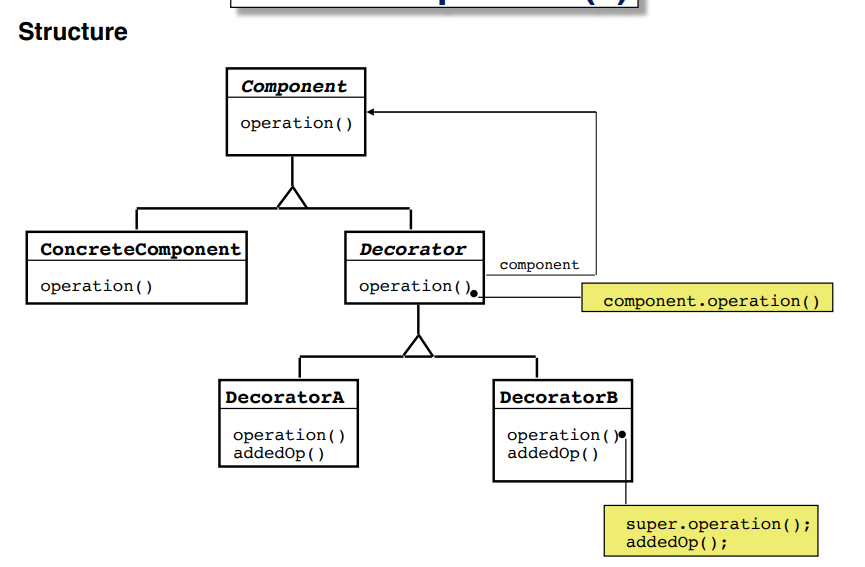
Applicazione



**Decorator Pattern**

Lavora sull’estensione di un oggetto, che è diversa dall’estensione di una classe, con le classi si devono prevedere a priori le classi di derivazione.

Con l’estensione di un oggetto si può estendere un oggetto a runtime, dando la possibilità di creare combinazioni di oggetti.

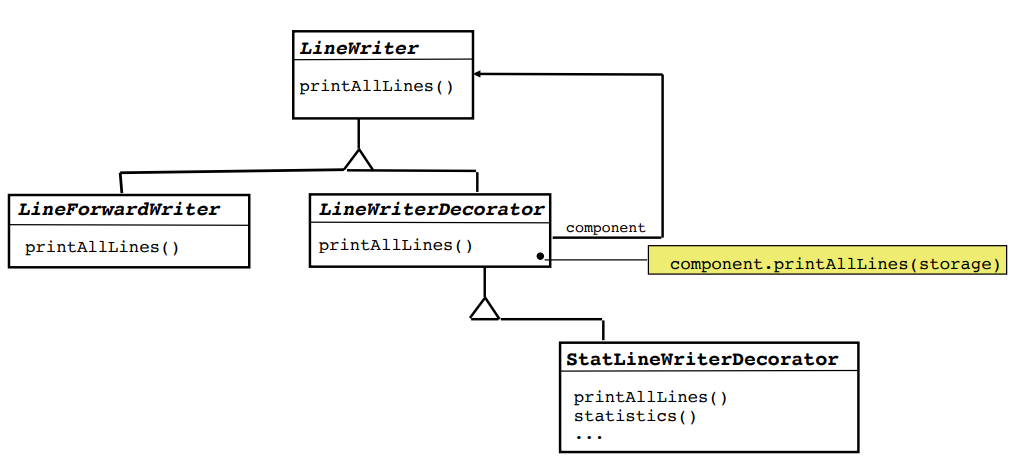


Responsabilità passate in modo dinamico

Combinazioni ricorsivo

Evita di avere un numero elevato di sottoclassi.

Esempio applicato



**Observer Pattern**

