Adapter & Facade

Alessandro Midolo, Ph.D. Student
alessandro.midolo@phd.unict.it

Tutorato Ingegneria del Software

A.A. 2021/2022

Adapter

Intento

- Convertire l'interfaccia di una classe in un'altra interfaccia che il client si aspetta. Permette la compatibilità delle interfacce di classi differenti le quali altrimenti non potrebbero lavorare insieme

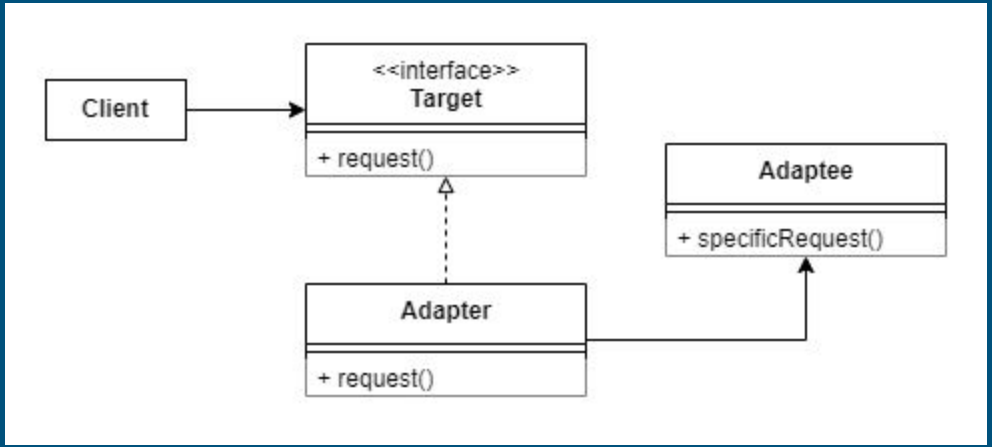
Problema

- Può capitare che una libreria esterna non può essere usata all'interno di un'applicazione perché incompatibile con l'interfaccia di quest'ultima (nome metodi, parametri, tipo parametri differenti)
- Non è possibile (non conviene) modificare il codice della libreria esterna
- Cambiare quale metodo invocare senza renderlo noto al chiamante

Soluzione

- **Target** è l'interfaccia che si aspetta il client
- **Adaptee** è l'oggetto della libreria da "adattare"
- **Adapter** si occupa di modificare la chiamata fatta dal client in modo da renderla compatibile con la libreria Tiene riferimento dell'**Adaptee** e sa come invocarlo, implementa le chiamate verso i metodi dell' **Adaptee**

Diagramma UML Adapter



Conseguenze Adapter

- Il client e l'Adaptee sono indipendenti tra loro → L'Adapter può modificare il comportamento dell'Adaptee
- Può inserire test di precondizione e postcondizione
- Dato che tutto le richieste fatte dal client devono passare dall'Adapter, potrebbe creare un collo di bottiglia (spesso trascurabile) e potrebbe portare ad una maggiore complessità del codice
- Può implementare la tecnica di **Lazy Initialization**

Facade

Intento

- Fornire un'interfaccia unificata per un insieme di interfacce in un sottosistema. Definisce un'interfaccia di **alto livello** per rendere il sottosistema più semplice da usare

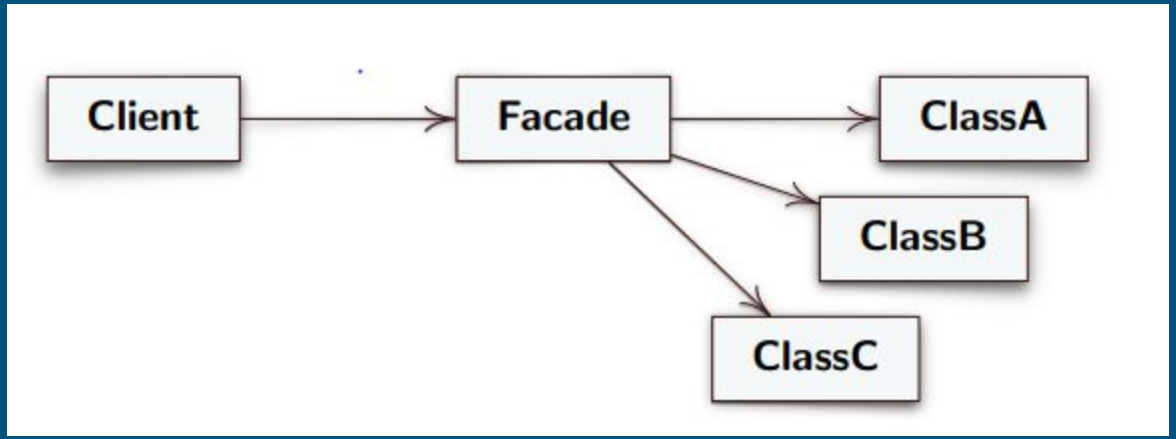
Problema

- Spesso capita di avere tante classi e l'insieme delle interfacce di queste può risultare complesso
- Potrebbe risultare difficile capire qual è l'interfaccia essenziale per l'insieme delle classi
- Si vogliono ridurre le comunicazioni e le dipendenze fra i client ed il sottosistema

Soluzione

- **Facade** fornisce l'interfaccia semplificata per il client nascondendo gli oggetti del sottosistema. Invoca i metodi degli oggetti che nasconde
- **Client** interagisce solo con l'oggetto **Facade**

Diagramma UML Facade



Conseguenze Facade

- Nasconde ai client l’implementazione del sottosistema
- Riduce le dipendenze di compilazione in sistemi grandi. Se si cambia una classe del sottosistema, questo non avrà ripercussioni nei vari client, ma solo nel Facade
- Non previene l’accesso da parte di client più complessi, se necessari, che vogliono accedere agli oggetti del sottosistema
- Promuove l’accoppiamento debole tra sottosistema e client