Strutturale

Design pattern Facade → Si vuole fornire un interfaccia che è semplificata al posto di un insieme di classi e sottosistemi complessi

tipo JAVA (CI PERMETTE DI DEFINIRE I VALORI DI RITORNO E PARAMETRI DELLE FUNZIONI)

così si semplifica la vita del programmatore

Facade è un design pattern strutturale. Si vuole fornire un'interfaccia che è semplificata al posto di un insieme di classi/ sottosistemi al fine di semplificare il lavoro al programmatore. L'interfaccia è il tipo java che non offre implementazioni di alcun metodo ma ci consente di definire i parametri e valori di ritorno delle funzioni. L'interfaccia ha anche un altro significato, ci si riferisce ad un punto di contatto tra l'utilizzatore e il sottosistema che si vuole utilizzare : interfaccia Facade.

Intento → FORNISCE INTERFACCIA SEMPLICE

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

Problema

Immagina di dover far funzionare il tuo codice con un ampio insieme di oggetti che appartengono a una libreria o un framework sofisticato. Normalmente, dovresti inizializzare tutti quegli oggetti, tenere traccia delle dipendenze, eseguire metodi nell'ordine corretto e così via. Di conseguenza, la logica aziendale delle tue classi diventerebbe strettamente accoppiata ai dettagli di implementazione delle classi di terze parti, rendendone difficile la comprensione e la manutenzione.

Quando costruiamo un sistema software per il discorso della modularità ci si ritrova a sviluppare tante classi e c'è un client che quindi ha bisogno di interagire con queste per ottenere servizi. Le dipendenze tra il client e le classi rendono difficile la comprensione del codice .

Soluzione

UTILIZZARE FACADE che è una classe che fornisce una semplice interfaccia a un sotto sistema complesso.

Utile quando devi integrare la tua app con una libreria sofisticata che ha dozzine di funzionalità, ma a cui ne interessano poche.

Una facade è una classe che fornisce una semplice interfaccia a un sottosistema complesso che contiene molte parti mobili. Una facade potrebbe fornire funzionalità limitate rispetto all'utilizzo diretto del sottosistema. Tuttavia, include solo quelle funzionalità che interessano davvero ai clienti. Avere una facade è utile quando devi integrare la tua app con una libreria sofisticata che ha dozzine di funzionalità, ma hai solo bisogno di una piccola parte delle sue funzionalità.

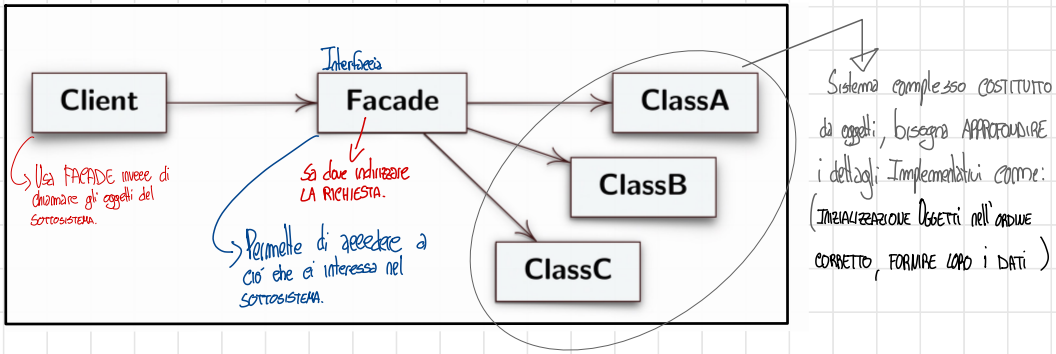
Ad esempio, un'app che carica brevi video divertenti con gatti sui social media potrebbe potenzialmente utilizzare una libreria di conversione video professionale. Tuttavia, tutto ciò di cui ha veramente bisogno è una classe con il metodo "encode(filename, format)" . Dopo aver creato una classe di questo tipo e averla collegata alla libreria di conversione video, avrai la tua prima facciata.

Quando chiami un negozio per effettuare un ordine telefonico, un operatore è la tua facciata verso tutti i servizi e i reparti del negozio. L'operatore fornisce una semplice interfaccia vocale al sistema di ordinazione, gateway di pagamento e vari servizi di consegna.

Facciamo in maniera tale che il client conosca solo Facade e l'interfaccia Facade conosce le varie classi con cui il client ha bisogno di interfacciarsi, in modo da rendere il codice più modulare. Si ha un vantaggio anche in termini di dipendenza in quanto il client dipende solo da Facade (ha solo una dipendenza). Il facade deve offrire un'interfaccia semplificata al client, significa che ci poniamo la domanda "cosa serve al client?", e forniamo metodi opportuni in modo tale da soddisfarne le richieste.

Se dovessimo modificare il nome di una classe del sottosistema non bisognerà modificare l'intero codice ma basterà modificare la sola interfaccia Facade.

Struttura



- L'interfaccia **Facade** fornisce un comodo accesso a una parte particolare della funzionalità del sottosistema. Sa dove indirizzare la richiesta del cliente e come azionare tutte le parti in movimento.

- Il sottosistema complesso è costituito da vari oggetti. Per fare in modo che tutti facciano qualcosa di significativo, devi approfondire i dettagli di implementazione del sottosistema, come inizializzare gli oggetti nell'ordine corretto e fornire loro i dati nel formato corretto.
  Le classi del sottosistema non sono a conoscenza dell'esistenza della facade. Operano all'interno del sistema e collaborano direttamente tra loro.

- Il client usa la facade invece di chiamare direttamente gli oggetti del sottosistema.

Conseguenze Nasconde ai client l'implementazione del sottosistema, Promuove Accoppiamento debole tra sottosistema e client.
Riduce le dipendenze di compilazione in sistemi grandi, Non prevede l'uso di client più complessi

• Nasconde ai client l'implementazione del sottosistema

• Promuove l'accoppiamento debole tra sottosistema e client

• Riduce le dipendenze di compilazione in sistemi grandi. Se si cambia una classe del sottosistema, si può ricompilare la parte di sottosistema fino al facade, quindi non i vari client

• Non previene l'uso di client più complessi, quando occorre, che accedono ad oggetti del sottosistema

PRIMA = Vi è una classe che istanzia oggetti di altre classi
Dopo = il Client comunica con il Facade e il Facade comunica con le VARIE sottoclassi.

Domanda : Applicazione prima e dopo

Prima -> Si crea un insieme di classi dove una classe a sua volta istanzia oggetti di altre classi

Dopo -> Si crea un solo client, un solo facade e le varie sottoclassi

Questo fa capire che tra prima e dopo soddisfiamo molto i requisiti dell'ingegneria del software ottenendo tutti i vantaggi scritti sopra

• Quando vi è bisogno di un interfaccia limitata ma semplice per un sottosistema più COMPLESSO.

Applicabilità

• Quando si vuole strutturare a livelli il sottosistema.

- Utilizzare il modello Facade quando è necessario disporre di un'interfaccia limitata ma semplice per un sottosistema complesso.

Spesso i sottosistemi diventano più complessi nel tempo. Anche l'applicazione di modelli di progettazione in genere porta alla creazione di più classi. Un sottosistema può diventare più flessibile e più facile da riutilizzare in vari contesti, ma la quantità di configurazione e codice boilerplate che richiede a un client cresce sempre di più. The Facade tenta di risolvere questo problema fornendo una scorciatoia alle funzionalità più utilizzate del sottosistema che soddisfano la maggior parte dei requisiti del cliente.

- Utilizzare facade quando si desidera strutturare un sottosistema in livelli.

Crea facciate per definire i punti di ingresso a ciascun livello di un sottosistema. È possibile ridurre l'accoppiamento tra più sottosistemi richiedendo loro di comunicare solo tramite facciate.

Ad esempio, torniamo al nostro framework di conversione video. Può essere suddiviso in due livelli: video e audio. Per ogni livello, puoi creare una facciata e quindi far comunicare tra loro le classi di ogni livello tramite quelle facciate.

Implementazione

Verificare se è possibile fornire un'interfaccia più semplice rispetto a quella già fornita da un sottosistema esistente. Sei sulla buona strada se questa interfaccia rende il codice client indipendente da molte delle classi del sottosistema.

Dichiara e implementa questa interfaccia in una nuova classe di facciata. La facciata dovrebbe reindirizzare le chiamate dal codice client agli oggetti appropriati del sottosistema. La facciata dovrebbe essere responsabile dell'inizializzazione del sottosistema e della gestione del suo ulteriore ciclo di vita, a meno che il codice client non lo faccia già.

Per ottenere il massimo vantaggio dal modello, fai in modo che tutto il codice client comunichi con il sottosistema solo tramite la facciata. Ora il codice client è protetto da qualsiasi modifica nel codice del sottosistema. Ad esempio, quando un sottosistema viene aggiornato a una nuova versione, dovrai solo modificare il codice nella facciata. Se la facciata diventa troppo grande , prendere in considerazione l'estrazione di parte del suo comportamento in una nuova classe di facciata raffinata.

Per rendere gli oggetti del sottosistema non accessibili al client le corrispondenti classi possono essere annidate dentro la classe Facade