**Шаблон Façade**

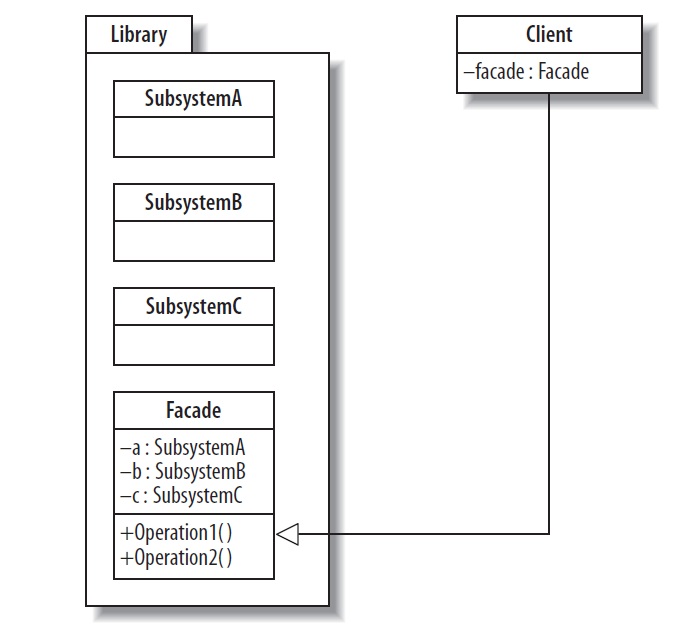
1. Описание

Ролята на шаблона Façade е да представи различен high-level поглед на подсистемите, чийто детайли са скрити за потребителите. Като цяло операциите, които могат да бъдат желани от гледна точка на потребителя, могат да бъдат направени чрез избор на различни части от подсистемата.

В реалния живот има много такива примери където прости интерфейси скриват сложни подсистеми. Те могат да бъдат създадени с цел честото използване на системата да става бързо или да се прави разлика между начинаещите и напредналите потребители. Един добър пример е опцията (или системата) “Bye now with 1-Click“ използвана от Amazon.com, опростяваща поръчване на продукти за добре познатите клиенти. Обикновено, след избор на продукт за покупка, клиента трябва да въведе информация за доставката и банковия акаунт преди да поръча. Ако тези детайли се съхраняват и клиента се идентификира по някакъв начин, 1 клик ще отведе клиента направо към касата (поръчването). Съхранените банков акаунт и начин на доставка се използват за покупката и по точи начин значително се ускорява (и опростява) пазаруването. Тази 1- Click опция представлява фасада на пълната система под нея.

1. Дизайн

Скриването на подробностите е ключово понятие при програмиране. Това което отличава фасадата от , да речем, Адаптер или Декоратор е, че интерфейсът който се построява може да бъде изцяло нов. Това не е зададено към съществуващите изисквания, нито пък трябва да отговаря на съществуващите интерфейси. Може да има няколко фасади изградени около съществуващ набор от подсистеми. Терминът “подсистема“ тук се използва съзнателно; ние говорим за по-високо ниво от класове. Нека разгледаме фигурата по-долу. Приема се, че подсистемите са групирани заедно, така че те могат да си взаимодействат в предварително приетите начини за създаване на операциите от най-високо ниво.



Тук ролите са:

Namespace 1 – Библиотека на подсистемите;

Subsystem – Конкретен клас предлагащ подробни операции

Façade – Класът който предлага операции от високо ниво като избор от подсистеми

Namespace 2 – там където пребивава клиента

Client – Извиква операциите от високо ниво от Namespace 1 във Фасадата

Както се вижда на диаграмата, кода на клиента не може да се обръща към класовете от имена на подсистемите; само получава достъп до техните операции чрез фасадата.

1. Имплементация

Шаблона Фасада е лесен за изпълнение . Той използва понятието в C# за namespaces. Класовете в namespaces имат възможност да определят достъпността за internal или public. Ако достъпността се определя като internal, елементите са видими само в асемблито където пространството от имена се компилира. При много голяма система, GUI на клиента ще бъде в namespaces от библиотеката, така че ние може да наложи използване на фасадата.

Namespace 1 - Amazon.com сървъра

Subsystems – Регистрация, количка, въвеждане на адрес, валидация на акаунта и др.

Façade – Прозореца за избор за начин при закупуване

Operation – 1-Click

Namespace 2 – Клиентския GUI

Client – купувач

=======

Пример:

1. Създаваме проект Class Library, с името FacadeLib. Той включва класовете :

* internal class SubsystemA
* internal class SubsystemB
* internal class SubsystemC
* public static class Facade

internal класовете се виждат единствено в пространството от имена FacadeLib. Връзката към външния свят е единствено през public класа Facade. Билдваме приложението и така ще имаме библиотека FacadeLib.dll която можем да използваме в друго приложение.

2. Създаваме втори проект - конзолен, за клиента с името FacadePattern. При него трябва да включим като референция горната библиотека библиотеката:

Избираме проекта -> десен бутон -> Add reference -> Browse -> FacadeLib.dll -> Add

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using FacadeLib;

namespace FacadePattern

{

class Program

{

//Client

static void Main(string[] args)

{

Console.WriteLine("Hello");

Facade.Operation1();

Facade.Operation2();

}

}

}