

Projektowanie obiektowe oprogramowania

Wykład 7 – wzorce czynnościowe (2)

Wiktor Zychla 2015

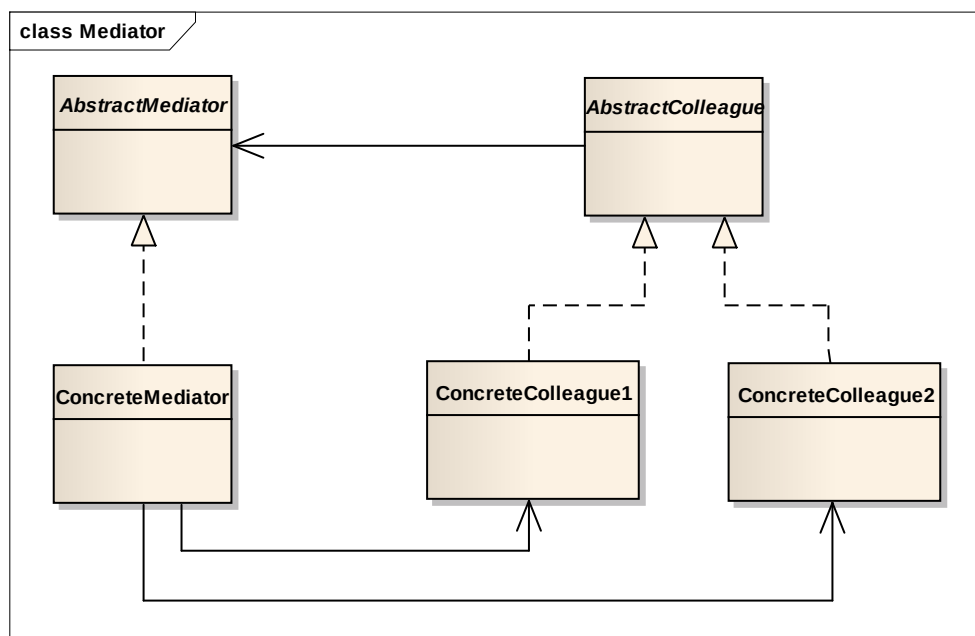
1 Mediator

Motto: Koordynator współpracy ściśle określonej grupy obiektów – dzięki niemu one nie odwołują się do siebie wprost (nie muszą nic o sobie wiedzieć), ale przesyłają sobie powiadomienia przez mediatora.

Kojarzyć: niby Observer bo też „powiadomienia”, ale nie – zbiór współpracujących obiektów jest tu ściśle określony. Mediator może więc wykorzystać ten fakt do wyboru różnych technik przesyłania powiadomień (bezpośrednio, na styl observera itp.).

Druga różnica między Mediatorem a Observerem jest taka że to kolaborujące obiekty przesyłają sobie powiadomienia o zmianie swojego stanu, a stan Mediatora nie ma nic do tego. W Observerze wszyscy zainteresowani nasłuchują powiadomień o zmianie stanu obiektu obserwowanego. Nie ma więc zupełnie analogii między mediatora a obserwowanym.

Przykład z życia: typowe okienka desktopowych technologii wytwarzania GUI są mediatorami między konkretnymi kontrolami, które są zagregowane wewnątrz (w środku okienka – Mediator, pomiędzy okienkami – Observer)



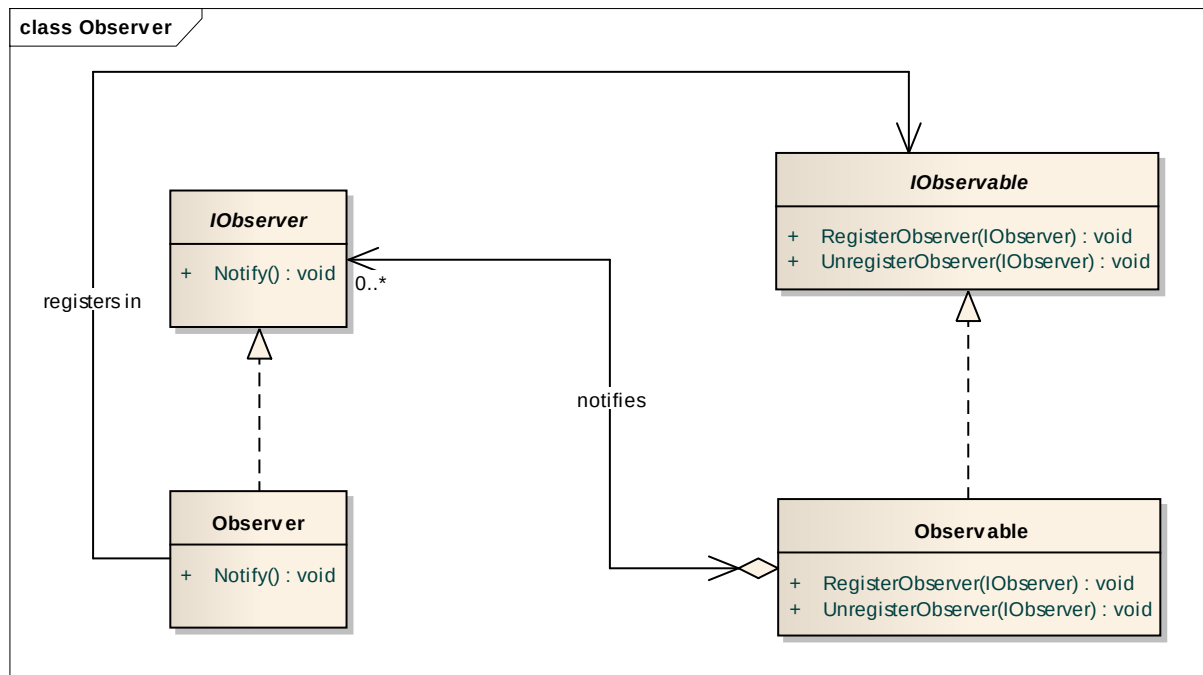
2 Observer

Motto: powiadamianie zainteresowanych o zmianie stanu, dzięki czemu nie odwołują się one do siebie wprost.

Kojarzyć: zdarzenia w C#

Przykład z życia: architektura aplikacji oparta o powiadomienia między różnymi widokami (w środku okienka – Mediator, pomiędzy okienkami – Observer)

Jeszcze inaczej – Observer ujednolica interfejs „Colleagues” Mediatora, dzięki czemu obsługuje dowolną liczbę „Colleagues”



Komentarz: kolejny wzorec który silnie wpływa na rozwój języków – C#-owe zdarzenia (events) to przykład uczynienia ze wzorca projektowego elementu języka.

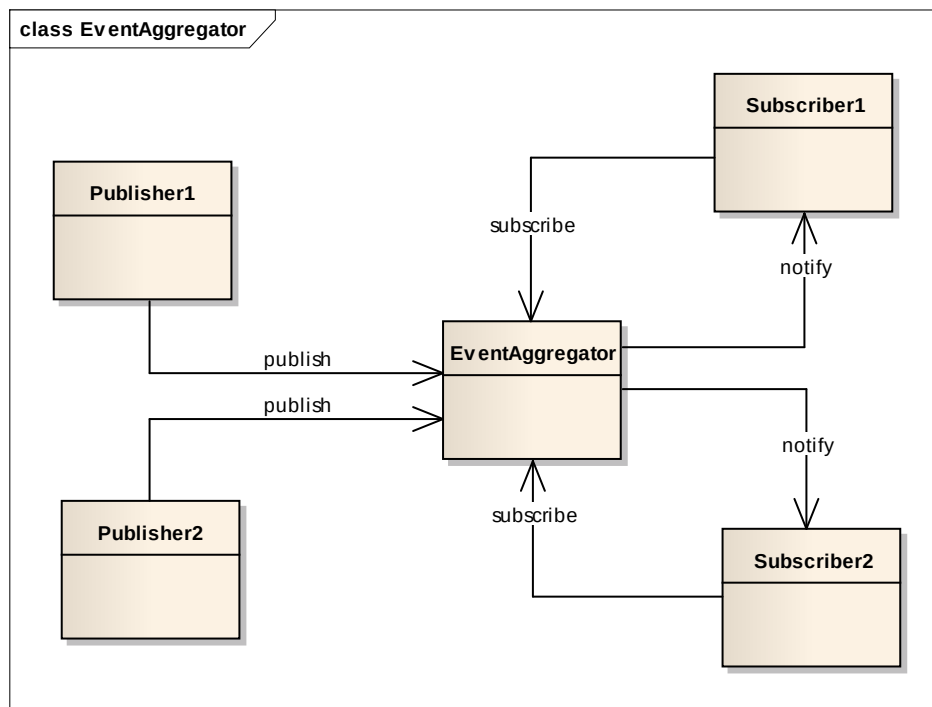
3 Event Aggregator

Motto: rozwiąż problem Observera ogólniej – jeden raz dla różnych typów powiadomień

Kojarzyć: ogólniejszy Observer, „hub” komunikacyjny (Observer zaimplementowany jako „słownik list” słuchaczy indeksowany typem powiadomienia)

Event Aggregator znosi najważniejsze ograniczenie Observera – klasy obserwatorów muszą tam znać klasę obserwowanego. W EventAggregatorze zarówno obserwowani jak i obserwujący muszą tylko wiedzieć gdzie szukać EventAggregatora. W efekcie klasy obserwowane i obserwujące mogą być zdefiniowane np. w niezależnych od siebie modułach (co jest niemożliwe w przypadku Observera).

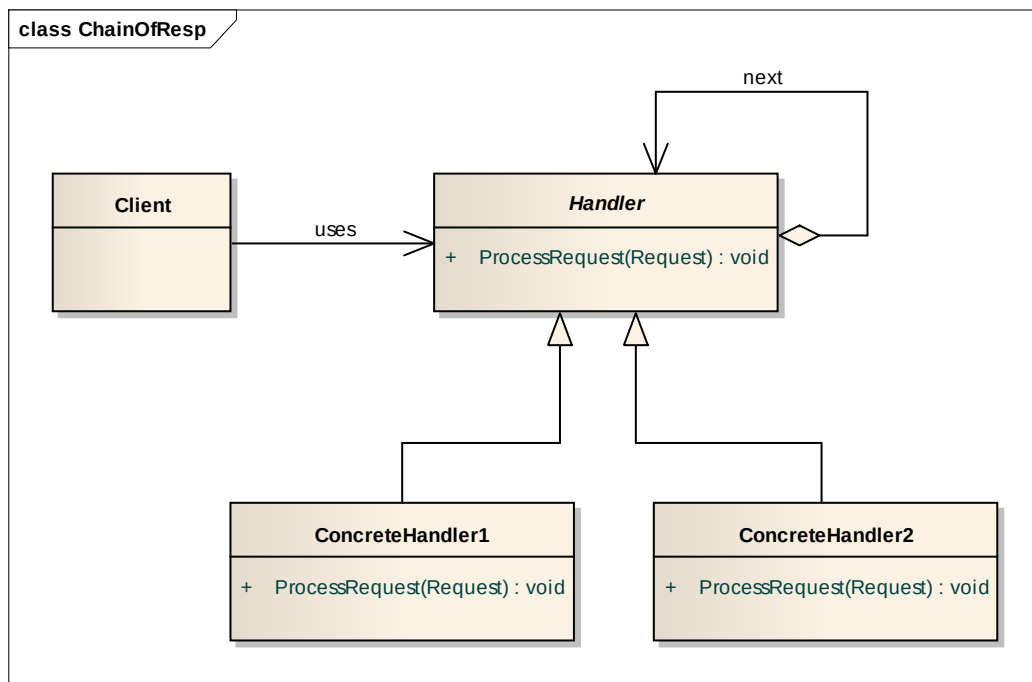
Uwaga: jeden z ważniejszych wzorców **dobrej architektury** aplikacji



4 Chain of Responsibility

Motto: uniknięcie związania obiektu wysyłającego żądanie z konkretnym odbiorcą w sytuacji gdy zbiór możliwych odbiorców jest dynamiczny

Kojarzyć: łańcuch niezależnych odbiorców wiadomości; przetwarzanie w skomplikowanej logice



5 Memento

Motto: Zapamiętuj i pozwalaj odzyskać stan obiektu

Uwaga: stan obiektu i stan pamiętki nie muszą być takie same. W szczególności duże obiekty mogą tworzyć małe, przyrostowe pamiętki

Kojarzyć z: Undo (i opcjonalnym Redo).

