Wzorce Projektowe

Janusz Jabłonowski

18 kwietnia 2007

Wzorce - dlaczego?

- Nie ma przymusu
- Sprawdzone
- Zgodne z podejściem obiektowym
- Słownik
- Uwaga: przykłady pochodzą z książki powstałej za granicą i to kilka lat temu, zatem wszelkie podobieństwa itd.

Strategia

- Na podstawie Head First Design Patterns (choć zmodyfikowane)
- Problem: gra *Symulator Kaczki*
- W grze występuje wiele różnych gatunków kaczek (dzika, krzyżówka, plaskonosa, ...)
- Przyjęte rozwiązanie:
  - nadklasa Kaczka (metody kwacz(), pływaj(), wyświetl())
  - podklasy DzikaKaczka, Krzyżówka, Płaskonos (metoda wyświetl)
- Działa ...
- ... do czasu

Modyfikacja nr 1 rozwiązanie nr 1

- Modyfikacje, podatki i śmierć
- Kaczki mają latać
- Przyjęte rozwiązanie:
  - w nadklasie Kaczka metoda leć()
- Działa ...
- ... aż za bardzo (w grze były też gumowe kaczki i nagle zaczęły latać ...)

Modyfikacja nr 1 rozwiązanie nr 2

- W klasie Kaczka metoda leć() lata, przedefiniowana w klasie KaczkaGumowa, tak by np. wypisywać stosowny komunikat dla gracza
- Działa ...
- ... ale pojawiają się inne nielatające kaczki (drewniana KaczaWabik np.) i trzeba powielać kod

Modyfikacja nr 1 rozwiązanie nr 3

- Podklasy klasy Kaczka: KaczkaLatająca i KaczkaNielatająca i stosowne rozdzialenie istniejących kaczek
- Działa ...
- ... do czasu

Modyfikacja nr 2 rozwiązanie nr 3

- Kaczki mają kwakać lub nie, latać lub nie stosownie do swojego rodzaju
- Rozwiązanie nr 3 przestaje działać, gdy pojawiają się kolejne cechy wspólne dla grup kaczek, kombinatoryczna ekslozja liczby klas, no i potrzebujemy wielodziedziczenia ...

Modyfikacja nr 2 rozwiązanie nr 4

- Tworzymy interfejsy:
  - KaczkiLatające, z metodą leć()
  - KaczkiKwaczące, z metodą kwacz()
- Kaczki implementują (lub nie) odpowiednie interfejsy
- Działa ...
- ... ale jest chyba najgorszym z dotychczasowych, bo wymaga powielania kodu (kwakanie, latanie) na ogromną skalę.
- Poza tym nie zadziała, gdy jednak wrócimy do wymagania, żeby nielatające kaczki sygnalizowały to stosownym komunikatem przy próbie zmuszenia do lotu (analogicznie z kwakaniem)

Modyfikacja nr 2 czarna rozpacz

- Co więc zrobić ???

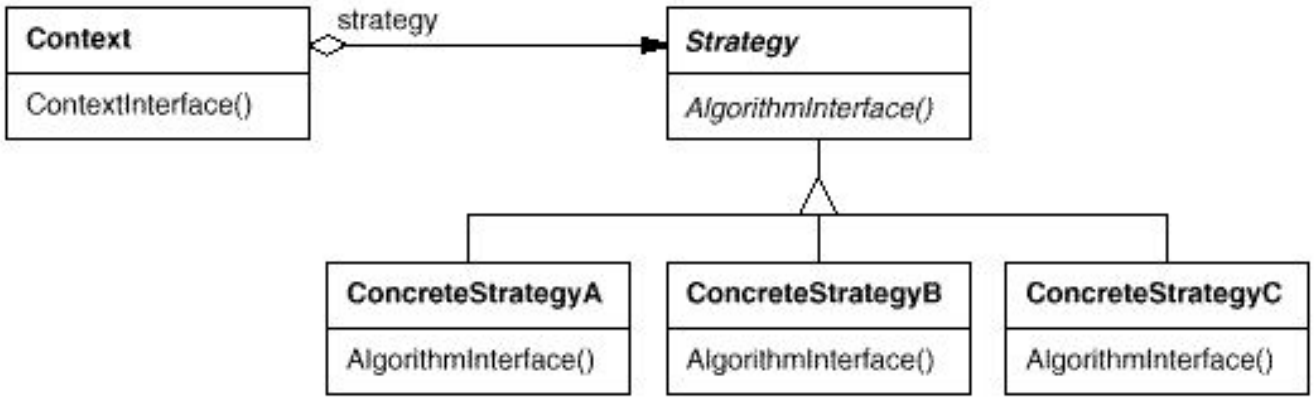
Stosujemy wzorzec strategia

- To co się zmienia wyciągamy z obiektu
- Hierarchię dziedziczenia w naszych klasach częściowo zastępujemy składaniem
- Dla nowych klas tworzymy ich hierarchię dziedziczenia (po interfejsie)

Rozwiązanie

- Wprowadzamy interfejsy ILatanie i IKwakanie
- Tworzymy klasy implementujące te interfejsy na wszystkie potrzebne nam sposoby
- W klasie Kaczka wprowadzamy atrybuty typu ILatanie i IKwakanie
- W konstruktorze klasy Kaczka ustalamy wartości tych atrybutów na podstawie parametrów podawanych w konstruktorach podklas
- Wywołania metod leć() i kwacz() realizujemy jako delegowanie żądania do atrybutu

Strategia w całej okazałości



Obserwator

- Problem: mamy obiekt zmieniający swój stan i chcemy mieć obiekty śledzące te zmiany
- Obiekt obserwowany nie powinien uzależniać swojego zachowania od liczby (czy w ogóle istnienia) obserwujących go obiektów
- luźne związki

Obserwator w całej okazałości