Wzorce projektowe, część II

Joanna Wnęk

22-10-2015

„Wzorzec opisuje problem, który powtarza się wielokrotnie w danym środowisku, oraz podaje istotę jego rozwiązania w taki sposób, aby można było je zastosować miliony razy bez potrzeby powtarzania tej samej pracy”

(Christopher Alexander *A pattern language*, 1977)

Wzorce w inżynierii oprogramowania:

- *wzorce architektoniczne – poziom integracji komponentów*
- ***wzorce projektowe** – poziom interakcji między klasami*
- *wzorce analityczne – poziom opisu rzeczywistości*
- *wzorce implementacyjne – poziom języka programowania*

Wzorzec projektowy identyfikuje i opisuje pewną abstrakcję, której poziom znajduje się powyżej poziomu abstrakcji pojedynczej klasy, instancji lub komponentu.

E. Gamma, R. Johnson, R. Helm, J. Vlissides, 1994

Systematyka wzorców projektowych:

- Wzorce kreacyjne (konstrukcyjne)
  - abstrakcyjne metody tworzenia obiektów
  - uniezależnienie systemu od sposobu tworzenia obiektów
- Wzorce strukturalne
  - sposób wiązania obiektów w struktury
  - właściwe wykorzystanie dziedziczenia i kompozycji
- Wzorce behawioralne
  - algorytmy i przydział odpowiedzialności
  - opis przepływu kontroli i interakcji

Wzorzec projektowy jest opisany przez:

- nazwę
- klasyfikację
- cel
- aliasy
- motywację
- zastosowania
- strukturę
- uczestników
- współdziałania
- konsekwencje
- implementację
- przykład
- pokrewne wzorce

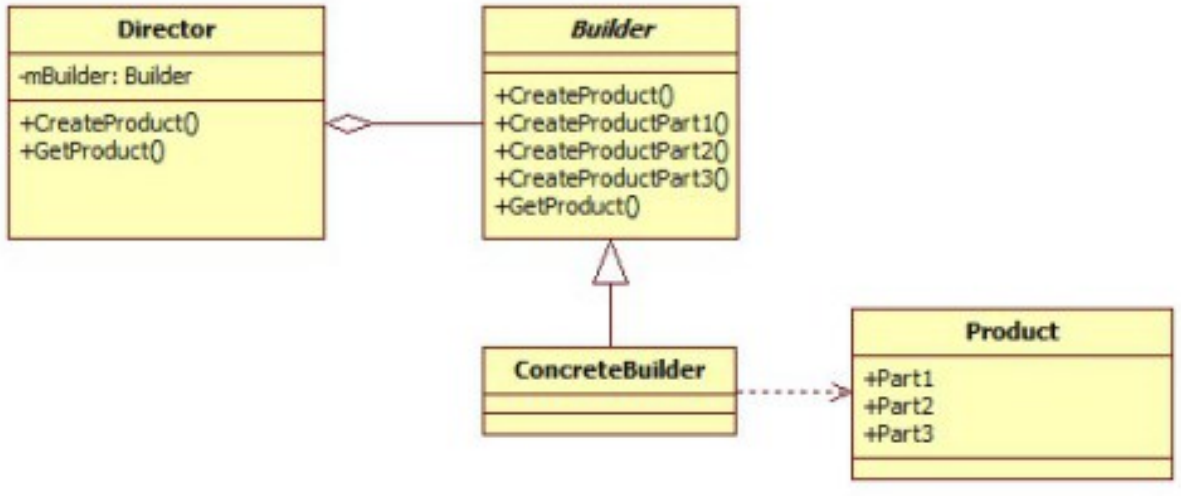
Przedstawione w części I:

- Singleton - kreacyjny, obiektowy
- Dekorator - strukturalny, obiektowy
- Obserwator - behawioralny, obiektowy
- Strategia - behawioralny, obiektowy
- Łańcuch Zobowiązań - behawioralny, obiektowy

Budowniczy (ang. builder)

Builder jest wzorcem strukturalnym i służy do tworzenia złożonych struktur obiektowych. Jego celem jest oddzielenie sposobu reprezentacji tych struktur od mechanizmu ich konstrukcji. Pozwala to także wykorzystać te same mechanizmy konstrukcyjne do tworzenia różnych struktur.
Builder wykorzystujemy w celu konstrukcji skomplikowanych obiektów z zachowaniem kolejności tworzenia ich składników.

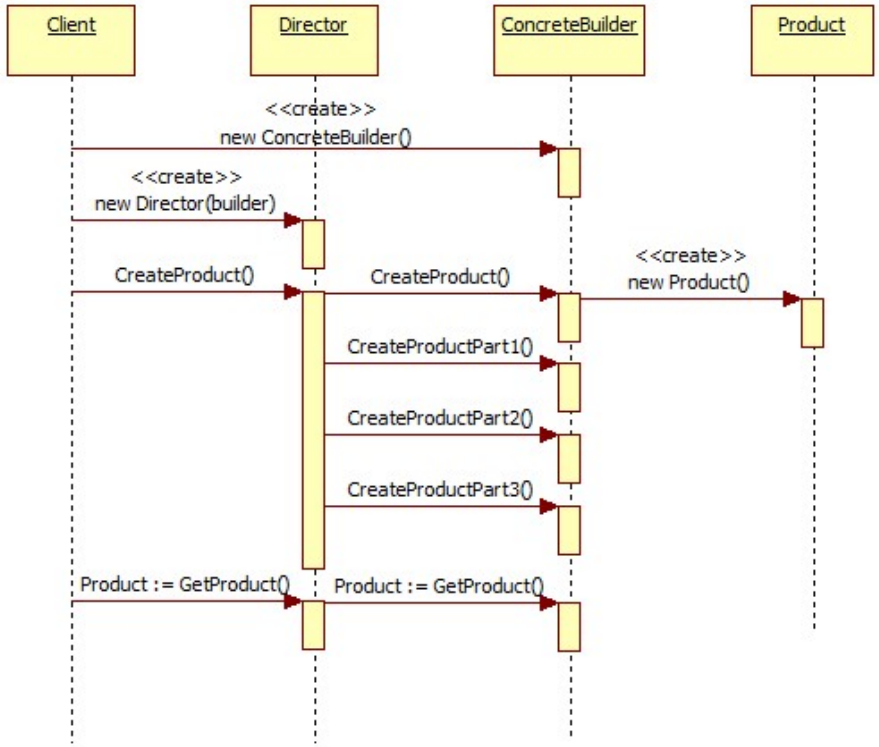
Podstawowym elementem mechanizmu jest interfejs Budowniczy udostępniający metody budowy składowych końcowego produktu. Dziedziczą po nim Konkretni budowniczowie decydujący o tym jak dokładnie dany produkt będzie tworzony. Całość procesu kontroluje Nadzorca uruchamiający metody budowniczego w odpowiedniej kolejności.



Rysunek : Diagram klas

Kolejne kroki scenariusza budowania produktu:

1. Tworzymy instancję Konkretnego budowniczego odpowiedniego do oczekiwanego wariantu produktu.
2. Tworzymy instancję Nadzorcy i przekazujemy do niego referencję budowniczego.
3. Uruchamiamy w Nadzorcy metodę konstrukcji produktu, która zapewni użycie odpowiedniej sekwencji metod Budowniczego.
4. Po stworzeniu produktu pobieramy go od Nadzorcy.



Rysunek : Diagram sekwencji.

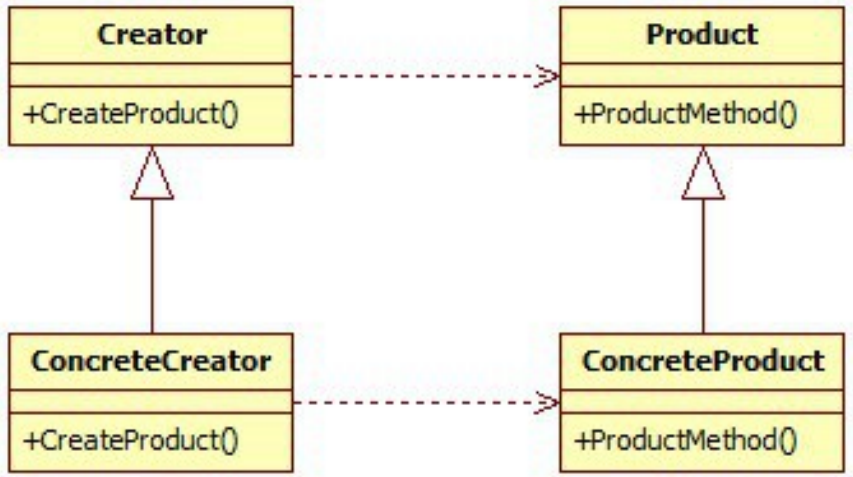
Kod źródłowy.

Bulider - konsekwencje

1. Zmiana implementacji obiektów Product nie wpływa na proces konstrukcji struktury
2. Odseparowanie reprezentacji i konstrukcji struktur obiektowych
3. Precyzyjna kontrola nad procesem konstrukcji struktury
4. Ułatwione testowanie elementów struktury

Metoda wytwórcza (ang. factory method)

Wzorzec Factory Method jest podstawowym wzorcem kreacyjnym. Jego celem jest zastąpienie prostych wywołań konstruktora dedykowanym interfejsem (metodą), która przejmie odpowiedzialność za tworzenie i ew. inicjację obiektu danej klasy. Podobnie jak w przypadku wzorca Singleton (który jest specjalizowaną wersją Factory Method, ograniczoną do tworzenia jednego obiektu), istnieje możliwość hermetyzacji wewnątrz tej metody sposobu wyboru klasy obiektu spośród jej podklas oraz użytego konstruktora.



Rysunek : Diagram klas.

Kod źródłowy.

Klient odwołuje się do dwóch interfejsów (lub klas abstrakcyjnych): Creator, zawierającej metodę tworzącą produkty, i Product, reprezentującą obiekty tworzone przez Factory Method. Oba interfejsy posiadają implementacje powiązane parami: obiekt klasy ConcreteCreator o przeciążonej metodzie factoryMethod() tworzy instancję obiektu ConcreteProduct. Dzięki temu, podczas zmiany obiektu Creator, jednocześnie zmieniany jest tworzony produkt. Warto zwrócić na dualizm hierarchii klas produktu i kreatora, który pozwala na abstrakcyjne traktowanie całego procesu tworzenia obiektów za pomocą wymiennych producentów, co jest niemożliwe przy bezpośrednim użyciu konstruktora.

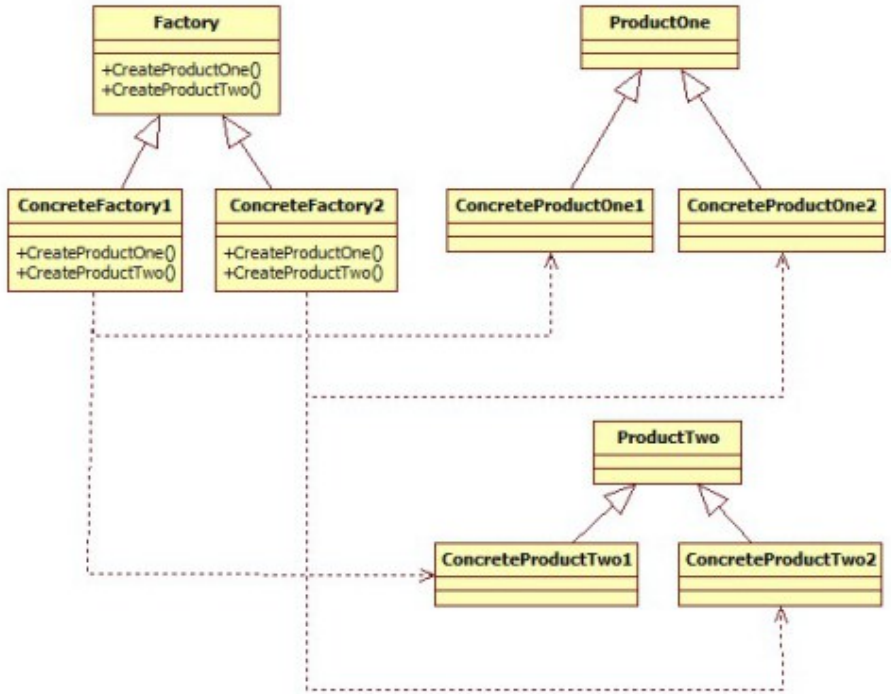
Z punktu widzenia klienta metoda factoryMethod() jest równoważna pod względem funkcjonalnym z konstruktorem: jej wywołanie powoduje utworzenie obiektu żądanego typu. Warto zwrócić uwagę, że wzorzec ten pozwala także na dodatkowy stopień swobody: bezpośrednie wywołanie konstruktora przez klienta zawsze powoduje utworzenie obiektu konkretnej klasy (konstruktory nie są metodami polimorficznymi), natomiast użycie wzorca Factory Method pozwala metodzie tworzącej obiekty na wybór klasy obiektu i sposobu jego tworzenia.

Factory Method - konsekwencje

1. Przeniesienie odpowiedzialności za tworzenie obiektów Product z klienta na obiekt Creator
2. Możliwość rozszerzania hierarchii klas Product niezależnie od klienta

Fabryka abstrakcji (ang. abstract factory)

Wzorzec Abstract Factory jest rozszerzeniem koncepcji znanej z Factory Method na całą rodzinę produktów, której zmiana na inną powinna odbywać się w postaci jednego kroku. Celem wzorca jest zdefiniowanie interfejsu do tworzenia takich rodzin obiektów, korzystając (podobnie jak w przypadku poprzedniego wzorca) z dedykowanych klas zajmujących się produkcją obiektów.



Rysunek : Diagram klas.

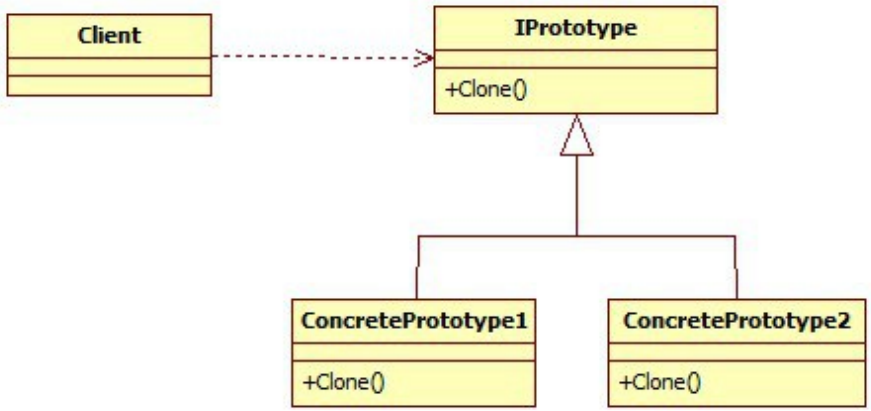
Kod źródłowy.

Abstract Factory - konsekwencje

1. Łatwa zmiana całych grup produktów poprzez zmianę używanej Concrete Factory
2. Wydzielenie interfejsu do tworzenia obiektów
3. Odseparowanie klienta od szczegółów implementacji obiektów Product
4. Utrudnione dodawanie kolejnych obiektów Product we wszystkich grupach

Prototyp (ang. prototype)

Wzorzec Prototype należy do grupy wzorców kreacyjnych, jednak sposób tworzenia przez niego obiektów jest zupełnie inny niż w przypadku innych rozwiązań z tej grupy, np. Factory Method czy Singletona. Celem jego stosowania jest tworzenie nowych obiektów poprzez klonowanie już istniejącego wzorcowego obiektu.



Rysunek : Diagram klas.

Kod źródłowy.

Obiekt poddający się klonowaniu, Prototype, posiada metodę clone(). Metoda ta jest implementowana we wszystkich jego obiektach potomnych w ten sposób, że tworzy ona dokładną kopię bieżącego obiektu. Jedyna różnica pomiędzy oryginałem i klonem polega na odrębnej tożsamości obiektu (w większości języków tożsamość jest rozstrzygana na podstawie referencji do tego obiektu). Klient, żądając utworzenia kopii obiektu Prototype, wywołuje w istniejącej instancji tego obiektu metodę clone(),która zwraca jego klon.

Prototyp - konsekwencje

1. Możliwość tworzenia obiektów poprzez przykład
2. Uproszczona konstrukcja podobnych obiektów
   - pominięcie wyboru konstruktora
   - ograniczenie liczby podklas w systemie

 M Grand, B. Merrill “Visual Basic .NET - Wzorce projektowe”. 2006.

 www.algorytm.org/wzorce-projektowe/

 www.dotnetpage.wordpress.com

 www.wazniak.mimuw.edu.pl

Dziękuję!