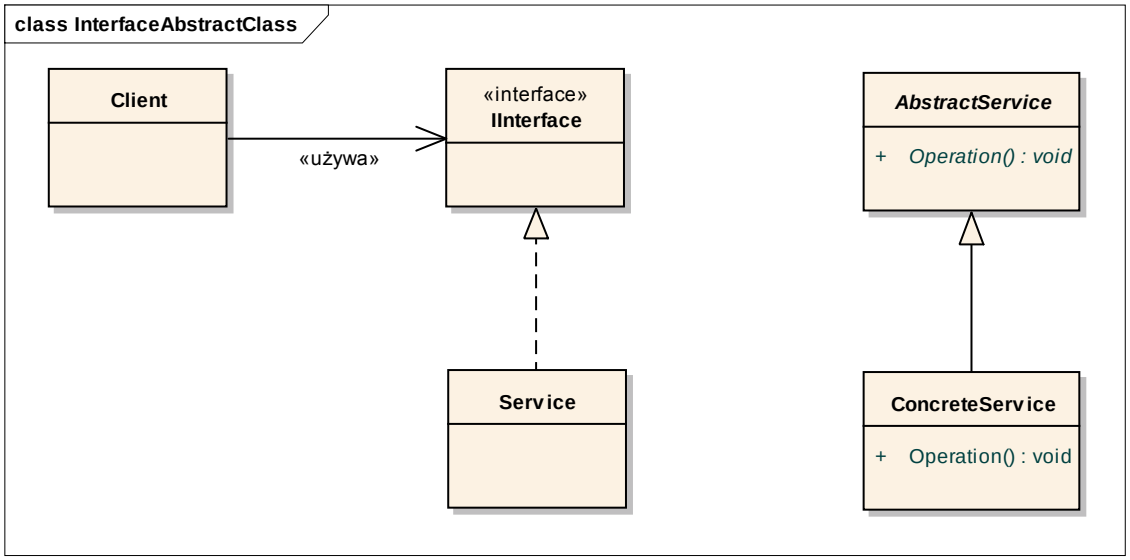
Projektowanie obiektowe oprogramowania
Wykład 4 – wzorce projektowe
cz.I. wzorce podstawowe i kreacyjne
Wiktor Zychla 2016

1 Wzorce podstawowe

1.1 Interface vs Abstract class



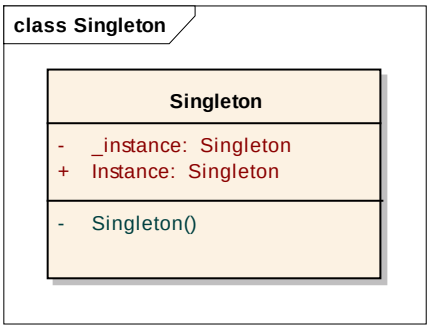
- Klasa abstrakcyjna może zawierać implementacje, interfejs nie
- Klasa może dziedziczyć tylko z jednej klasy abstrakcyjnej iwielu interfejsów
- Przykłady IEnumerable vs Stream

1.2 Delegation (Prefer Delegation over Inheritance)

- Dziedziczenie jest relacją statyczną, delegacja może być dynamiczna
- Delegujący obiekt może ukrywać metody delegowanego, co jest niemożliwe w przypadku dziedziczenia
- klasa domeny nie dziedziczymy z klas użytkowych (Person nie dziedziczy z Hashtable), ale delegacja jest ok.
- delegacja powoduje że jest więcej kodu

2 Wzorce kreacyjne

2.1 Singleton



- Jedna i ta sama instancja obiektu dla wszystkich klientów
- Często punkt wyjścia dla innych elementów architektury aplikacji

Zalety:

- Uniwersalność
- „Leniwa” konstrukcja

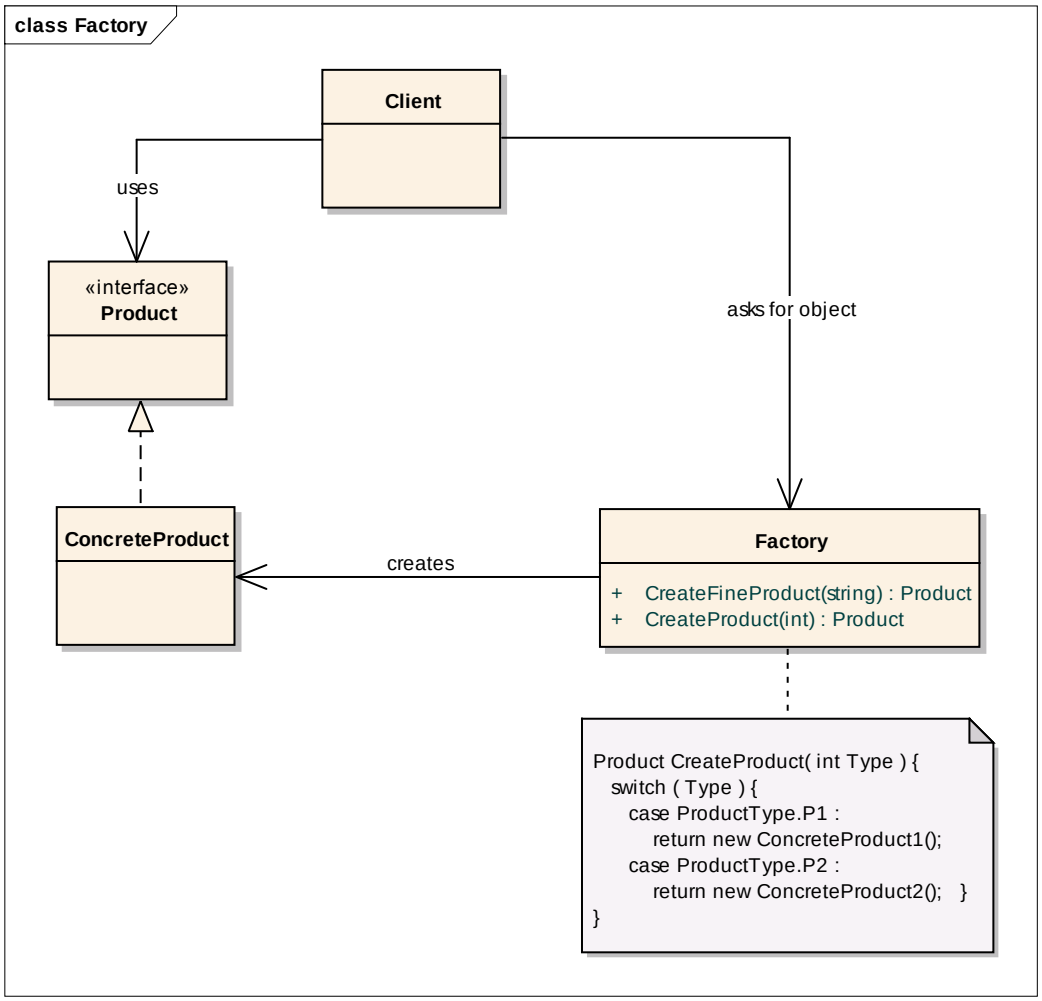
Rozszerzenia:

- Możliwość sterowania czasem życia obiektu „wspierającego” (pseudosingleton, singleton z określoną polityką czasu życia)
- Singleton z parametrami

2.2 Monostate

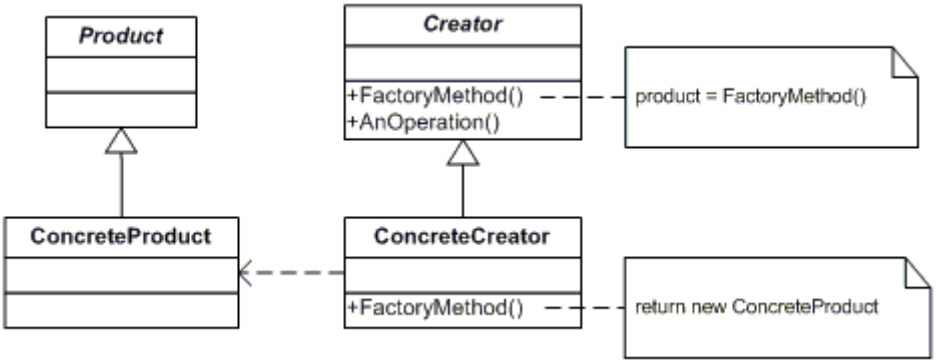
- Usuwa ograniczenie liczby instancji w Singletonie, pozostawia właściwość współdzielenia stanu

2.3 (Parametrized) Factory



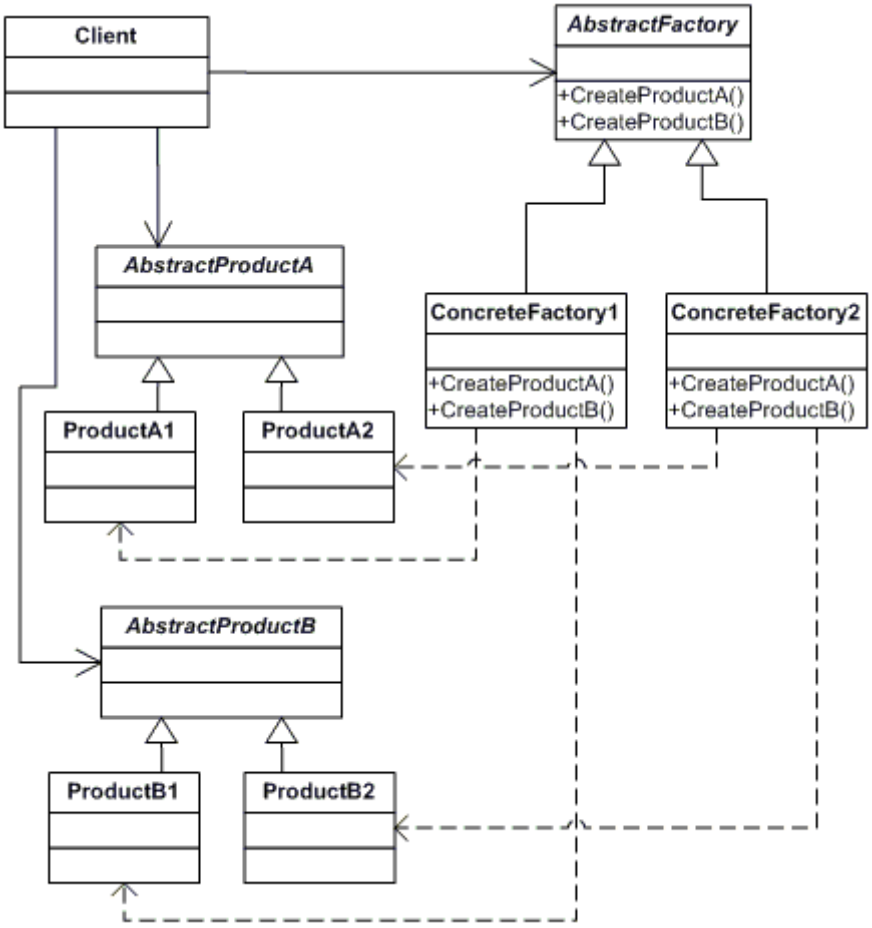
- To jeden z częściej stosowanych wzorców, realizacja odpowiedzialności Creator z GRASP
- Interfejs klasy fabryki może mieć wiele metod, ułatwiających tworzenie konkretnych obiektów (parametryzacja przez typ metody fabryki, przez wiele metod jednej fabryki); fabryka może też zwracać obiekt typu pochodnego względem oczekiwanego, w ten sposób być przygotowana na zmiany funkcjonalności – klient spodziewa się obiektu typu A, dostaje B dziedziczące z A i korzysta z niego jak z A, ale w rzeczywistości B realizuje swoją odpowiedzialność być może inaczej niż A
- Fabryka może kontrolować czas życia tworzonych obiektów (zwracając obiekty o różnych czasach życia)
- Fabryka może być przygotowana na rozszerzenia, w ten sposób realizując postulat OCP (przykład z wykładu: Factory+FactoryWorker)
- W praktyce – zamiast z singletonów i monostates lepiej używać fabryki, jest dużo bardziej uniwersalna

2.4 Factory Method



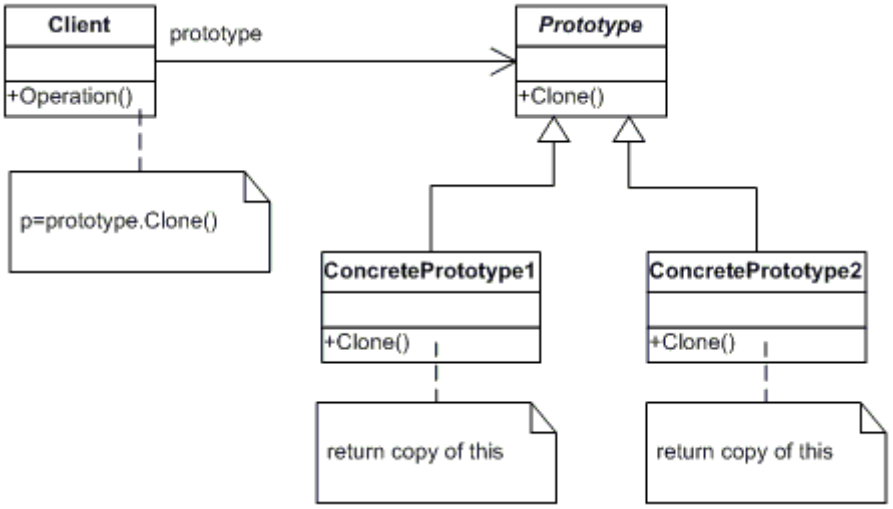
- Rozszerzenie wzorca Factory
- Deleguje utworzenie obiektu użytkowego do metody tworzącej
- Podklasy specyfikują konkretną funkcjonalność użytkową

2.5 Abstract Factory



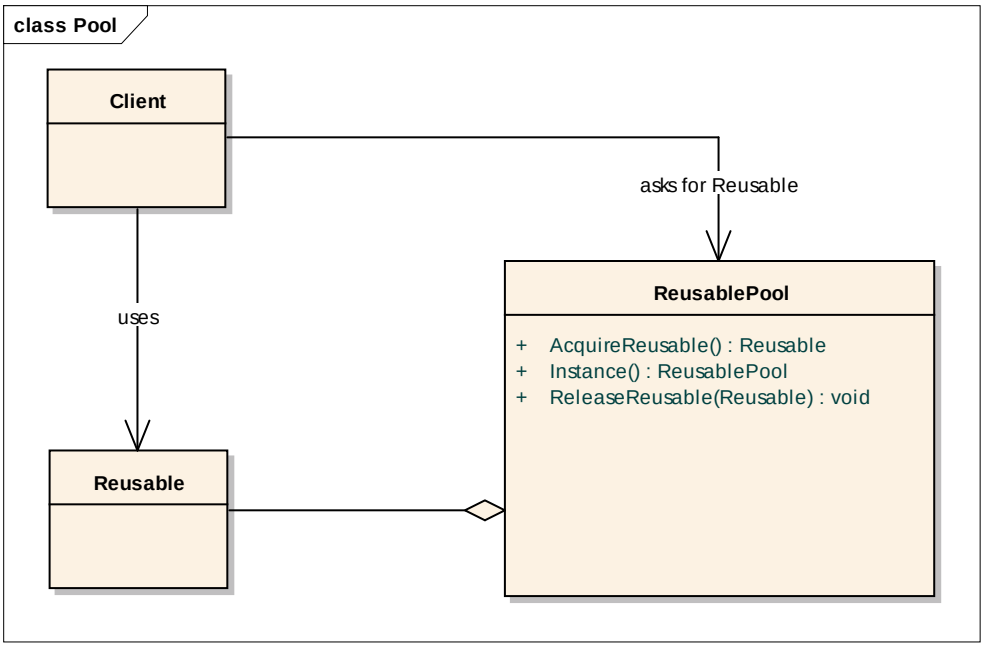
- Nazywany też „tookit”
- Abstrakcyjna fabryka całej rodziny obiektów (może być opisana przez interfejs)

2.6 Prototype



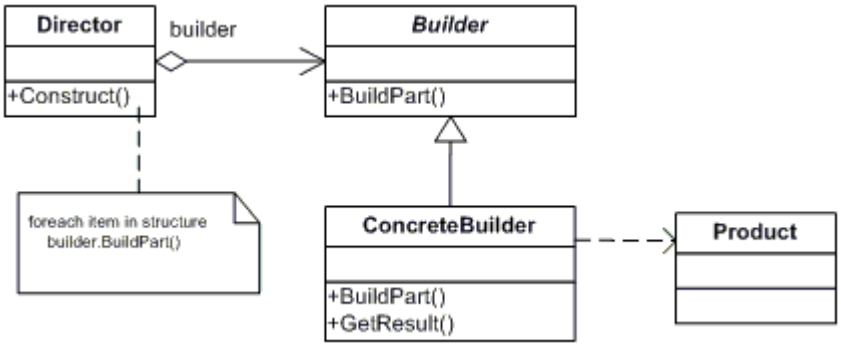
- Istnieje kilka prototypowych instancji obiektów
- Tworzenie nowych polega na kopiowaniu prototypów
- Nie ma znaczenia kto i jak wyprodukował instancje prototypów

2.7 Object Pool



- Reużywanie / współdzielenie obiektów które są kłopotliwe w tworzeniu (np. czasochłonne)
- Metoda tworzenia/pobierania obiektu bywa parametryzowana

2.8 Builder



- Ukrywanie szczegółów kodu służącego do kreowania obiektu/obiektów
- Ukrywanie wewnętrznej struktury obiektu
- Przykład – **XmlTextWriter**
- Przykład z wykładu: http://www.wiktorzychla.com/2012/02/simple-fluent-and-recursive-tag-builder.html