Paradygmaty Programowania

Wzorce projektowe

dr inż. Marcin Bączyk

Wykład 7

11 kwietnia 2022

Treść dzisiejszego wykładu

Wzorce projektowe

- Wzorce behawioralne:
  - strategia,
  - polecenie.
- Wzorce strukturalne:
  - kompozyt,
  - dekorator.
- Wzorce kreacyjne:
  - budowniczy,
  - fabryka abstrakcyjna.

Literatura

- Eric Gamma, Richard Helm, Ralph Johnson, John Vlissides: “Wzorce projektowe”, Helion, 2010
- Robert C. Martin: “Zwinne wytwarzanie oprogramowania. Najlepsze zasady, wzorce i praktyki”
- https://refactoring.guru/pl/design-patterns
- https://sourcemaking.com/design_patterns
- https://www.oodesign.com

Wzorce projektowe – wprowadzenie

Wprowadzenie

W programowaniu (zwłaszcza) obiektowym istnieją klasy problemów, z którymi programiści (projektanci) spotykają się stosunkowo często. Ze względu na ich powtarzalność opracowano szereg standardowych mechanizmów pozwalających na poprawne oraz stosunkowo szybkie rozwiązanie danego problemu.

Wzorzec projektowy

„Wzorzec projektowy jest sprawdzonym w praktyce rozwiązaniem często pojawiających się problemów i powtarzalnych problemów projektowych. Pokazuje powiązania i zależności pomiędzy klasami oraz obiektami i ułatwia tworzenie modyfikację oraz utrzymanie kodu źródłowego. Jest opisem rozwiązania."

Elementy wzorca projektowego

- Nazwa – jednoznacznie identyfikuje konkretne rozwiązanie. W kodzie często pojawiają się nazwy klas odnoszące się do nazw wykorzystanych w dokumentacji wzorca, np. interfejs **ThreadFactory** w bibliotece standardowej Javy wykorzystuje wzorzec Fabryka Abstrakcyjna.
- Opis problemu – określa, kiedy stosować dany wzorzec. Wyjaśnia, o co dokładnie chodzi w danym wzorcu.
- Opis rozwiązania – określa elementy składające się na projekt, ich związki, zobowiązania i współpracę.
- Konsekwencje stosowania – efekty oraz wady i zalety użycia danego wzorca.

Wzorce projektowe – wprowadzenie

Podział wzorców projektowych ze względu na rodzaj

- **Kreacyjne** – związane są z procesem tworzenia obiektów.
- **Strukturalne** – opisują sposób składania klas lub obiektów w większą całość, realizującą pewną specyficzną funkcję.
- **Czynnościowe (behawioralne)** – definiują sposób, w jaki klasy lub obiekty oddziałują na siebie i przekazują między sobą zobowiązania (odpowiedzialność).

Podział wzorców projektowych ze względu na zakres

- Klasowe – zajmują się związkami między klasami i ich podklasami. Są ustalane statycznie.
- Obiektowe – zajmują się związkami między obiektami. Są ustalane dynamicznie.

Wzorce projektowe – wprowadzenie

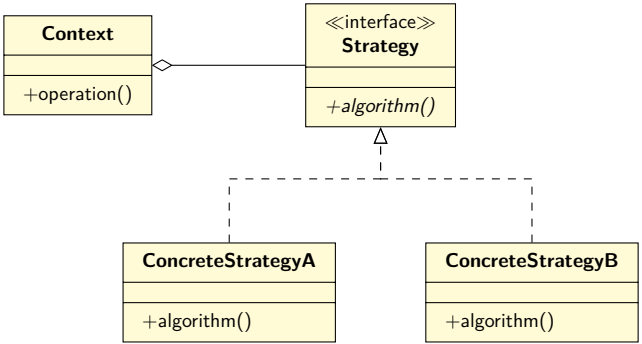
		Rodzaj		
		kreacyjne	strukturalne	czynnościowe
Zakres	klasy	metoda wytwórcza	adapter	interpreter metoda szablonowa
	obiekty	**budowniczy** **fabryka abstrakcyjna** prototyp singleton	adapter **dekorator** fasada **kompozyt** most pełnomocnik pyłek	iterator łańcuch zobowiązań mediator *obserwator* odwiedzający pamiątka **polecenie** stan **strategia**

Wzorce projektowe – wprowadzenie

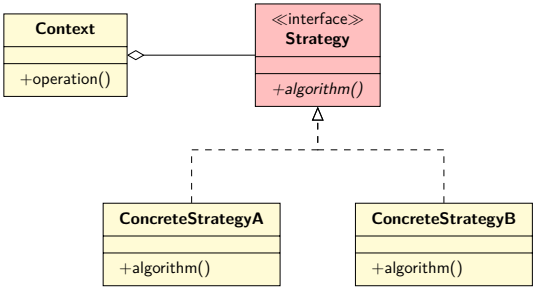
Wzorce projektowe:

- stanowią abstrakcyjny opis pomiędzy klasami,
- przyczyniają się do standaryzacji kodu,
- pozwalają lepiej komunikować programistom ich zamiary,
- zwiększają czytelność i niezawodność kodu,
- przyspieszają proces rozwoju oprogramowania,
- mogą być łączone – np. fabryka abstrakcyjna może być zaimplementowana jako singleton.

Wzorzec projektowy «strategia»



Wzorzec projektowy «strategia»

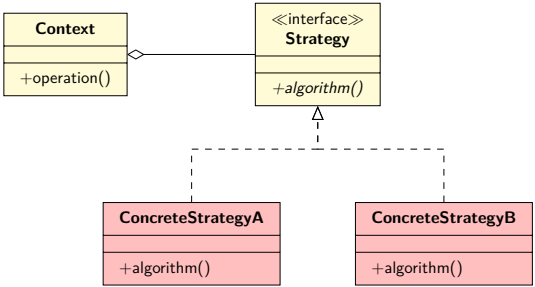


Strategia

Deklaruje interfejs wspólny dla wszystkich obsługiwanych algorytmów.

Strategia – może być zaimplementowana jako interfejs lub klasa abstrakcyjna. Definiuje interfejs metody, która musi być zdefiniowana w poszczególnych **Strategiach Konkretnych**. **Kontekst** wykorzystuje ten interfejs do wykonywania algorytmu zdefiniowanego przez **Konkretną Strategię**.

Wzorzec projektowy «strategia»

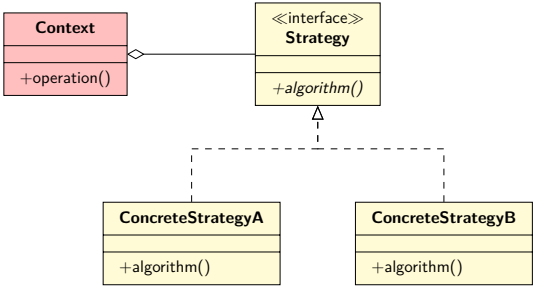


Strategia Konkretna

Implementuje algorytm, wykorzystując interfejs z klasy **Strategia**.

Kontekst zawsze wykorzystuje obiekt klasy **Konkretna Strategia**.

Wzorzec projektowy «strategia»



Kontekst

- Jest konfigurowany za pomocą obiektu klasy **Strategia Konkretna**.
- Utrzymuje odwołanie do obiektu typu **Strategia**.
- Może zdefiniować interfejs umożliwiający **Strategii** uzyskanie dostępu do jego danych.

W niektórych przypadkach **Kontekst** może zarządzać / być konfigurowanym wieloma obiektami różnych klas **Konkretna Strategia**.

Wzorzec projektowy «strategia»

Obiekty klas **Strategia** i **Kontekst** współdziałają w celu zaimplementowania wybranego algorytmu. Dane niezbędne do wykonania algorytmu mogą być przekazane w trakcie wywołania lub później poprzez referencję do obiektu **Kontekstu**.

Kontekst przekazuje żądania od swoich klientów do konkretnej(-ch) strategii. Klienci mogą bezpośrednio konfigurować obiekt **Kontekstu** lub posługiwać się obiektem skonfigurowanym przez innych aktorów/klientów.

Wzorzec projektowy «strategia»

Obszar zastosowań

Stosuj wzorzec strategii, gdy:

- wiele powiązanych ze sobą klas różni się jedynie zachowaniem,
- potrzebne są różne warianty jakiegoś algorytmu,
- w algorytmie są używane dane, o których klienci nie powinni wiedzieć,
- klasa definiuje wiele zachowań, które w operacjach są uwzględnione w postaci wielokrotnych instrukcji warunkowych.

Wzorzec projektowy «strategia» – przykład zastosowania

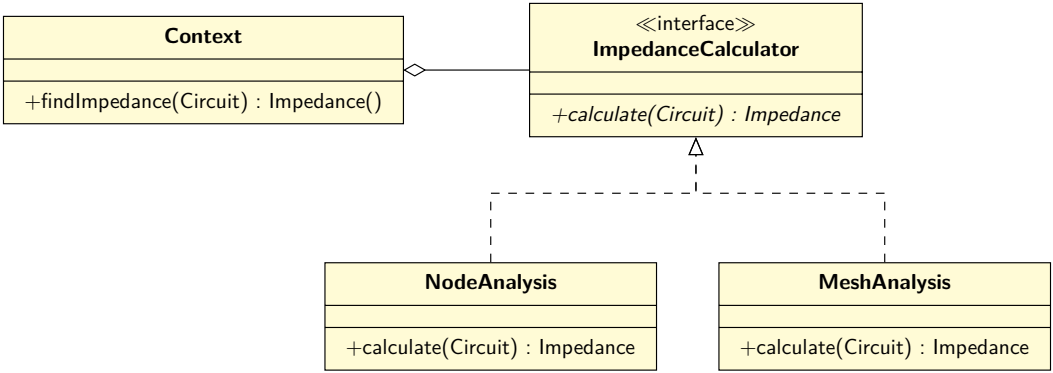
Przykładowy problem

Dany jest symulator układów analogowych wraz ze wszystkimi klasami niezbędnymi do reprezentacji poszczególnych elementów dyskretnych.
Cechy symulatora:

- Dla danego układu/podukładu istnieje możliwość wyznaczenia impedancji zastępczej.
- Istnieje możliwość wyboru metody użytej do wyznaczenia impedancji zastępczej.

Należy zaprojektować mechanizm, który pozwoli na przełączenie metody wyznaczania impedancji zastępczej danego układu bez wpływu na swoich klientów.

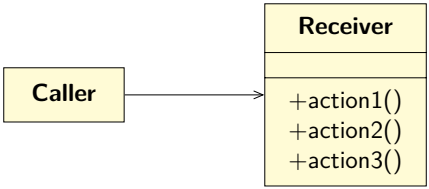
Wzorzec projektowy «strategia» – przykład zastosowania



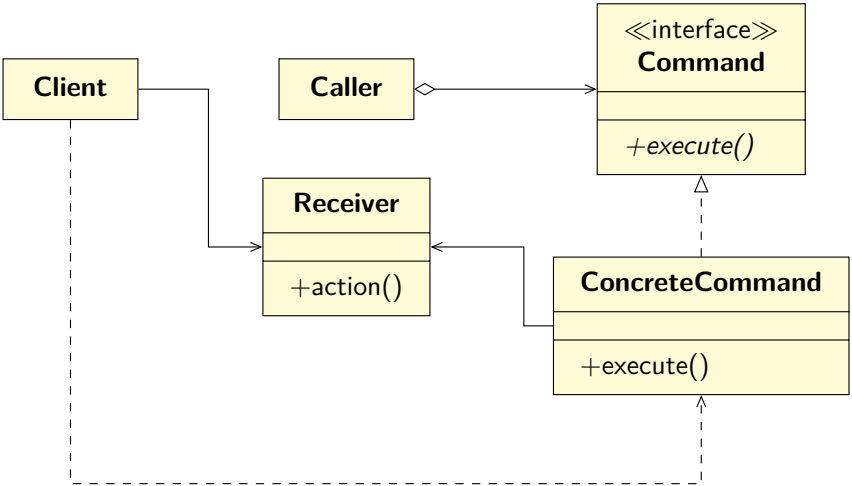
Wzorzec projektowy «polecenie» – wstęp

Bez wzorca «polecenie»

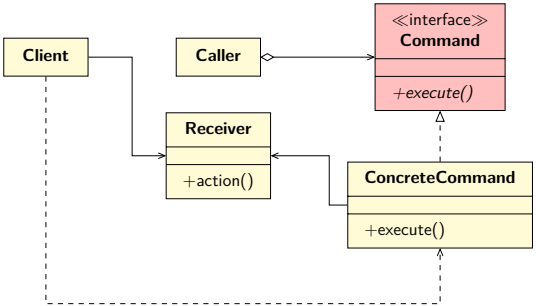
- Najprostszy sposób interakcji między dwiema klasami: zlecającą wykonanie czynności i wykonującą ją.
- Problemy:
  - jak zapamiętywać polecenia,
  - jak je cofać, powtarzać itp.
- Rozwiązanie: samo polecenie powinno być obiektem.



Wzorzec projektowy «polecenie»



Wzorzec projektowy «polecenie»

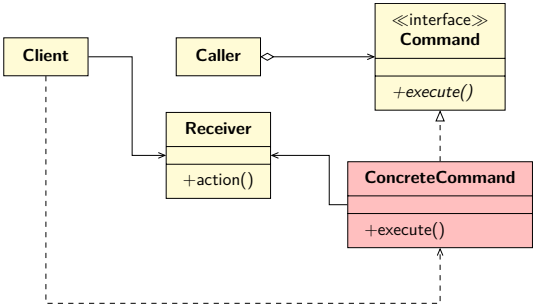


Polecenie

Definiuje interfejs do wykonywania operacji.

Polecenie może deklarować jedynie metodę do jednorazowego wykonania działania. Jeżeli to możliwe i uzasadnione, **Polecenie** może również zawierać deklarację metody do cofania zmian oraz ich ponownego wykonywnia.

Wzorzec projektowy «polecenie»

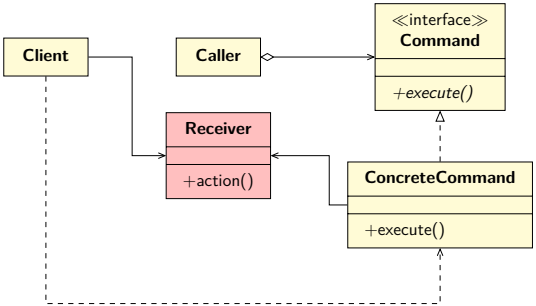


KonkretnePolecenie

- Definiuje powiązanie między **Odbiorcą** i akcją.
- Implementuje operację *wykonaj* (*execute*), wykorzystując odpowiednie operacje **Odbiorcy**.

KonkretnePolecenie implementuje konkretne działanie, które może być odłożone w czasie (nawet jeżeli ten czas jest bardzo krótki). Zazwyczaj, jeżeli w systemie implementowany jest ten wzorzec, to istnieje bardzo dużo różnych implementacji **KonkretnychPoleceń**.

Wzorzec projektowy «polecenie»

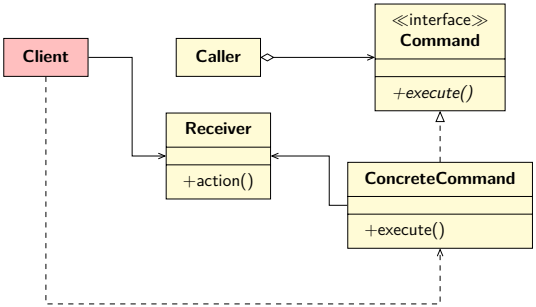


Odbiorca

Wie, jak wykonać konkretne operacje związane ze spełnieniem żądania.

Odbiorca najczęściej jest obiektem, który ulega zmianie w związku z wykonywanym **KonkretnymPoleceniem**. **Odbiorca** implementuje całą funkcjonalność związaną ze spełnieniem żądania oraz (jeśli to możliwe i uzasadnione) jego cofnięciem.

Wzorzec projektowy «polecenie»

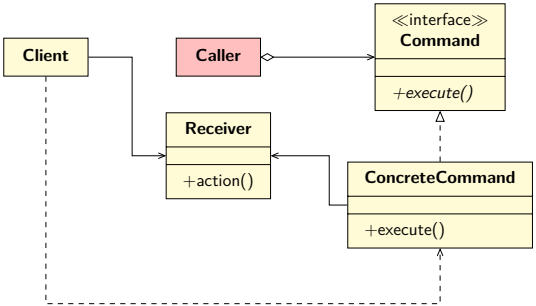


Klient

- Tworzy **KonkretnePolecenie** i ustala jego **Odbiorcę**.

Klient ustala, na którym obiekcie będzie wykonywane polecenie, przekazując referencję do **Odbiorcy KonkretnemuPoleceniu**. Utworzone **KonkretnePolecenie Klient** przekazuje **Wywołującemu**.

Wzorzec projektowy «polecenie»



Wywołujący

Prosi **Polecenie** o spełnienie żądania.

Wywołujący, jeżeli jest implementowany mechanizm cofania poleceń, zazwyczaj przechowuje **Polecenia** w kolekcji umożliwiającej przechodzenie sekwencyjne po jej elementach.

Wzorzec projektowy «polecenie»

Klient tworzy **KonkretnePolecenie**, przekazując mu **Odbiorcę** tego polecenia, a następnie przekazuje je **Wywołującemu**.

Wywołujący przechowuje **Polecenie** tak długo, jak jest mu ono potrzebne.
Wywołujący zgłasza żądanie, wywołując operację *wykonaj()* **Polecenia**.

KonkretnePolecenie wywołuje metody swojego **Odbiorcy** w celu spełnienia żądania.

Wzorzec projektowy «polecenie» – przykładowe zastosowanie

Przykładowy problem

Dany jest graficzny edytor dla symulatora układów elektronicznych złożonych z elementów dyskretnych.

Cechy edytora:

- Pozwala na zapamiętanie ostatnio wykonanych akcji i umożliwia ich cofanie.
- Pozwala również na ponawianie wcześniej cofniętych akcji.

Należy zaprojektować mechanizm pozwalający w prosty sposób zaimplementować cofanie oraz ponawianie wprowadzanych zmian.

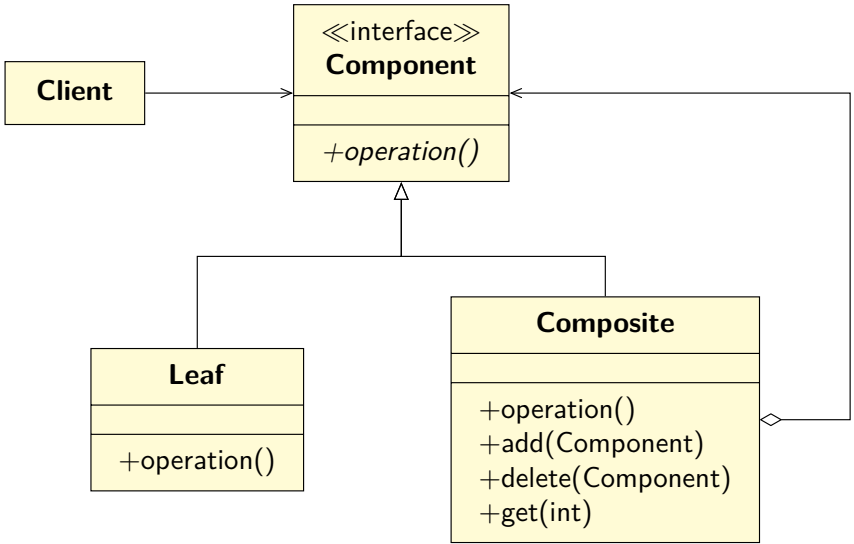
Wzorzec projektowy «polecenie» – przykładowe zastosowanie

Obszar zastosowań

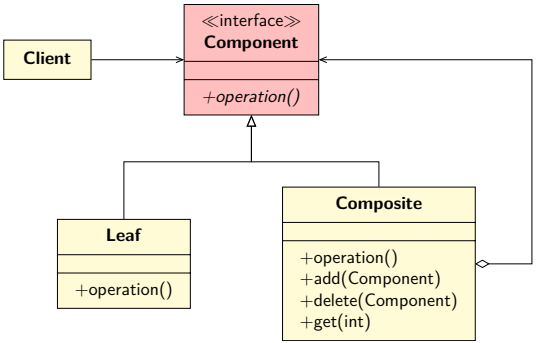
Stosuj wzorzec polecenia, gdy:

- W systemie istnieje potrzeba, aby niektóre akcje mogły być enkapsulowane w postaci obiektów w celu ich późniejszego wykonania.
- Okres życia tych obiektów zazwyczaj jest niezależny od pierwotnego żądania wykonania akcji.
- Formułowanie żądań i ich wykonanie może być rozdzielone w czasie.
- Istnieje potrzeba logowania wykonywanych akcji lub ich kolejkowania.
- Istnieje potrzeba wprowadzenia możliwości anulowania poleceń.

Wzorzec projektowy «kompozyt»



Wzorzec projektowy «kompozyt»

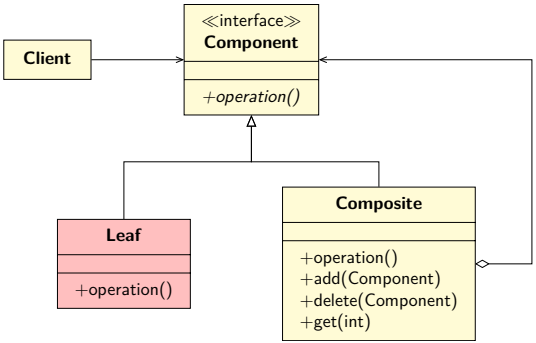


Komponent

Deklaruje interfejs składanych obiektów.

Klienci, używając poszczególnych komponentów, wykorzystują metody "z dziedziny problemu" zdefiniowane w klasie bazowej. Tutaj są one symbolizowane przez *operation*.

Wzorzec projektowy «kompozyt»

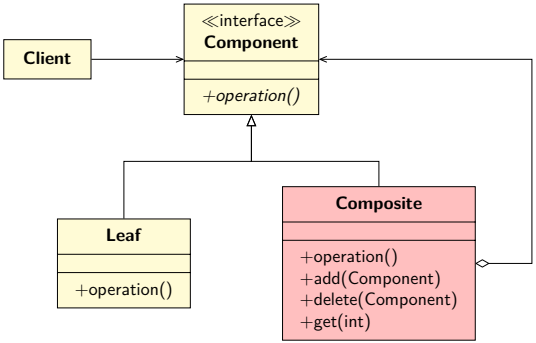


Liść

Definiuje działanie obiektów pierwotnych w strukturze.

W danej strukturze może istnieć wiele różnych klas będących liśćmi. Poszczególne liście mogą implementować zachowanie właściwe dla swojego rodzaju.

Wzorzec projektowy «kompozyt»



Kompozyt

- Definiuje działanie **Komponentów** składających się z innych **Komponentów**.
- Przechowuje **Komponenty**, z których jest złożony.
- Udostępnia operacje związane z zarządzaniem składowymi (tu: *add* i *delete*).

Gdy wywołanie konkretnej operacji zdefiniowanej w Komponencie zwraca wynik, Kompozyt składa wyniki działania swoich Liści w spójną całość. Dotyczy to również tak zwanych efektów ubocznych.

Wzorzec projektowy «kompozyt»

Poszczególni **Klienci** nie wiedzą, czy mają do czynienia z pojedynczym obiektem, czy całą strukturą. W ten sposób uproszczona zostaje ich budowa.

Dodawanie nowych rodzajów **Komponentów** jest proste. W językach o statycznym typowaniu (C++) dodanie nowego typu **Komponentu** nie wymaga ponownej kompilacji kodu **Klientów**.

Istnieją implementacje wzorca projektowego kompozyt, którym klasa **Komponentu** implementuje również operacje związane z zarządzaniem składowymi klasy **Kompozytu**. W takim przypadku **Klienci** również mają możliwość modyfikowania drzewiastej struktury **Kompozytu**. Od projektanta systemu zależy, która implementacja będzie wykorzystywana.

Wzorzec projektowy «kompozyt»

Obszar zastosowań

Stosuj wzorzec kompozytu, gdy:

- W systemie istnieją rodziny obiektów, które mogą być przedstawiane w postaci hierarchii.
- Klienci wykorzystujący obiekty powinni móc ignorować różnicę między prostym obiektem a złożonym.

Klasy implementujące wzorzec **Polecenie** często mają strukturę kompozytu w celu rejestracji bardziej złożonych poleceń.

Wzorzec projektowy «kompozyt» – przykład zastosowania

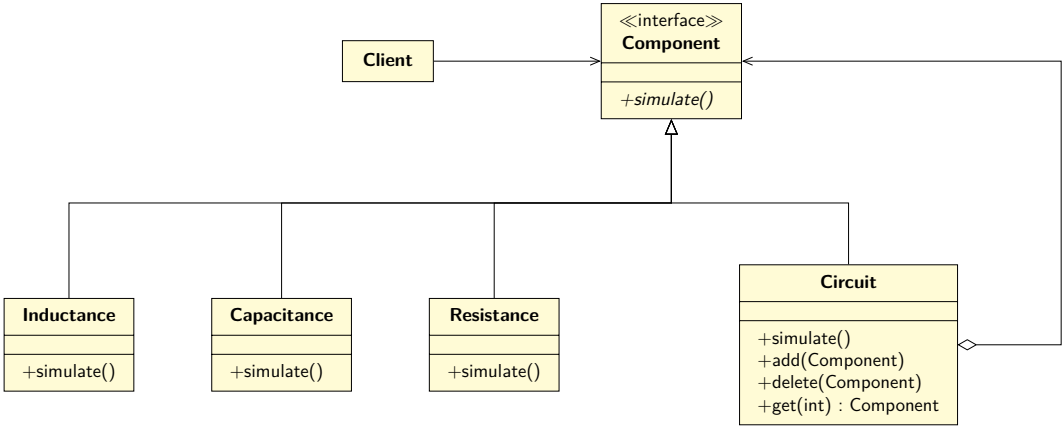
Przykładowy problem

Dany jest symulator układów analogowych wraz ze wszystkimi klasami niezbędnymi do reprezentacji poszczególnych elementów dyskretnych.
Cechy symulatora:

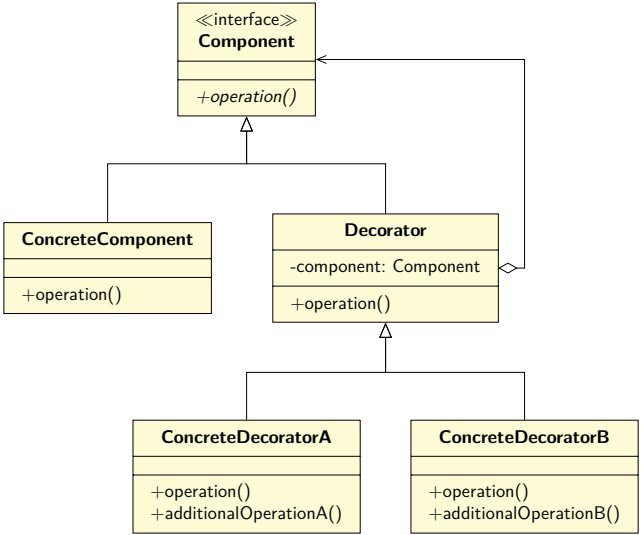
- Pozwala na budowanie podukładów będących dwójnikami, które później mogą być wielokrotnie wykorzystywane do budowania bardziej złożonych systemów.
- Budowane układy i podukłady mogą być wielokrotnie zagnieżdżane.

Należy zaprojektować taką strukturę reprezentacji poszczególnych elementów i układów w systemie, aby symulator nie musiał ingerować w to, z jakiego rodzaju elementem czy podukładem ma do czynienia.

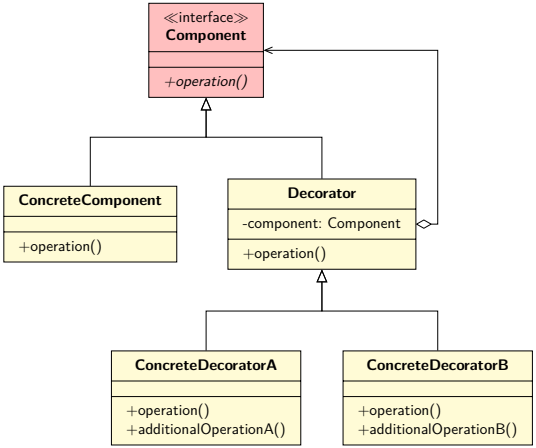
Wzorzec projektowy «kompozyt» – przykład zastosowania



Wzorzec projektowy «dekorator»



Wzorzec projektowy «dekorator»

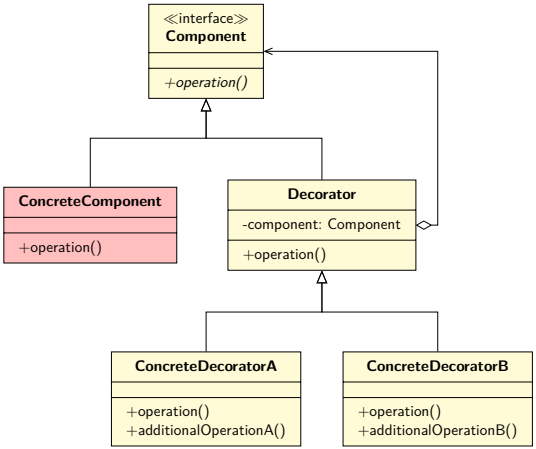


Komponent

Deklaruje interfejs obiektów, do których można dynamicznie dołączyć zobowiązanie (dynamicznie = w trakcie wykonywania programu).

Komponent definiuje, jakie metody muszą być zdefiniowane w **Konkretnych Komponentach** oraz poszczególnych **Dekoratorach**. Klienci korzystający z **Komponentów** nie mają świadomości, czy korzystają **Konkretny Komponent** został udekorowany czy nie.

Wzorzec projektowy «dekorator»

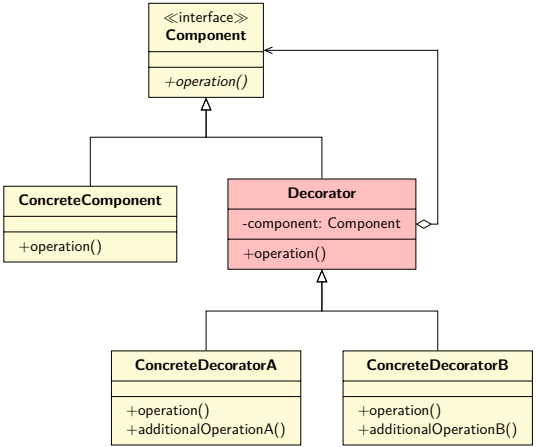


KonkretnyKomponent

Definiuje obiekt, do którego można dołączyć dodatkowe zobowiązanie.

Konkretny Komponent zazwyczaj implementuje zdefiniowane z interfejsie operacje w sposób podstawowy. Dodatkowe aspekty tej funkcjonalności definiują **Konkretne Dekoratory**.

Wzorzec projektowy «dekorator»

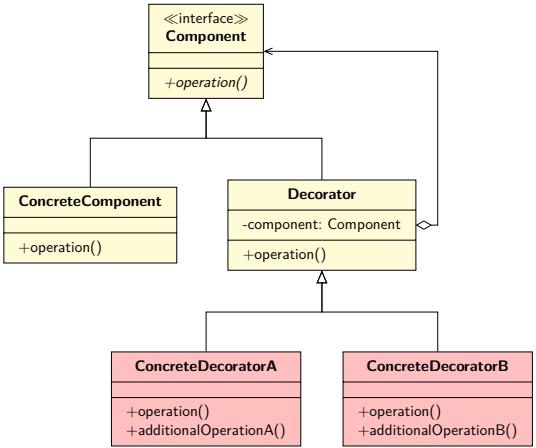


Dekorator

- Zarządza odwołaniem do obiektu **Komponent**.
- Definiuje interfejs dopasowany do interfejsu **Komponentu**.

Dekorator oddziela **Konkretnych Dekoratorów** od **Konkretnych Komponentów**. Zarządza obiektem komponentu. Może być jego właścicielem lub jedynie użytkownikiem.

Wzorzec projektowy «dekorator»



DekoratorKonkretny

Dodaje zobowiązania do **Komponentu**.

Dekorator Konkretny pozwala na dodanie nowej operacji (metody) do istniejącego już obiektu. Zazwyczaj istnieje wielu różnych dekoratorów i każdy z nich dodaje nowe funkcjonalności. W systemie mogą pojawiać obiekty "udekorowane" w rozmaity sposób.

Wzorzec projektowy «dekorator»

Dekorator przesyła żądania wykonania operacji do swojego **Komponentu**. **Dekorator** przed przesłaniem żądania lub po może wykonać dodatkowe operacje.

Obszar zastosowań

Stosuj wzorzec dekoratora, gdy:

- Istnieje konieczność dodawania w sposób dynamiczny zobowiązań do istniejących obiektów bez wpływu na ich klientów.
- Ze względu na dużą ilość nowych odpowiedzialności stworzenie wszystkich ich kombinacji jest niepraktyczne.

Wzorzec projektowy «dekorator» – przykład zastosowania

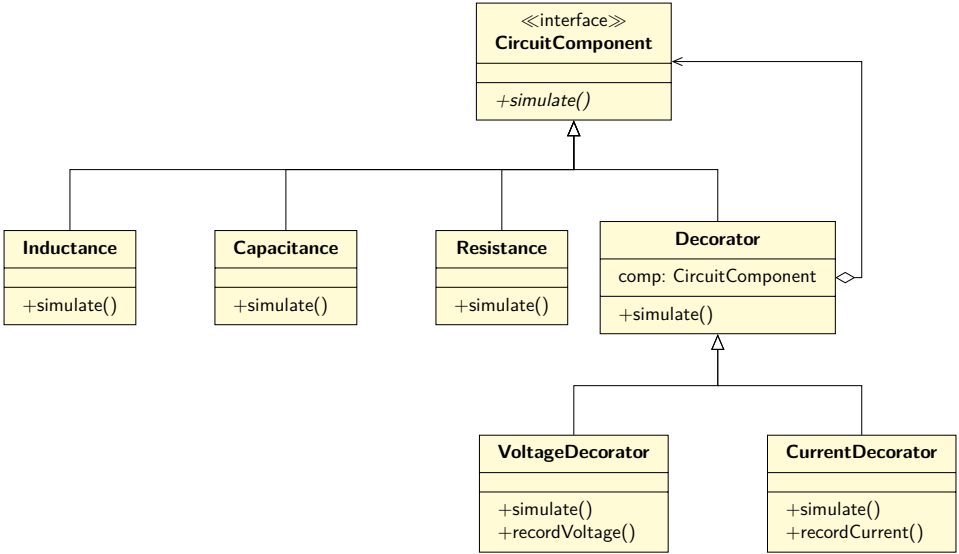
Przykładowy problem

Dany jest symulator układów analogowych wraz ze wszystkimi klasami niezbędnymi do reprezentacji poszczególnych elementów dyskretnych.
Cechy symulatora:

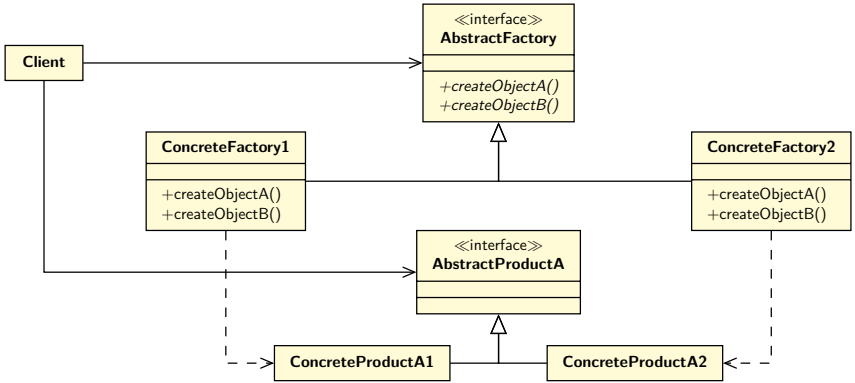
- Dla symulacji czasowej pozwala wybrać elementy, dla których rejestrowane jest napięcie na zaciskach i/lub prąd przepływający przez te element.
- Mechanizmy rejestracji mogą być dodawane niezależnie od siebie oraz niezależnie od wybranego typu elementu.

Należy zaprojektować mechanizm, który niezależnie od rodzaju elementu lub podukładu pozwoli na dynamiczne dodawanie do obiektu opcji rejestracji napięcia na zasiskach tego elementu lub prąd przez niego przepływający.

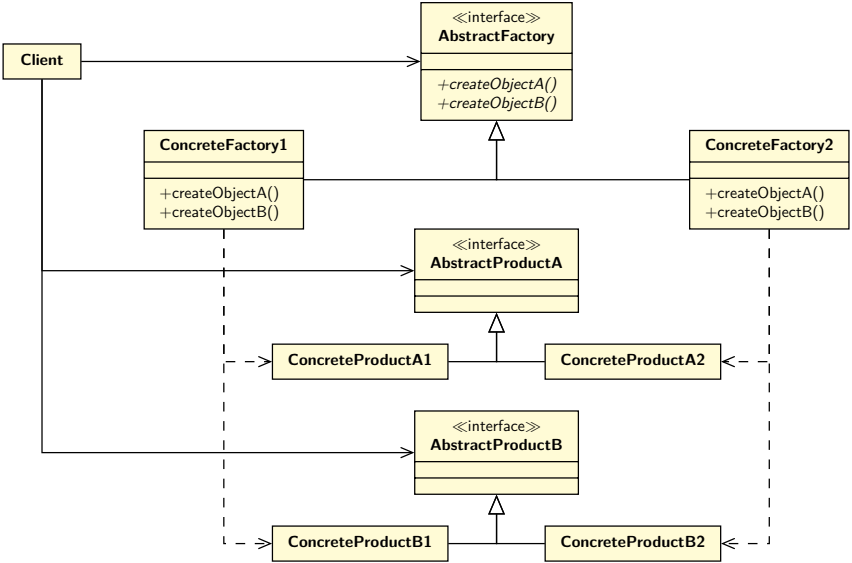
Wzorzec projektowy «dekorator» – przykład zastosowania



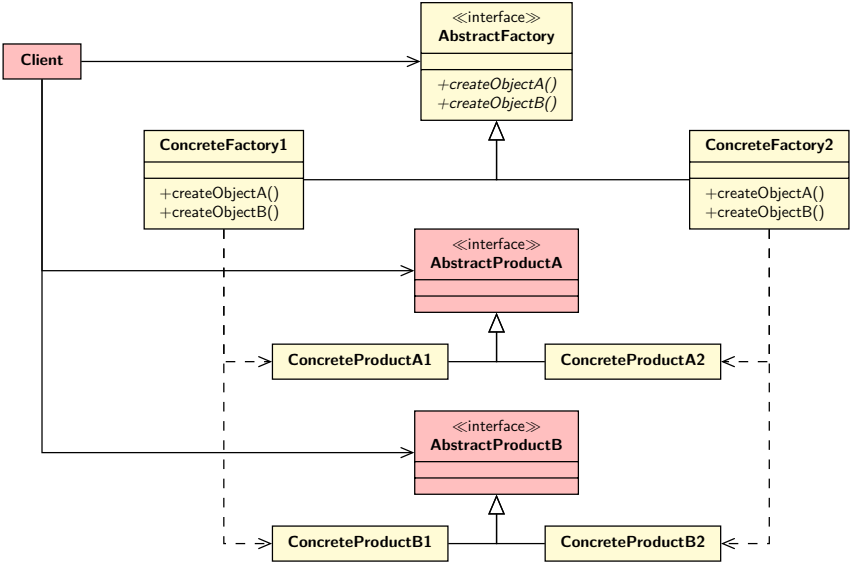
Wzorzec projektowy «fabryka abstrakcyjna» – trywialna



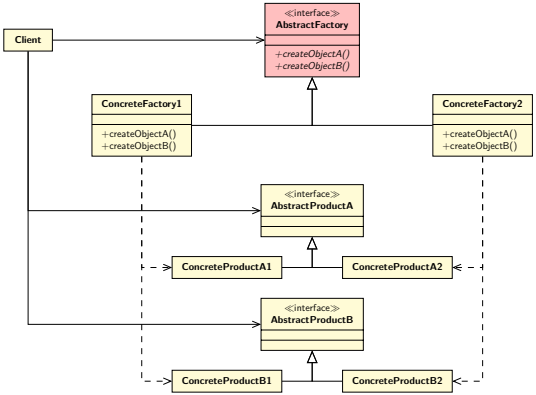
Wzorzec projektowy «fabryka abstrakcyjna» – w praktyce



Wzorzec projektowy «fabryka abstrakcyjna»



Wzorzec projektowy «fabryka abstrakcyjna»

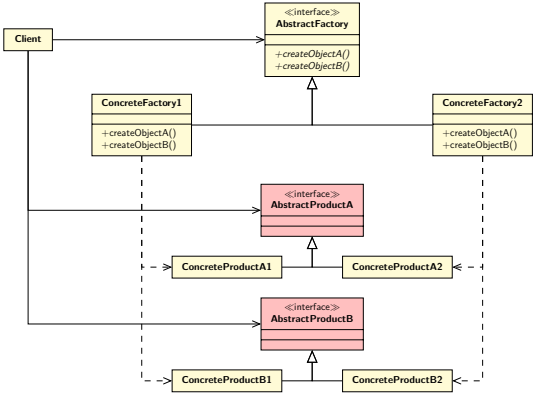


FabrykaAbstrakcyjna

Deklaruje interfejs tworzenia **Produktów Abstrakcyjnych**.

Fabryka Abstrakcyjna definiuje, jakiego typu **Produkty Abstrakcyjne** są tworzonej przez jej konkretne realizacje. Tworzone w ramach fabryki produkty należą do tej samej dziedziny, łączą się w pewien logiczny sposób. **Klienci Fabryki Abstrakcyjnej** nie wiedzą, z którą **Konkretną Fabryką** współpracują. **Fabryka Abstrakcyjna** oddziela **Klientów** od jej konkretnej implementacji.

Wzorzec projektowy «fabryka abstrakcyjna»

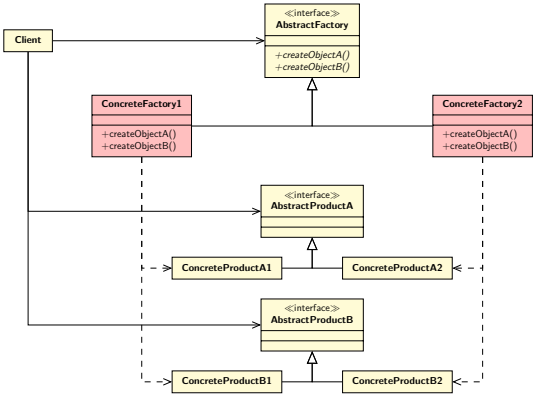


ProduktAbstrakcyjny

Deklaruje interfejs dla pewnego rodzaju produktu.

Produkt Abstrakcyjny definiuje, czym jest obiekt z danej rodziny produktów.

Wzorzec projektowy «fabryka abstrakcyjna»

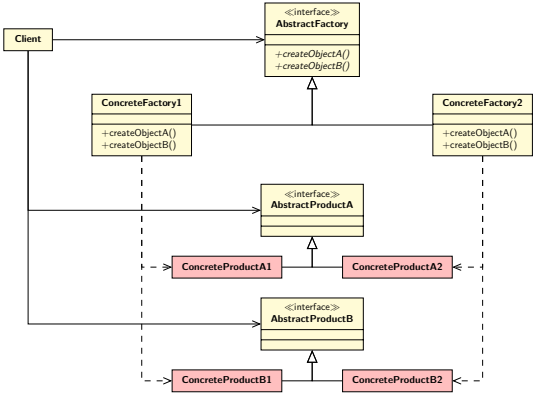


KonkretnaFabryka

Implementuje operację tworzenia pewnego rodzaju produktu.

Konkretna Fabryka tworzy **Konkretne Produkty**. Sama **Konkretna Fabryka** tworzona jest w oderwaniu od **Klienta**. **Klienci**, którzy używają **Fabryki Abstrakcyjnej** nie wiedzą, z którą **Konkretną Fabryką** współpracują.

Wzorzec projektowy «fabryka abstrakcyjna»



KonkretnyProdukt

- Definiuje obiekt będący produktem.
- Implementuje interfejs klasy **Produkt Abstrakcyjny**.

Konkretne Produkty tworzone są przez **Konkretne Fabryki**. **Klienci** nie wiedzą, którego konkretnego produktu używają.

Wzorzec projektowy «fabryka abstrakcyjna»

Klienci korzystają z interfejsu fabryki do tworzenia konkretnych obiektów. **Klienci** nie wiedzą z jaką konkretnie fabryką współpracują, nie wiedzą też, jaki typ mają konkretne produkty tworzone przez fabrykę.

Klienci zazwyczaj nie mają wpływu na to, z której **KonkretnejFabryki** korzystają. **Klienci** uniezależnieni od konkretnych implementacji obiektów i fabryk skuteczniej mogą implementować swoje odpowiedzialności.

Wzorzec projektowy «fabryka abstrakcyjna»

Obszar zastosowań

Stosuj wzorzec fabryki abstrakcyjnej, gdy:

- W systemie istnieje wiele powiązanych ze sobą rodzin produktów.
- System konfigurowany jest przy użyciu jednej z tych rodzin.
- Obiekty z poszczególnych rodzin produktów powinny być używane wspólnie.
- System powinien być izolowany od mechanizmu tworzenia poszczególnych produktów.

Fabryka abstrakcyjna – przykład zastosowania

Przykładowy problem

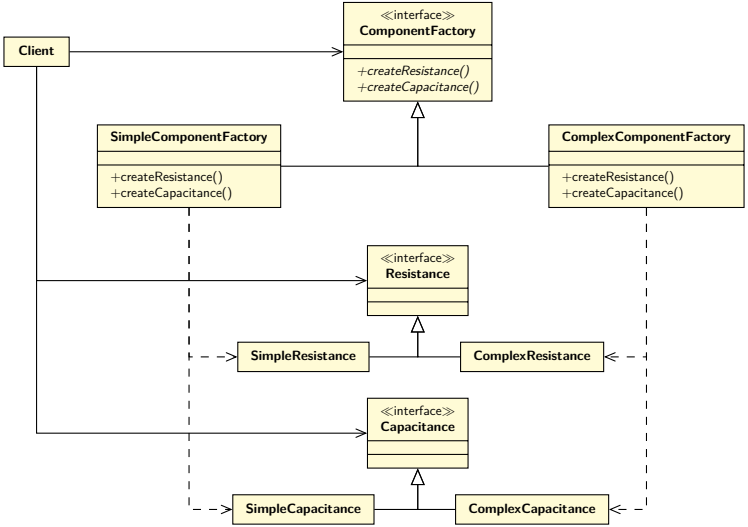
Dany jest symulator układów analogowych wraz ze wszystkimi klasami niezbędnymi do reprezentacji poszczególnych elementów dyskretnych.

Cechy symulatora:

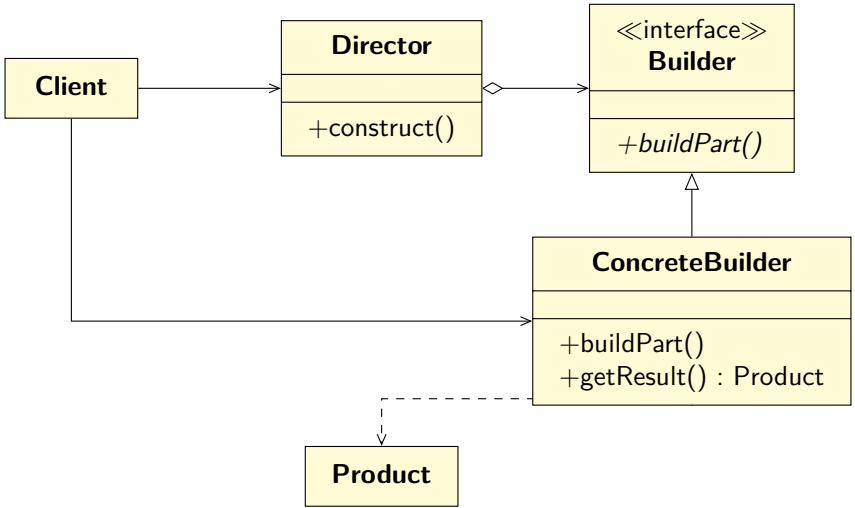
- Poszczególne elementy dyskretne mogą być modelowane w sposób:
  - uproszczony, gdzie elemetny cechują się jedynie podstawowymi parametrami – pojemnością, rezystancją oraz indukcyjnością,
  - pogłębiony, gdzie oprócz podstawowych parametrów uwzględnione są na przykład pojemność zwojów cewki.
- Ma możliwość przeprowadzania symulacji uproszczonych oraz dokładnych w zależności od rodzaju wykorzystanych elementów.

Należy zbudować mechanizm tworzenia obiektów reprezentujących elementy dyskretne, który w zależności od trybu przeprowadzanej symulacji będzie dostarczał odpowiednie elementy.

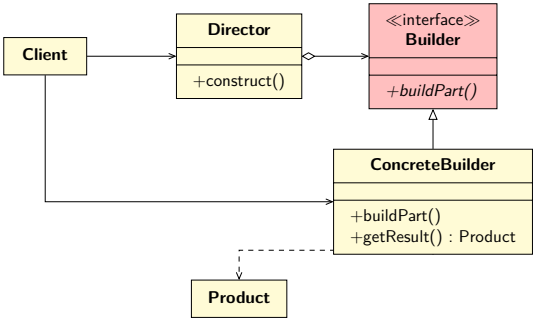
Fabryka abstrakcyjna – przykład zastosowania



Wzorzec projektowy «budowniczy»



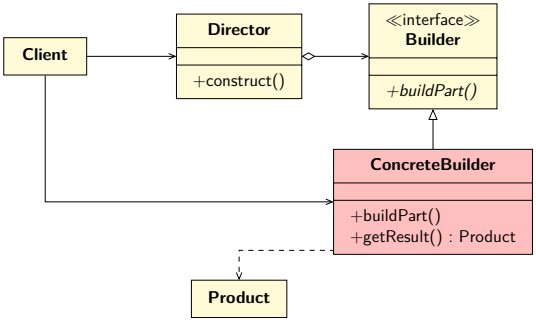
wzorzec projektowy «budowniczy»



Budowniczy

Określa interfejs abstrakcyjny do tworzenia części składowych **Produktu**.

Budowniczy definiuje, jakich metod mogą używać **Kierownik/cy** i jakie metody muszą być zdefiniowane w konkretnych **Budowniczych**.

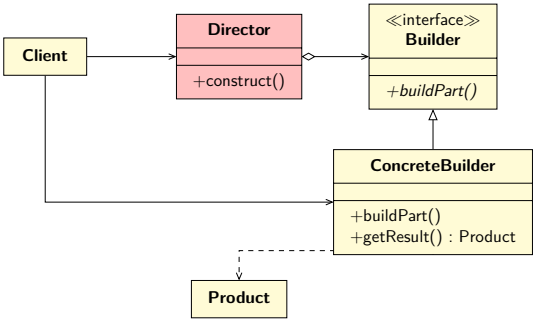


KonkretnyBudowniczy

- Konstruuje i zestawia ze sobą części **Produktu** przez implementowanie interfejsu **Budowniczego**.
- Definiuje i kontroluje tworzoną przez siebie reprezentację.

Wytwarzany **Produkt** może być inny dla różnych **Budowniczych**. Poszczególne **Produkty** nie muszą należeć do jednej hierarchii. Dom z klocków Lego może powstawać według tych samych ogólnych założeń co dom rodzinny.

Wzorzec projektowy «budowniczy»



Kierownik

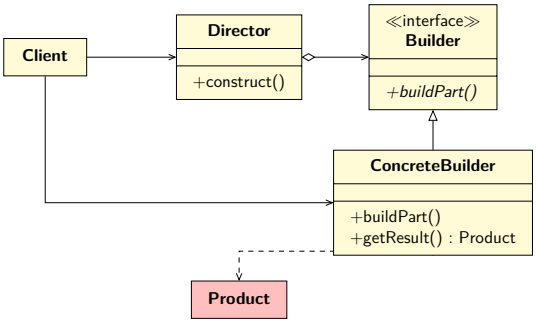
Konstruuje **Produkt**, używając interfejsu **Budowniczego**.

Kierownik wie:

- jakiego *rodzaju* elementy mają być utworzone (ale bez znajomości konkretnej klasy),
- w jakiej kolejności mają być tworzone i łączone.

Kierownik nie musi wiedzieć:

- z jakim konkretnie **Budowniczym** współpracuje,
- jaki konkretnie **Produkt** otrzyma.



Produkt

- Reprezentuje obiekt złożony.
- Zawiera obiekty definiujące jego części składowe.

Różni **Konkretni Budowniczowie** budują **Produkty** różnego rodzaju, niekoniecznie należące do wspólnej hierarchii klas – dlatego **Klient** odbiera **Produkt** od **Konkretnego Budowniczego**, nie od **Kierownika**.

Wzorzec projektowy «budowniczy»

Klient tworzy obiekt **Kierownika** i konfiguruje go konkretnym **Budowniczym**. **Kierownik** definiuje, co ma być utworzone i informuje **Budowniczego** o potrzebie zbudowania części **Produktu**.

Budowniczy przetwarza żądania **Kierownika** i dodaje niezbędne części do **Produktu**. **Klient** pobiera gotowy **Produkt** od **Budowniczego**.

Algorytm tworzenia złożonego **Produktu** jest niezależny od części składowych konkretnego obiektu i sposobu ich tworzenia. Konstruowany **Produkt** może mieć różne reprezentacje w zależności od wybranego **Budowniczego**

Wzorzec projektowy «budowniczy»

Obszar zastosowań

Stosuj wzorzec budowniczego, gdy:

- Dany jest algorytm tworzenia obiektu złożonego z wielu elementów składowych.
- Mogą występować różne niepowiązane ze sobą obiekty, które tworzone są z użyciem tego samego algorytmu.
- Należy zapewnić jednolity sposób budowania wszystkich obiektów, który jest niezależny od wewnętrznej reprezentacji tych obiektów.

Wzorzec projektowy «budowniczy» – przykład zastosowania

Przykładowy problem

Dany jest symulator układów analogowych wraz ze wszystkimi klasami niezbędnymi do reprezentacji poszczególnych elementów dyskretnych.

Cechy symulatora:

- Posiada dwa niezależne moduły symulacji: w dziedzinie czasu i częstotliwości.
- Każdy z modułów symulacji posiada własną hierarchię klas odpowiadających symulowanym elementom dyskretnym.
- Istnieje jeden format pliku utrwalającego strukturę symulowanego układu analogowego.

Należy zbudować mechanizm odczytywania struktury pliku, który będzie w stanie budować odpowiednie układy, niezależnie od typu symulacji.

Wzorzec projektowy «budowniczy» – przykład zastosowania