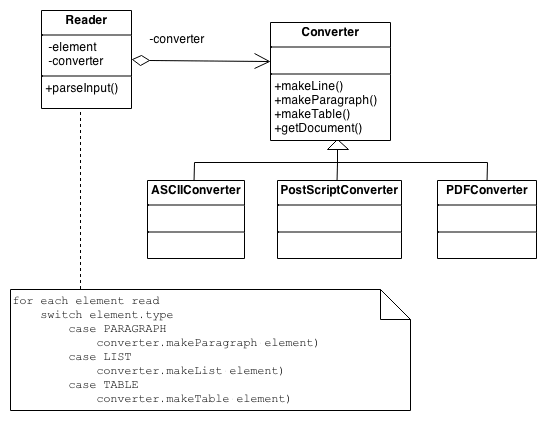
**Wzorce kreacyjne**

**Builder** (Budowniczy) – rozdziela sposób budowy obiektów od ich reprezentacji

Dokładny przebieg procesu budowania przedstawia znajdujący się obok diagram sekwencji:

1. klient używający wzorca konstruuje obiekt budowniczego,
2. klient konstruuje nadzorcę, przekazując mu referencję do obiektu budowniczego, z którego ma korzystać,
3. klient zleca skonstruowanie produktu,
4. nadzorca zleca budowniczemu wykonanie w odpowiedniej kolejności wszystkich czynności niezbędnych do stworzenia produktu,
5. klient pobiera gotowy produkt od budowniczego.

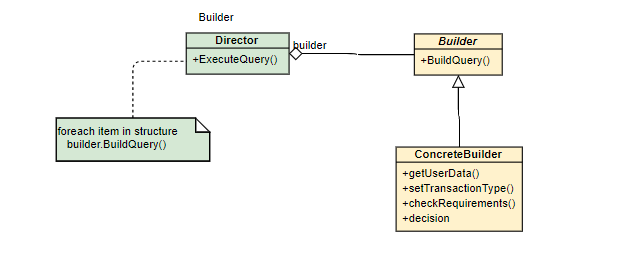


Zalety:

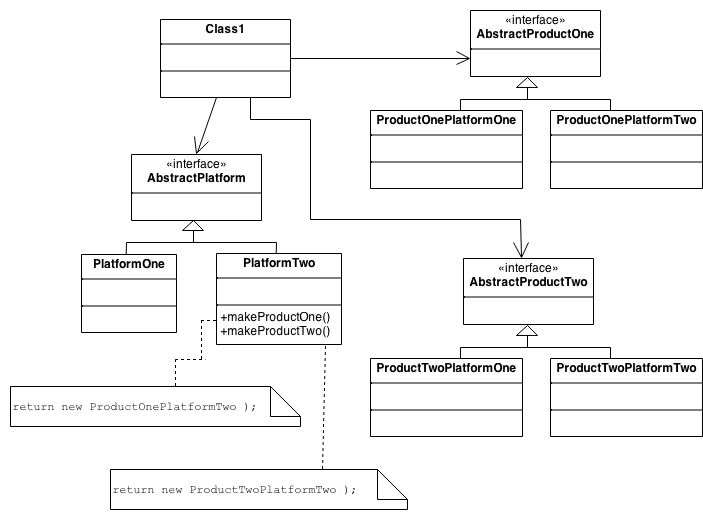
- Jasne oddzielenie budowy od reprezentacji obiektu

- Lepsza kontrola na procesem tworzenia obiektu

Budowniczy w Aplikacji Bankowej



**Abstract factory** (Fabryka abstrakcyjna) – interface do tworzenia rodzin jednego typu bez specyfikowania ich konkretnych klas.

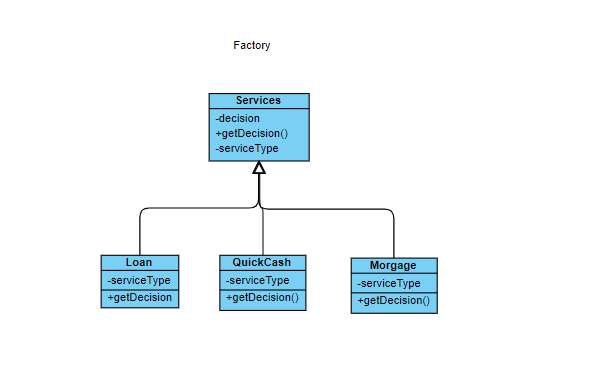


Zalety:

- Możliwość ukrycia szczegółów implementacyjnych przed klientem

- Klient widzi tylko interface

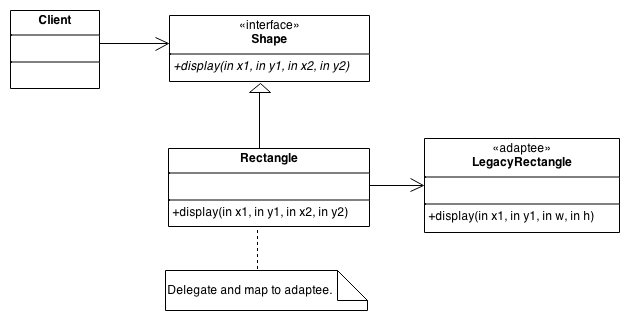
Fabryka abstrakcyjna w Aplikacji Bankowej



**Wzorce strukturalne**

**Adapter** (wrapper) – umożliwia współpracę dwóm klasom o niekompatybilnych interfacach poprzez przekształcenie jednej z klas na interface drugiej klasy. Umożliwia też opakowanie interfacu w nowy.

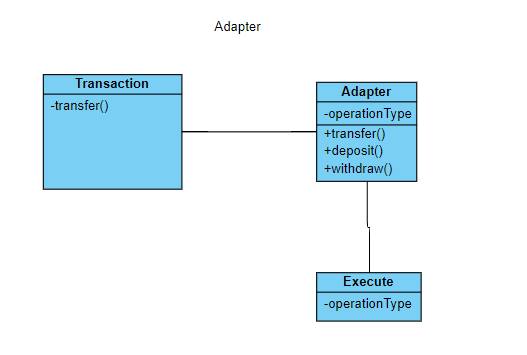
Adapter:



Konsekwencje stosowania:

- brak możliwości adaptowania klasy wraz z jej podklasami,

- możliwość przeładowania metod obiektu adaptowanego.

Adapter w Aplikacji Bankowej

**Most** (bridge) - pozwala oddzielić abstrakcję obiektu od jego implementacji.

- odseparować implementację od interfejsu,

- poprawić możliwości rozbudowy klas, zarówno implementacji, jak i interfejsu (m.in. przez dziedziczenie),

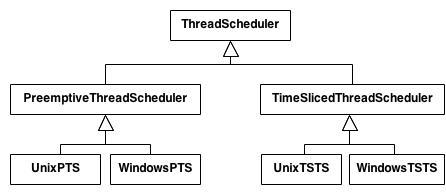
- ukryć implementację przed klientem, co umożliwia zmianę implementacji bez zmian interfejsu.

Zalety:

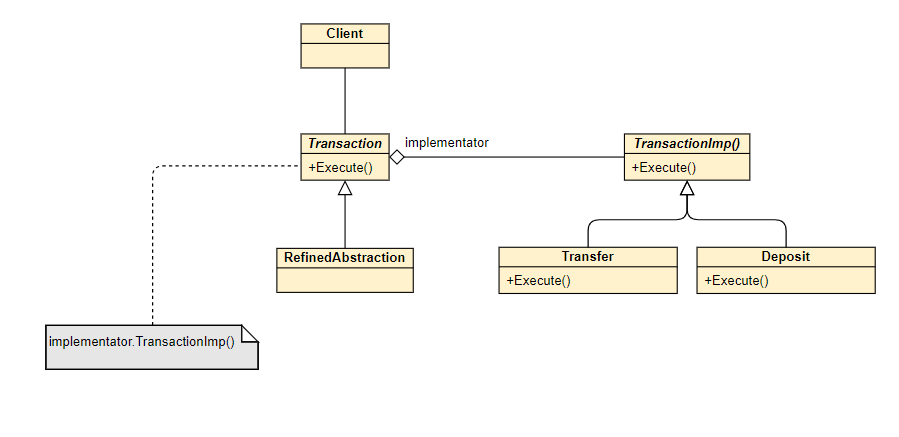
- decoupling interface obiektu

- poprawia możliwości rozbudowy

- ukrywa detale przed klientem

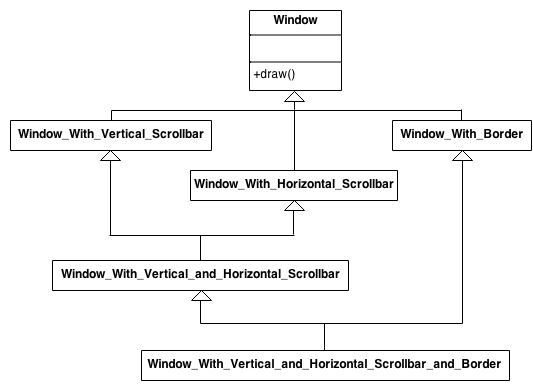


Most w Aplikacji Bankowej



**Dekorator** (Decorator) - pozwala na dodanie nowej funkcji do istniejących klas dynamicznie podczas działania programu. Wzorzec dekoratora polega na opakowaniu oryginalnej klasy w nową klasę "dekorującą". Zwykle przekazuje się oryginalny obiekt jako parametr konstruktora dekoratora, metody dekoratora wywołują metody oryginalnego obiektu i dodatkowo implementują nową funkcję.

Przykład użycia



Zalety:

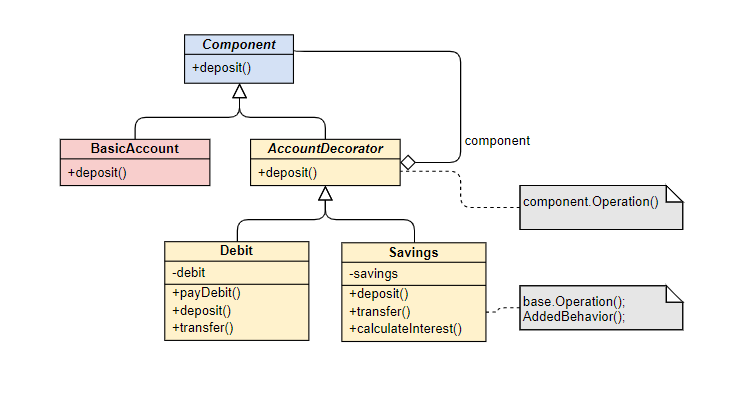
- Zapewnia większą elastyczność niż statyczne dziedziczenie.

- Pozwala uniknąć tworzenia przeładowanych funkcjami klas na wysokich poziomach hierarchii.

- Dekorator i powiązany z nim komponent nie są identyczne.

- Powstawanie wielu małych obiektów.

Dekorator w Aplikacji Bankowej:



**Wzorce behawioralne**

**Strategia (Strategy)** - Umożliwia wymienne stosowanie każdego z nich w trakcie działania aplikacji niezależnie od korzystających z nich użytkowników.

Zalety:

- wzorzec pozwala na ścisłe, formalne zdefiniowanie rozszerzalnych rodzin algorytmów dzięki wprowadzeniu interfejsu Strategia,

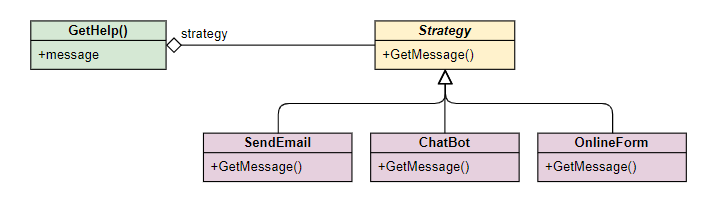
- bazuje na koncepcji kompozycji, a nie na dziedziczeniu — nie ma sztywnego powiązania między algorytmem a miejscem jego wykorzystania. Może on być wymieniany w trakcie działania programu

Wady:

- dodatkowy koszt komunikacji między klientem a strategią (wywołania metod, przekazywanie danych),

- zwiększenie liczby obiektów.

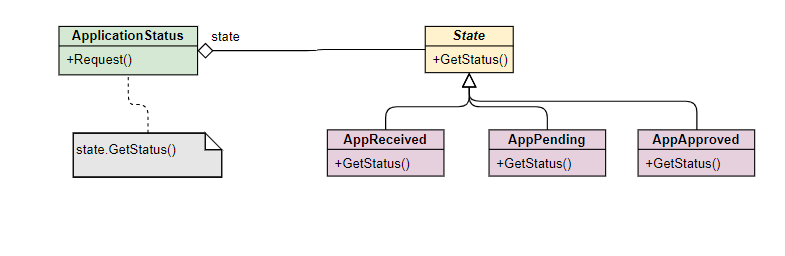
Strategia w aplikacji bankowej:



**Stan (State) -** umożliwia zmianę zachowania obiektu poprzez zmianę jego stanu wewnętrznego, uzależnia sposób działania obiektu od stanu w jakim się aktualnie znajduje.

Do plusów korzystania z tego wzorca należy możliwość łatwego dodawania kolejnych narzędzi.

Stan w Aplikacji Bankowej



**Interperter** - celem jest zdefiniowanie opisu gramatyki pewnego języka interpretowalnego, a także stworzenie dla niej interpretera, dzięki któremu będzie możliwe rozwiązanie opisanego problemu. Można go wykorzystać w sytuacjach, gdy zdania, zapisane w pewnym interpretowalnym języku, mogą być reprezentowane jako drzewa składniowe oraz istnieje prosta gramatyka opisująca ten język. Do przykładowych zastosowań tego wzorca należy interpretacja rzymskiego systemu liczbowego, interpretacja wyrażeń zapisanych w odwrotnej notacji polskiej oraz sprawdzanie poprawności pewnych reguł.

Interpreter w Aplikacji Bankowej:

