

WZORCE PROJEKTOWE

WZORCE
KREACYJNE

WZORCE
STRUKTURALNE

WZORCE
CZYNNOŚCIOWE

WZORCE KREACYJNE

Singleton Pattern

Zapewnia, że klasa posiada
pojedynczą instancję

ConfigManager

Get()

Set()

GetInstance()

ConfigManager

Get()

Set()

- ConfigManager()

ConfigManager

Get()

Set()

- ConfigManager()

- instance

ConfigManager

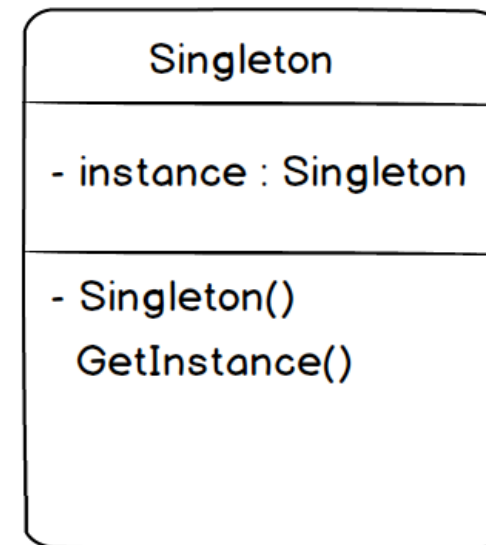
Get()

Set()

- ConfigManager()

- instance

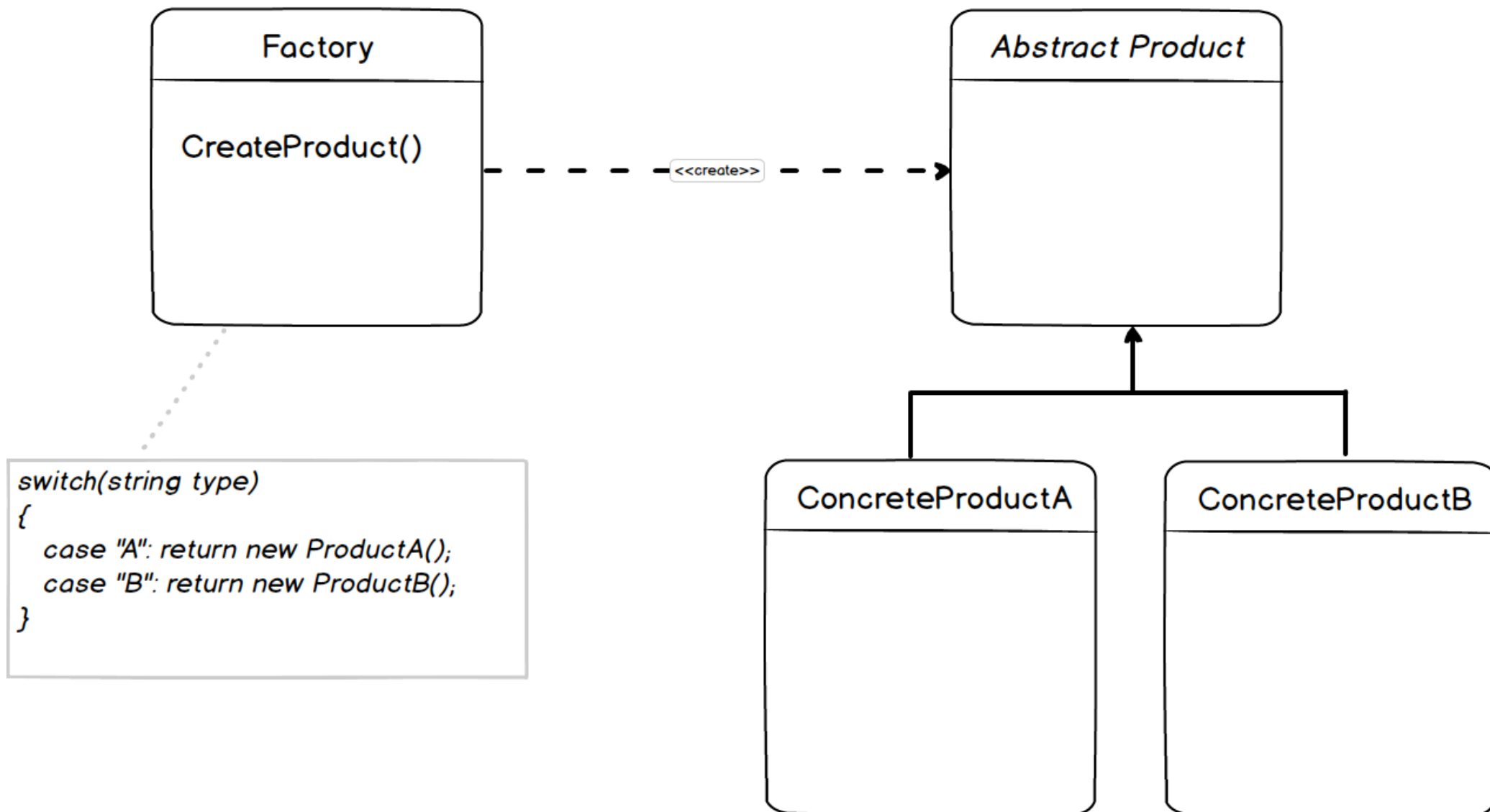
GetInstance()



Pozwala zachować pewność, że istnieje wyłącznie **jedna instancja** danej klasy.

Simple Factory Pattern

Zapewnia tworzenie obiektu na podstawie wielu warunków



Abstract Factory Pattern

Zapewnia interfejs do tworzenia
rodziny powiązanych obiektów

INTERFACE

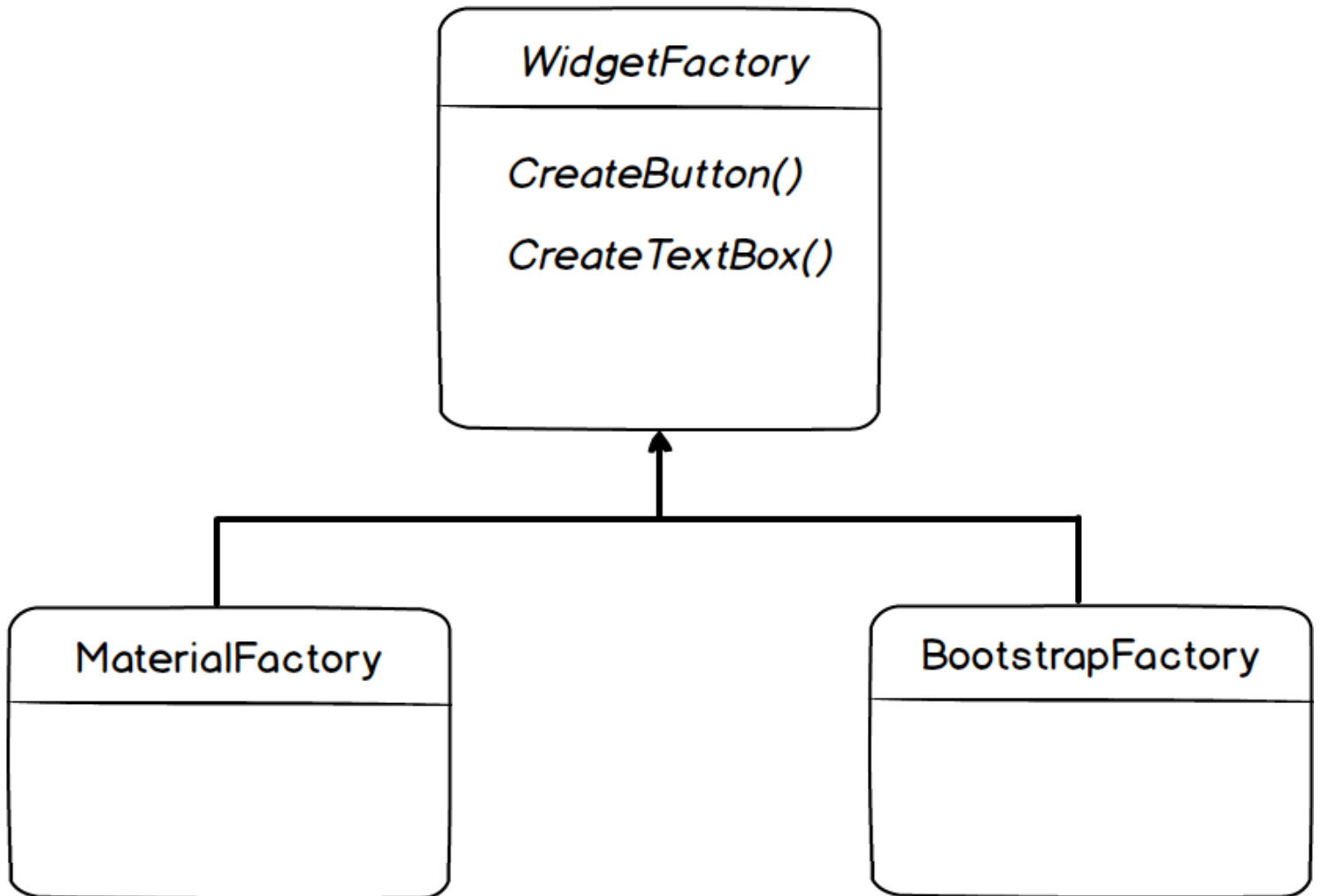
WidgetFactory

CreateButton()

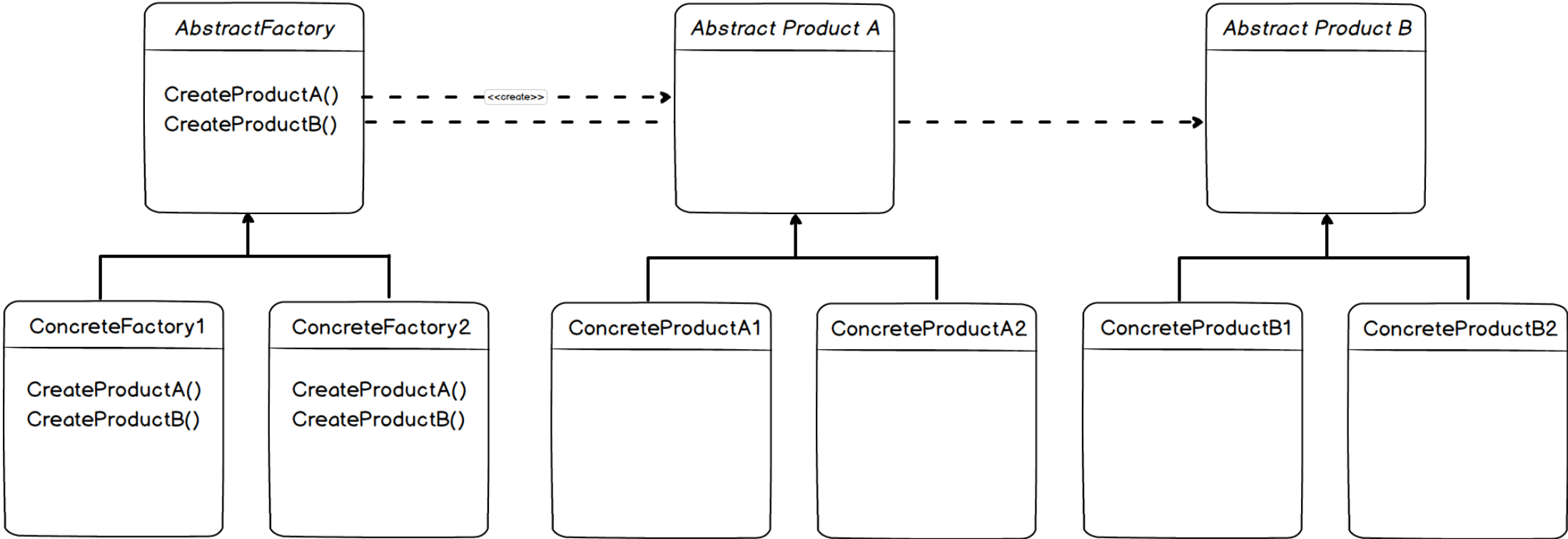
CreateTextBox()

MaterialFactory

BootstrapFactory



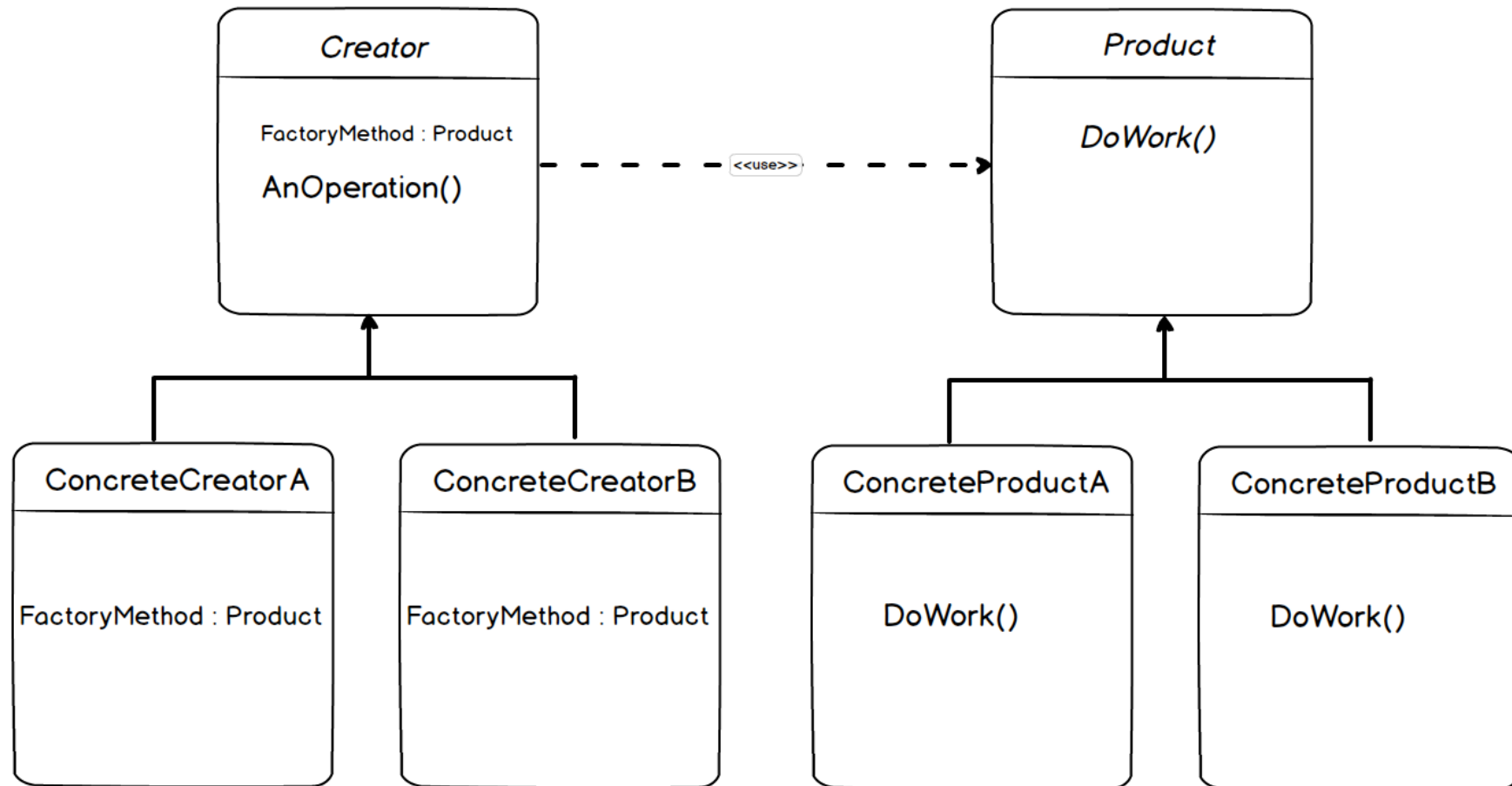
INTERFACE



Factory Method Pattern

Odkłada tworzenie obiektu
do **podklas**

INTERFACE

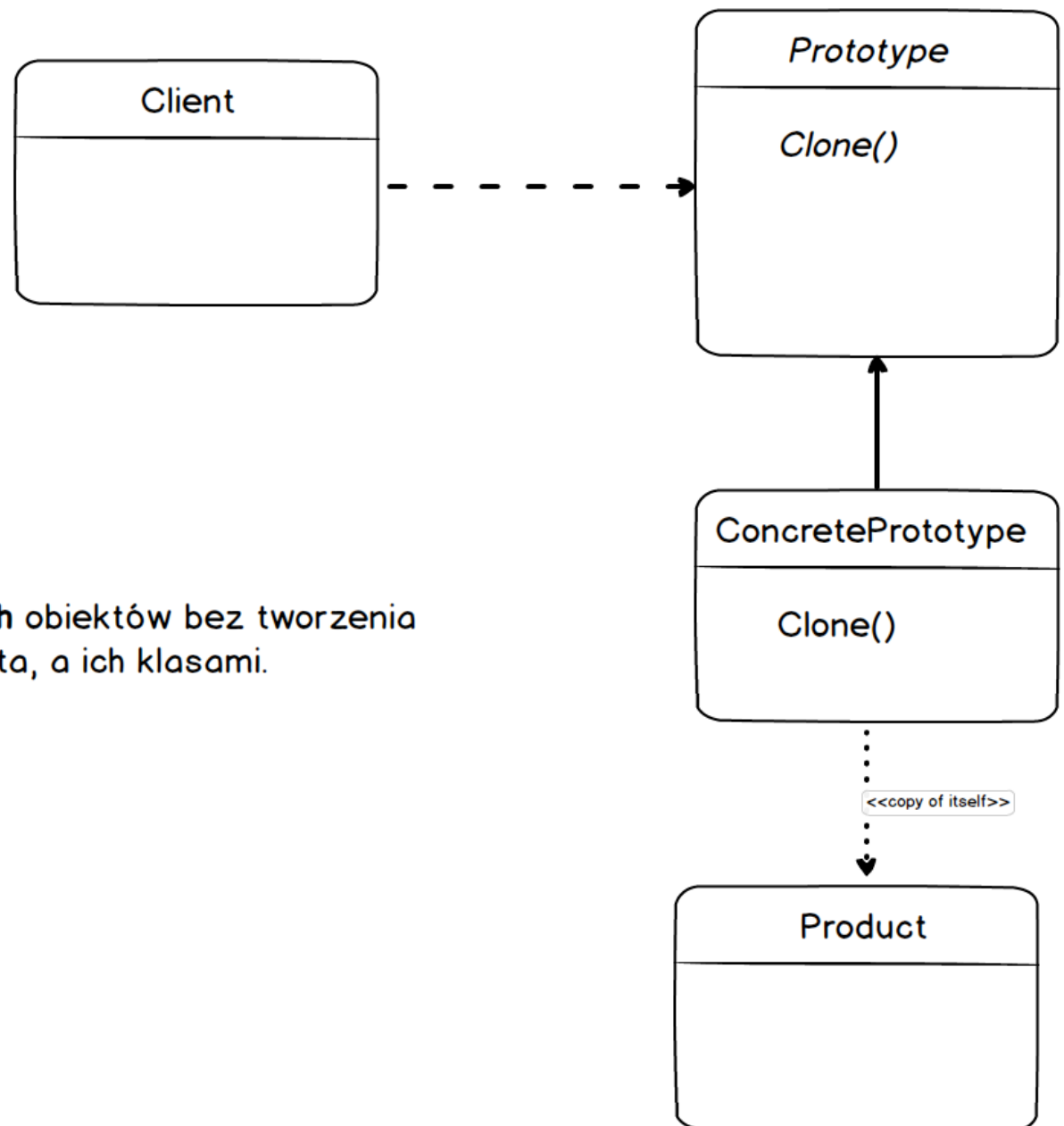


Prototype Pattern

Tworzy nowe obiekty
kopiując istniejący obiekt

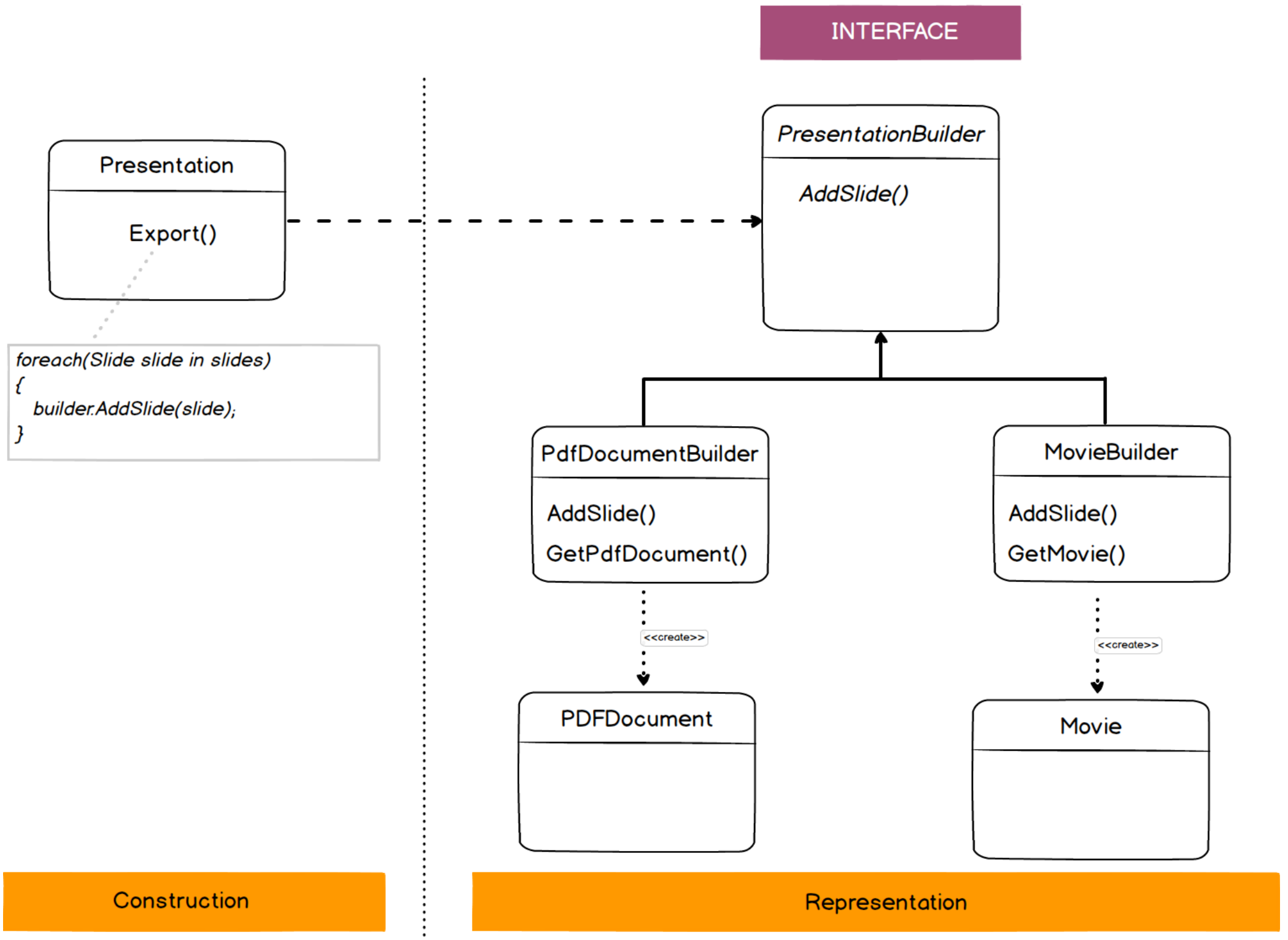
INTERFACE

Umożliwia kopiowanie **istniejących** obiektów bez tworzenia zależności pomiędzy kodem klienta, a ich klasami.

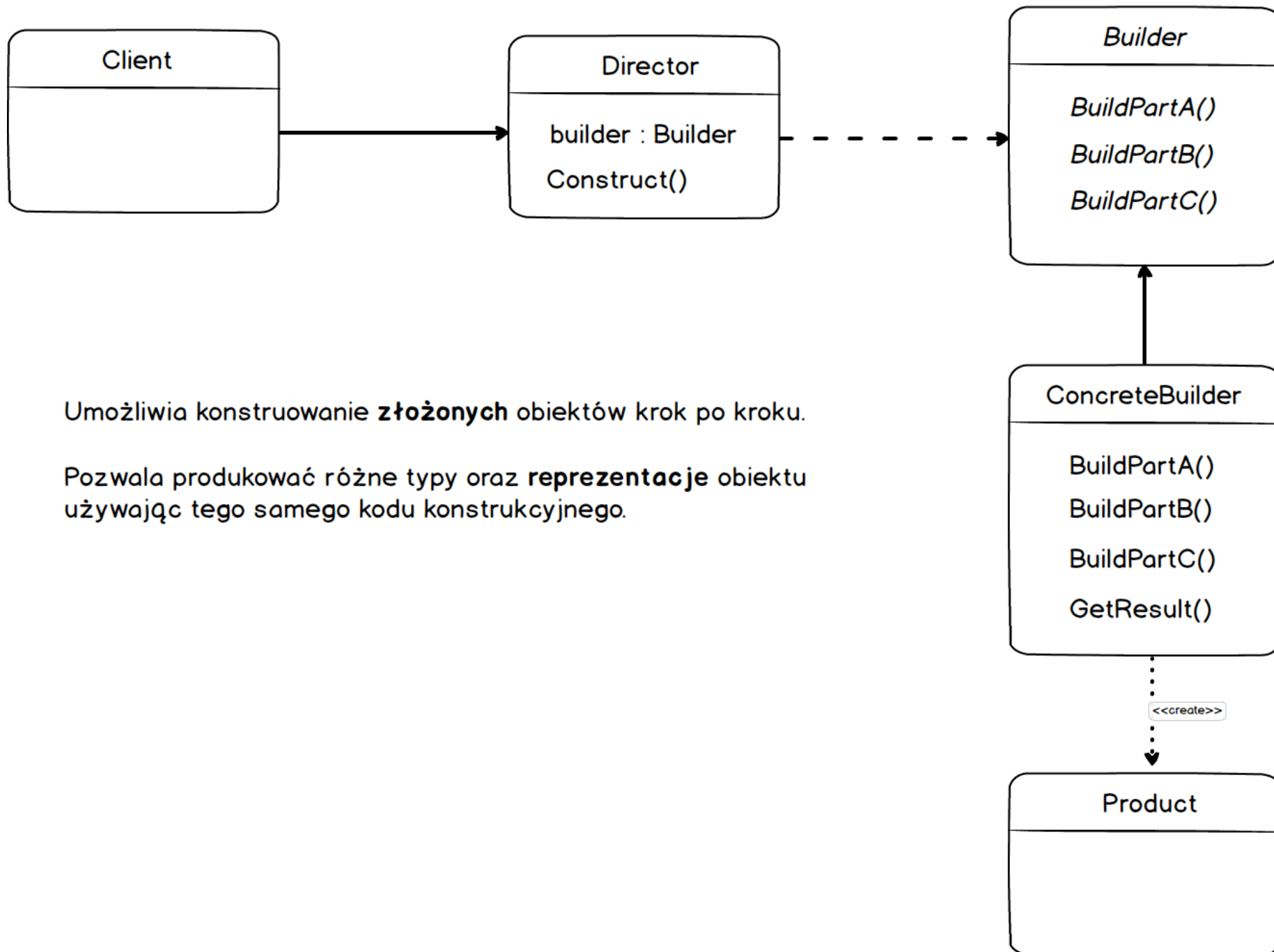


Builder Pattern

Oddziela konstrukcję obiektu
od jego reprezentacji



INTERFACE



Umożliwia konstruowanie **złożonych** obiektów krok po kroku.

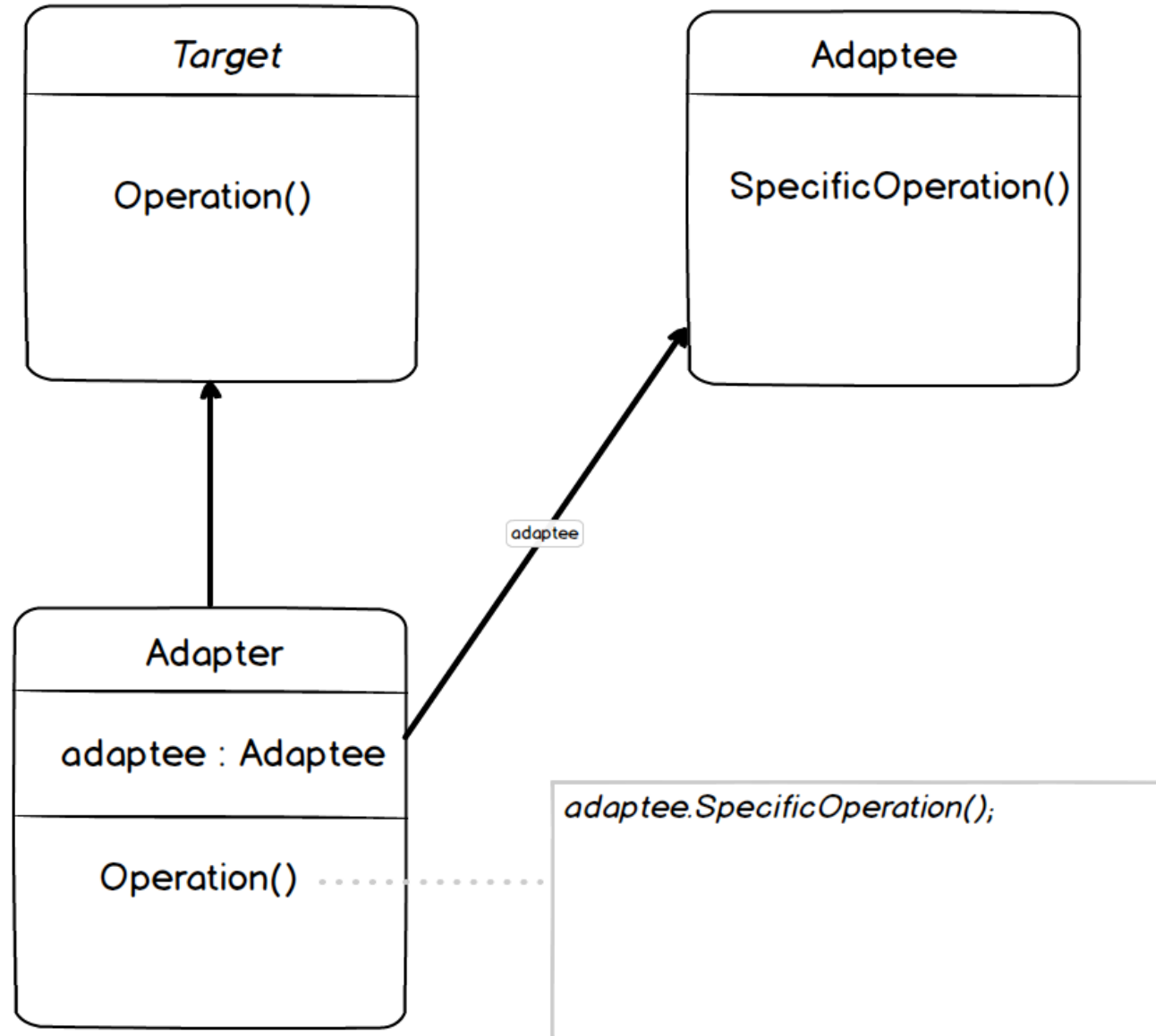
Pozwala produkować różne typy oraz **reprezentacje** obiektu używając tego samego kodu konstrukcyjnego.

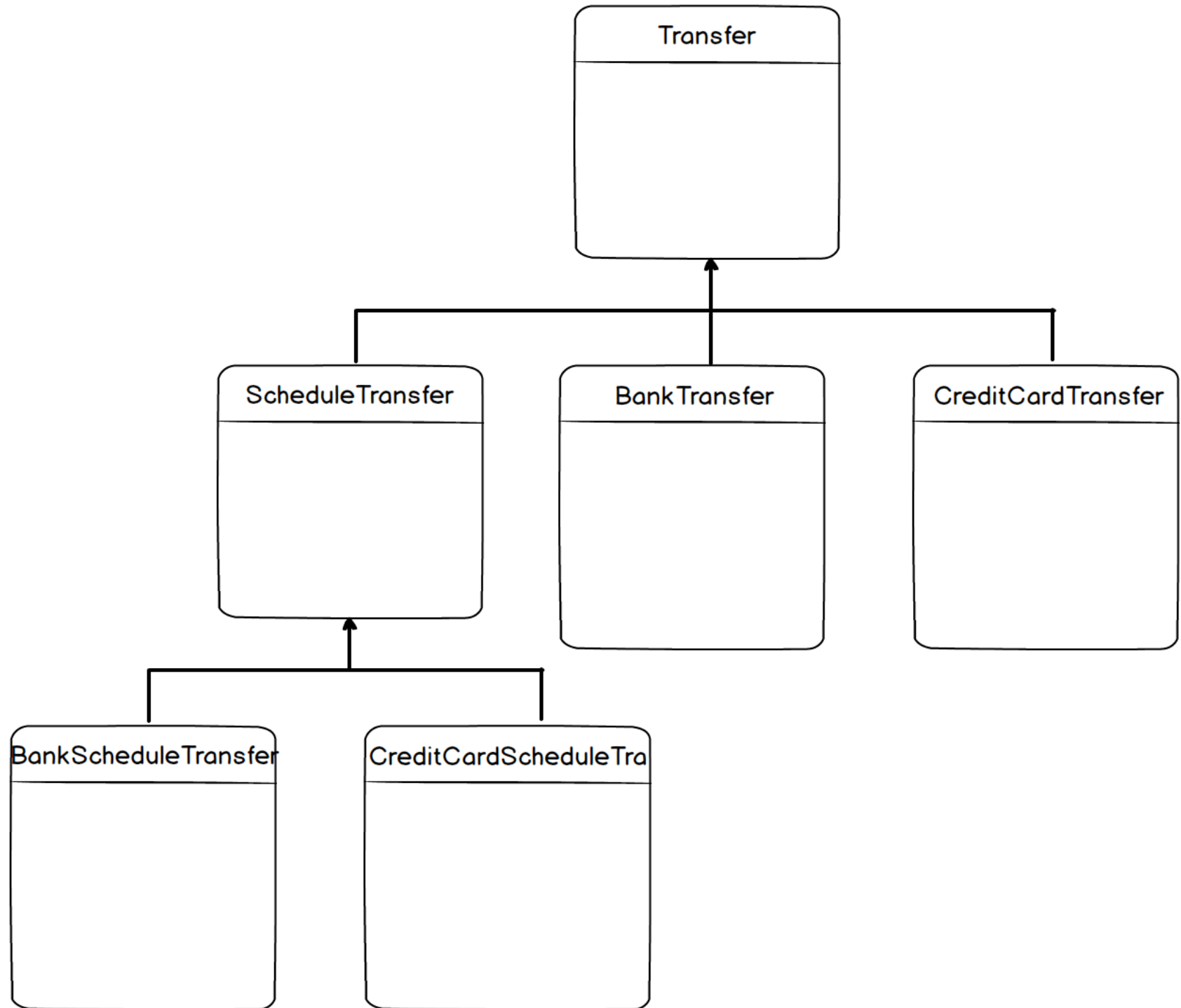
WZORCE STRUKTURALNE

Adapter Pattern

Pozwala na współdziałanie ze sobą
obiektów o niekompatybilnych interfejsach

Object Adapter





FEATURE

Transfer

ScheduleTransfer

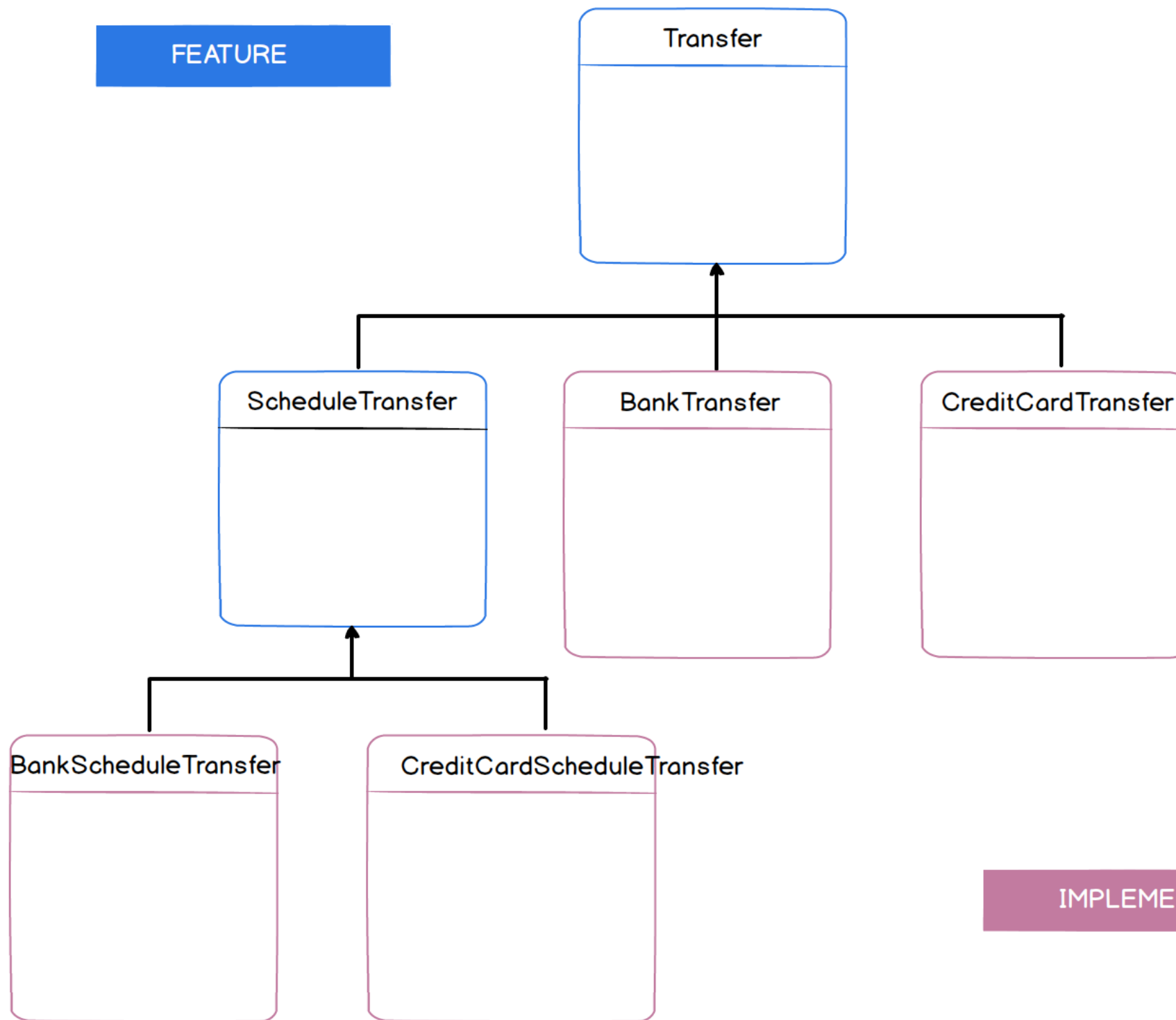
BankTransfer

CreditCardTransfer

BankScheduleTransfer

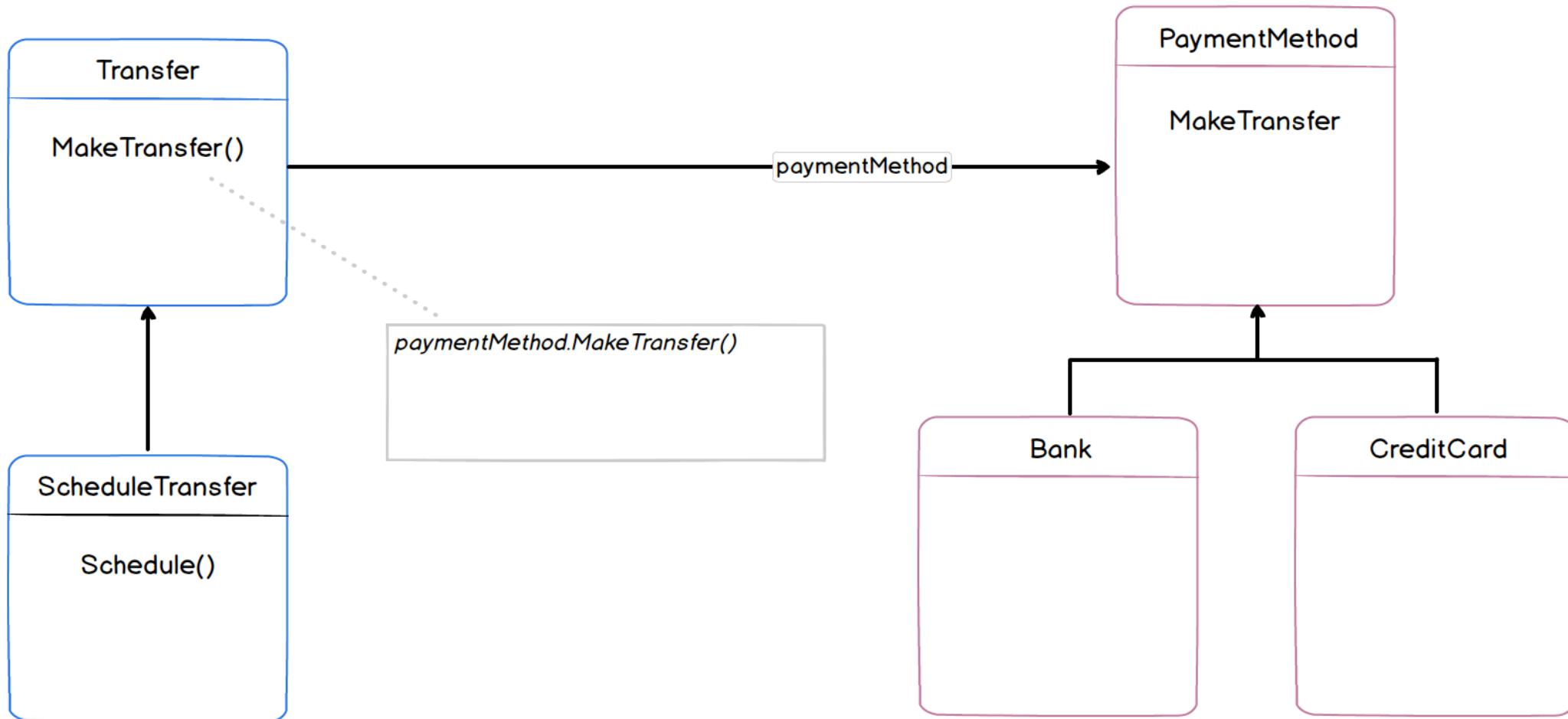
CreditCardScheduleTransfer

IMPLEMENTATION



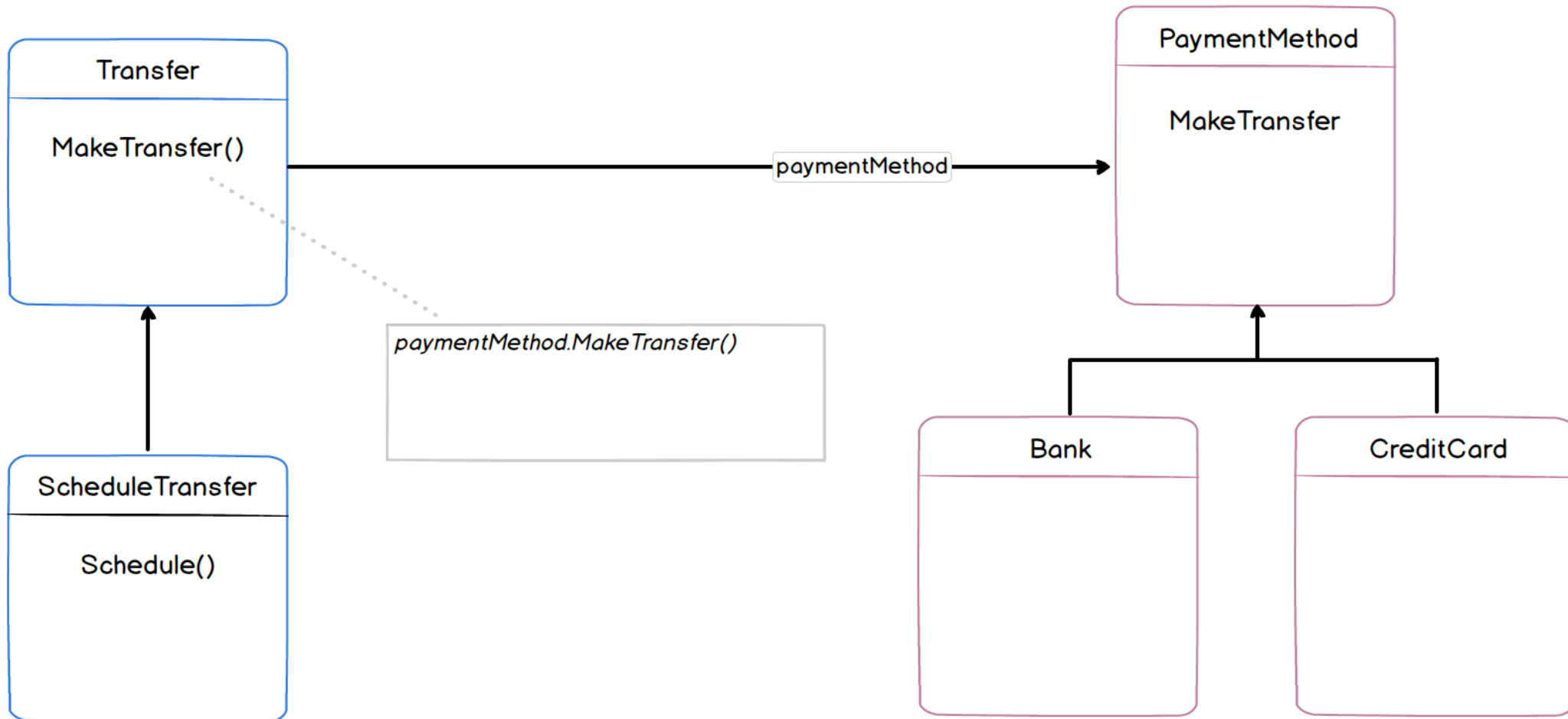
FEATURE

IMPLEMENTATION



FEATURE

IMPLEMENTATION



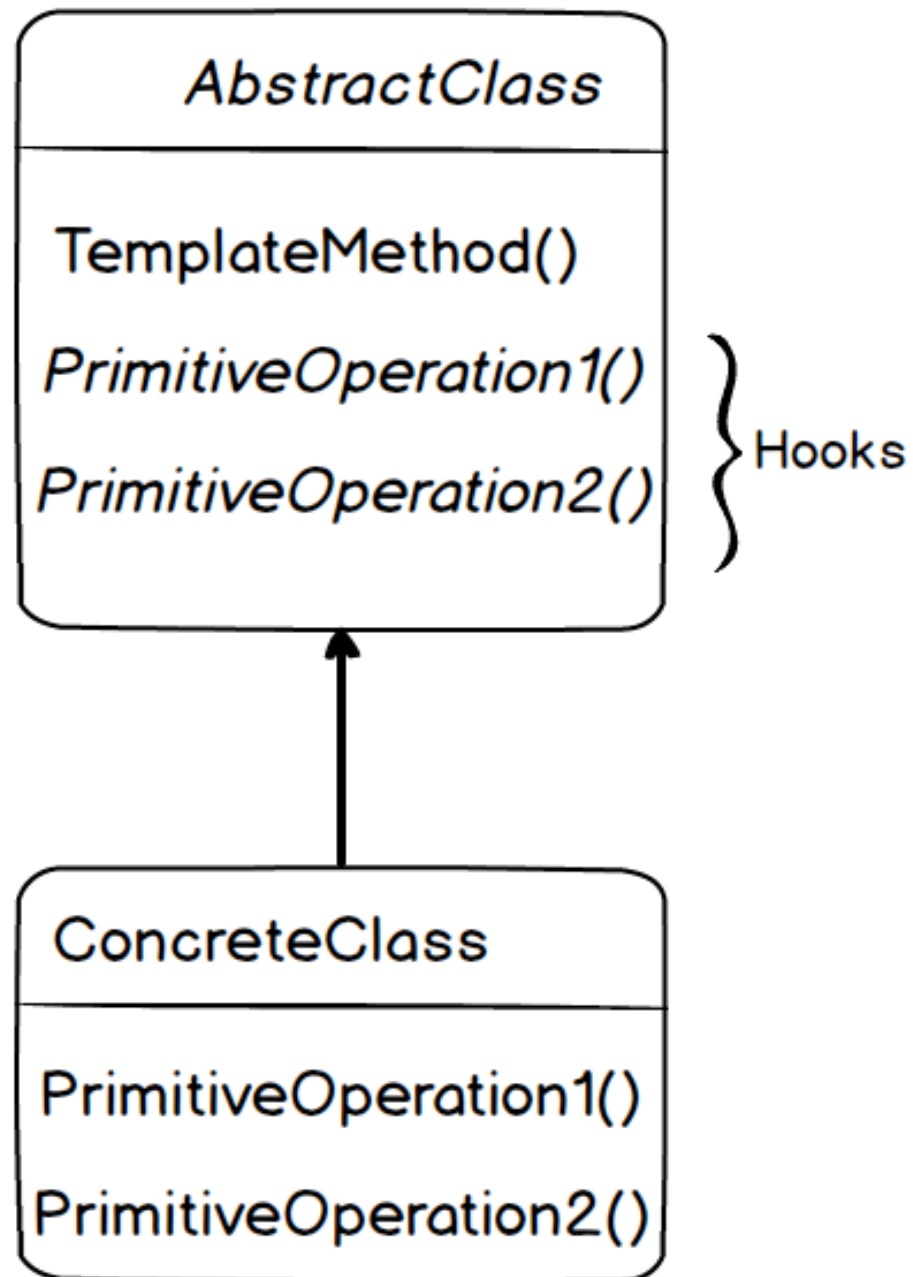
Decorator Pattern

Przedkładaj kompozycję nad dziedziczenie

WZORCE CZYNNOŚCIOWE

Template Method

Metoda, która jest **szkieletem** algorytmu,
a jego dokładna implementacja jest w klasach pochodnych.



Strategy Pattern

Definiuje rodzinę wymiennych
algorytmów.

Fasada Pattern

Ukrywa złożony zestaw klas
w uproszczony interfejs.

Command Pattern

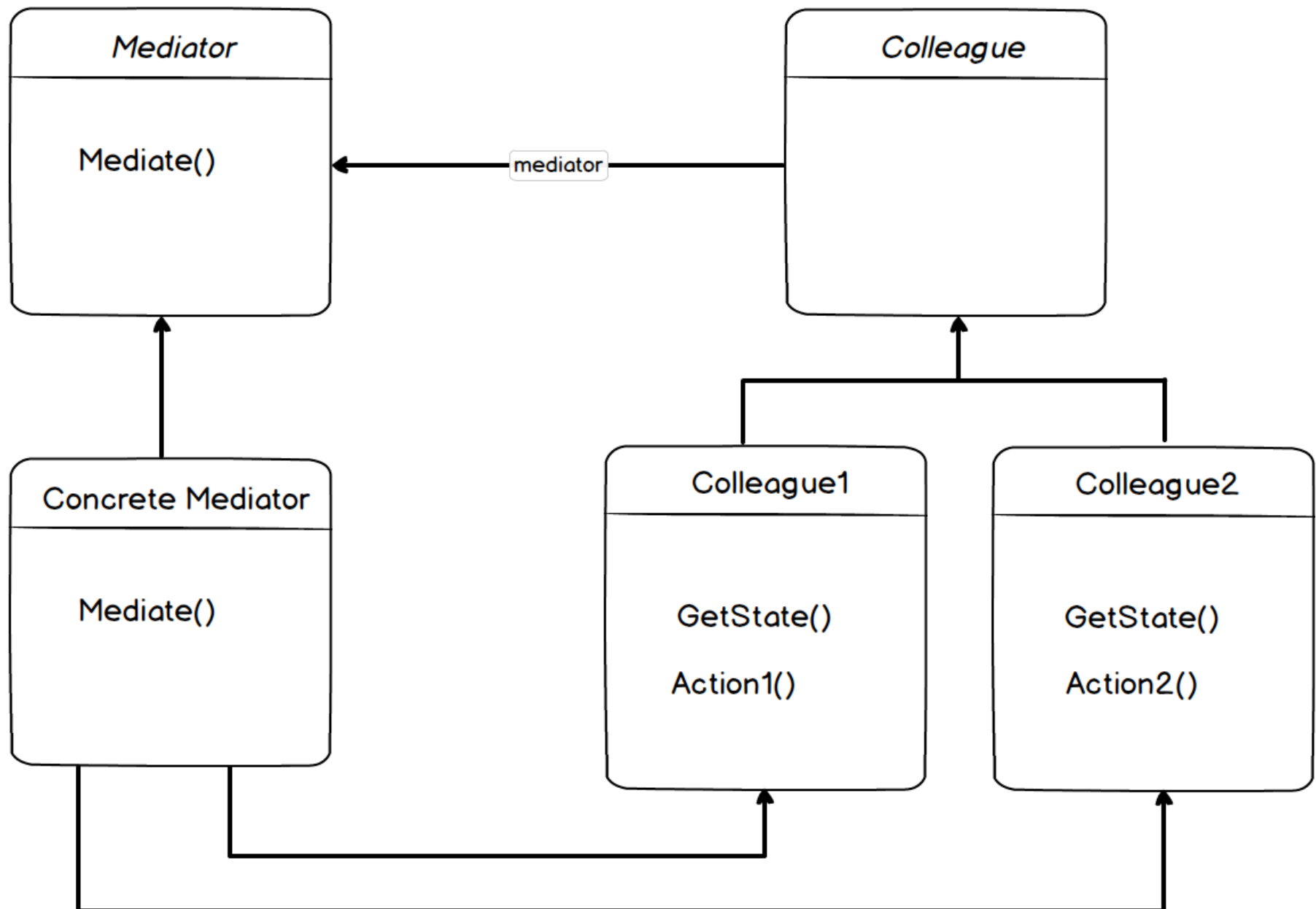
Traktujący żądanie wykonania
czynności jako **obiekt**.

Chain of Responsibility Pattern

Pozwala przekazywać żądania wzdłuż
łańcucha obiektów obsługujących

Mediator Pattern

Zapewnia jednolity interfejs do różnych elementów danego podsystemu



Interpreter Pattern

Umożliwia zdefiniowanie opisu gramatyki
języka interpretowalnego

Composite Pattern

Komponuje obiekty w
struktury drzewiaste

Iterator Pattern

Umożliwia dostęp do podobiektów
zgrupowanych w większym obiekcie

State Pattern

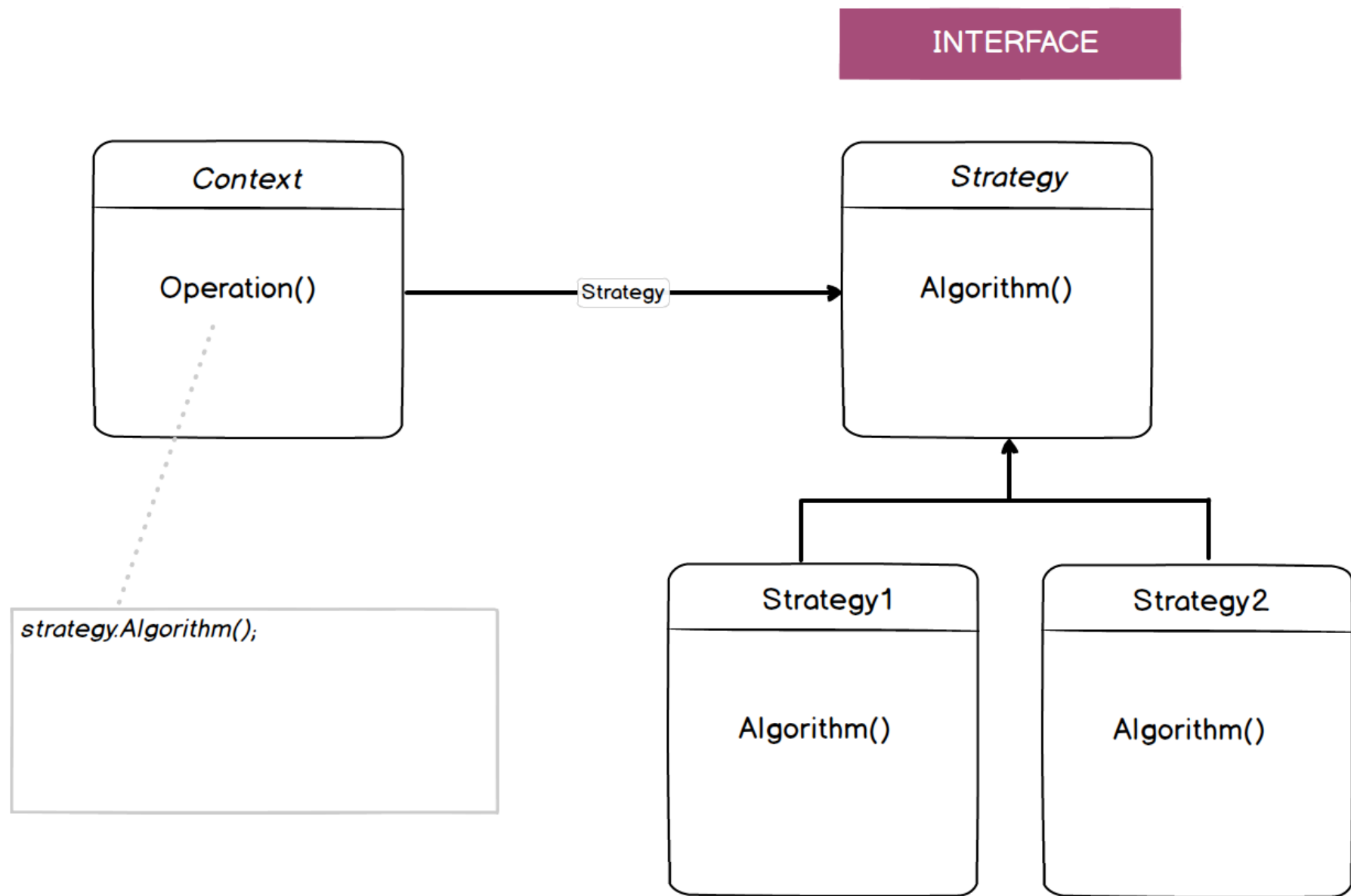
Pozwala obiektowi zmieniać swoje zachowanie
gdy zmieni się jego **stan wewnętrzny**

Memento Pattern

Umożliwia **zapamiętanie** stanu obiektu i
przywrócenie zapamiętanego stanu obiektu

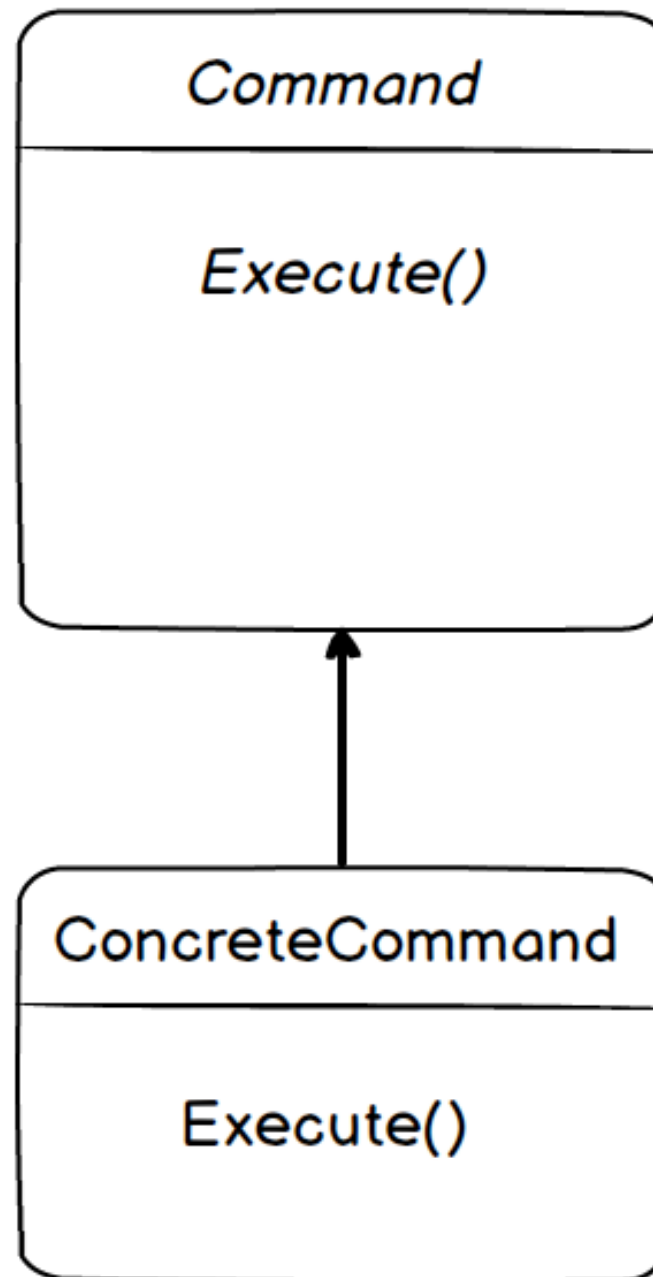
Null Object Pattern

Umożliwia uniknięcie sprawdzenia,
czy wartość jest różna od null



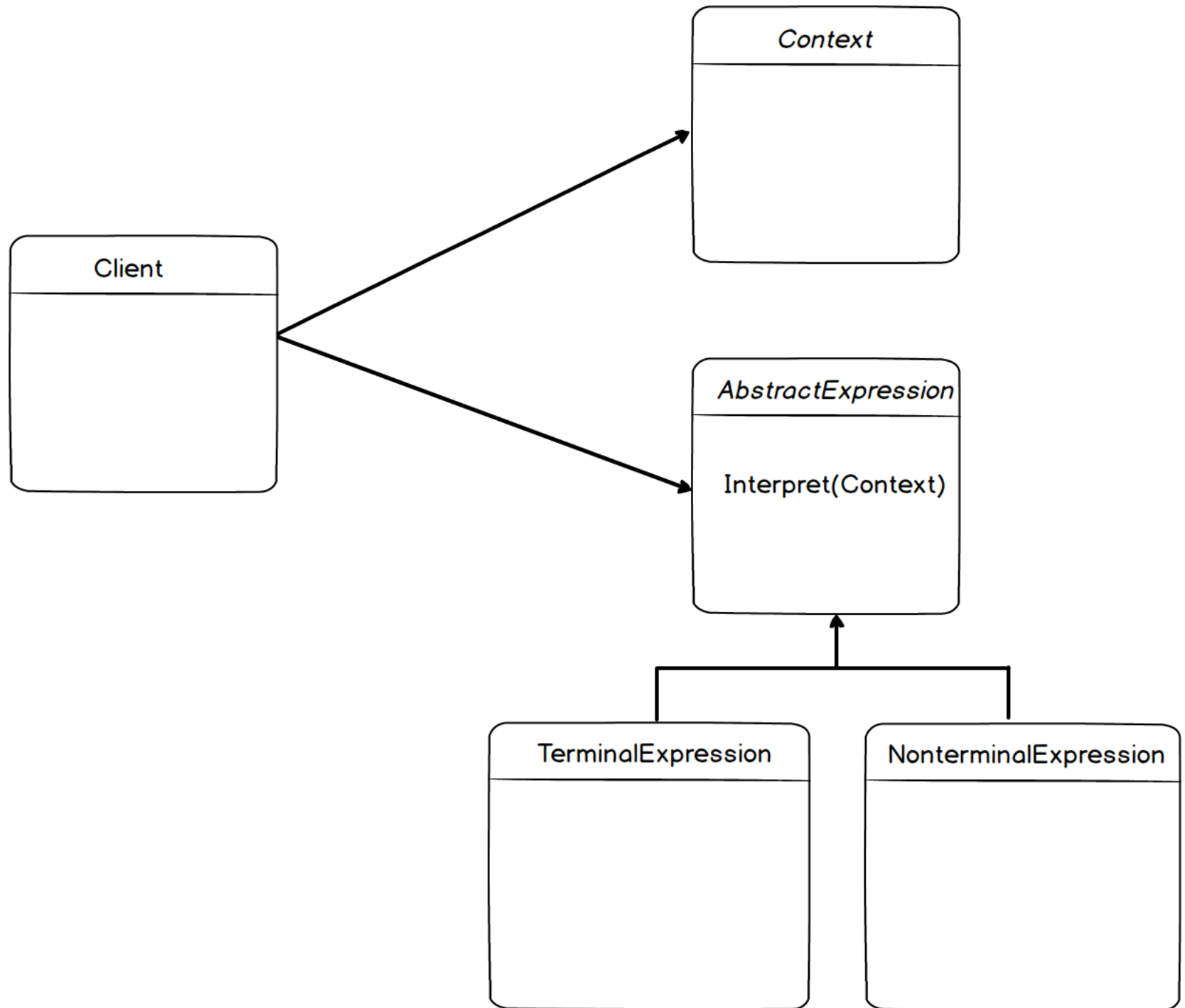
Observer Pattern

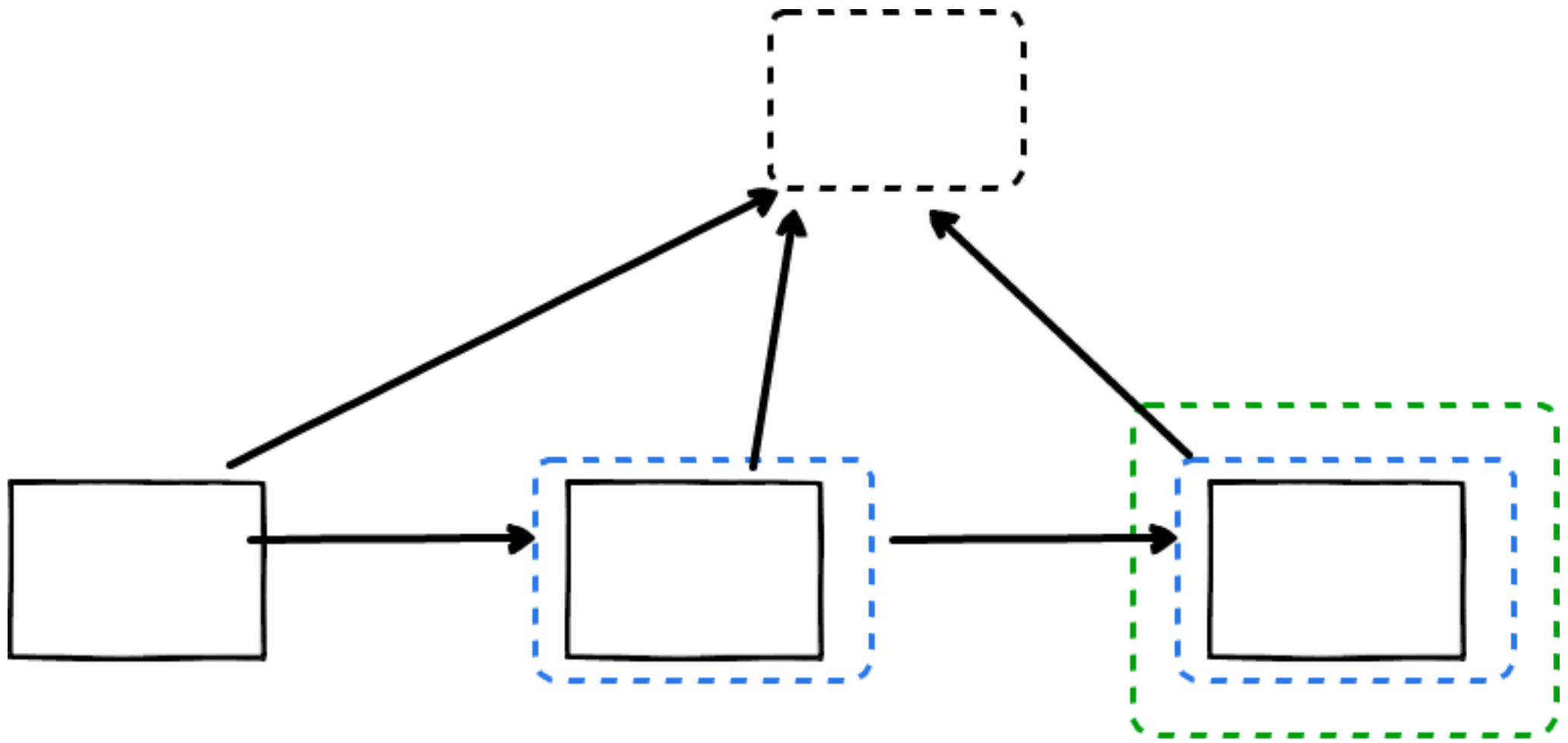
Powiadamiania zainteresowane obiekty
o zmianie stanu innego obiektu



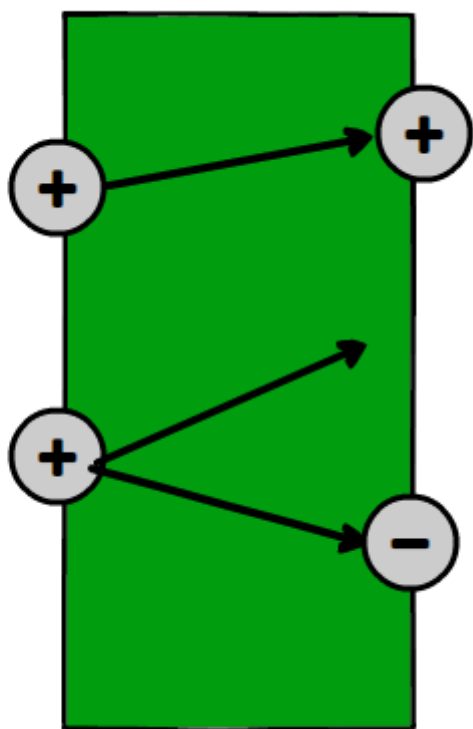
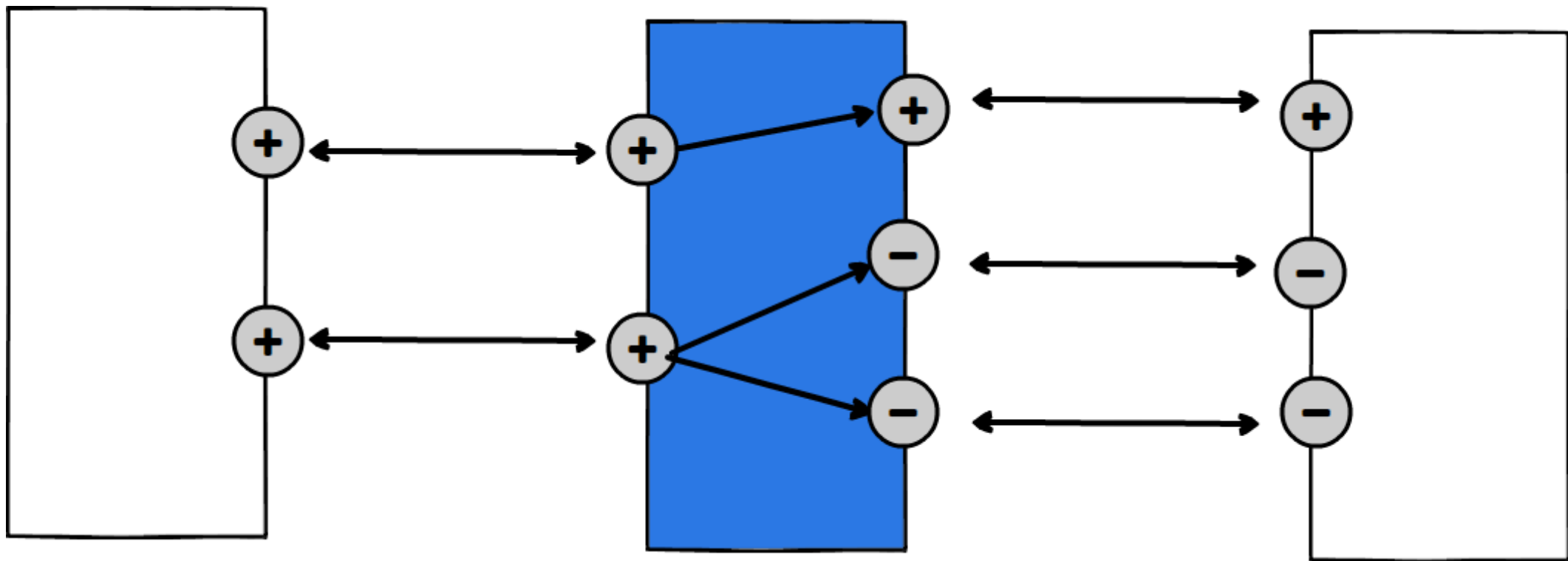
Visitor Pattern

Umożliwia odseparowanie algorytmu od struktury obiektowej na której operuje



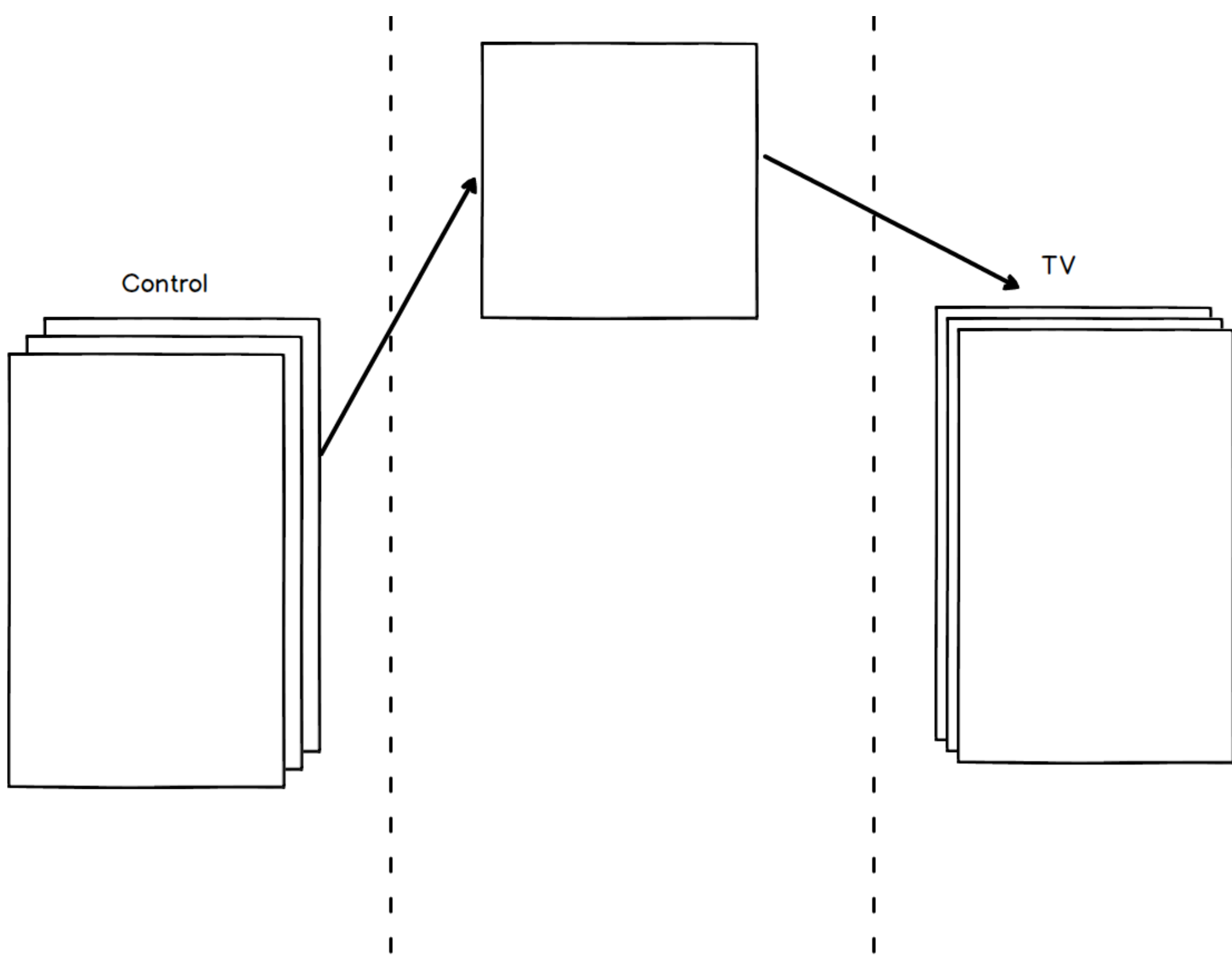


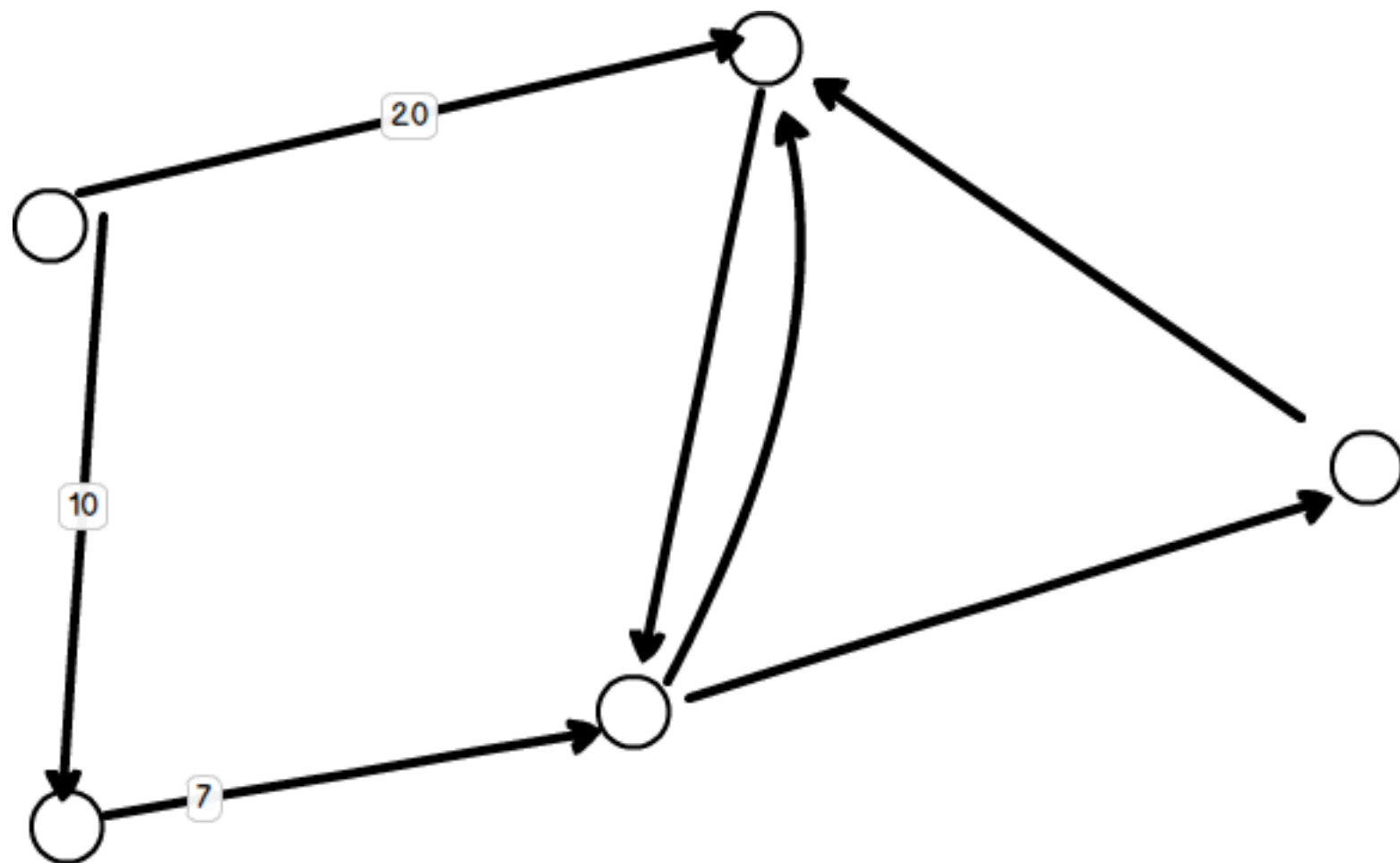
ls | grep .txt | foo.exe



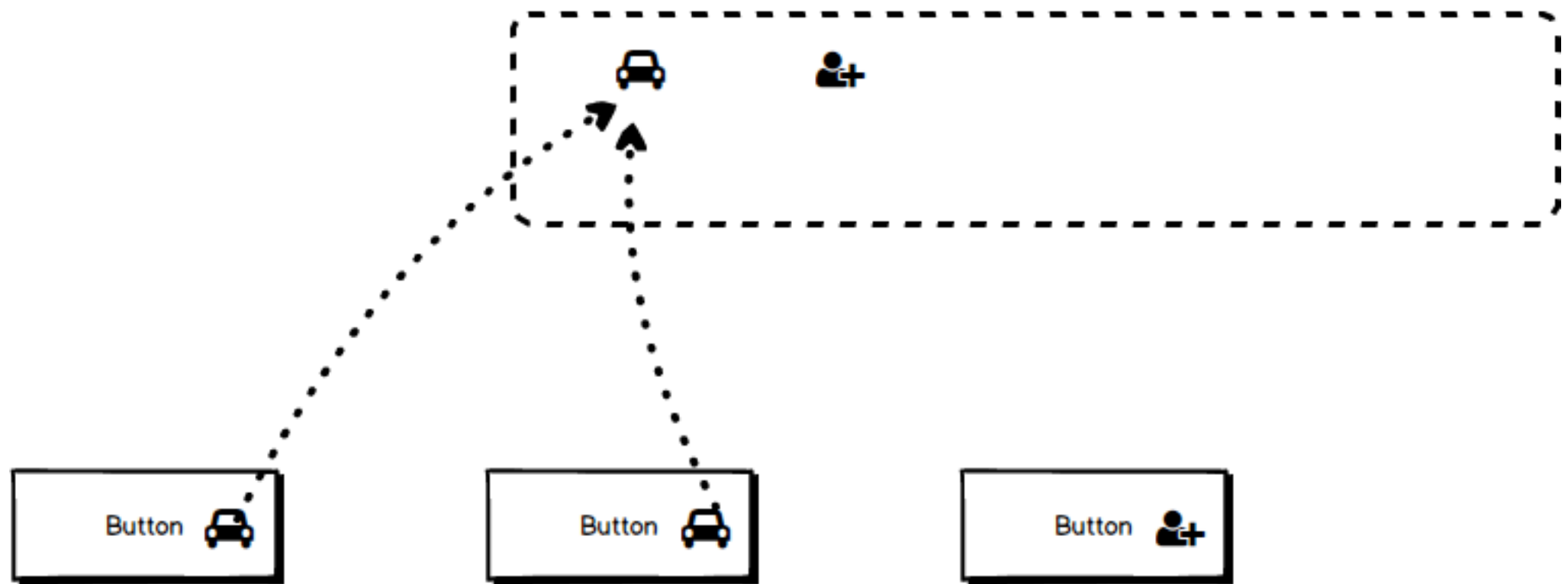
T Factory.Create<T>()

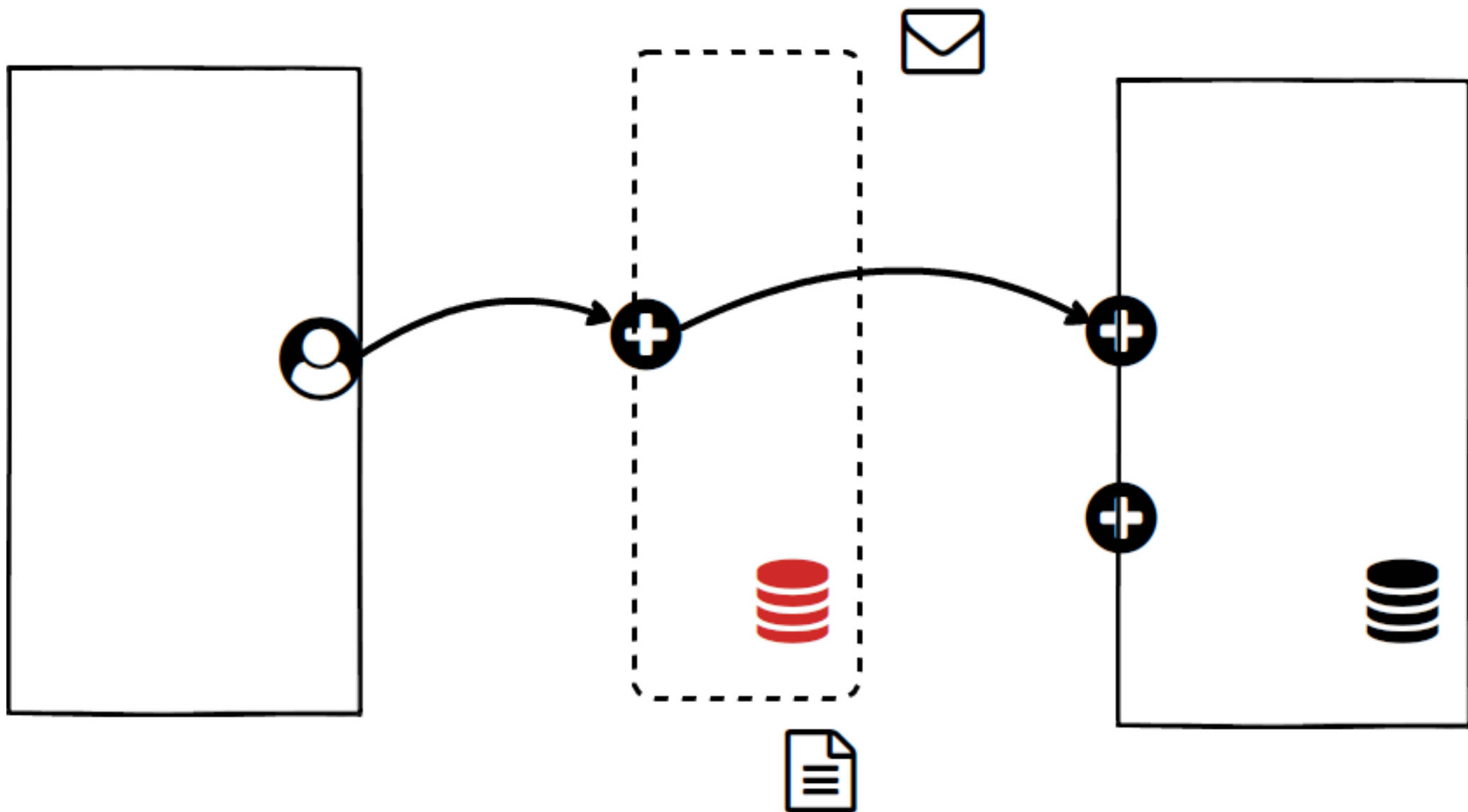
Type	TargetType	Cykl życia
ConfigManager	ConfigManager	Singleton
IProductService	ProductService	Transient



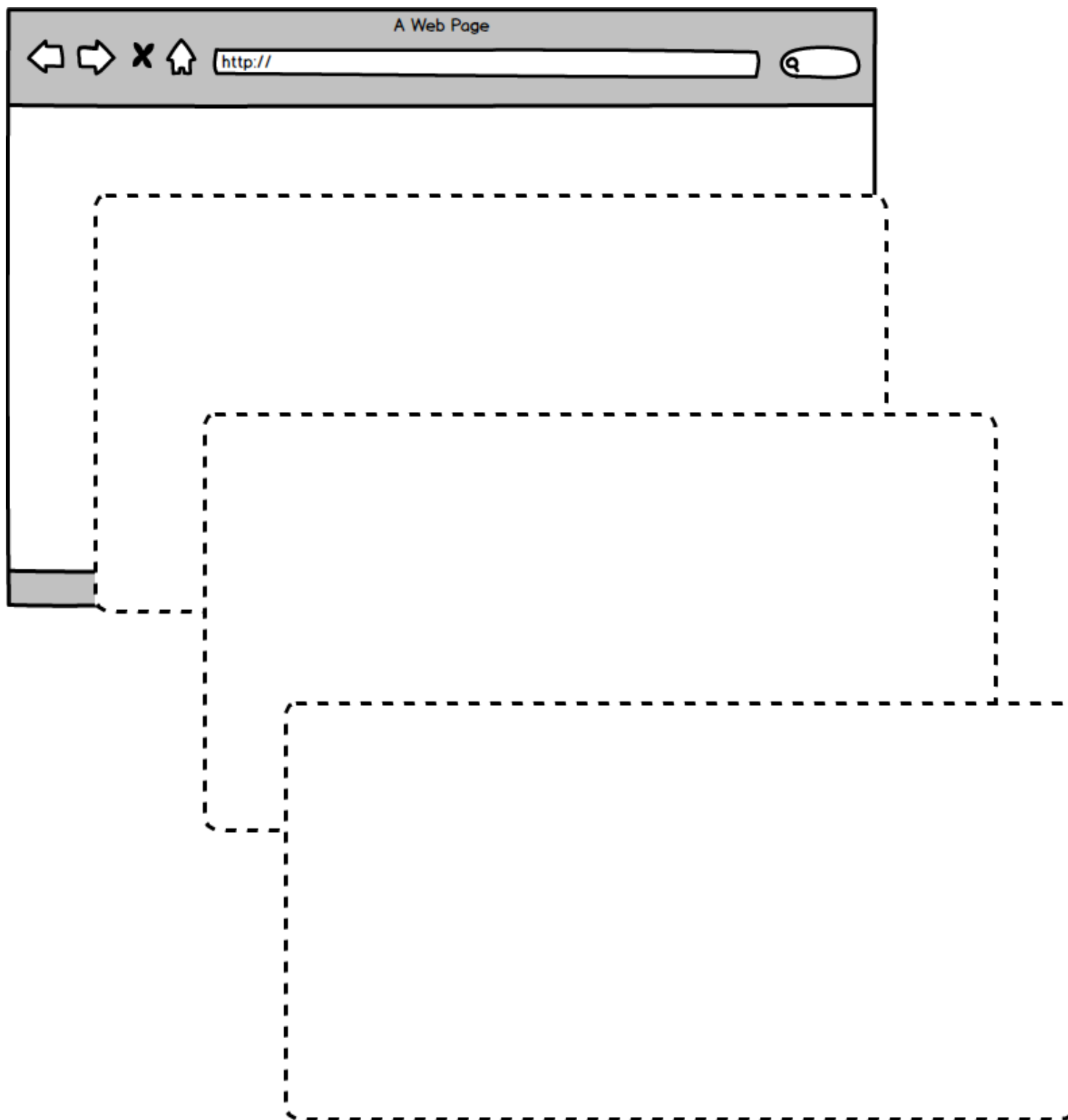


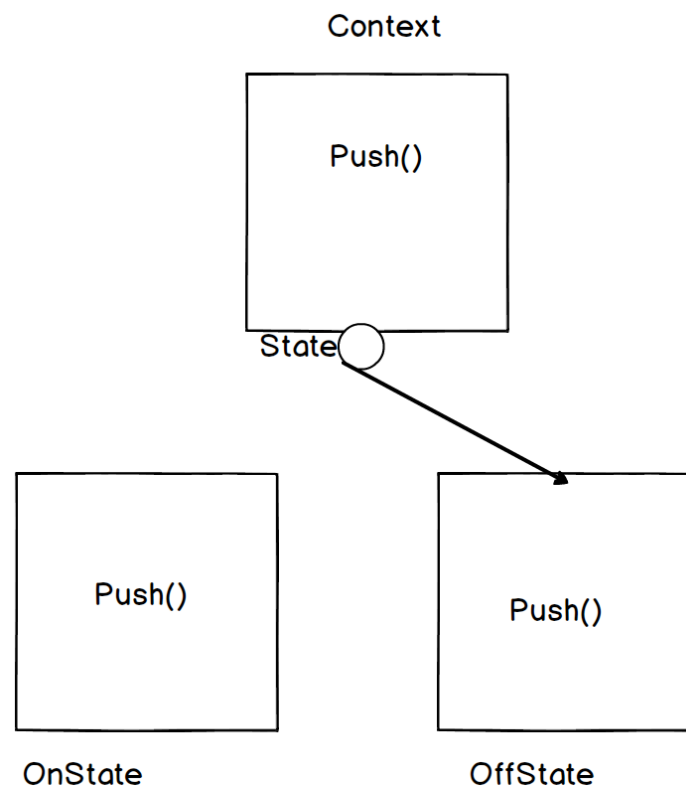
StaticResource



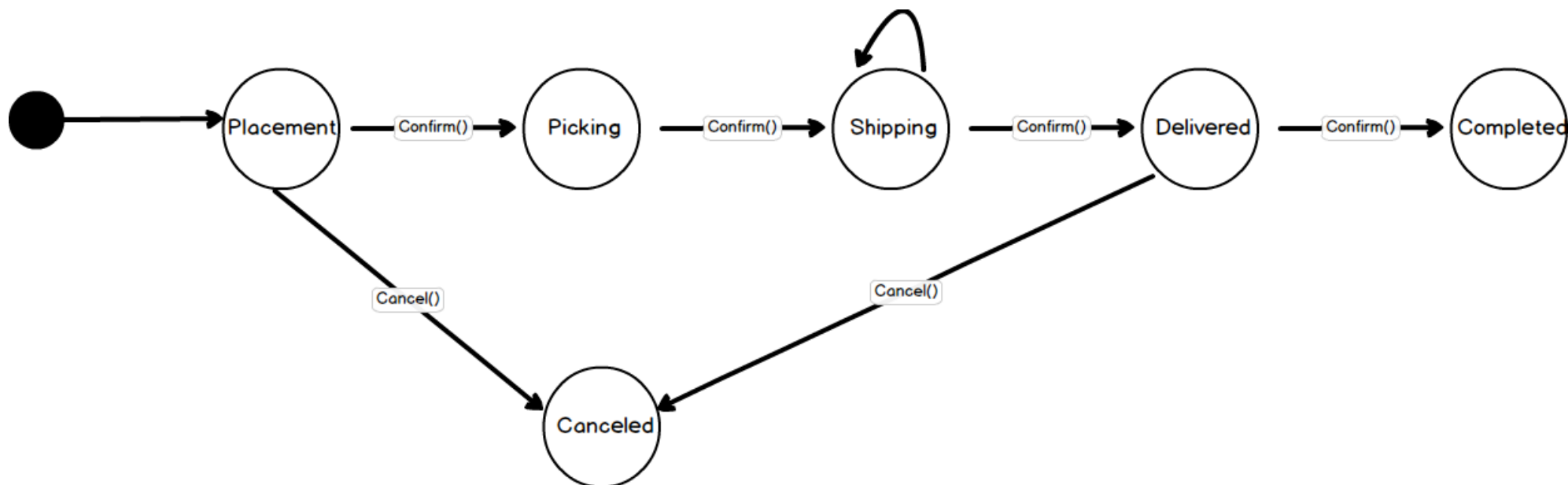


Order.Customer.Photo

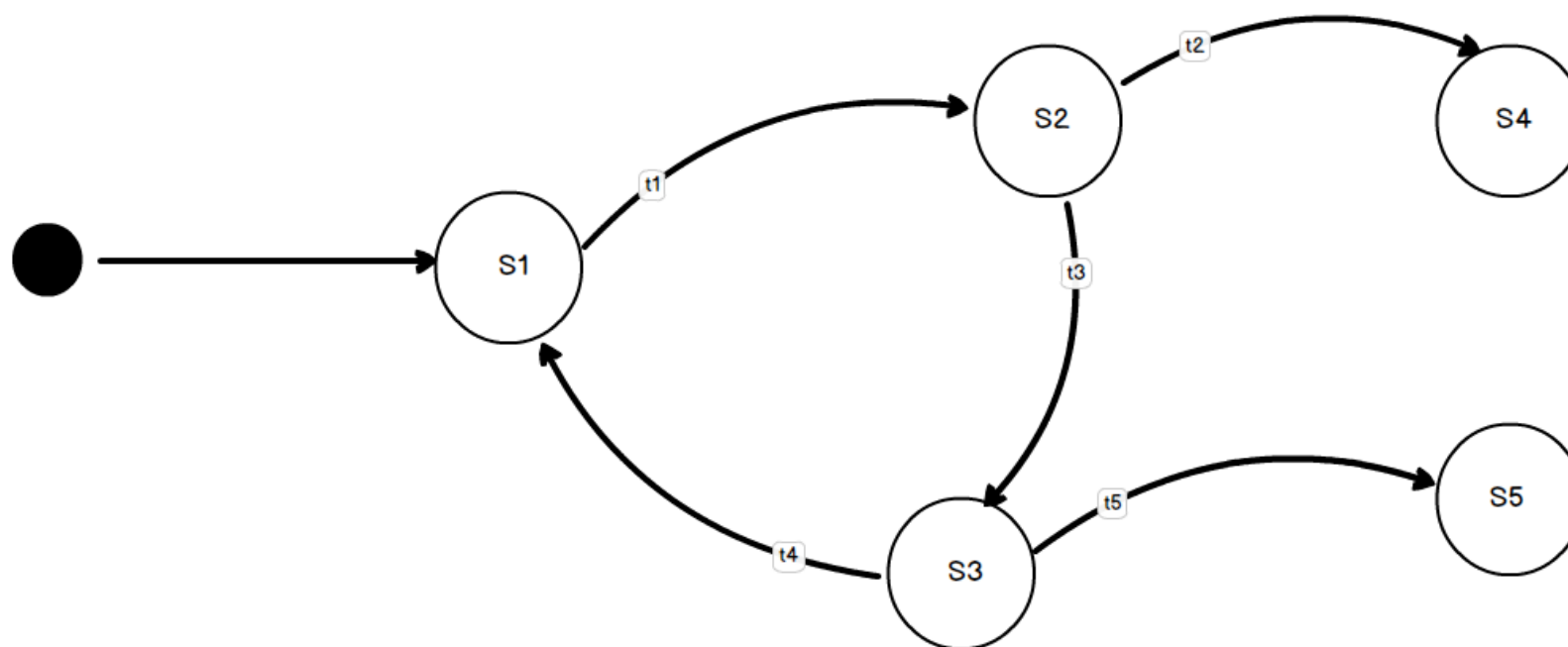


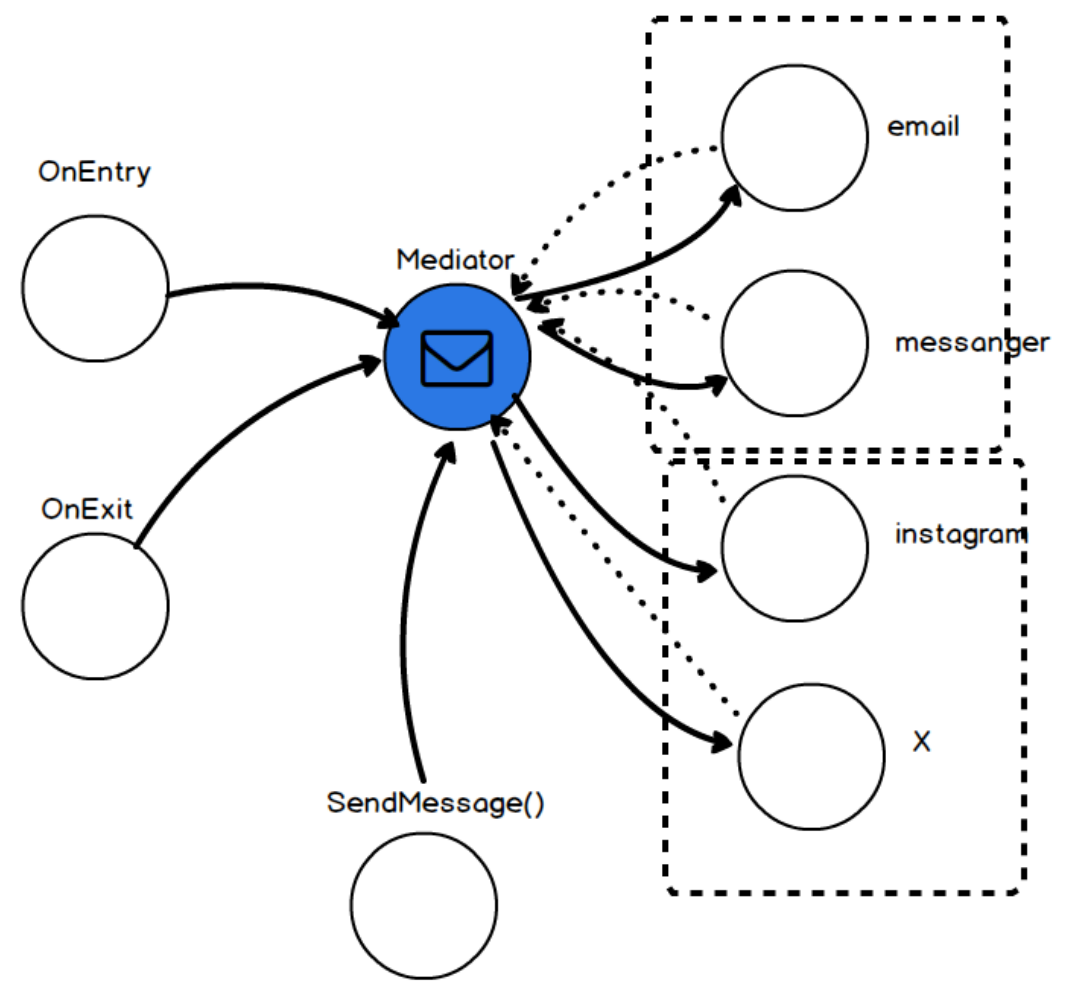
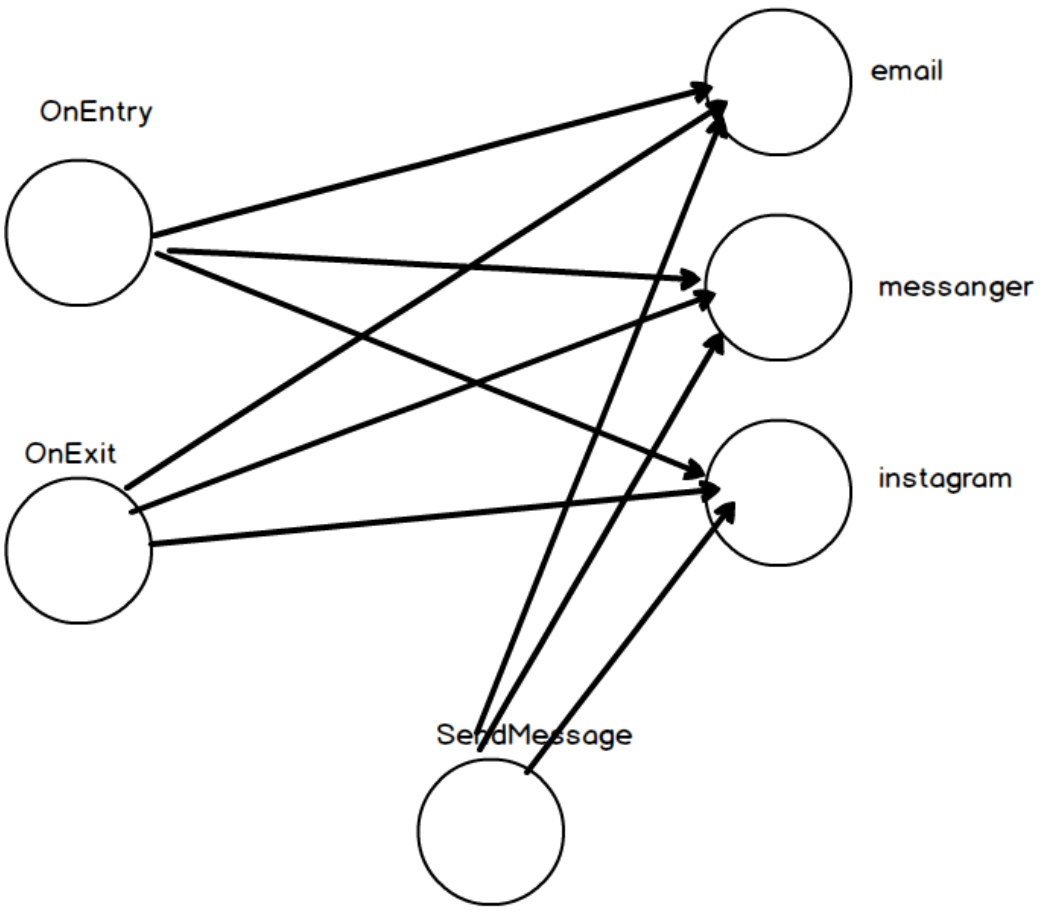


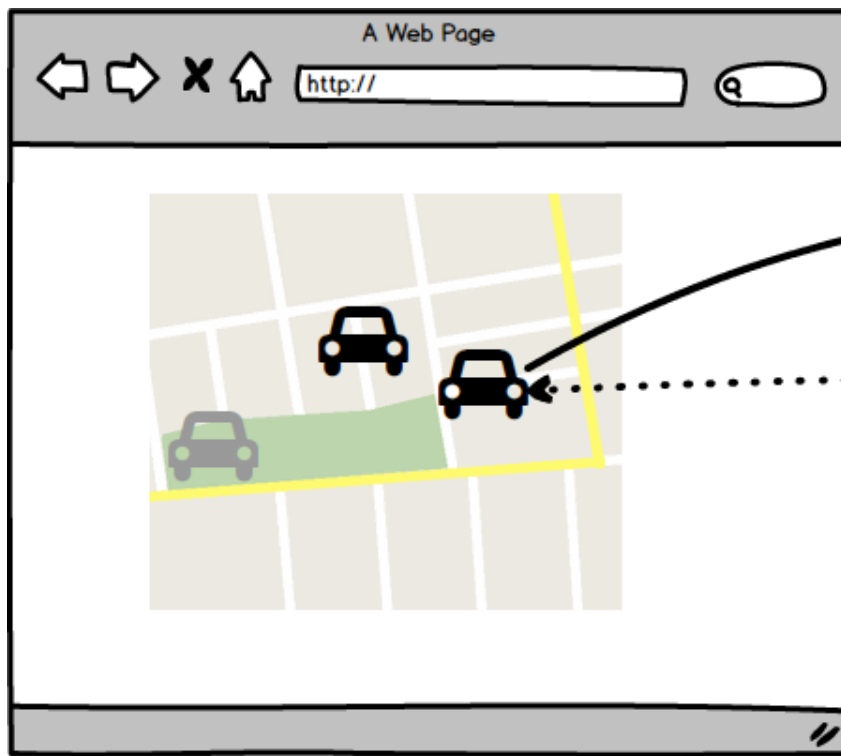
PlacementState



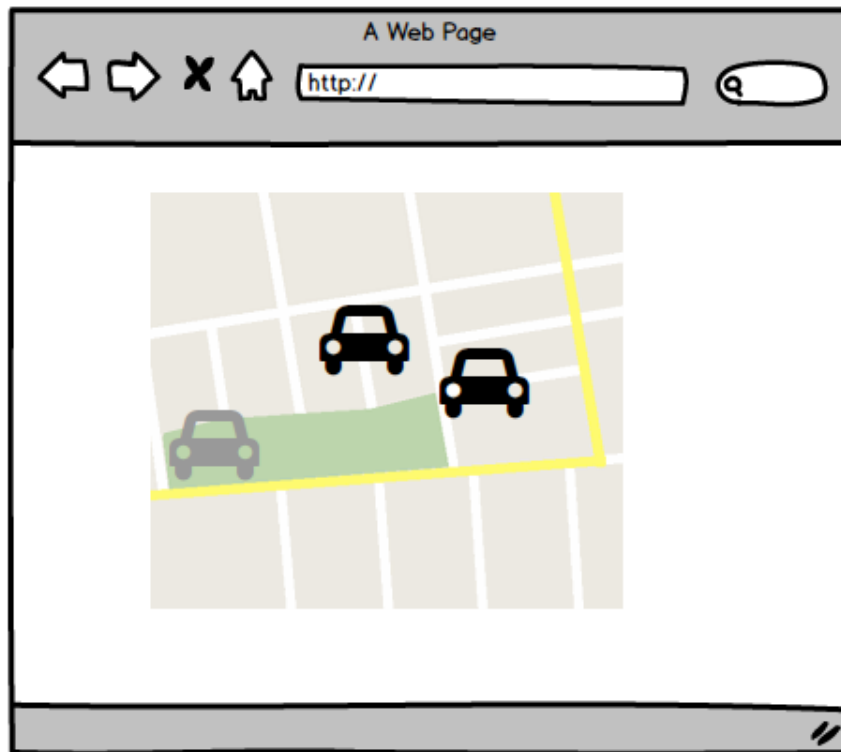
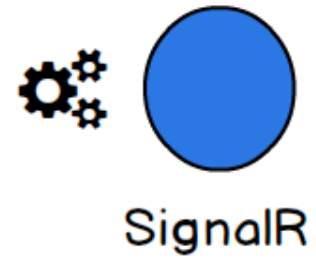
Maszyna stanów skończonych (automat skończony)





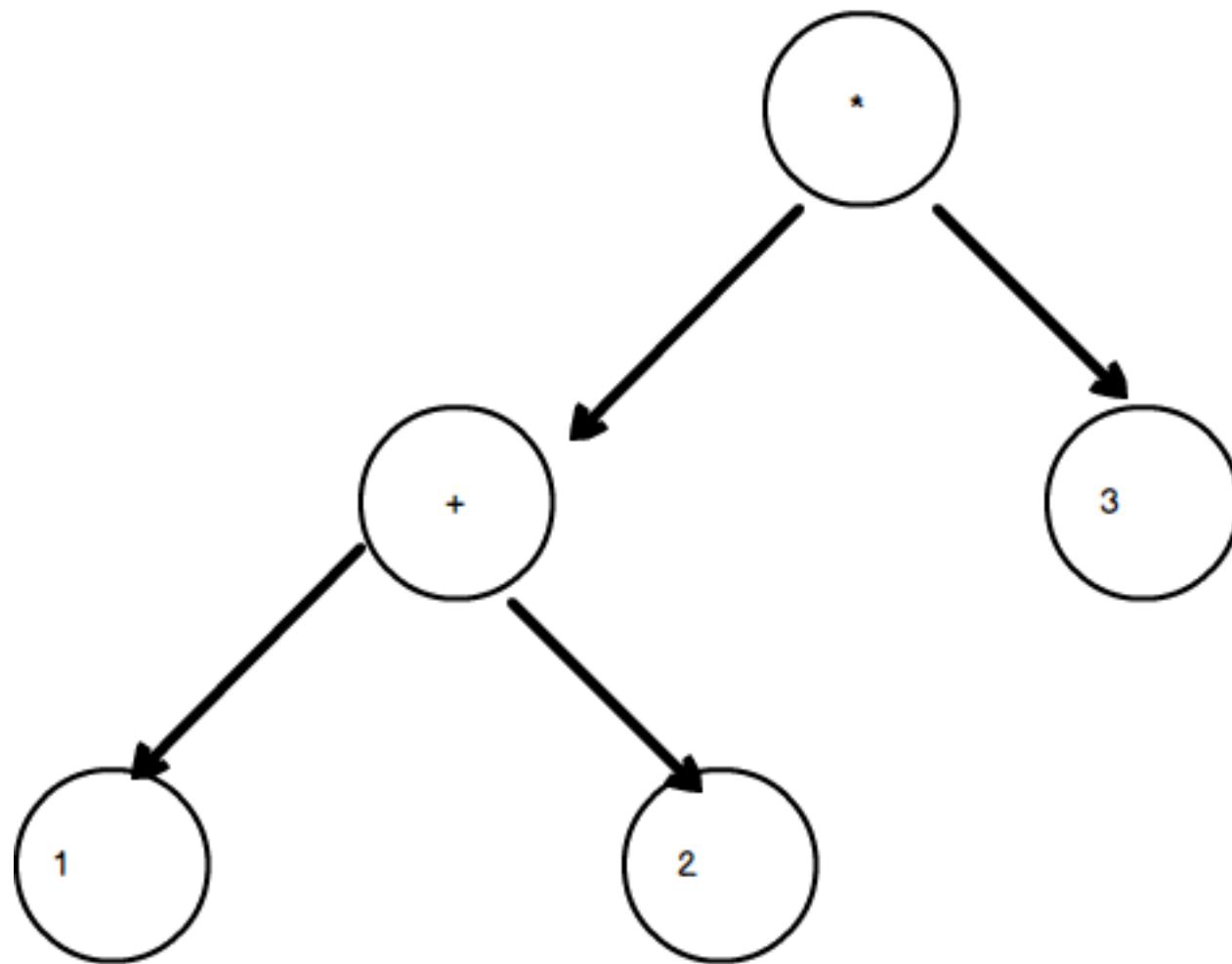


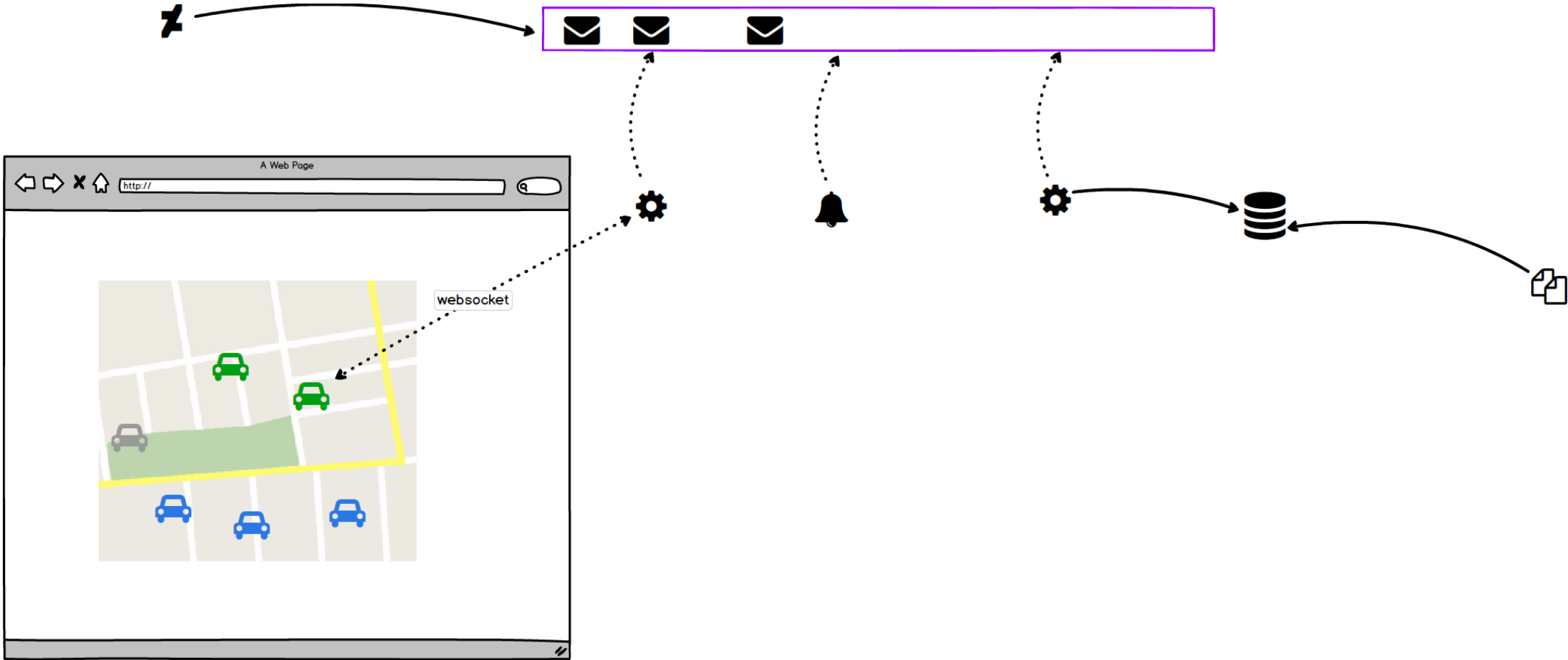
websocket

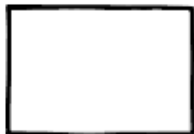


IQueryable

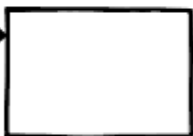
Expression



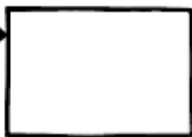




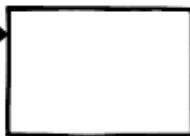
next



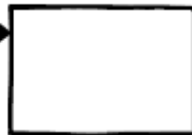
next

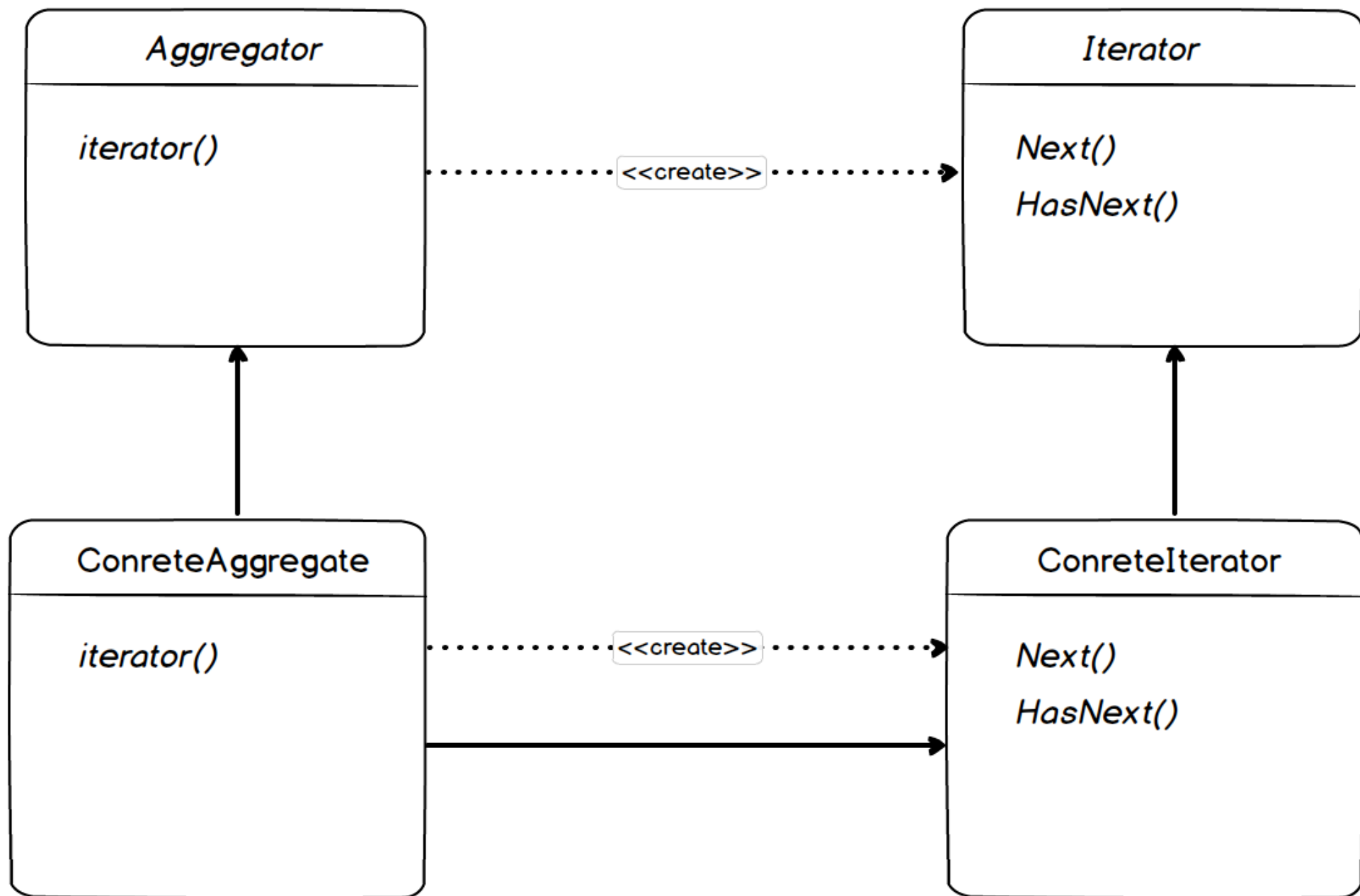


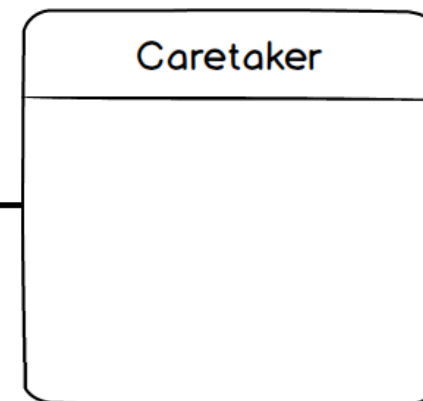
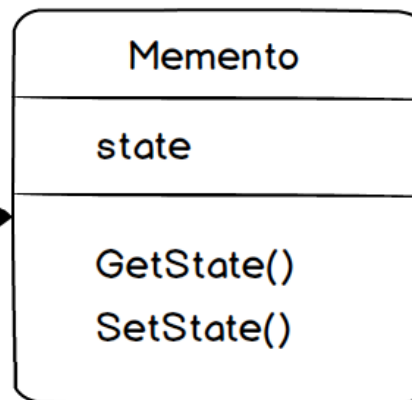
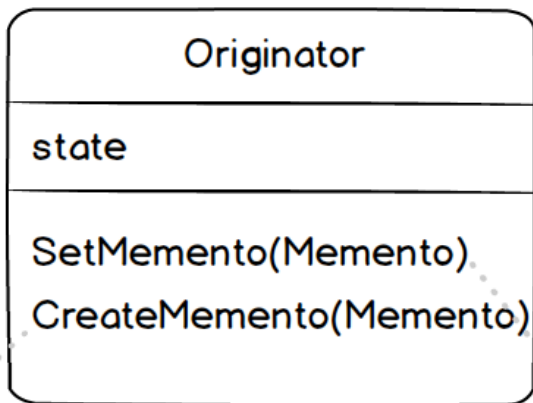
next



next

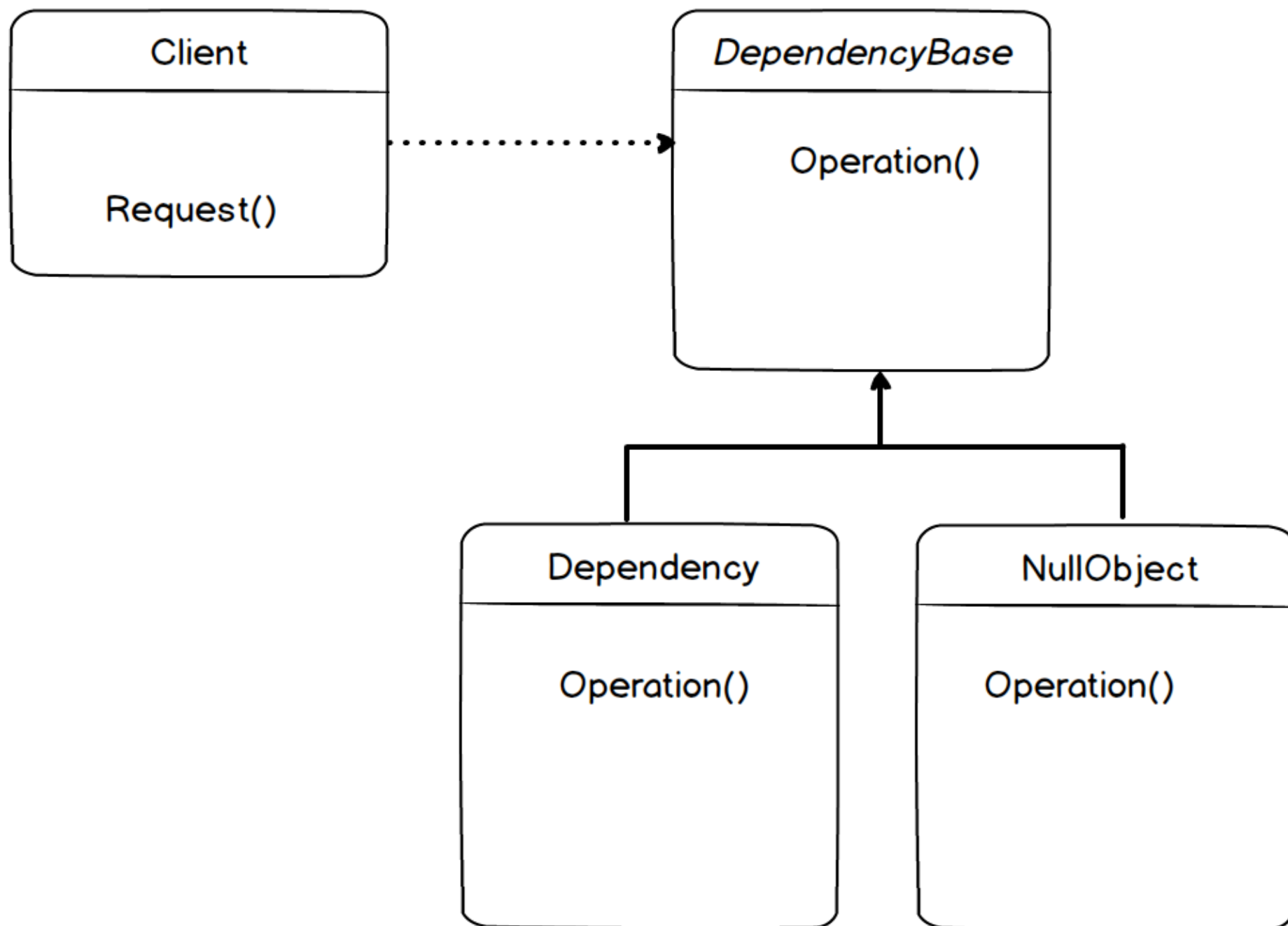


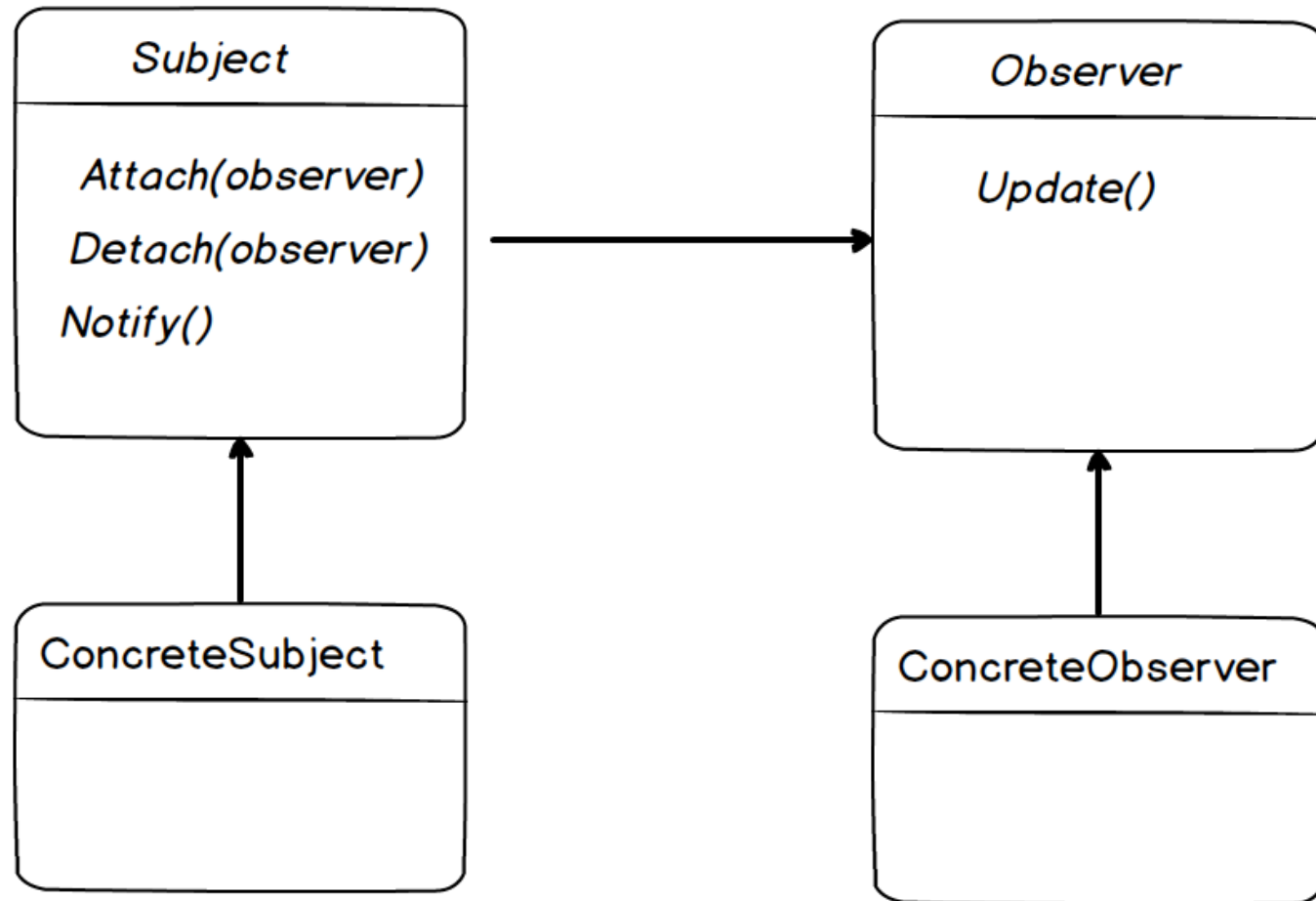




return new Memento(state)

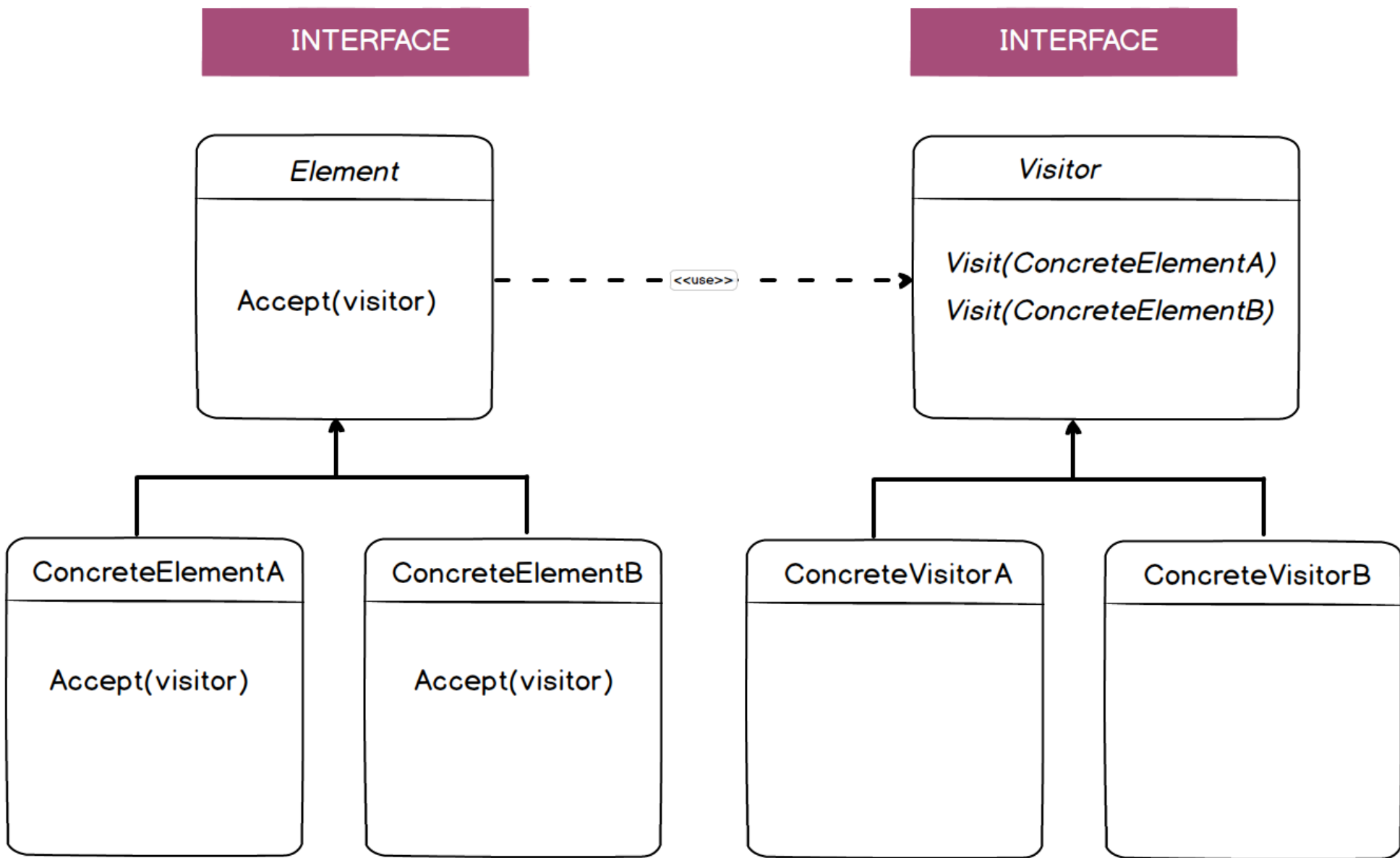
state = memento.GetState()

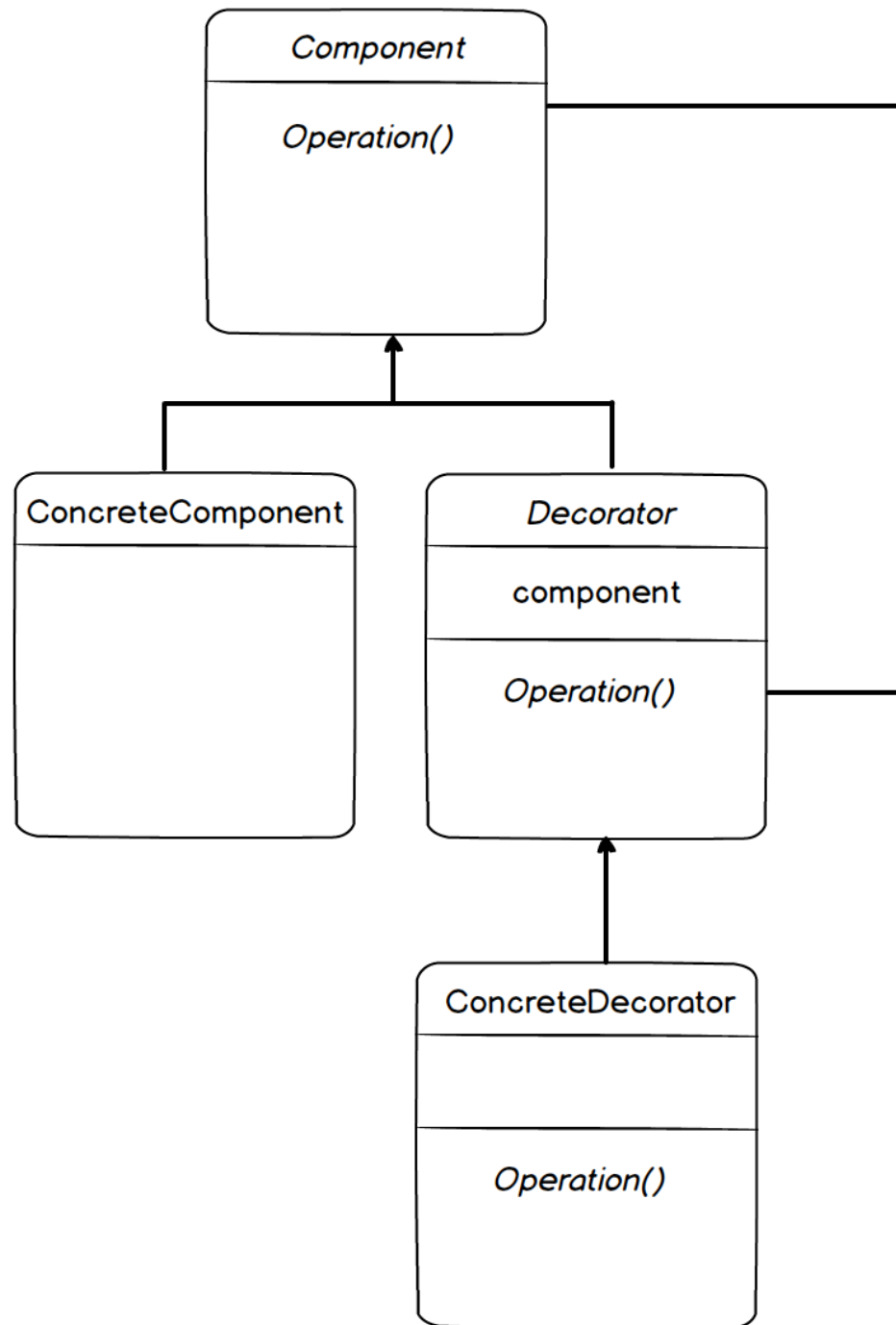




Flyweight Pattern

Pozwala zmieścić więcej obiektów w pamięci poprzez współdzielenie części opisu ich stanów





Proxy Pattern

Pozwala na stworzenie obiektu zastępczego
w miejsce innego obiektu.

INTERFACE

