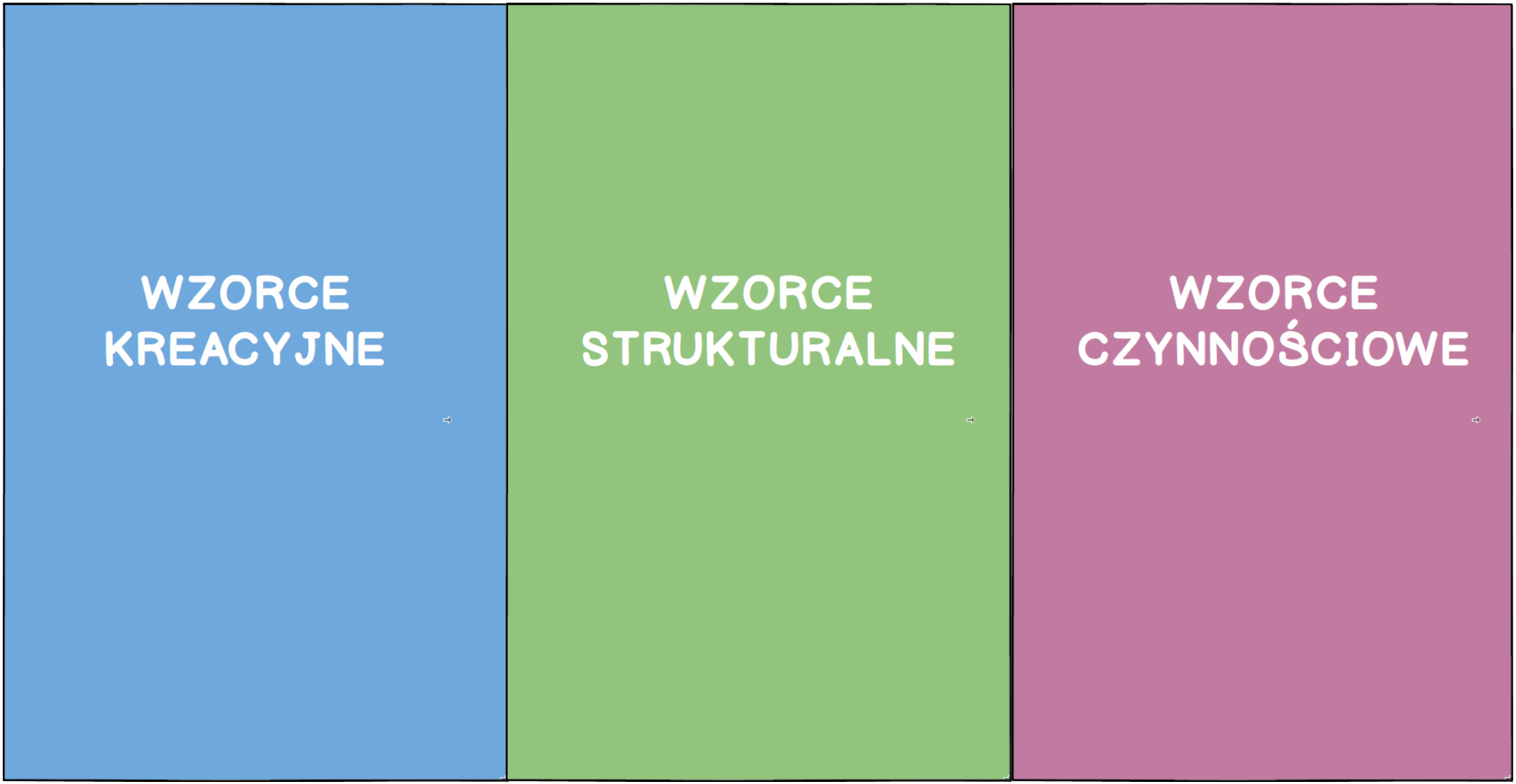
WZORCE PROJEKTOWE



DRY

Don't Repeat Yourself

Nie powtarzaj się

KISS

Keep It Simple, Stupid

Zachowaj prostotę, głupcze

YAGNI

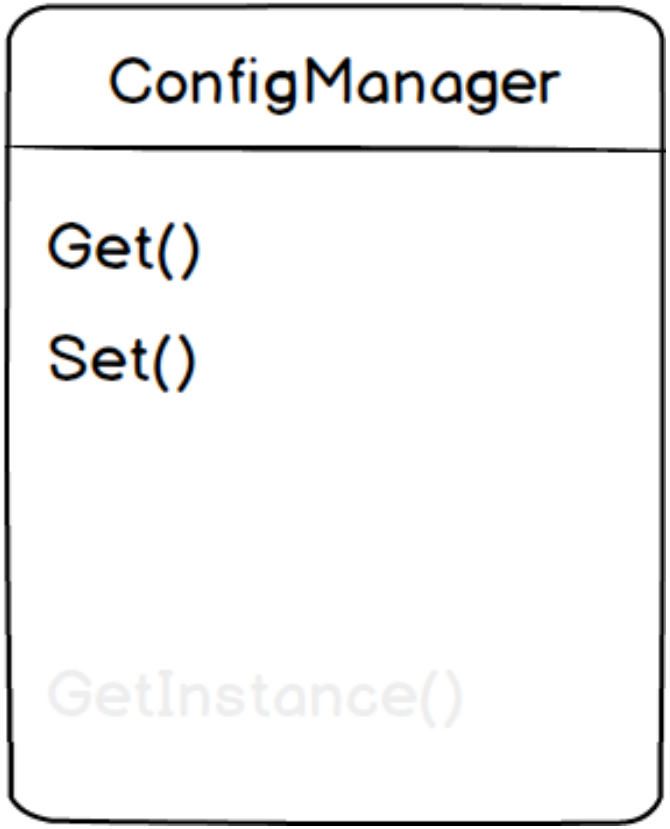
Aren't Going to Need It

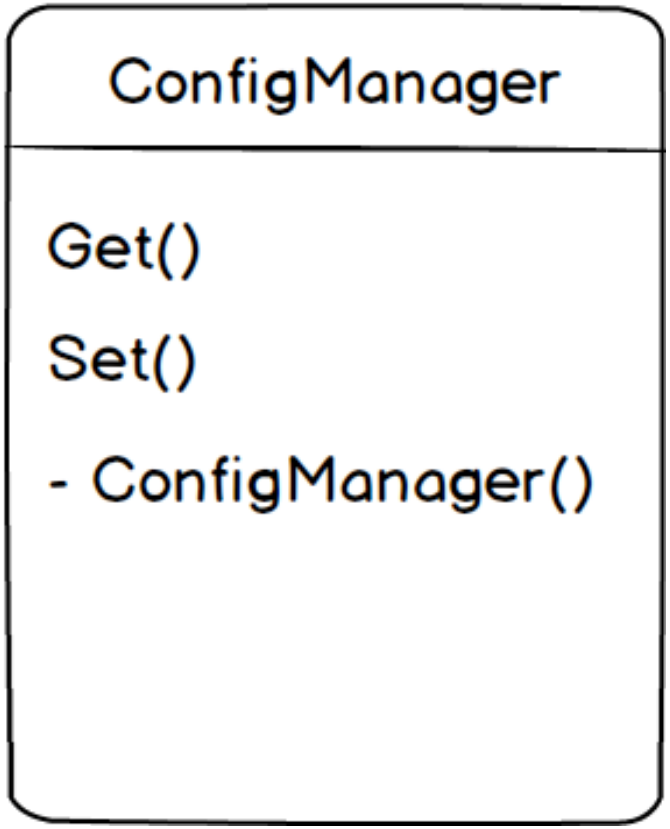
Nie będziesz tego potrzebować

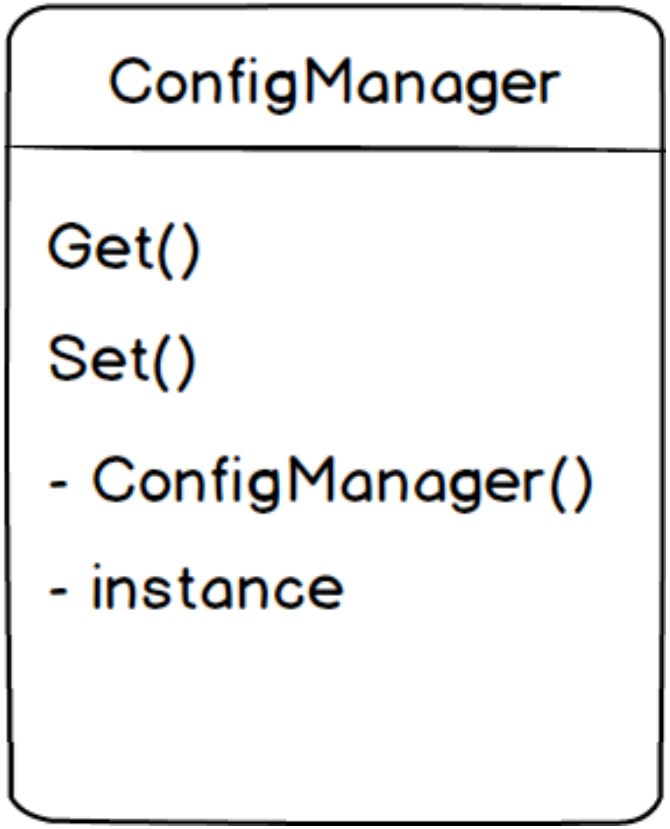
WZORCE KREACYJNE

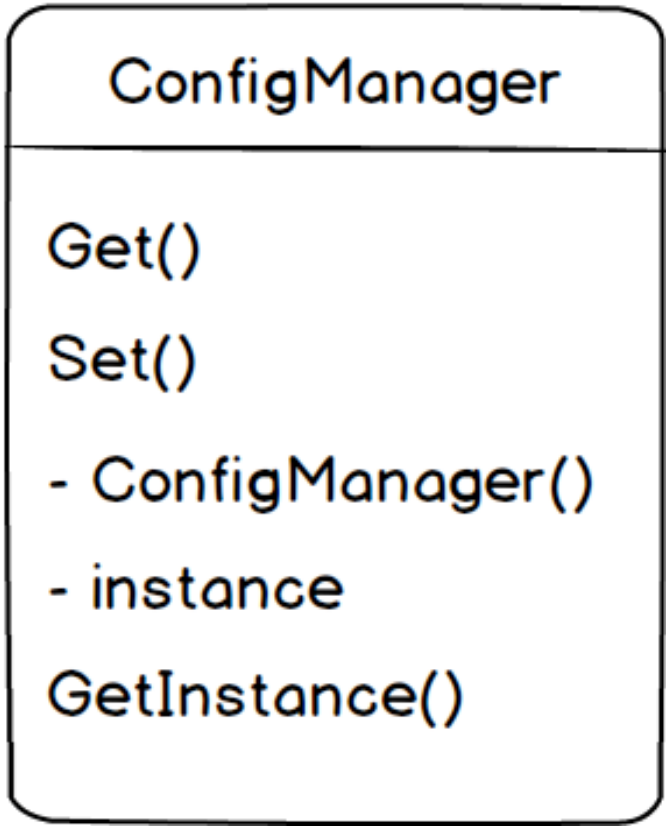
Singleton Pattern

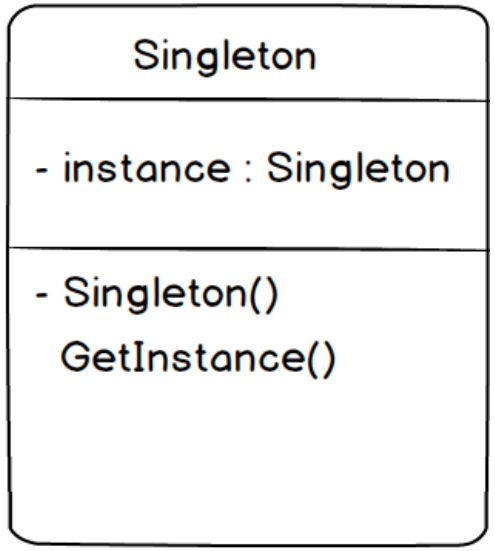
Zapewnia, że klasa posiada
pojedynczą instancję


ConfigManager
Get()
Set()
GetInstance()


ConfigManager
Get()
Set()
- ConfigManager()


ConfigManager
Get()
Set()
- ConfigManager()
- instance

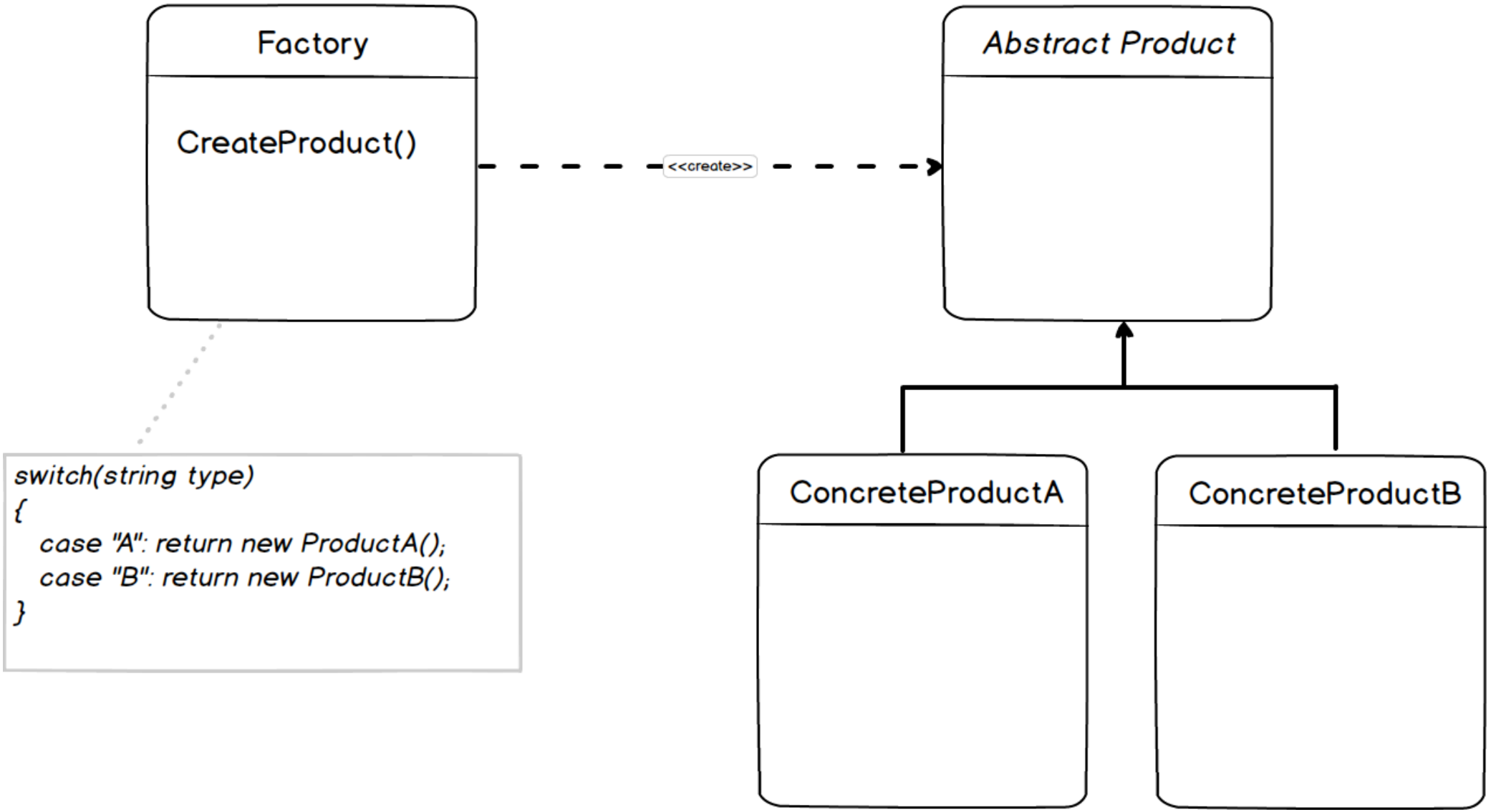

ConfigManager
Get()
Set()
- ConfigManager()
- instance
GetInstance()



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

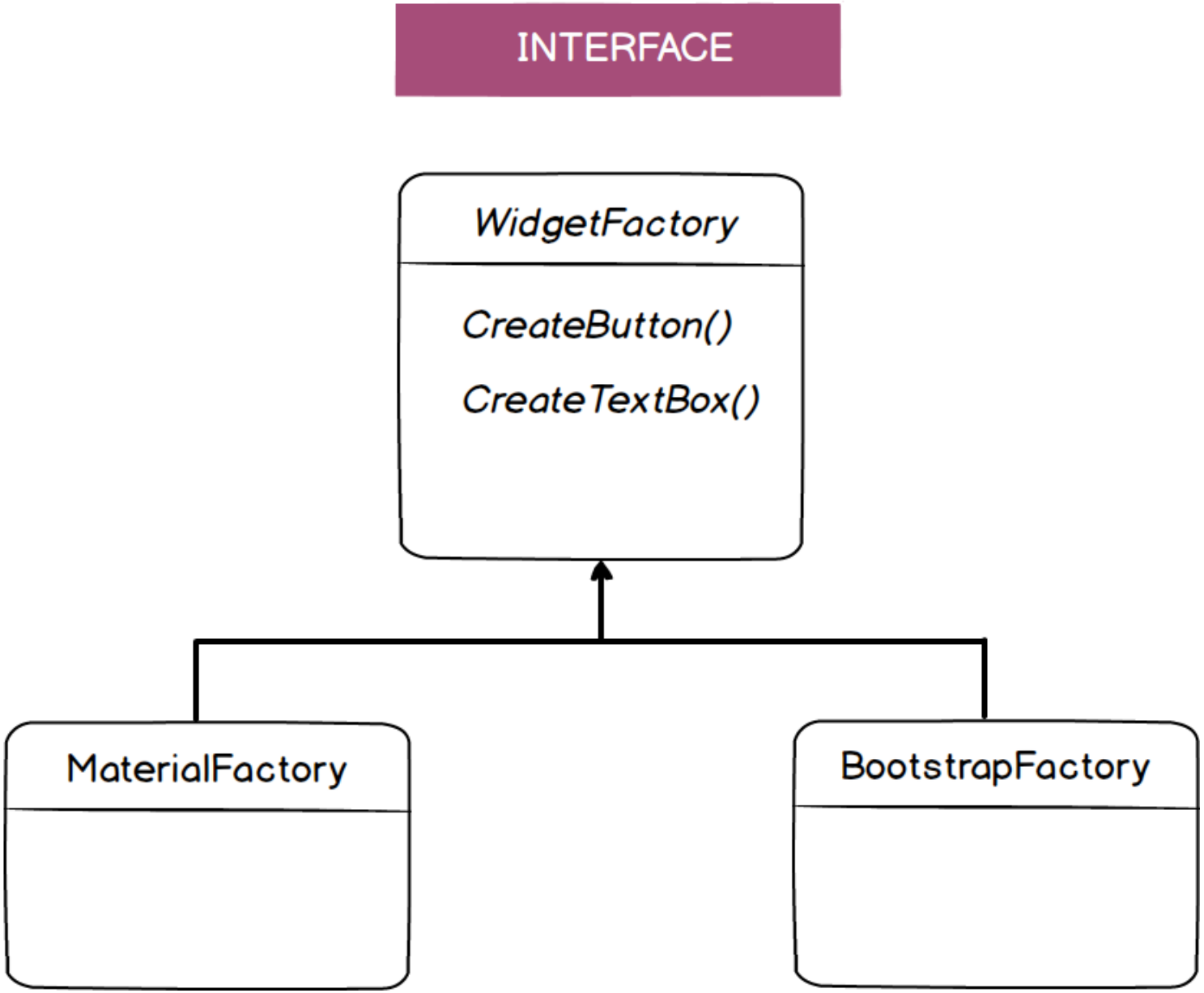
Simple Factory Pattern

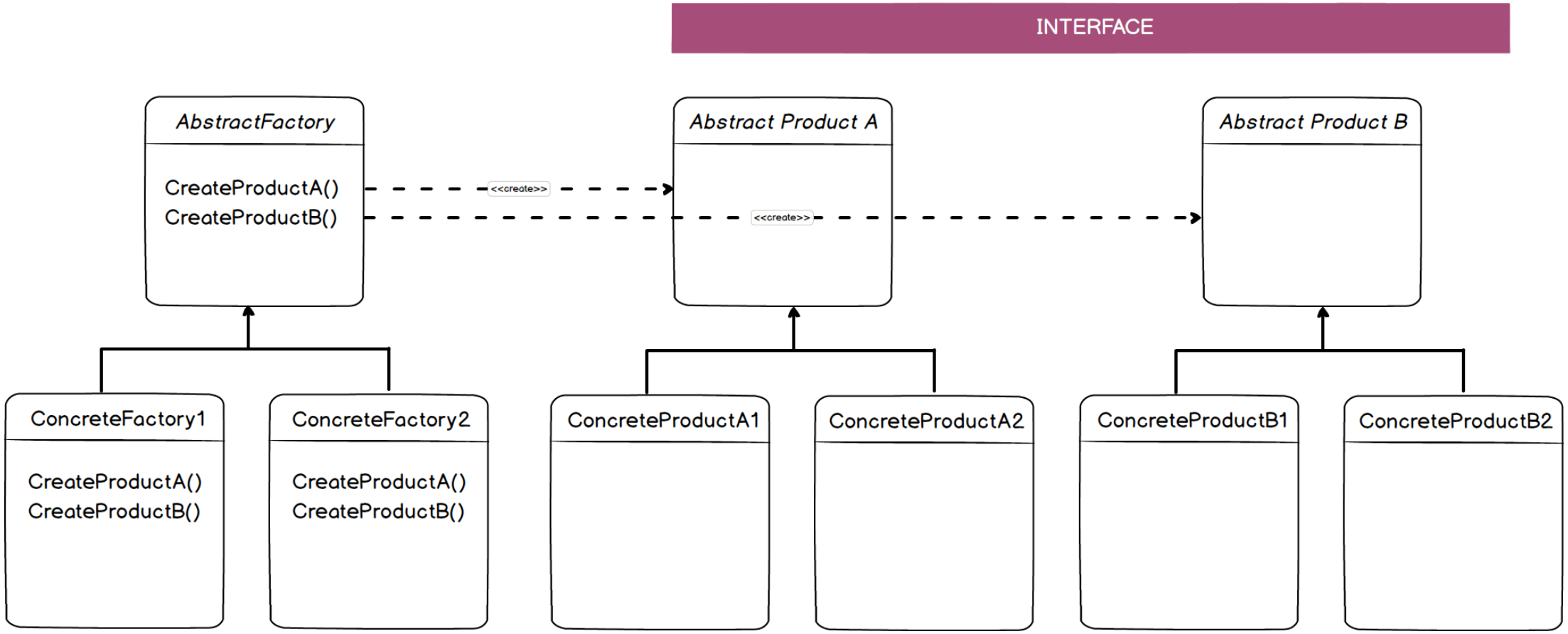
Zapewnia tworzenie obiektu na podstawie wielu **warunków**


Factory
CreateProduct()
<<create>>
Abstract Product
switch(string type)
{
case "A": return new ProductA();
case "B": return new ProductB();
}
ConcreteProductA
ConcreteProductB

Abstract Factory Pattern

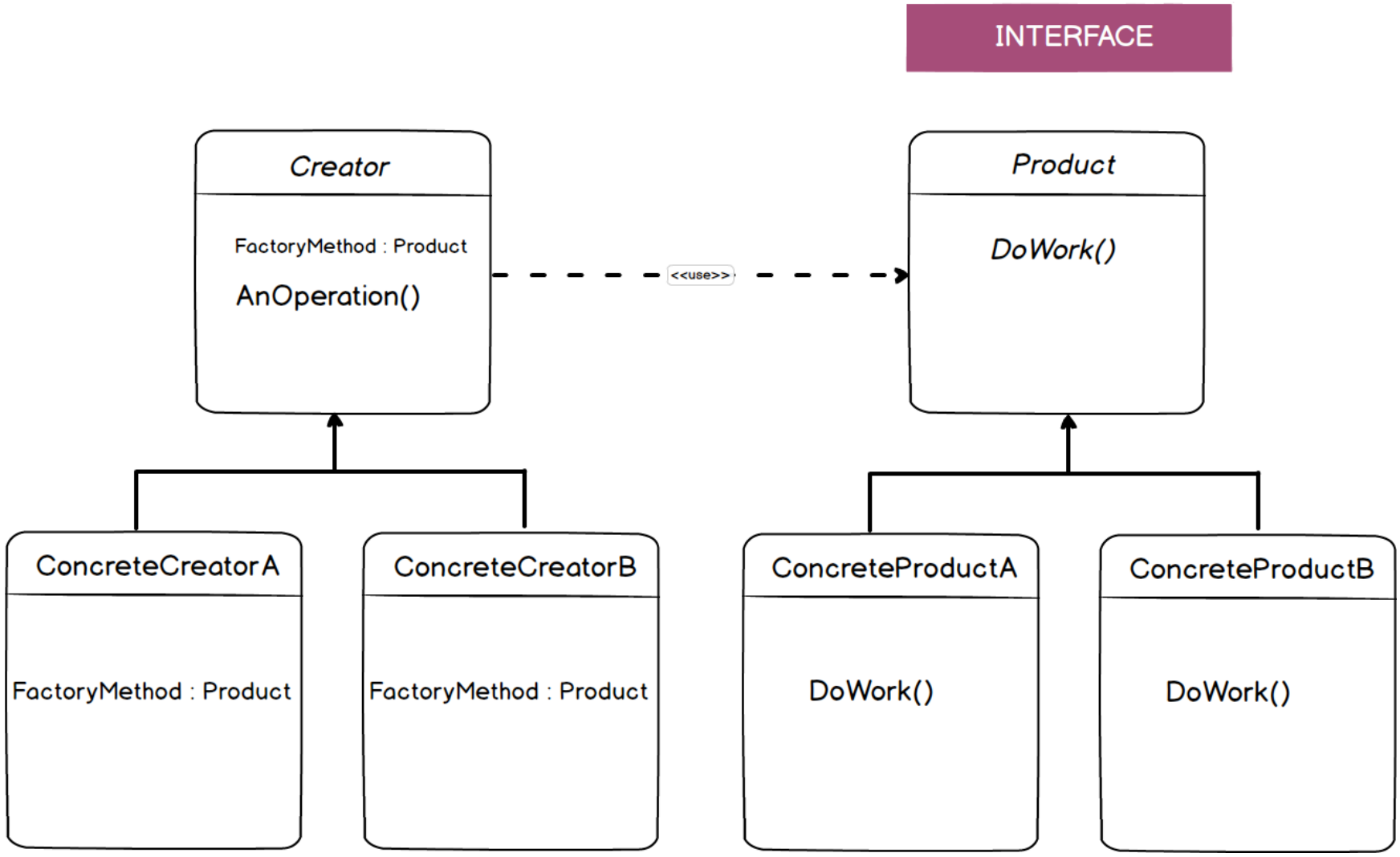
Zapewnia interfejs do tworzenia
rodziny powiązanych obiektów


INTERFACE
WidgetFactory
CreateButton()
CreateTextBox()
MaterialFactory
BootstrapFactory


INTERFACE
AbstractFactory
CreateProductA()
CreateProductB()
<<create>>
<<create>>
Abstract Product A
Abstract Product B
ConcreteFactory1
CreateProductA()
CreateProductB()
ConcreteFactory2
CreateProductA()
CreateProductB()
ConcreteProductA1
ConcreteProductA2
ConcreteProductB1
ConcreteProductB2

Factory Method Pattern

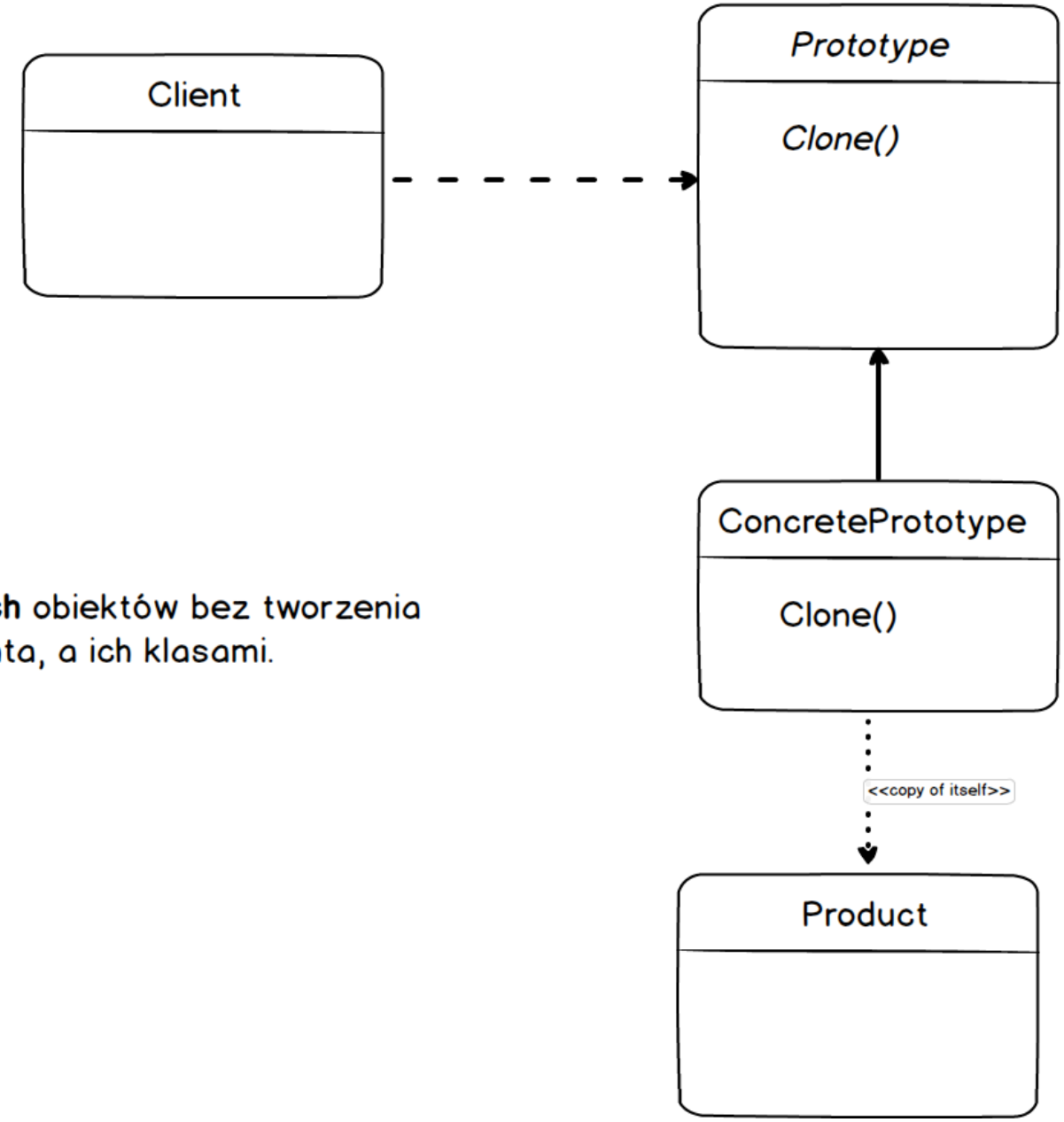
Odkłada tworzenie obiektu
do **podklas**


INTERFACE
Creator
FactoryMethod : Product
AnOperation()
<<use>>
Product
DoWork()
ConcreteCreatorA
FactoryMethod : Product
ConcreteCreatorB
FactoryMethod : Product
ConcreteProductA
DoWork()
ConcreteProductB
DoWork()

Prototype Pattern

Tworzy nowe obiekty
kopiując **istniejący** obiekt

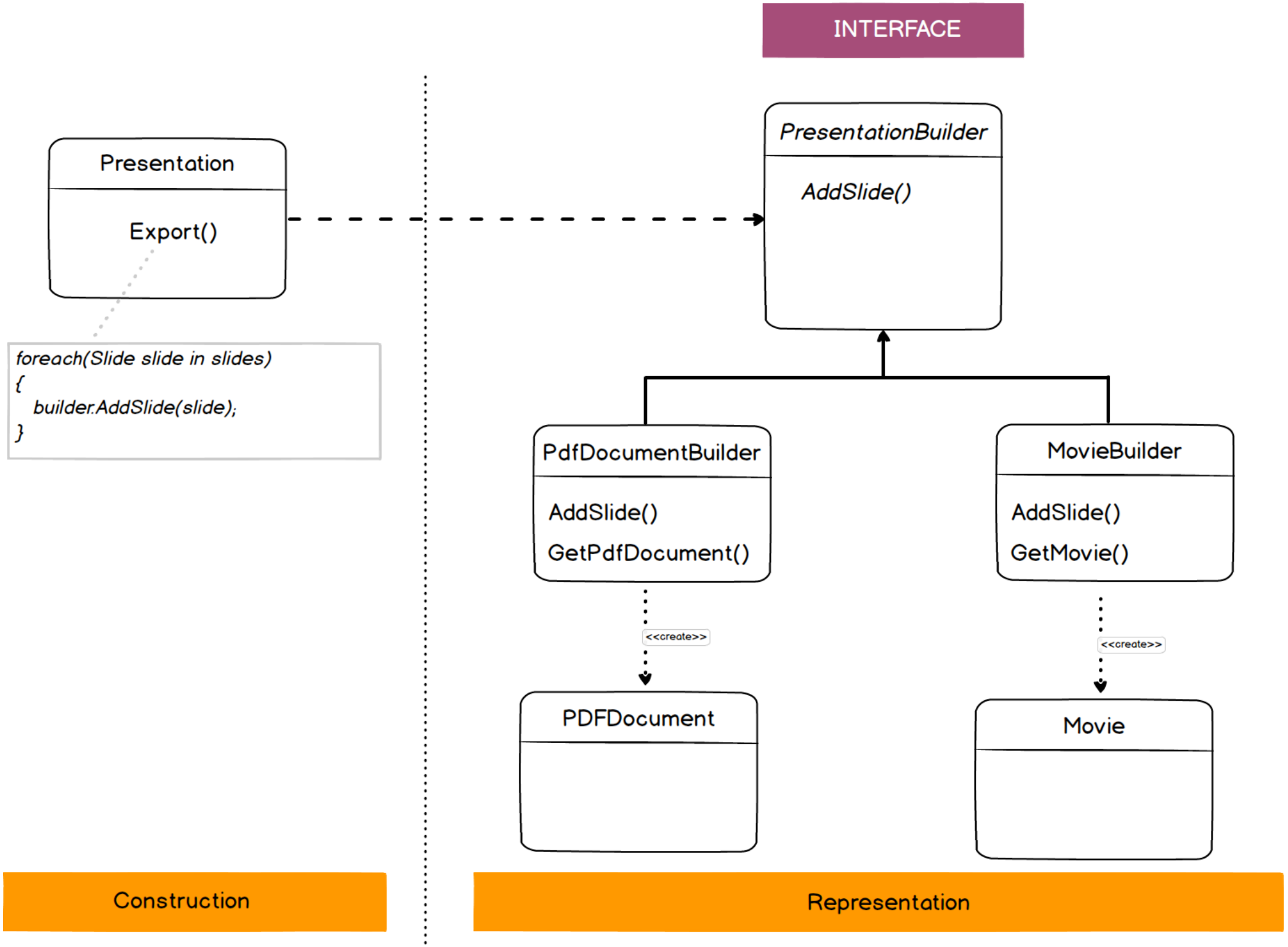
INTERFACE

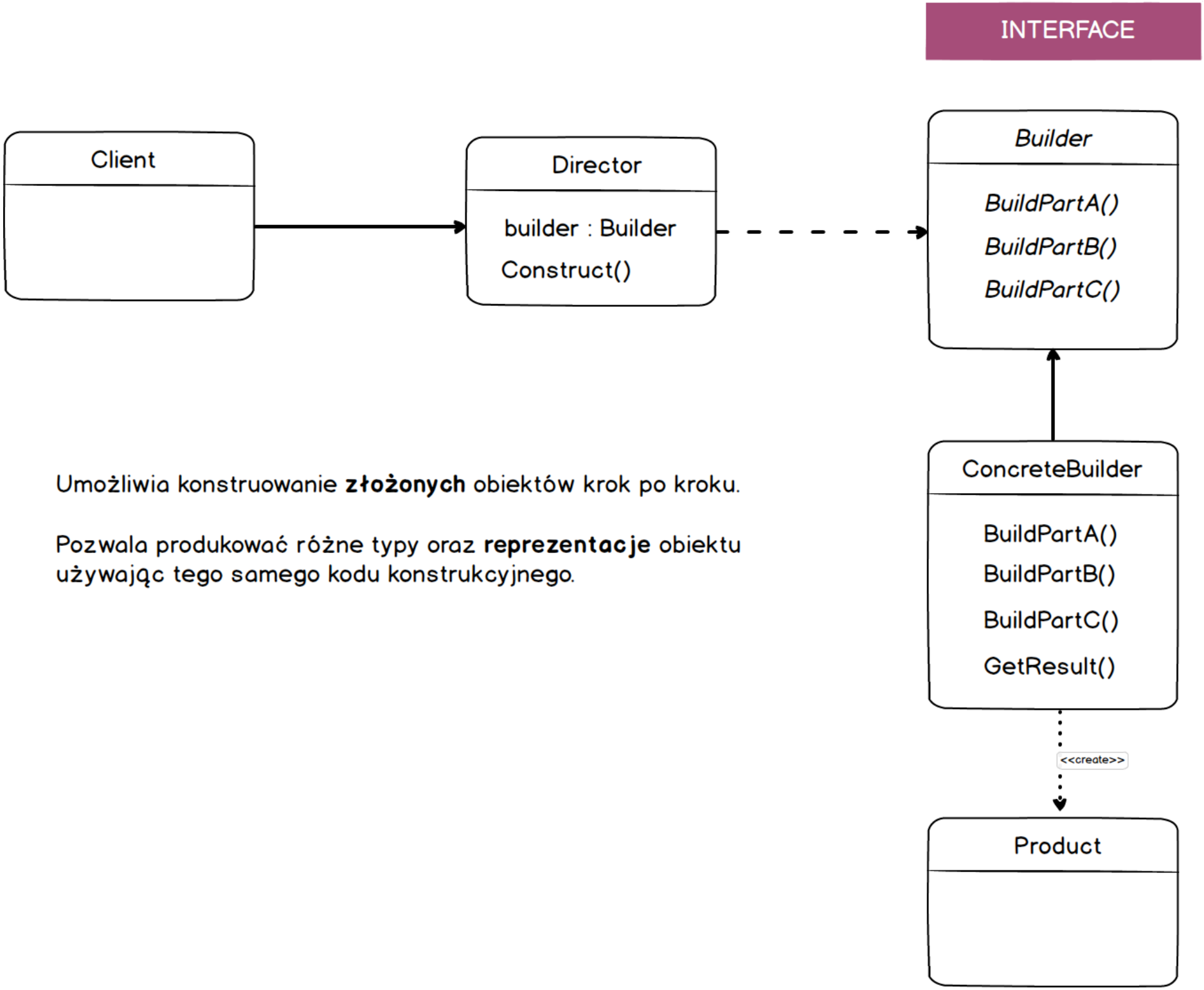


Umożliwia kopiowanie **istniejących** obiektów bez tworzenia zależności pomiędzy kodem klienta, a ich klasami.

Builder Pattern

Oddziela **konstrukcję** obiektu
od jego **reprezentacji**


INTERFACE
Presentation
Export()
foreach(Slide slide in slides)
{
builder.AddSlide(slide);
}
PresentationBuilder
AddSlide()
PdfDocumentBuilder
AddSlide()
GetPdfDocument()
MovieBuilder
AddSlide()
GetMovie()
<<create>>
<<create>>
PDFDocument
Movie
Construction
Representation



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

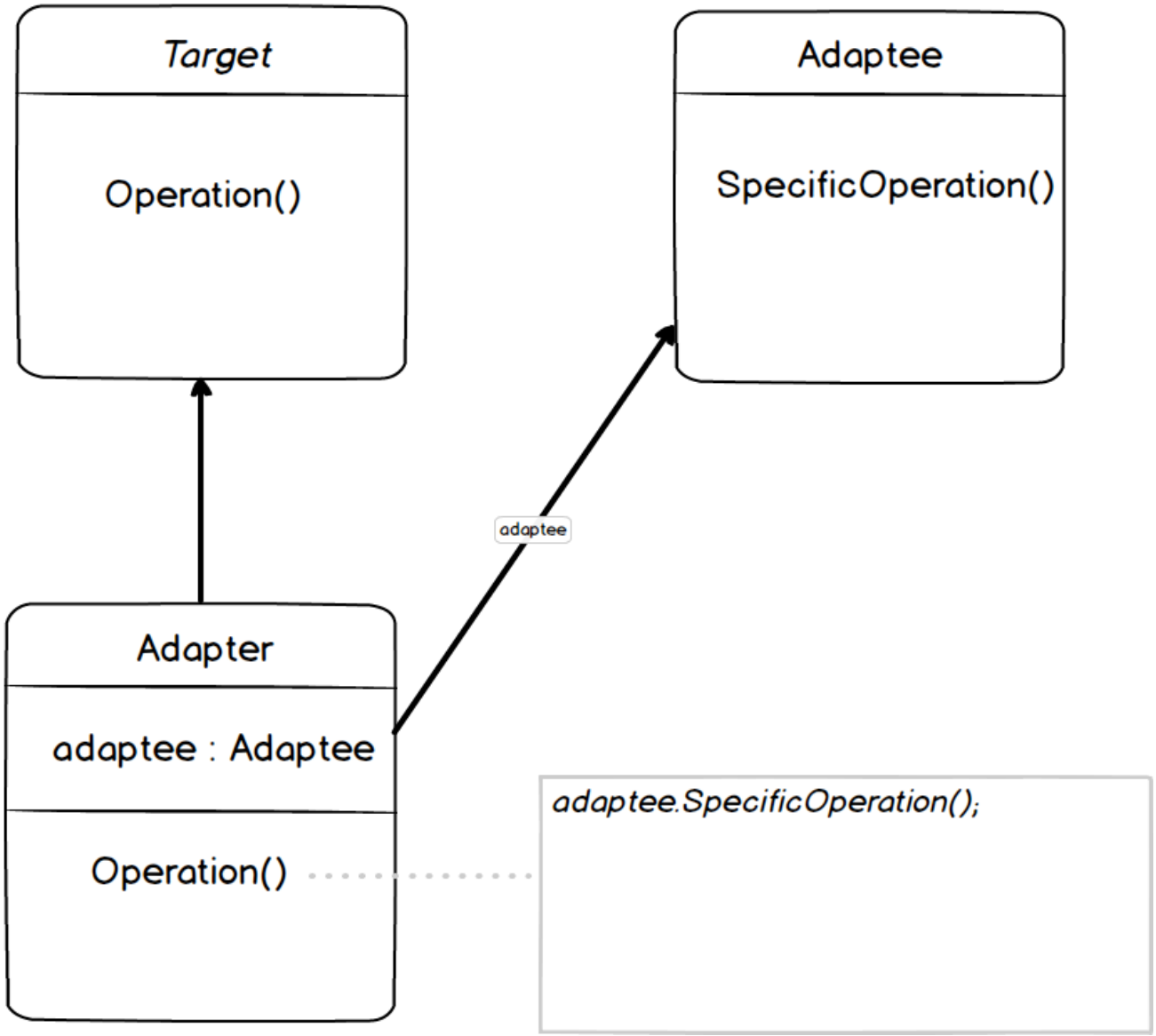
Pozwala produkować różne typy oraz **reprezentacje** obiektu używając tego samego kodu konstrukcyjnego.

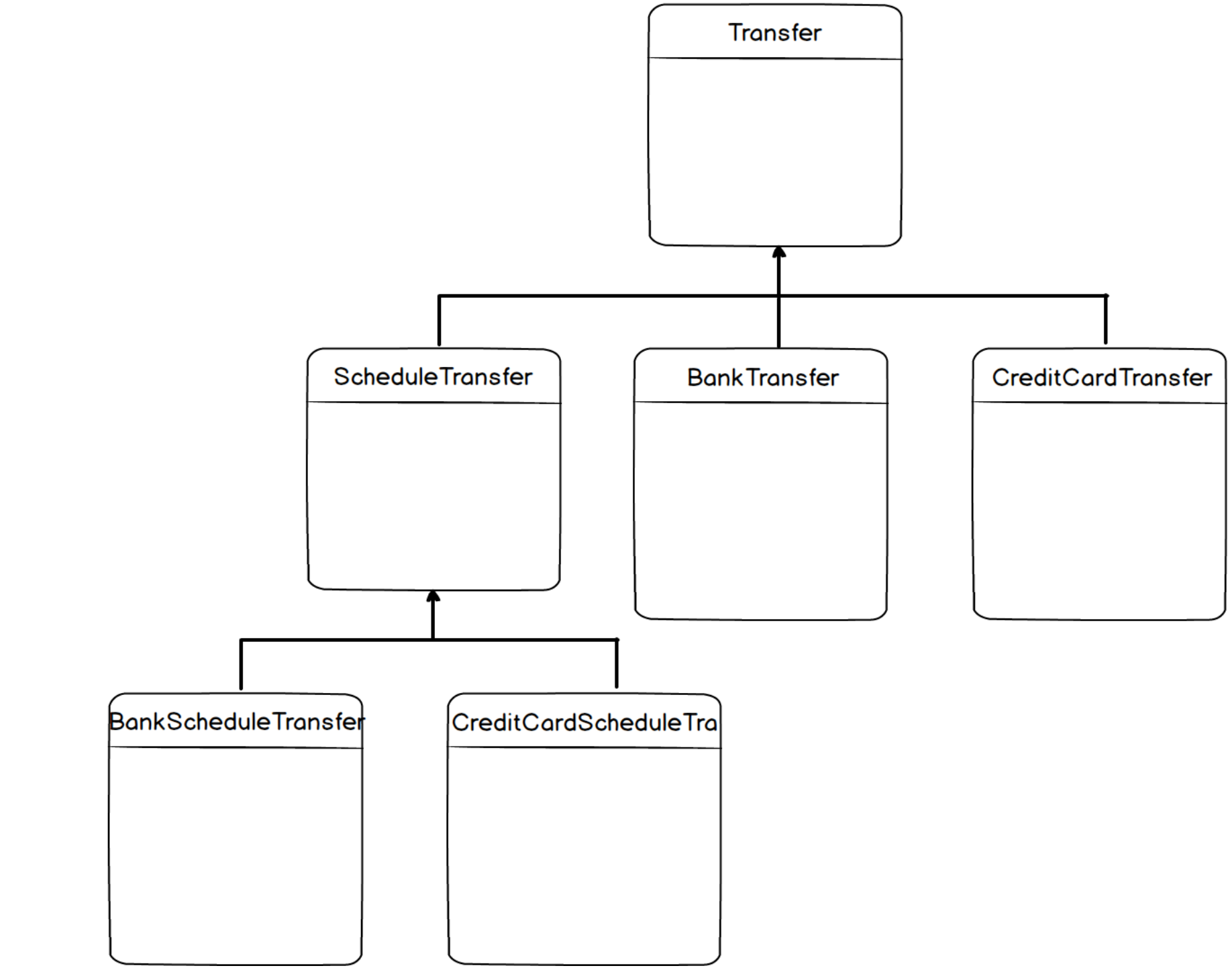
WZORCE STRUKTURALNE

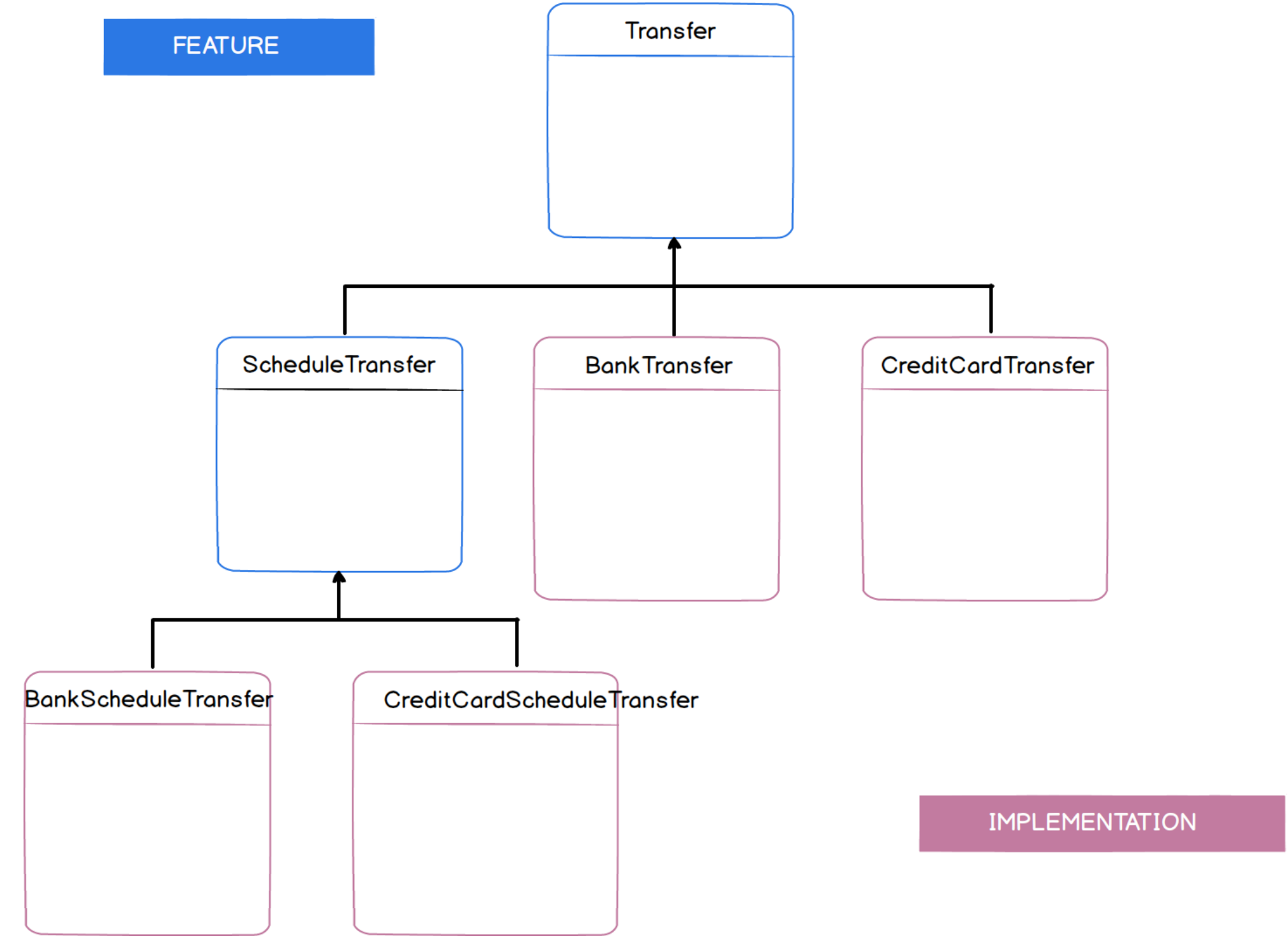
Adapter Pattern

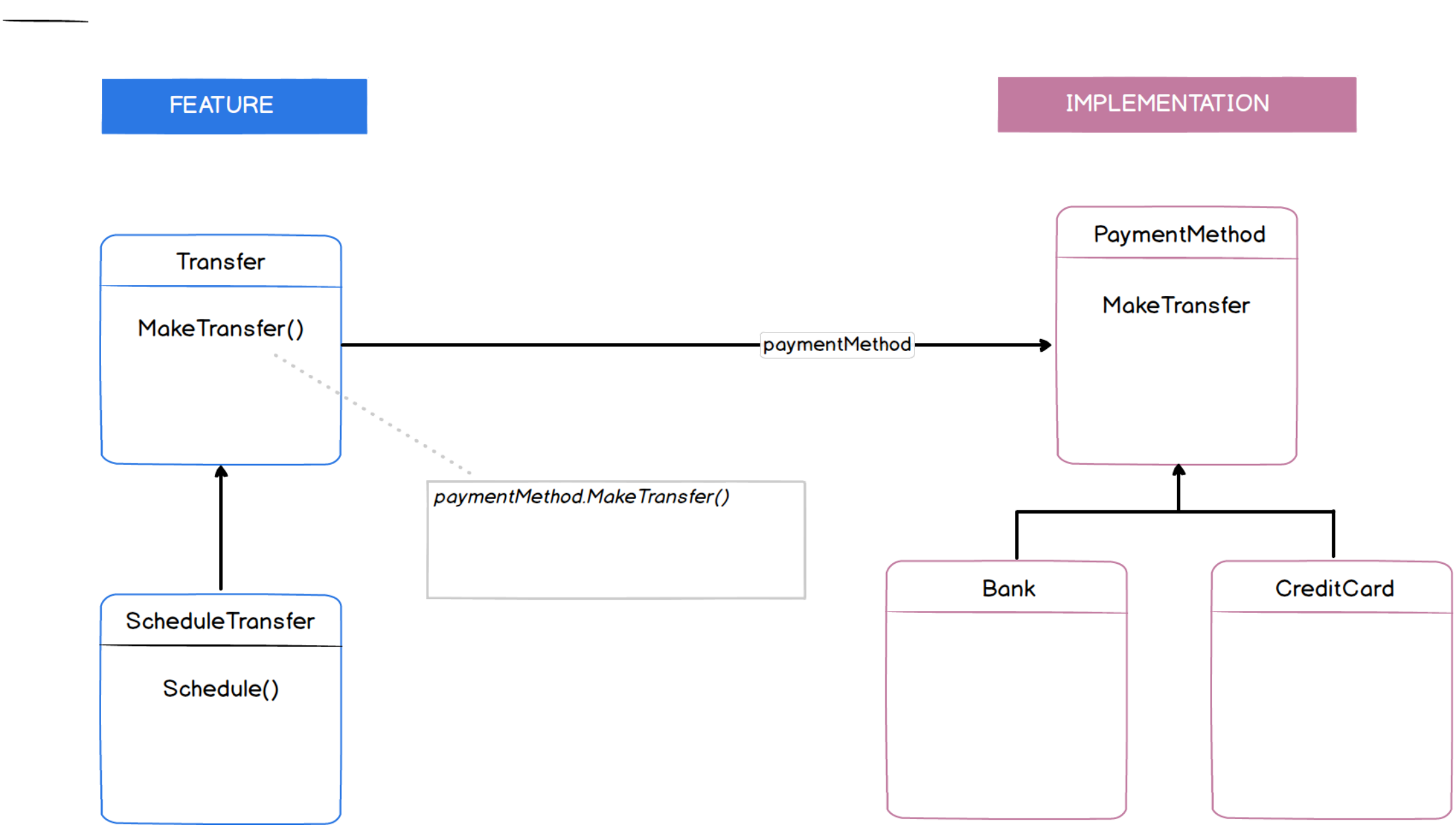
Pozwala na współdziałanie ze sobą
obiektów o **niekompatybilnych** interfejsach

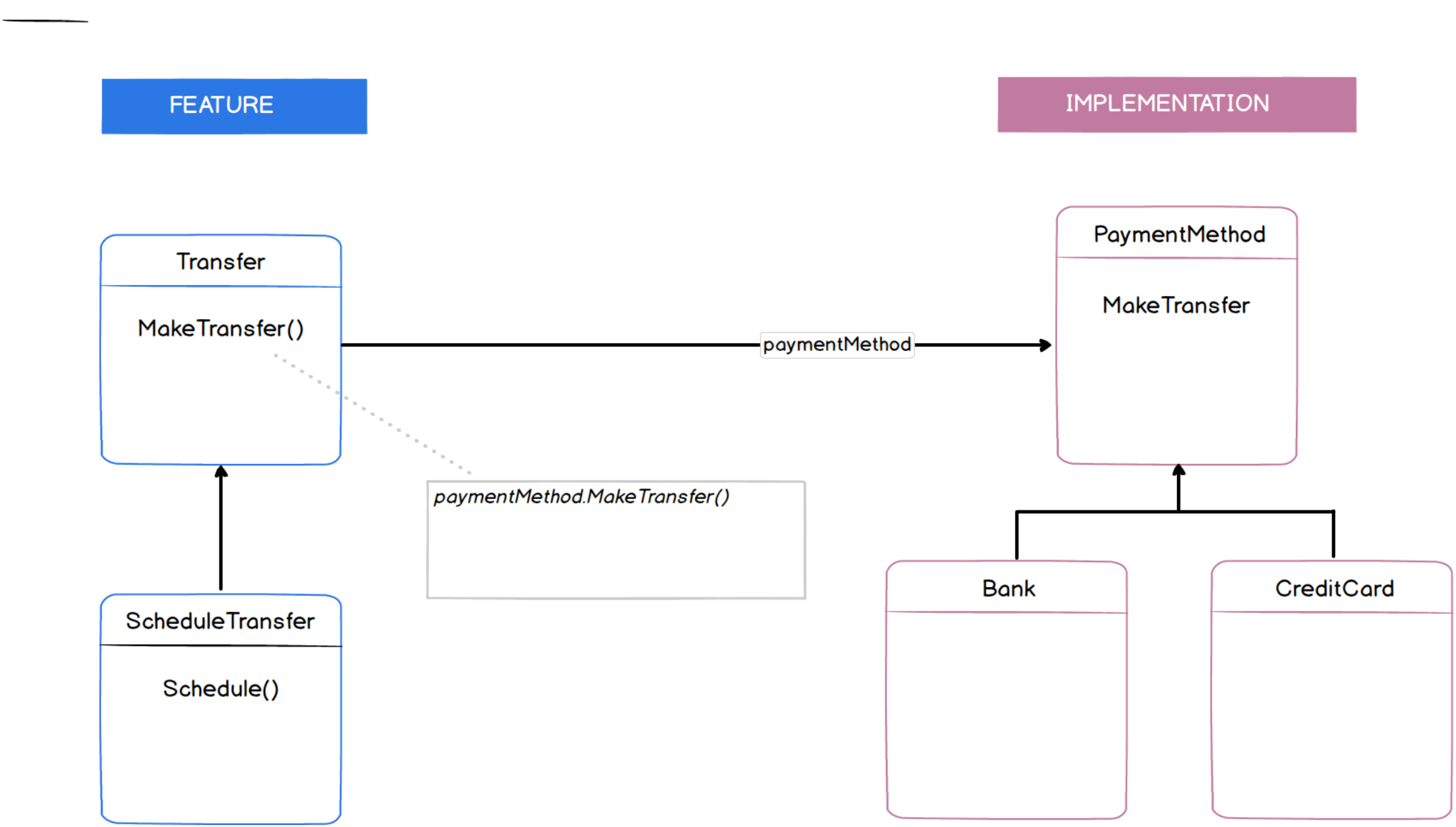
Object Adapter




Transfer
ScheduleTransfer
BankTransfer
CreditCardTransfer
BankScheduleTransfer
CreditCardScheduleTra

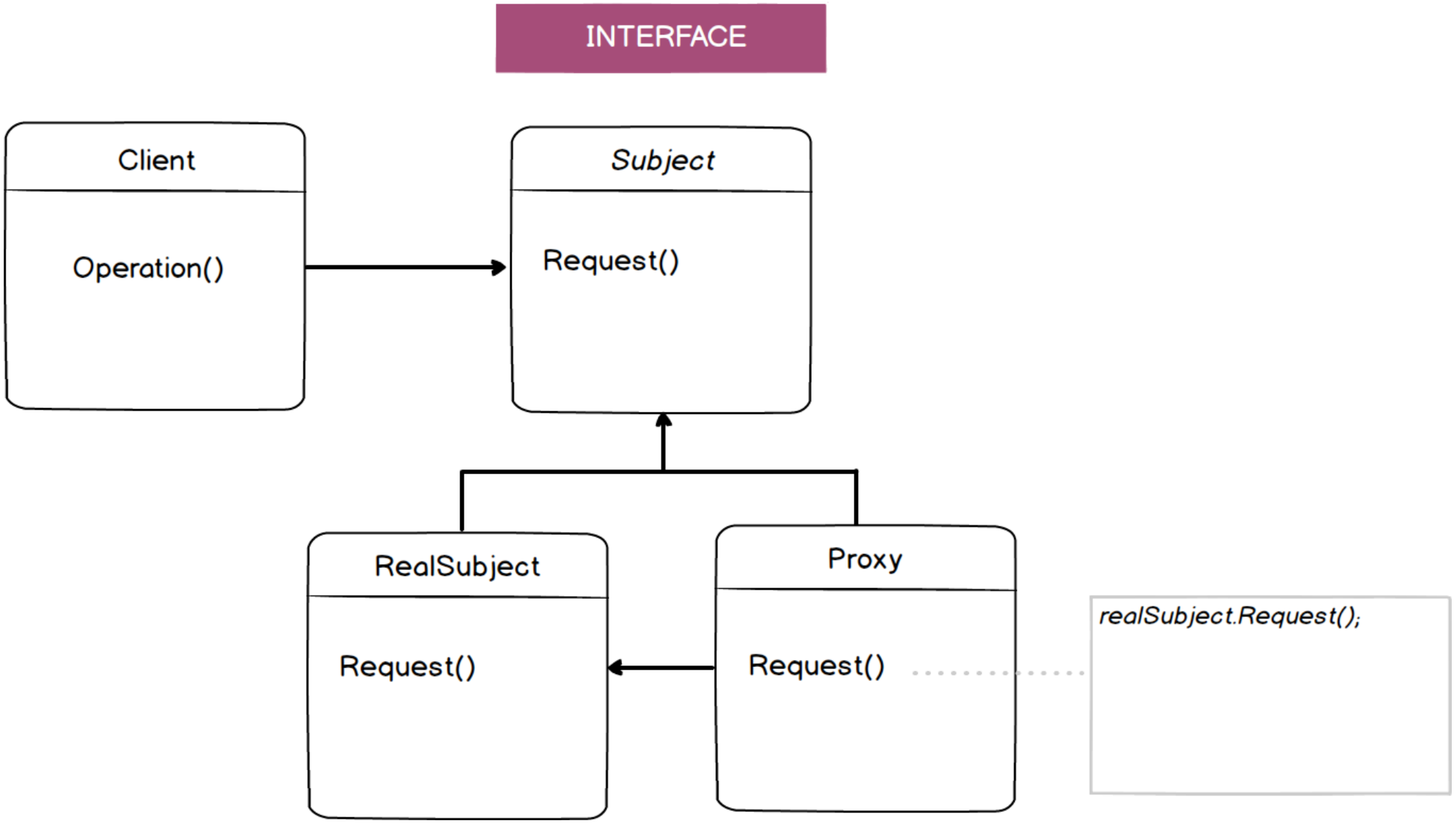

FEATURE
Transfer
ScheduleTransfer
BankTransfer
CreditCardTransfer
BankScheduleTransfer
CreditCardScheduleTransfer
IMPLEMENTATION


FEATURE
IMPLEMENTATION
Transfer
MakeTransfer()
paymentMethod
PaymentMethod
MakeTransfer
paymentMethod.MakeTransfer()
ScheduleTransfer
Schedule()
Bank
CreditCard


FEATURE
IMPLEMENTATION
Transfer
MakeTransfer()
paymentMethod
PaymentMethod
MakeTransfer
paymentMethod.MakeTransfer()
ScheduleTransfer
Schedule()
Bank
CreditCard

Proxy Pattern

Pozwala na stworzenie obiektu **zastępczego**
w miejsce innego obiektu.


INTERFACE
Client
Operation()
Subject
Request()
RealSubject
Request()
Proxy
Request()
realSubject.Request();

Decorator Pattern

Przedkładaj **kompozycję** nad dziedziczenie

WZORCE CZYNNOŚCIOWE

Fasada Pattern

Ukrywa złożony zestaw klas
w **uproszczony** interfejs.

Chain of Responsibility Pattern

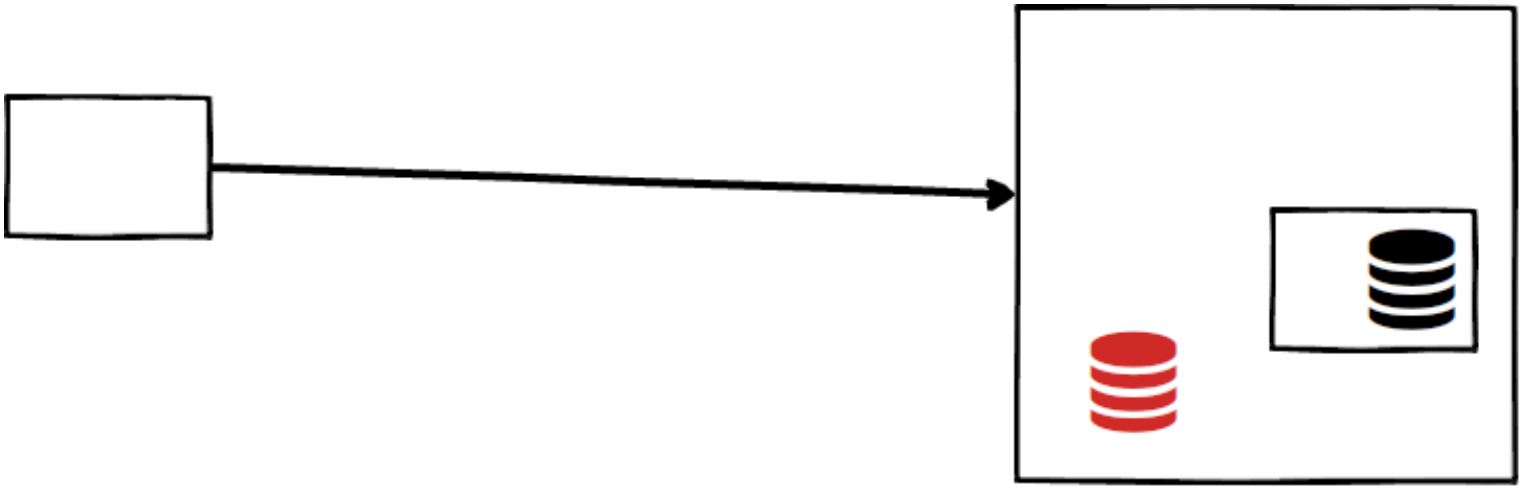
Pozwala przekazywać żądania wzdłuż
łańcucha obiektów obsługujących

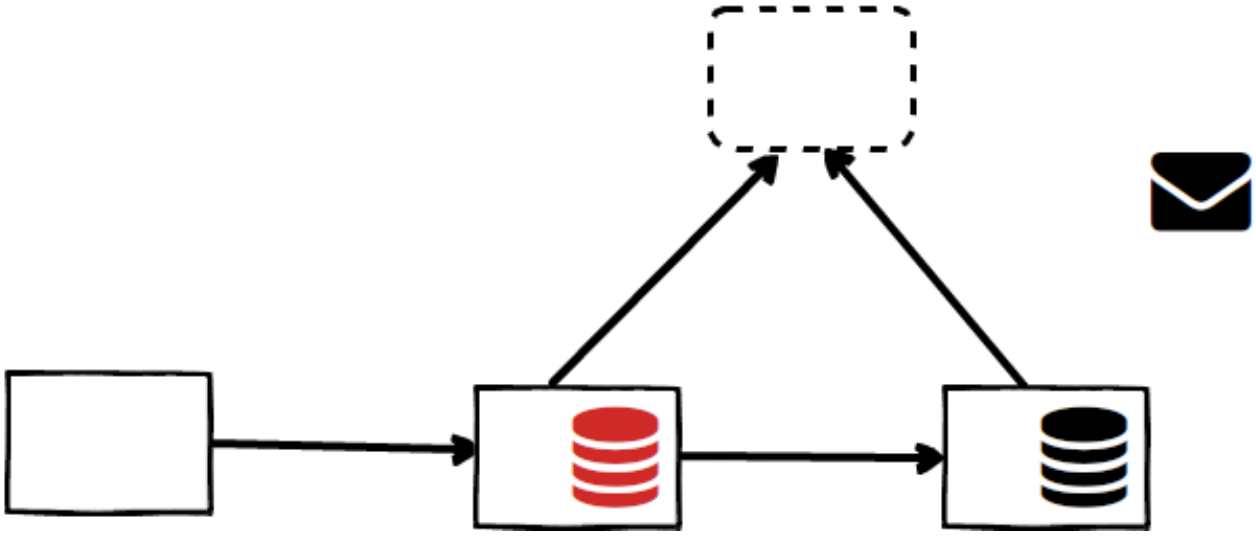
Composite Pattern

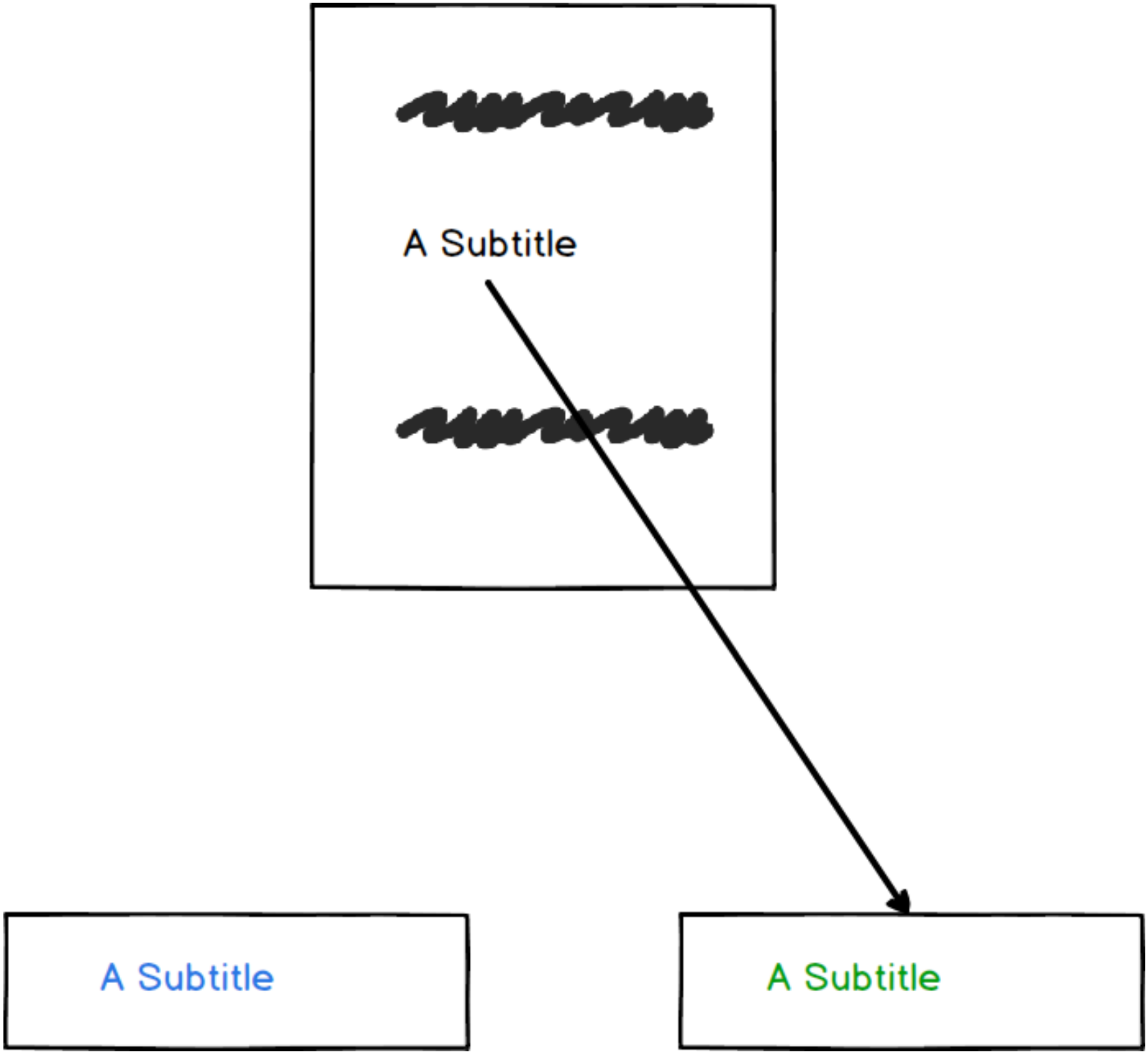
Komponuje obiekty w
struktury drzewiaste

State Pattern

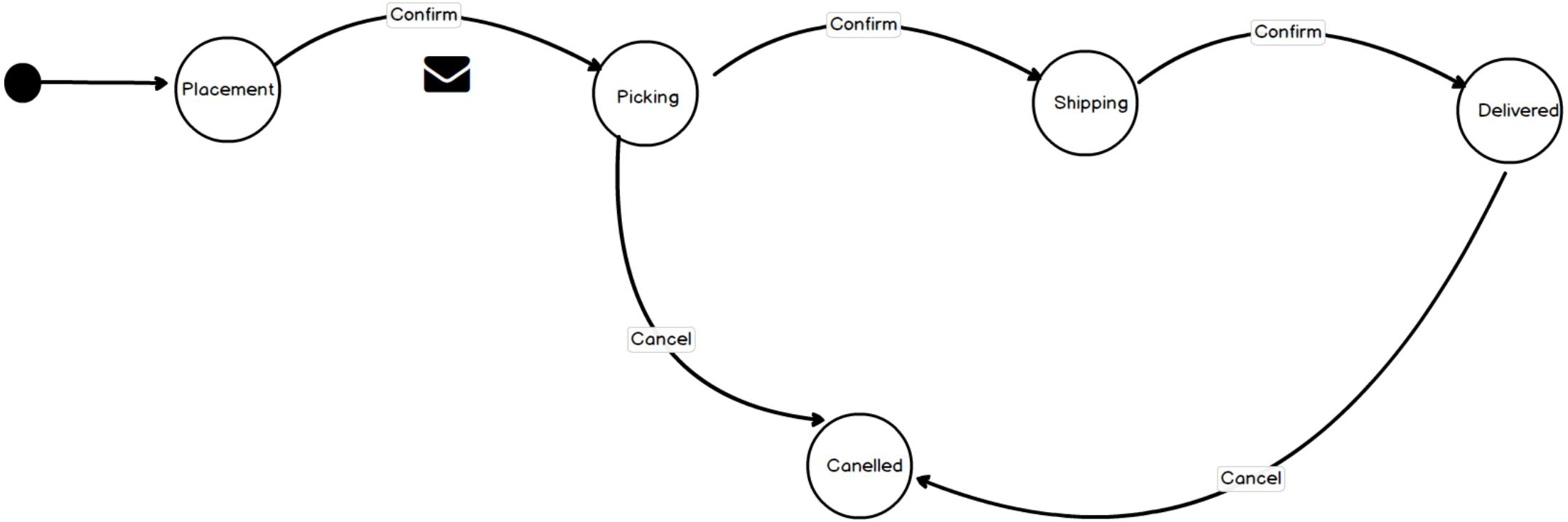
Pozwala obiektować zmieniać swoje zachowanie
gdy zmieni się jego **stan wewnętrzny**

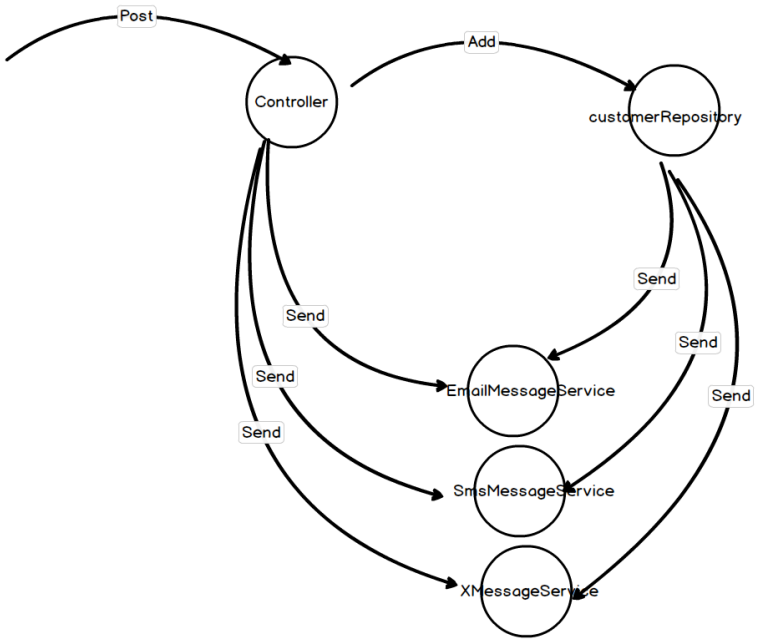


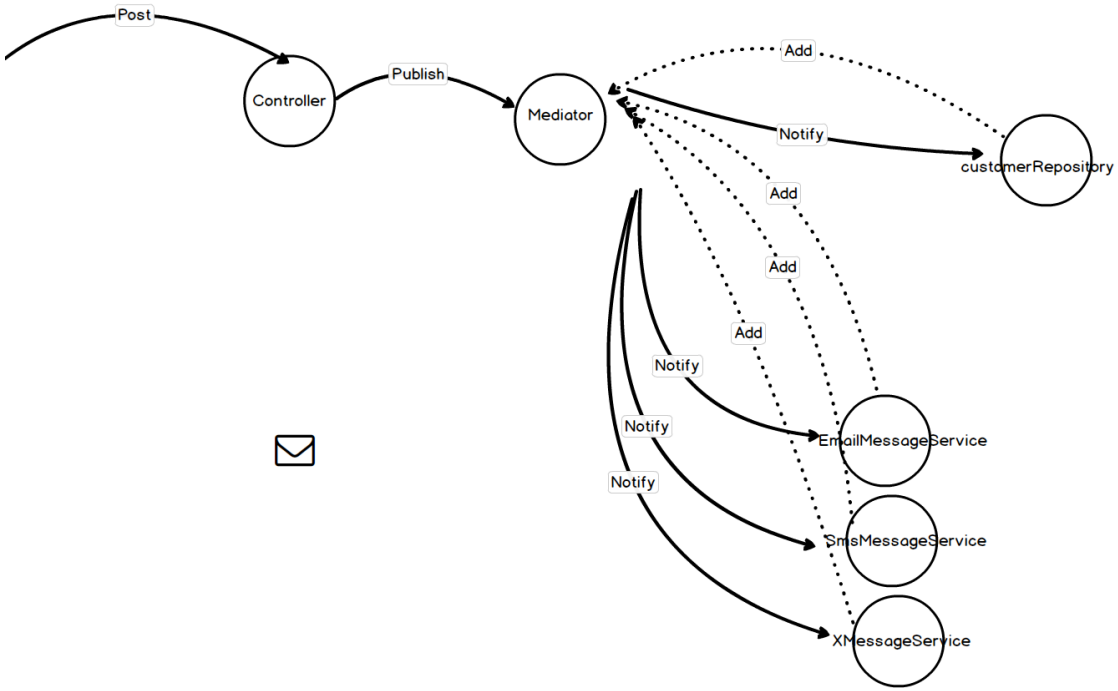


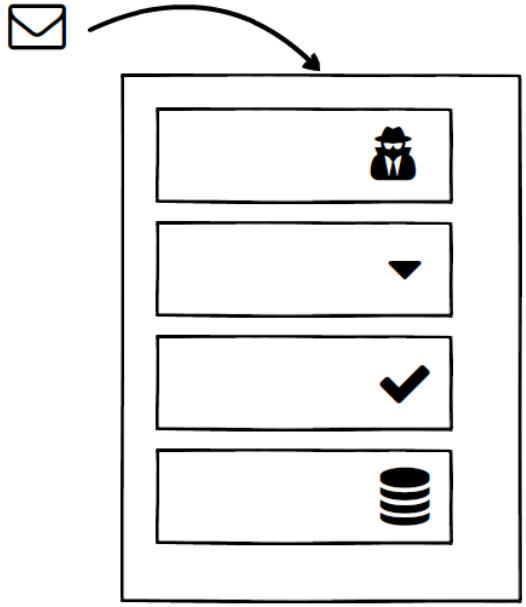

A Subtitle
A Subtitle
A Subtitle

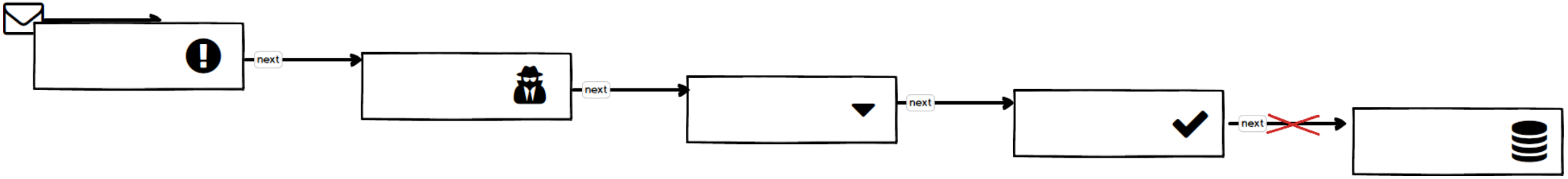
Maszyna stanów skończonych




Post
Controller
Add
customerRepository
Send
Send
Send
Send
Send
Send
EmailMessageService
SmsMessageService
XMessageService

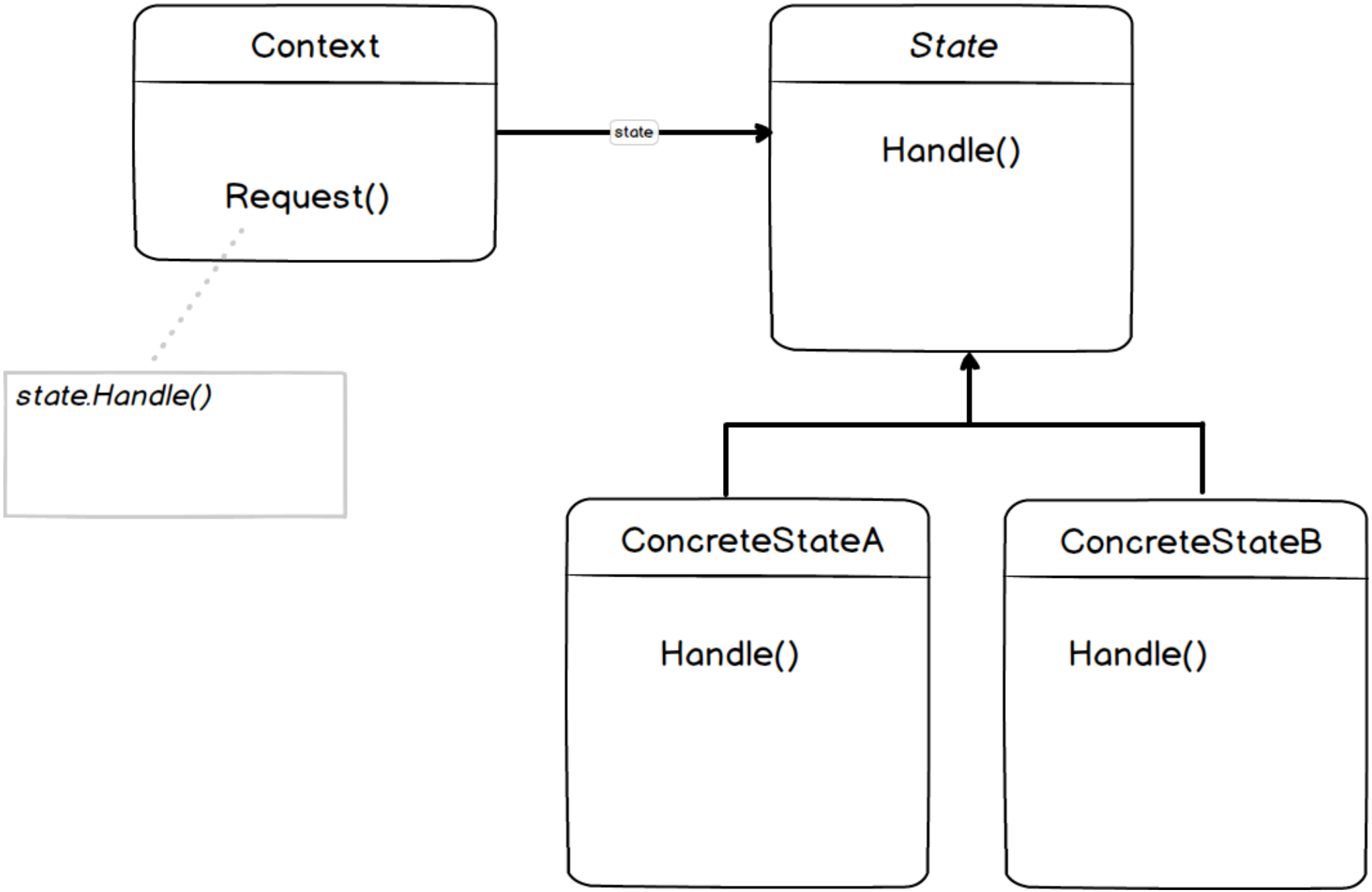

Post
Controller
Publish
Mediator
Add
Notify
customerRepository
Add
Add
Add
Notify
Notify
Notify
EmailMessageService
SmsMessageService
XMessageService




next
next
next
next

Flyweight Pattern

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


Context
Request()
state
State
Handle()
state.Handle()
ConcreteStateA
Handle()
ConcreteStateB
Handle()