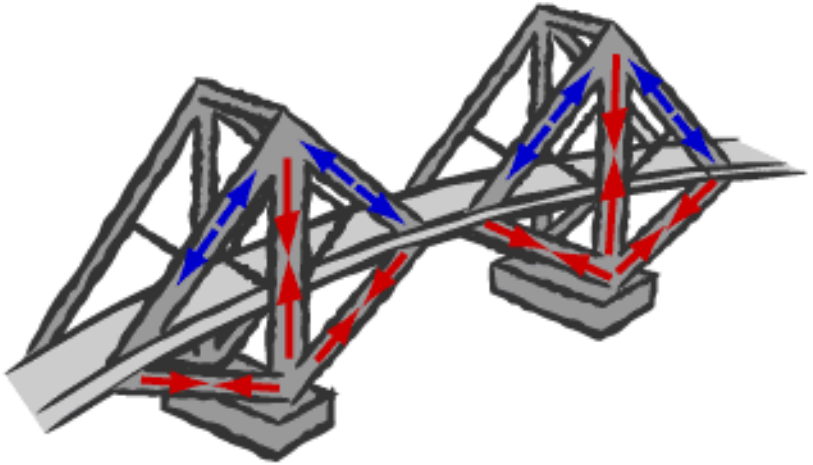
Wstęp

- Historia
- Czym są
- Rodzaje



bottega
IT SOLUTIONS

Hierarchia wzorców(Bruce Eckel, TIP)

- **Idiom**
- **Specyficzny projekt (specific design)**
- **Standardowy projekt (standard design)**
- **Wzorzec projektowy (design pattern)**

Elementy wzorca projektowego

- **Nazwa wzorca**
  - Wchodzi na stałe do słownika opisu archiektury
- **Problem**
  - Opisuje, kiedy należy użyć dany wzorzec
- **Rozwiązanie**
  - Abstrakcyjny opis projektu, ogólna struktura elementów rozwiązania (klas, obiektów)
- **Konsekwencje**
  - Korzyści i koszty użycia wzorca

Purpose: Reflects What the Pattern Does

Scope: Domain Where Pattern Applies

	Creational	Structural	Behavioral
Class	Factory Method	Adapter (class)	Interpreter Template Method
Object	Abstract Factory Builder Prototype Singleton	Adapter (object) Bridge Composite Decorator Flyweight Façade Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

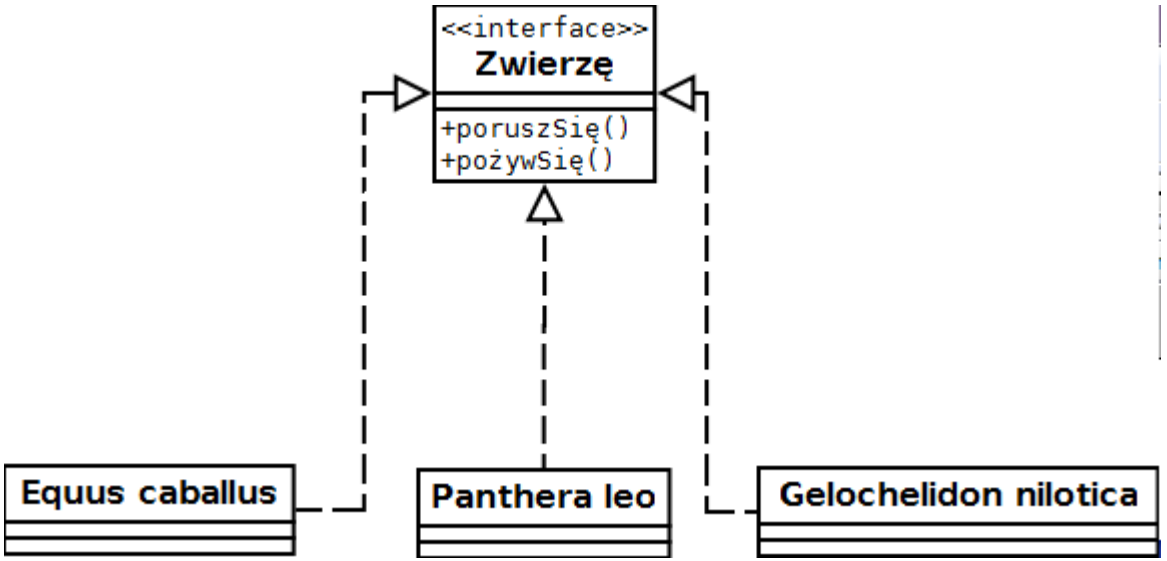
Przykład – odkrywanie wzorców

bottega IT SOLUTIONS

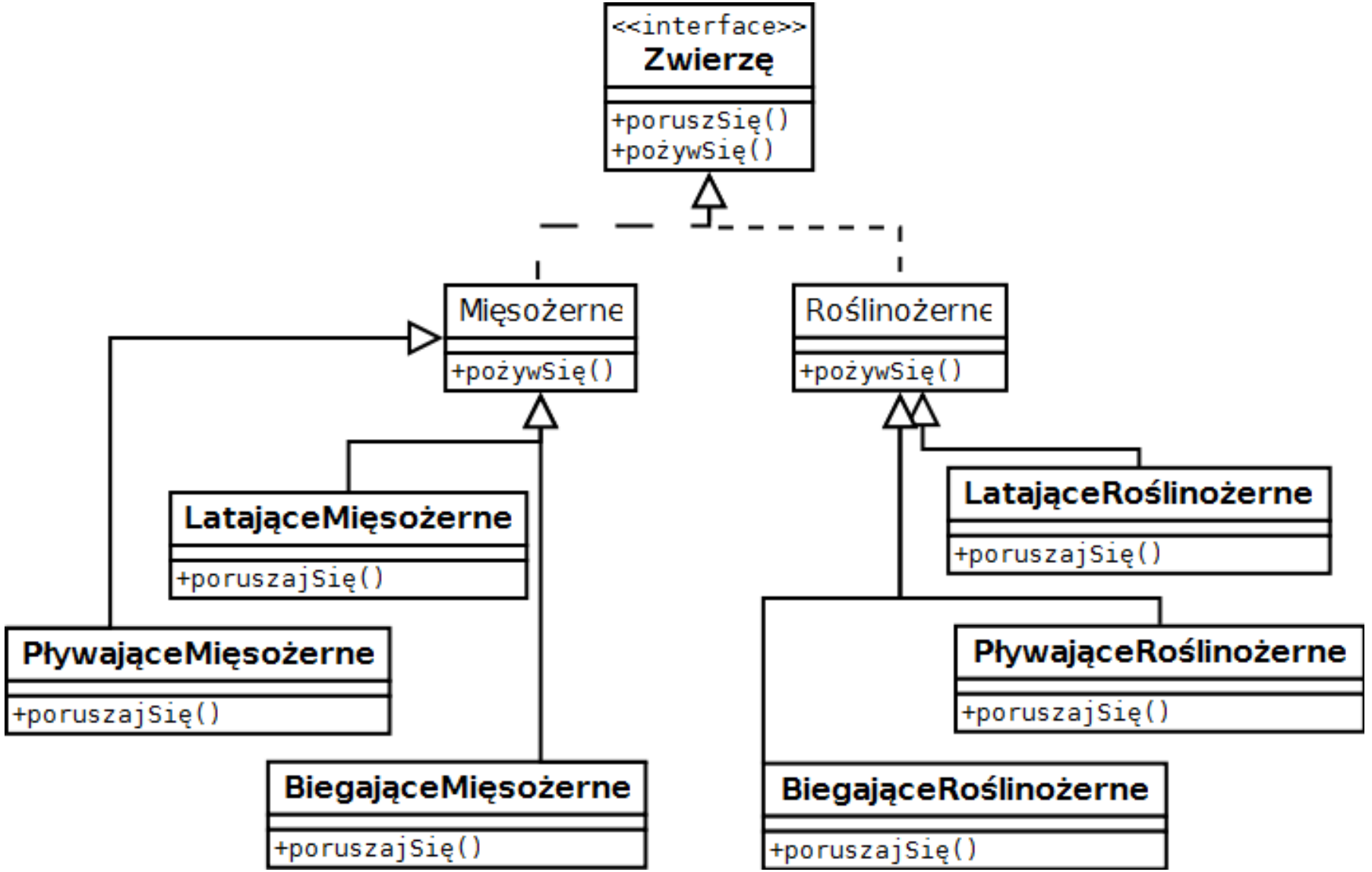
Rozwiązanie 1 - działające

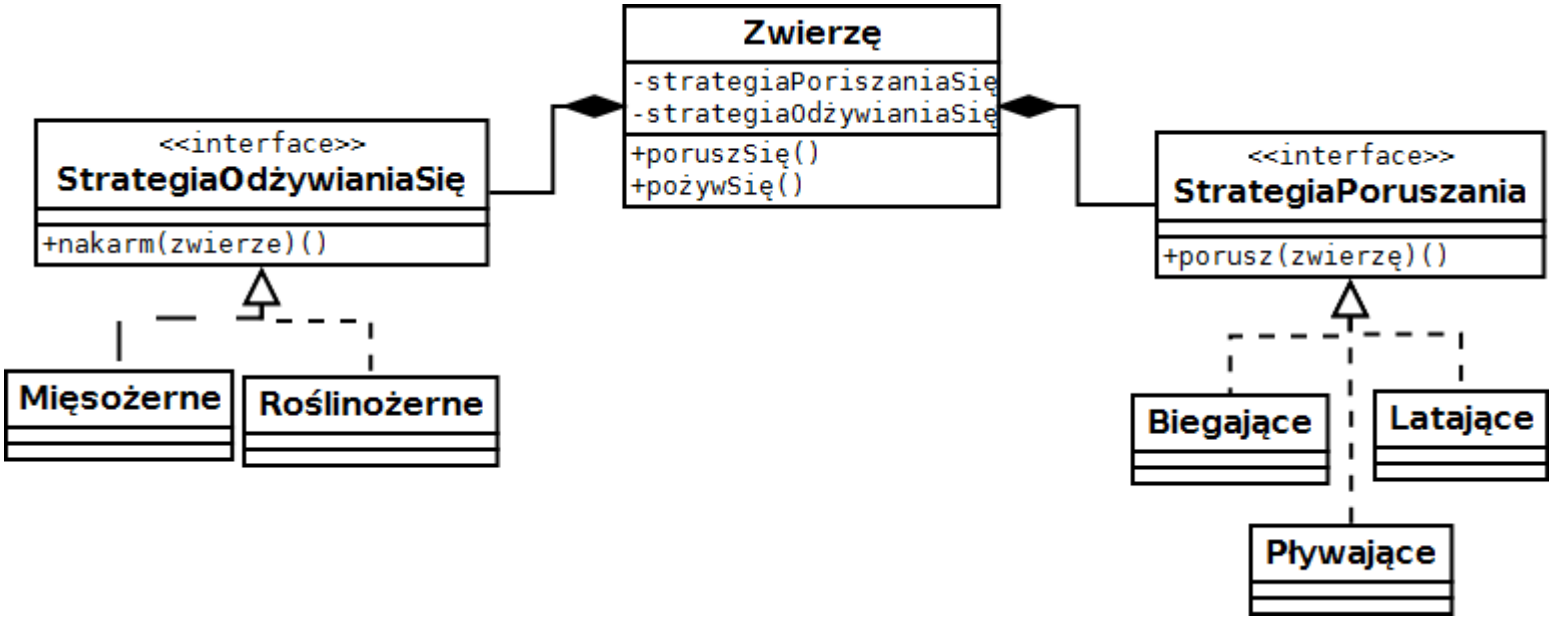
```
public class ZwierzęUtils{
  public static void załatwSprawę(...){
      if...
      if...
      ...
  }
}
```

Rozwiązanie 2 - Redundancja



Roz. 3 - Eksplozja kombinatoryczna






```
public void poruszSię(){
  this.StrategiaPoruszaniaSię.Porusz(this);
}

public void pożywSię(){
  this.StrategiaOdżywianiaSię.Nakarm(this);
}
```

bottega IT SOLUTIONS

Strategia

Kontekst

- Istnieją odmiany zachowania
- Posiadają wspólny kontrakt

Implementacja

- Agregacja zamiast dziedziczenia
  - Dziedziczenie może się przydać w hierarchii samej strategii
- Hermetyzacja zmienności poza stabilny interfejs

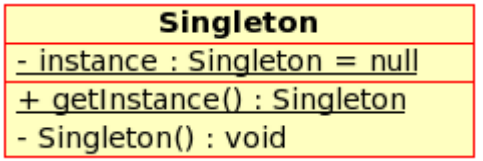
Siły

- Spójna odpowiedzialność
- Otwartość na rozbudowę bez modyfikacji
- **Stosowalność jedynie dla stałego "API"**

bottega IT SOLUTIONS

Singleton

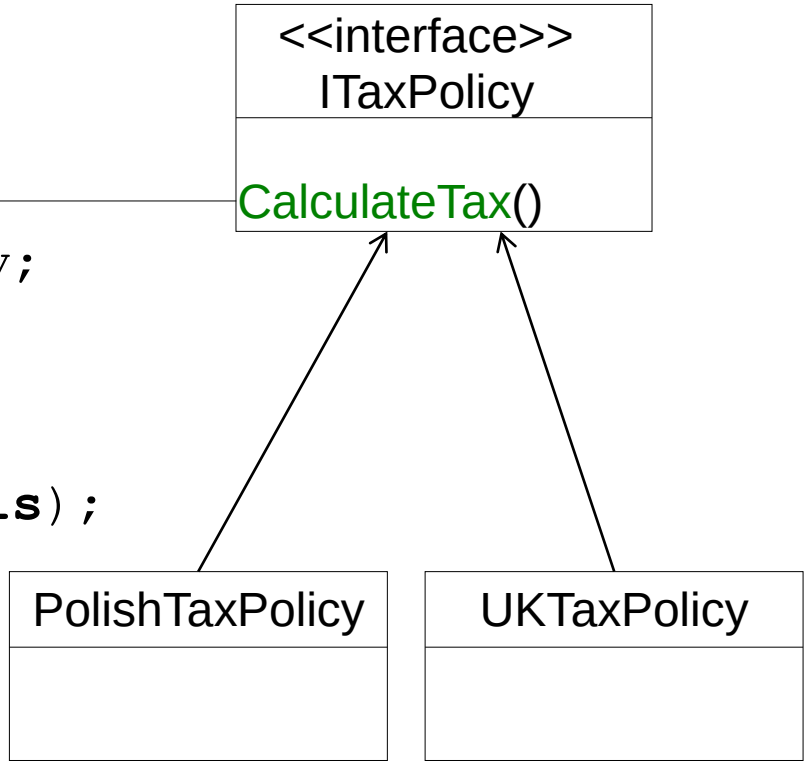
- Co robi?
- Przykłady użycia
- Zalety
- Wady
- Alternatywa
- Zadanie



Strategia

```
public class Order
{
   private ITaxPolicy _taxPolicy;
   private IRebatePolicy _rebatePolicy;

   public void Submit(){
      …
      _rebatePolicy.CalculateRebate(this);
      …
      _taxPolicy.CalculateTax(this);
      …
   }
}
```



Zadanie

bottega
IT SOLUTIONS

Strategia podsumowanie

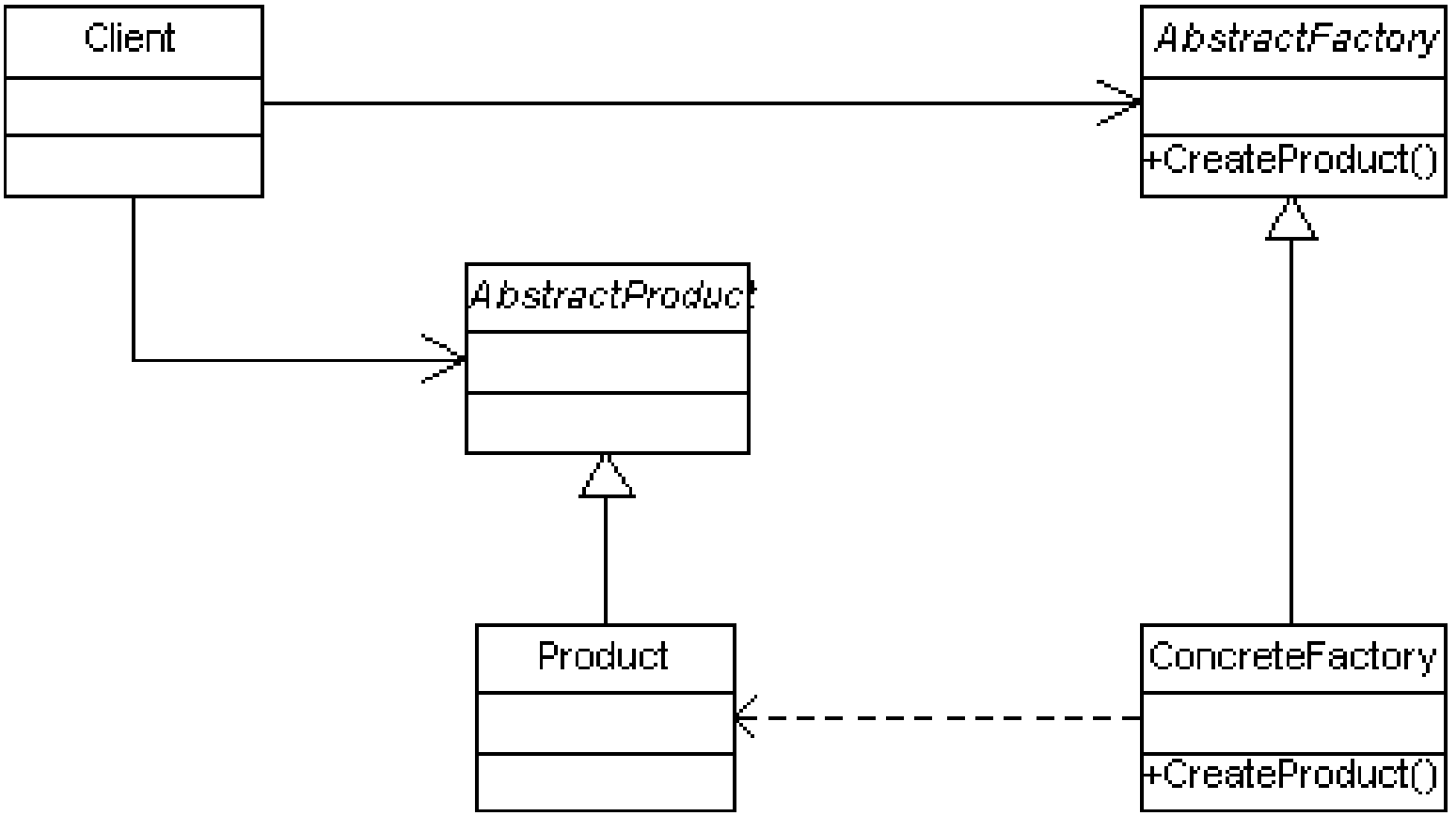
- Gdzie wywołać metodę „Generate”
- Czy w nazwie powinna być słowo strategia
- W którym momencie wstrzyknąć ICostCalculator

Fabryka (idiom)

```
public class TaxPolicyFactory{
        public static TaxPolicy CreateTaxPolicy(string country){
                //logika, zwykle if, switch etc.
                //ew. czytanie konf.
        }
}
```

- Ulepszenia:
  - kontekst jako parametr specyfikujący
  - enum jako parametr specyfikujący
    - interfejs, który może być implementowany przez np. wiele enumów (większa rozszerzalność)

Fabryka abstrakcji - struktura



bottega IT SOLUTIONS

Fabryka abstrakcji (Ilustracja)

- Abstrakcyjna Fabryka AGD (interfejs)
  - potrafi produkować
    - odkurzacze (interfejs)
    - telewizory (interfejs)
- Konkretna Fabryka "Predom"
  - potrafi produkować
    - odkurzacze Predom i telewizory Predom
- Konkretna Fabryka "Łucznik"
  - potrafi produkować
    - odkurzacze Łucznik i telewizory Łucznik

Fabryka abstrakcji (Przykład)

```
public interface OrderPoliciesFactory{
        public TaxPolicy createTaxPolicy();
        public RebatePolicy createRebatePolicy();
}
```

- Konkretna fabryka zwraca konkretne polityki
- Pobranie konkretnej fabryki wymaga użycia np. idomu Fabryki
- ```
  MainFactory.CreateOrderPoliciesFactory("klientA")
        .CreateTaxPolicy();
  ```
  ```

Zadanie

bottega IT SOLUTIONS

Fabryka podsumowanie

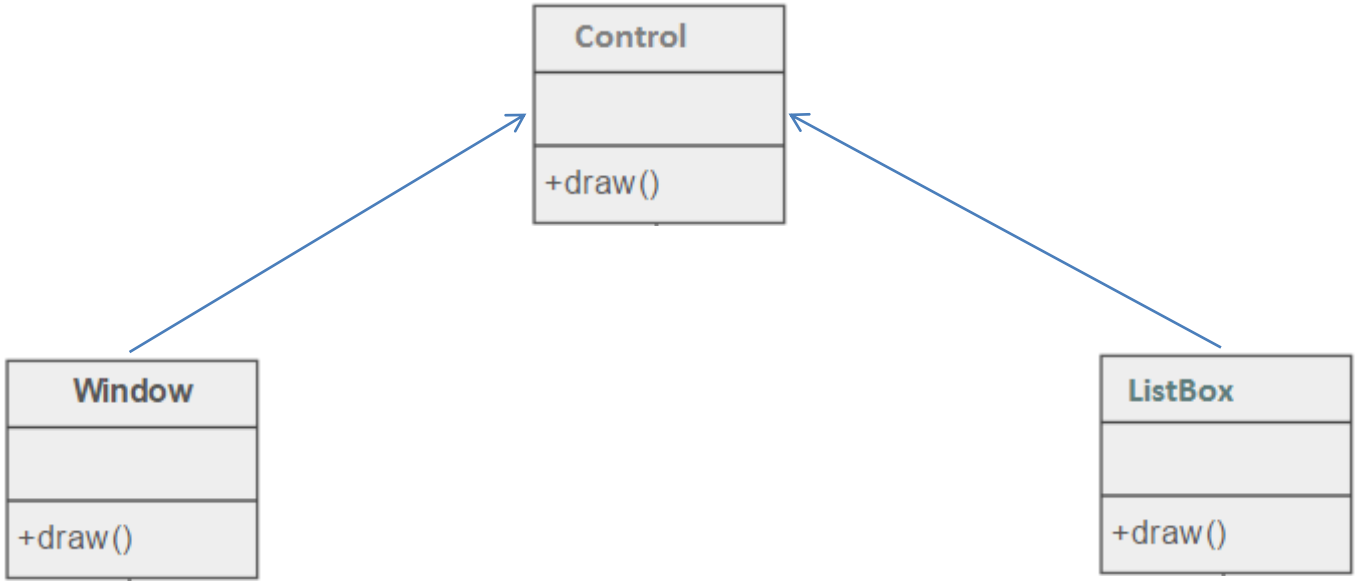
- Może fabryka dokumentów ?
- Widoczność elementów
- Czy wstrzykiwać
- Czy jedna fabryka czy wiele
- Widoczność assembly
- SOLID

Zadanie

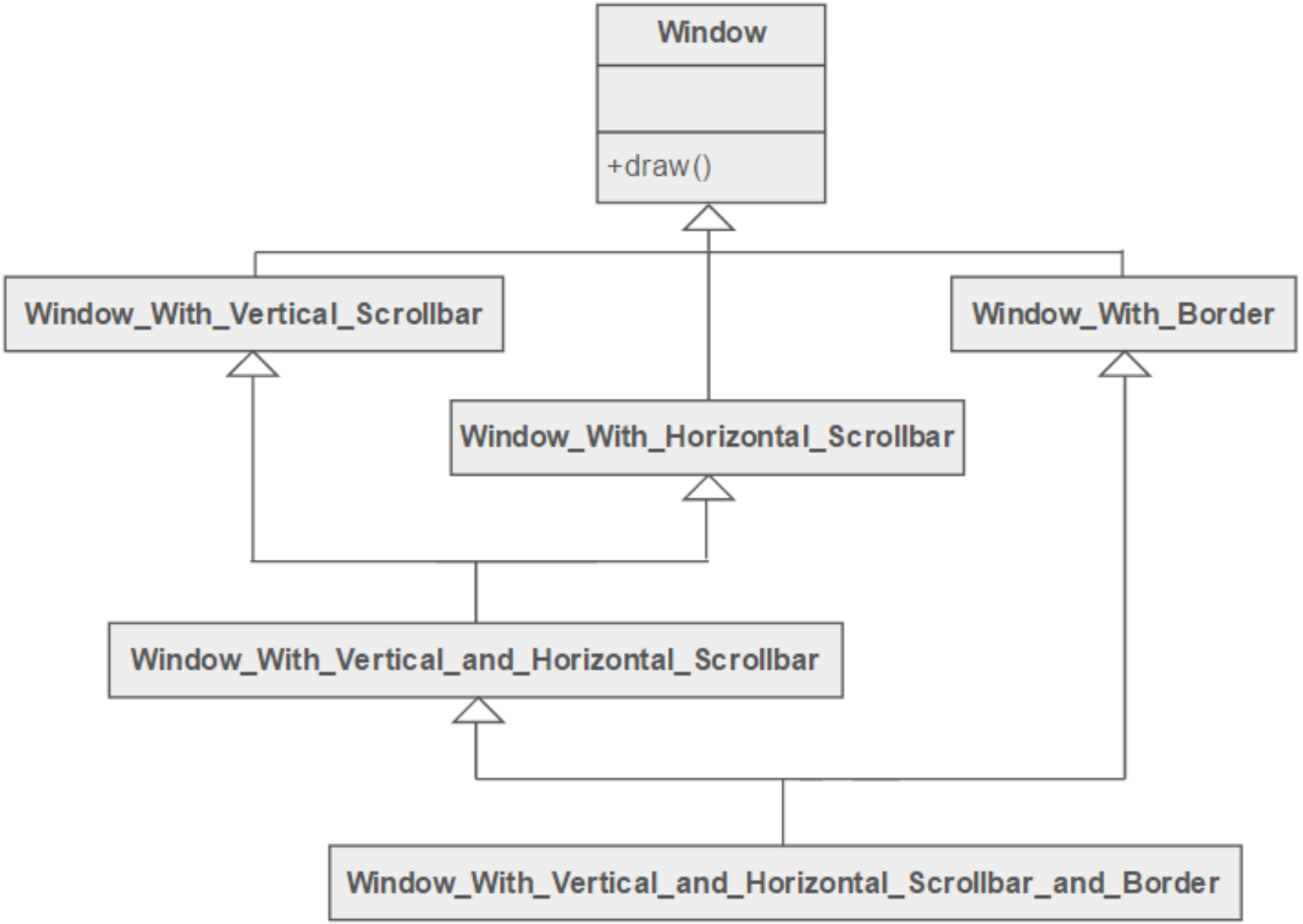
Jak zamodelować rysowanie elementu graficznego (kontrolki) tak aby można było rozszerzać o:

- Suwak (pion / poziom)
- Ramka

Z powinna mieć charakter addytywny – nie zmieniamy istniejącego kodu

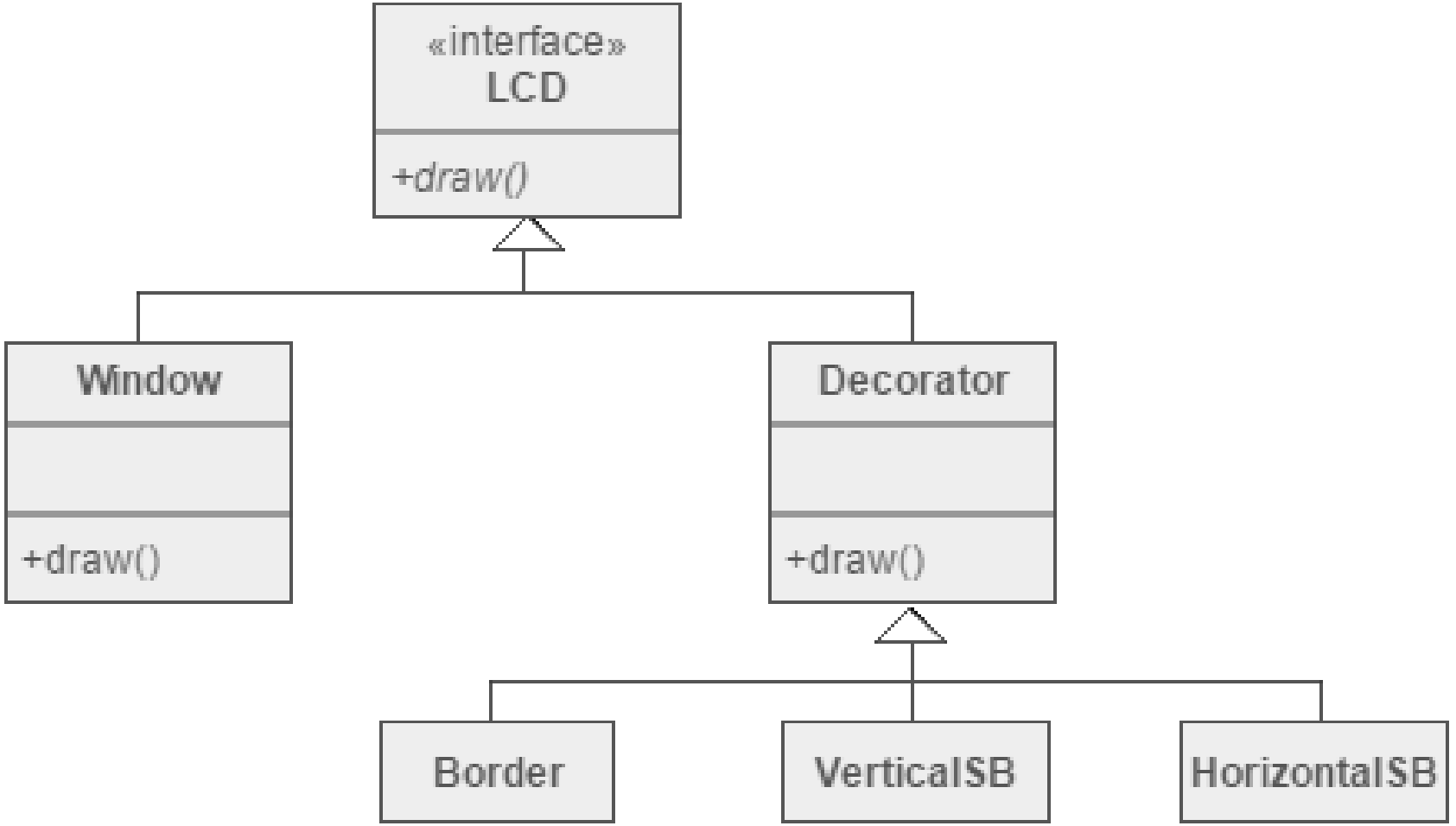


Dziedziczenie – eksplozja kombinatoryczna



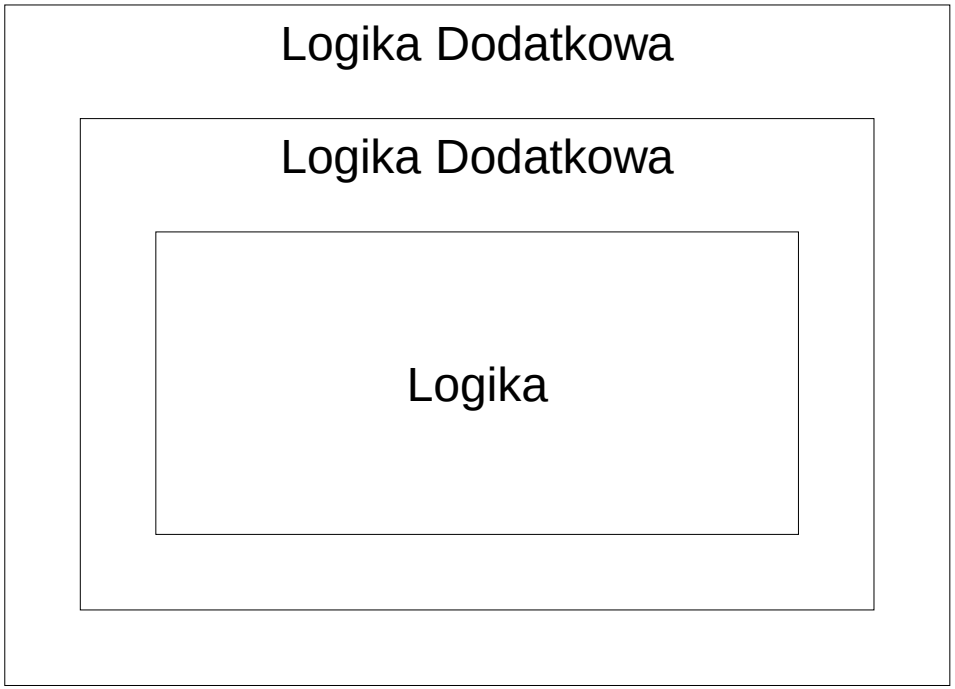
bottega IT SOLUTIONS

Dekorator

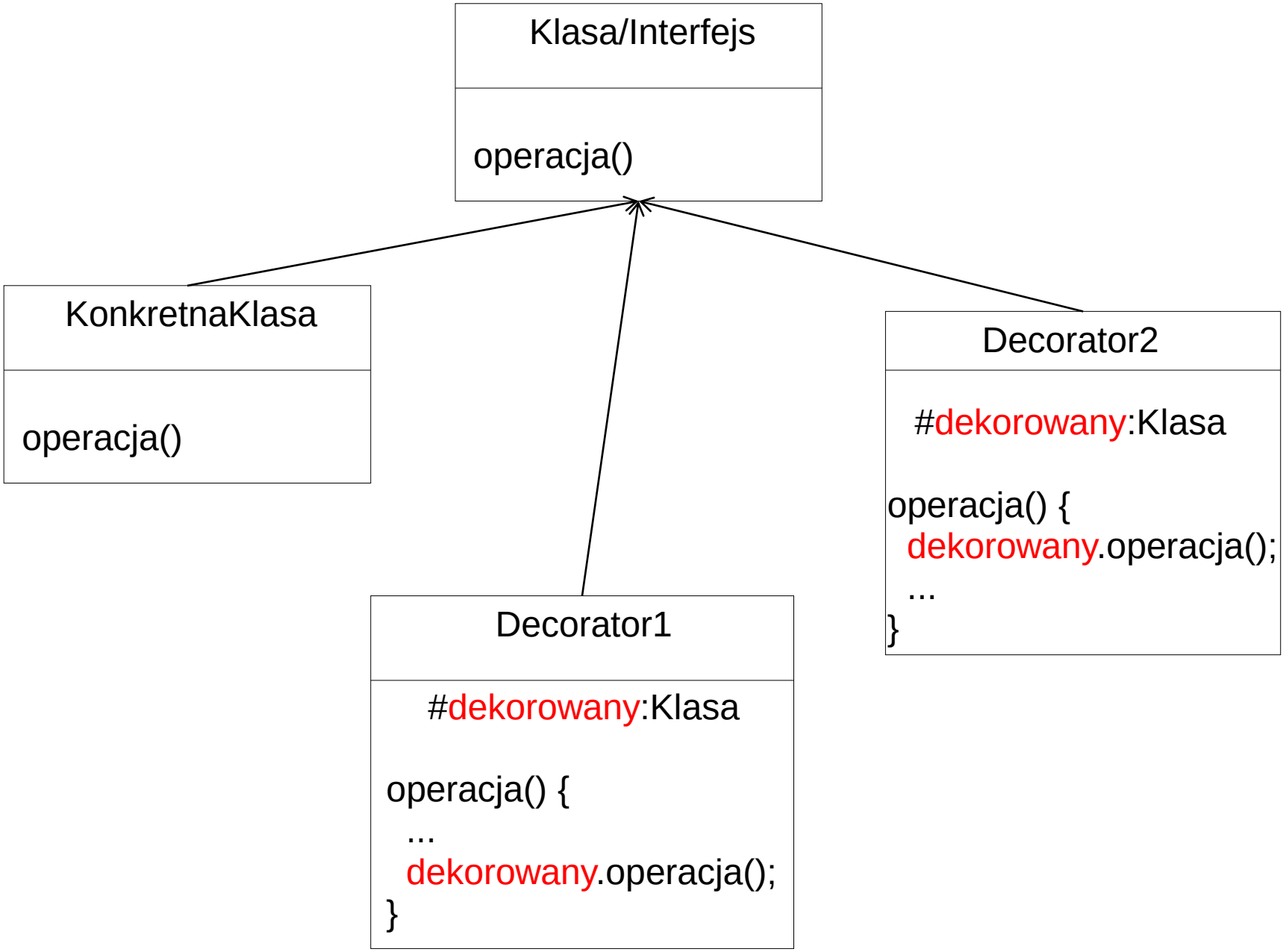


bottega IT SOLUTIONS

Dekorator



Dekorator - struktura

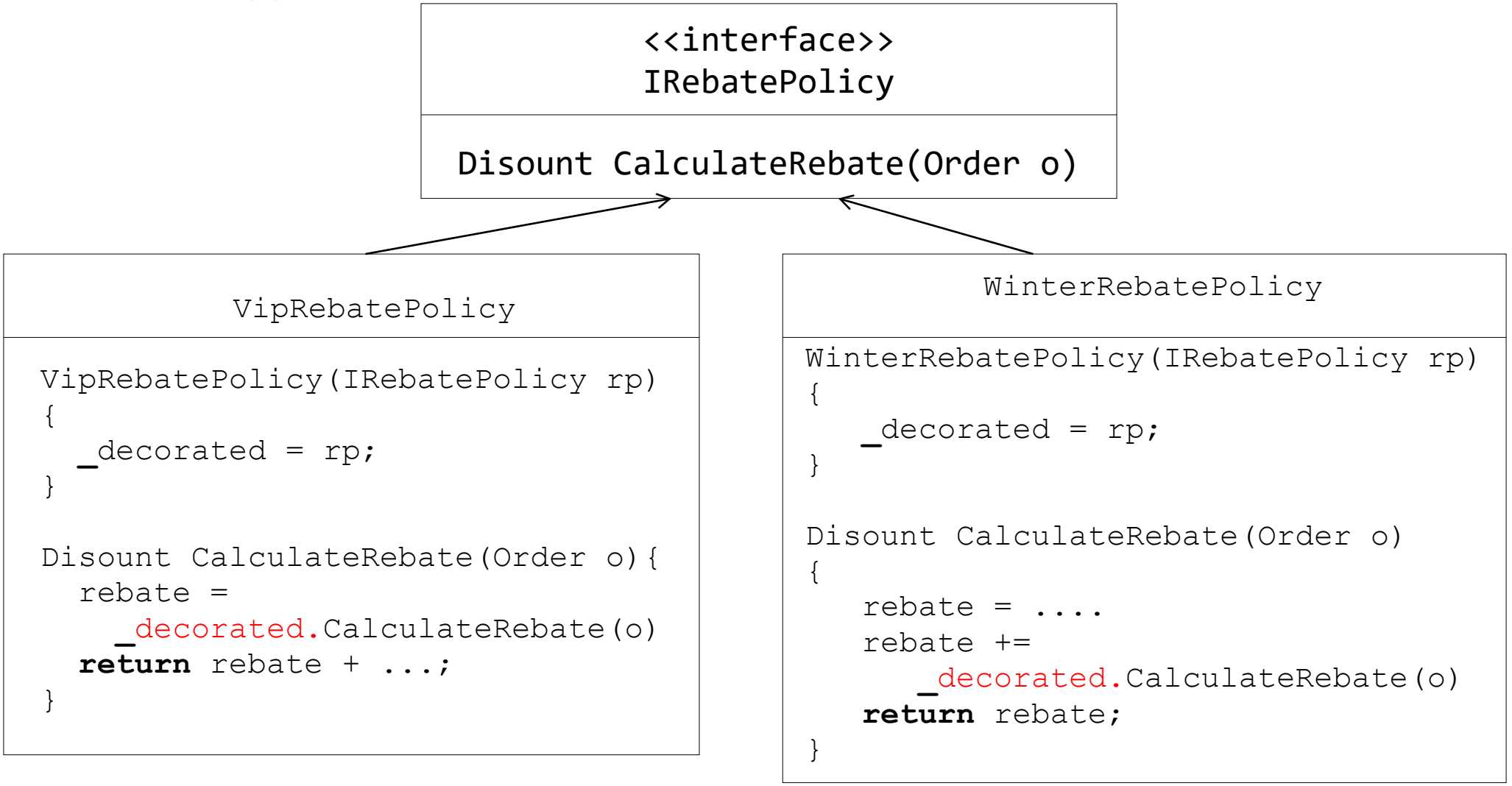


bottega IT SOLUTIONS

Dekorator - przykład

- Naliczanie rabatu może uwzględniać:
  - Promocję z okazji zimy
  - Status VIP
  - ew. inne uwarunkowania
  - Dowolną kombinację powyższych

Dekorator - przykład



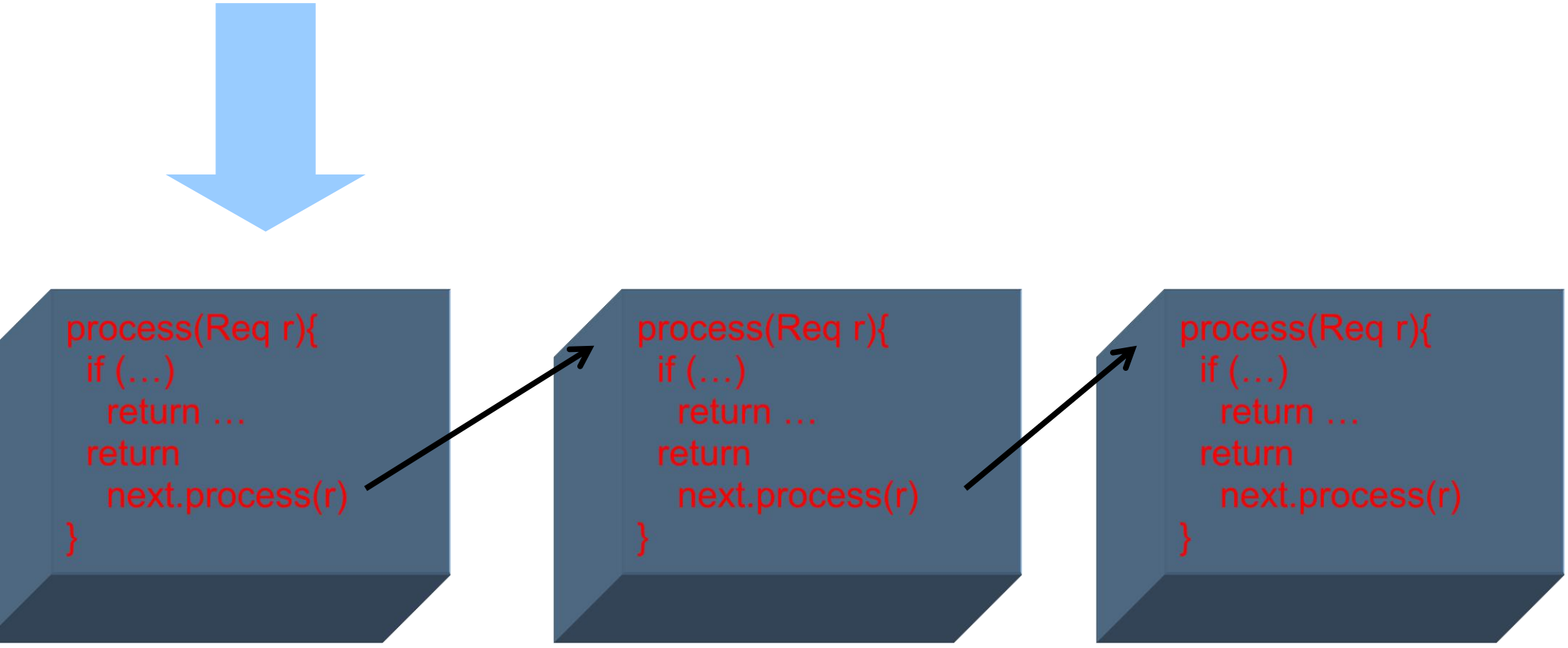

```
RebateCounter rc = new VipRebatePolicy(new WinterRebatePolicy(new RabatPolicy()));
rc.CountRebate(order);
```

Zadanie

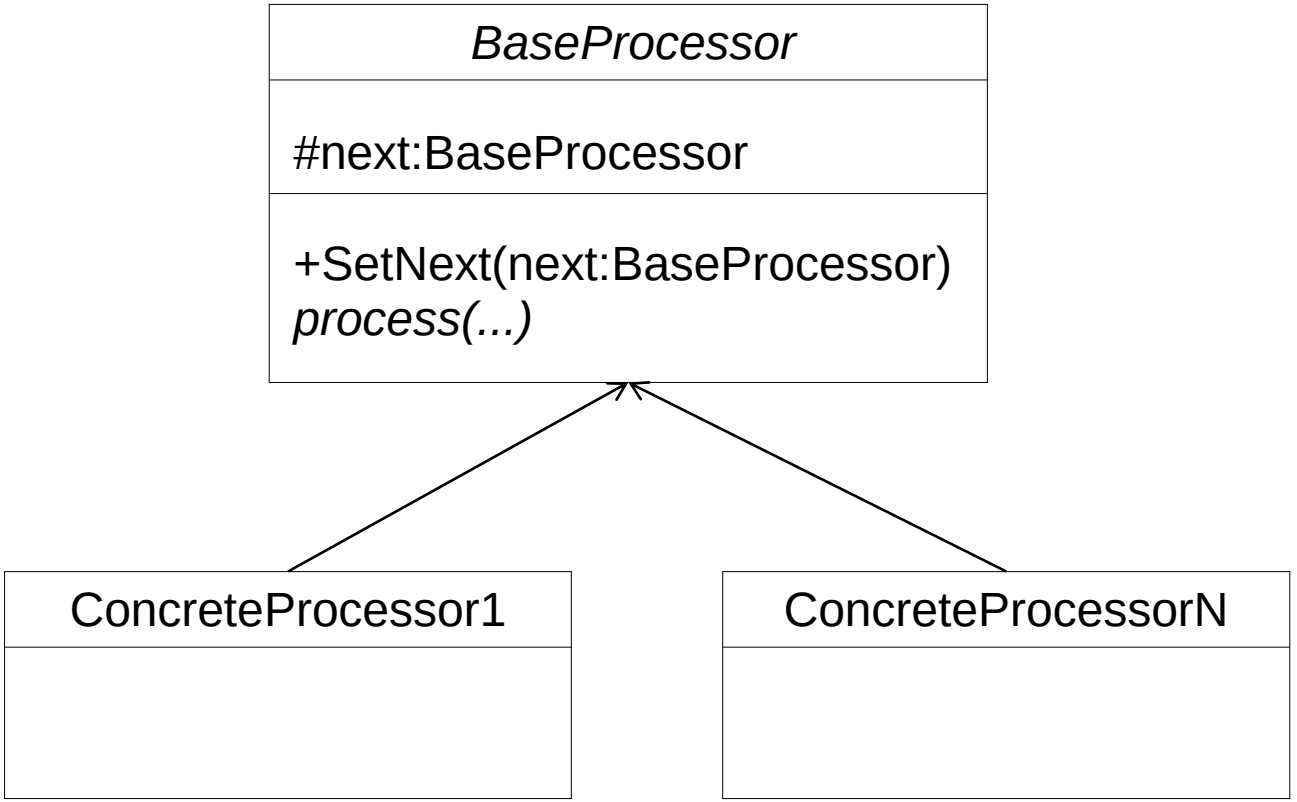
bottega
IT SOLUTIONS

Techsupport

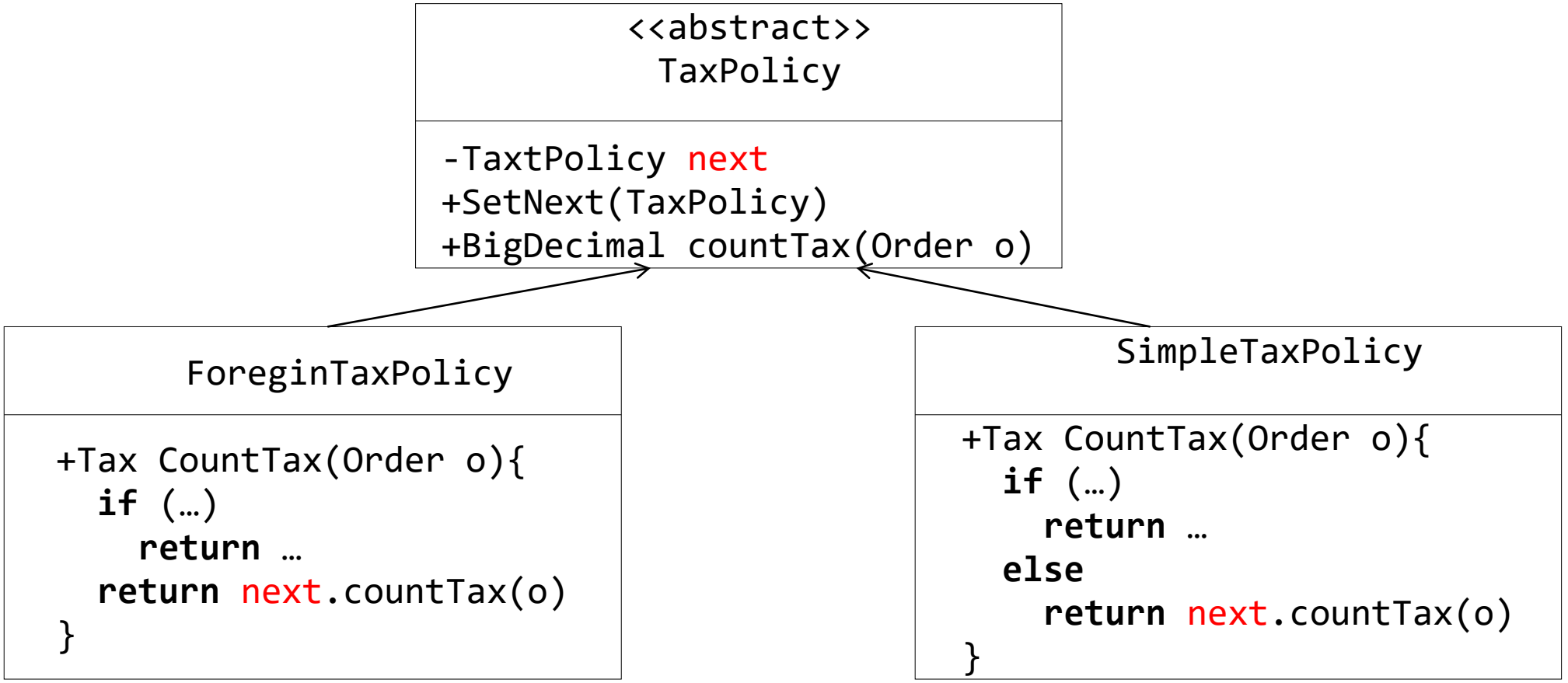
Chain of Responsibility



Chain of Responsibility - Struktura



Chain of Responsibility - Przykład biznesowy




```
TaxPolicy tp = new ForeginTaxPolicy();
tp.SetNext(new SimpleTaxPolicy());
...
tp.CountTax(order);
```

Chain of responsibility - przykład

```
public class SoulutionExplorer
{
  private object _currentItem;

  public void ItemClicked()
  {
    if(_currentItem is FolderItem)
    {
      FolderHandler.Handle(_currentItem as FolderItem);
    }
    else if((_currentItem is ProjectItem)
    {
      FolderHandler.Handle(_currentItem as ProjectItem);
    }
    else if(_currentItem is FileItem)
    {
      FolderHandler.Handle(_currentItem as FileItem);
    }
  }
}
```

Chain of responsibility - przykład

Visual Studio – Solution Explorer

```
public class SolutionExplorer
{
  private IHandler _handler;
  private object _currentItem;

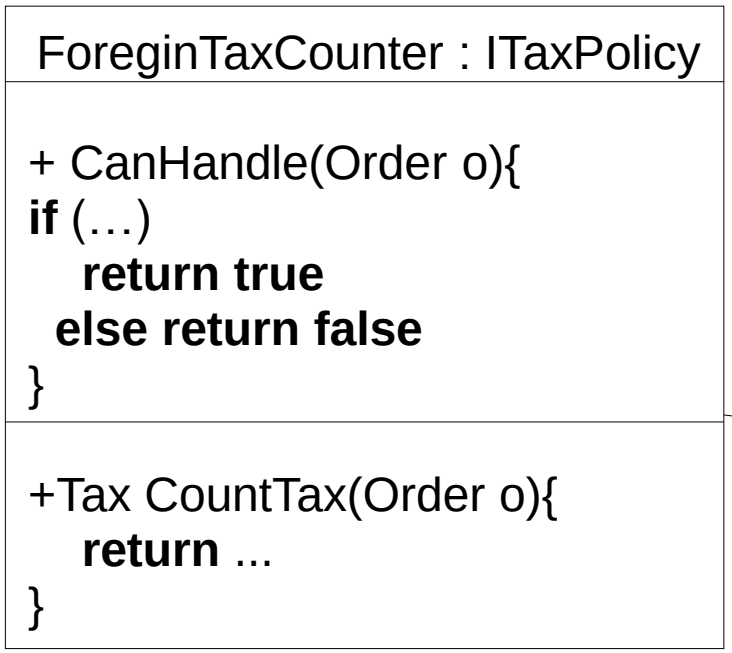
  private void AddHandler(IHandler newHandler)
  {
    newHandler.SetNext(_handler);
    _handler = newHandler;
  }

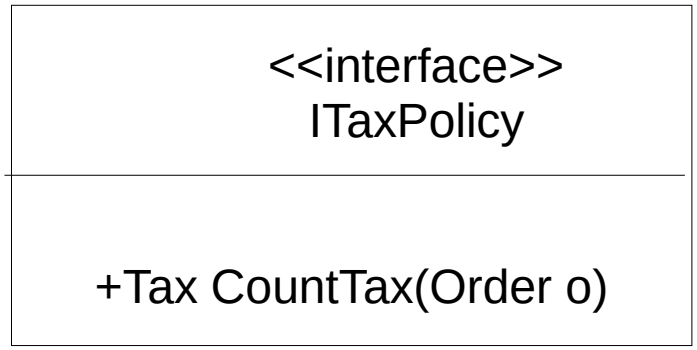
  public void ItemClicked()
  {
    _handle.Handle(_currentItem);
  }
}
```

Chain of Responsibility

Zadanie 1

Chain of Responsibility - Modyfikacja



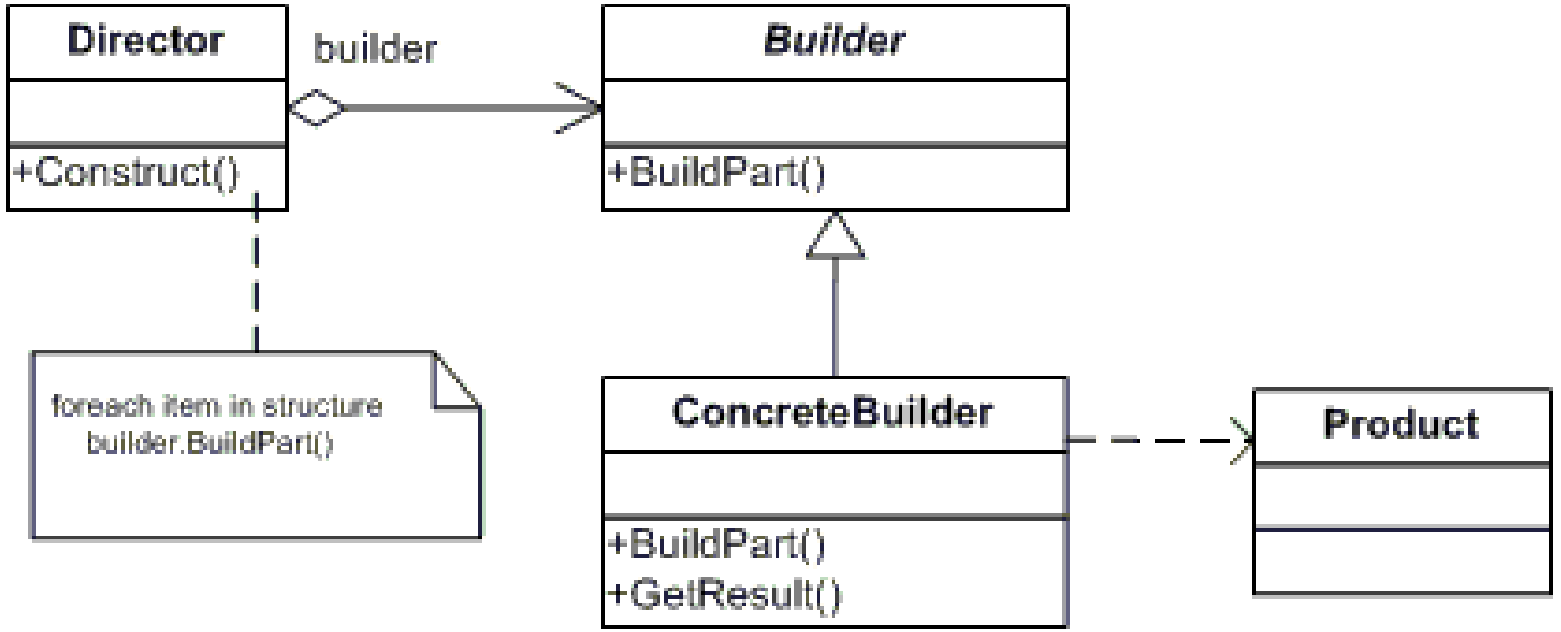



```
public class TaxPolicyChain : TaxPolicy
{
   private List<ITaxPolicy> _policies
   public Tax count(Order o)
   {
      foreach(TaxCounter in _policies)
      {
         if (tc.CanHandle(o))
           return tc.CountTax(o);
      }
      return new BigDecimal(0);
   }
}
```

Zadanie 2

Builder

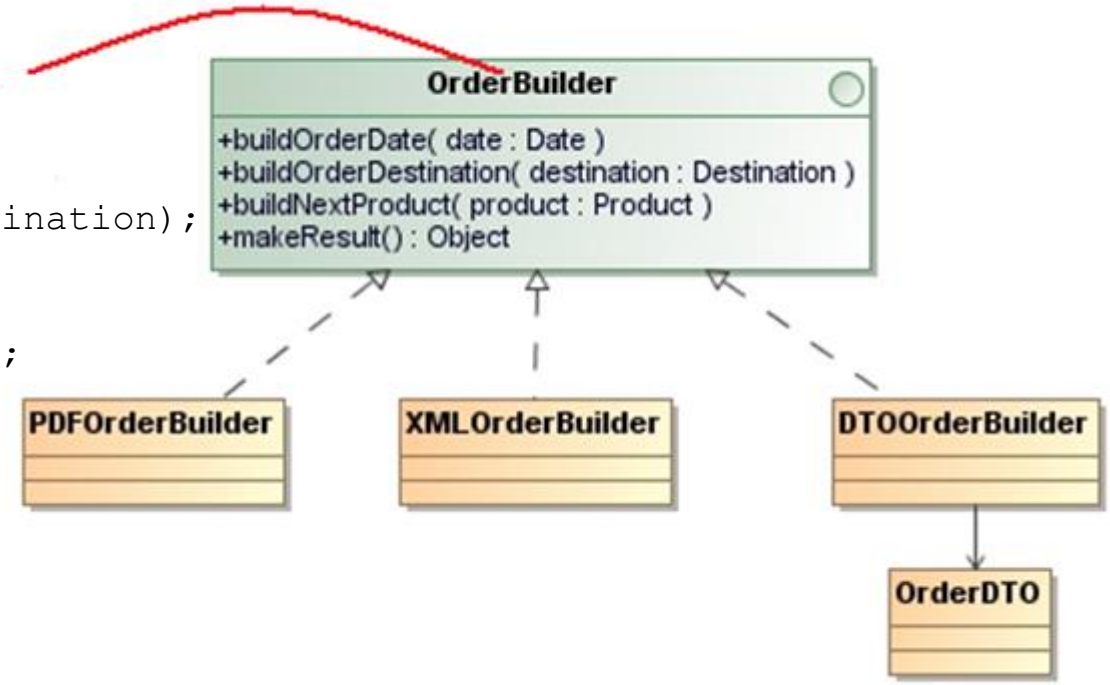
- Kierownik budowy
  - wie z czego budować – jak pozyskać surowce
- Budowniczy
  - wiec jak budować konkretny produkt



Builder – Przypadek Biznesowy

```
public class Order
{
  private DateTime _date;
  private List<Product> _products;
  private Destination _destination;

  object Export(OrderBuilder builder)
  {
    builder.BuildOrderDate(_date);
    builder.BuildOrderDestination(_destination);
    foreach (var product in _products)
    {
      builder.BuildNextProduct(product);
    }
    return builder.GetResult();
  }
}
```



Zadanie

Bankomat - problem

```
public void InsertCard()
    {
      switch (_currentState)
      {
        case MACHINE_STATE.INITIAL:
          _currentState = MACHINE_STATE.CARD_INSERTED;
          break;
        case MACHINE_STATE.CARD_INSERTED:
        case MACHINE_STATE.PIN_ENTERED:
        case MACHINE_STATE.CASH_WITHDRAWN:
          throw new InvalidOperationException("Card already inserted");
        default:
          throw new ArgumentOutOfRangeException();
      }
    }

    public void EnterPin(Pin pin)
    {
      switch (_currentState)
      {
        case MACHINE_STATE.INITIAL:
          throw new InvalidOperationException("No card inserted");

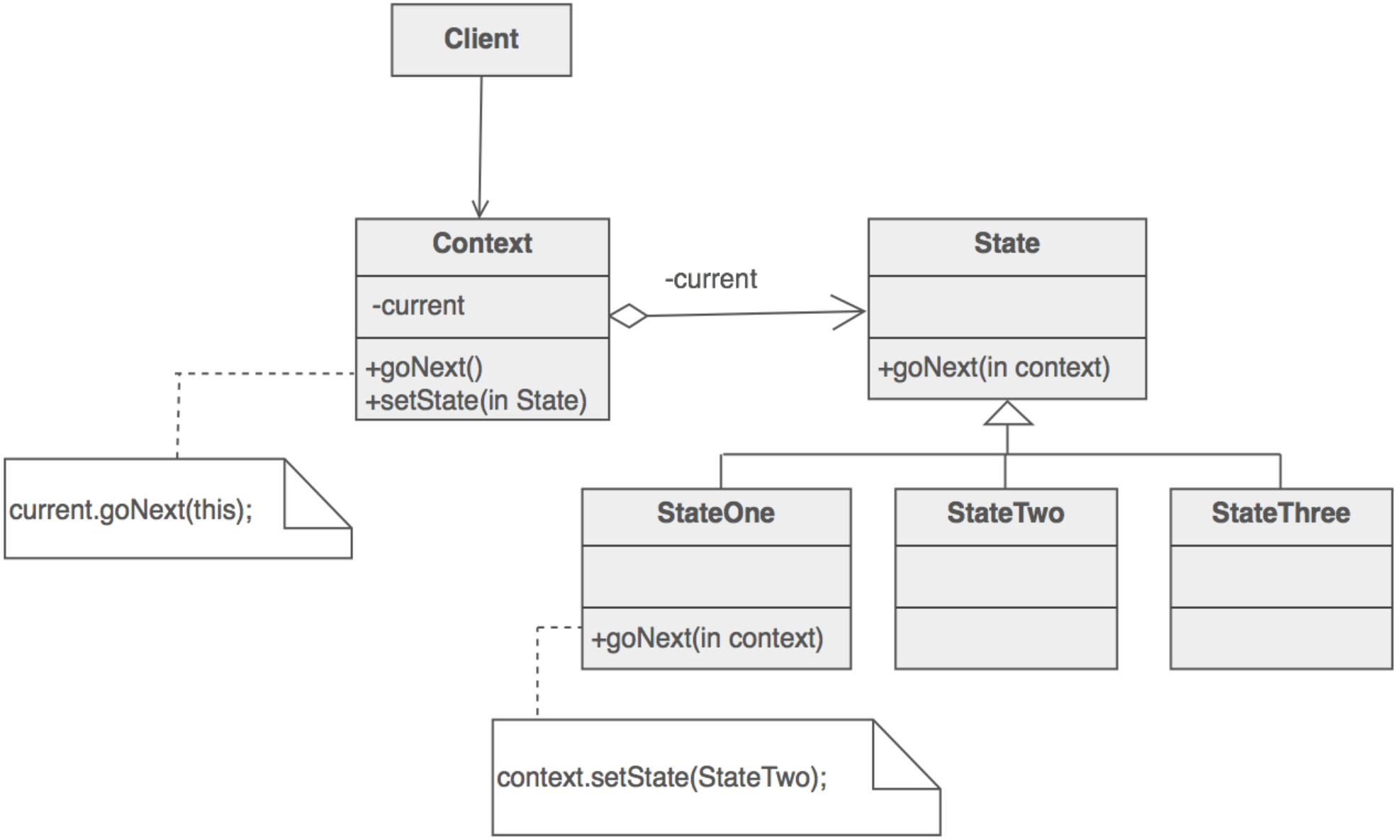
        case MACHINE_STATE.CARD_INSERTED:
          if (pin != 1234)throw new InvalidOperationException("incorect pin");
          _currentState = MACHINE_STATE.PIN_ENTERED;
          break;

        case MACHINE_STATE.PIN_ENTERED:
        case MACHINE_STATE.CASH_WITHDRAWN:
          throw new InvalidOperationException("Pin already entered");

        default:
          throw new ArgumentOutOfRangeException();
      }
    }
```

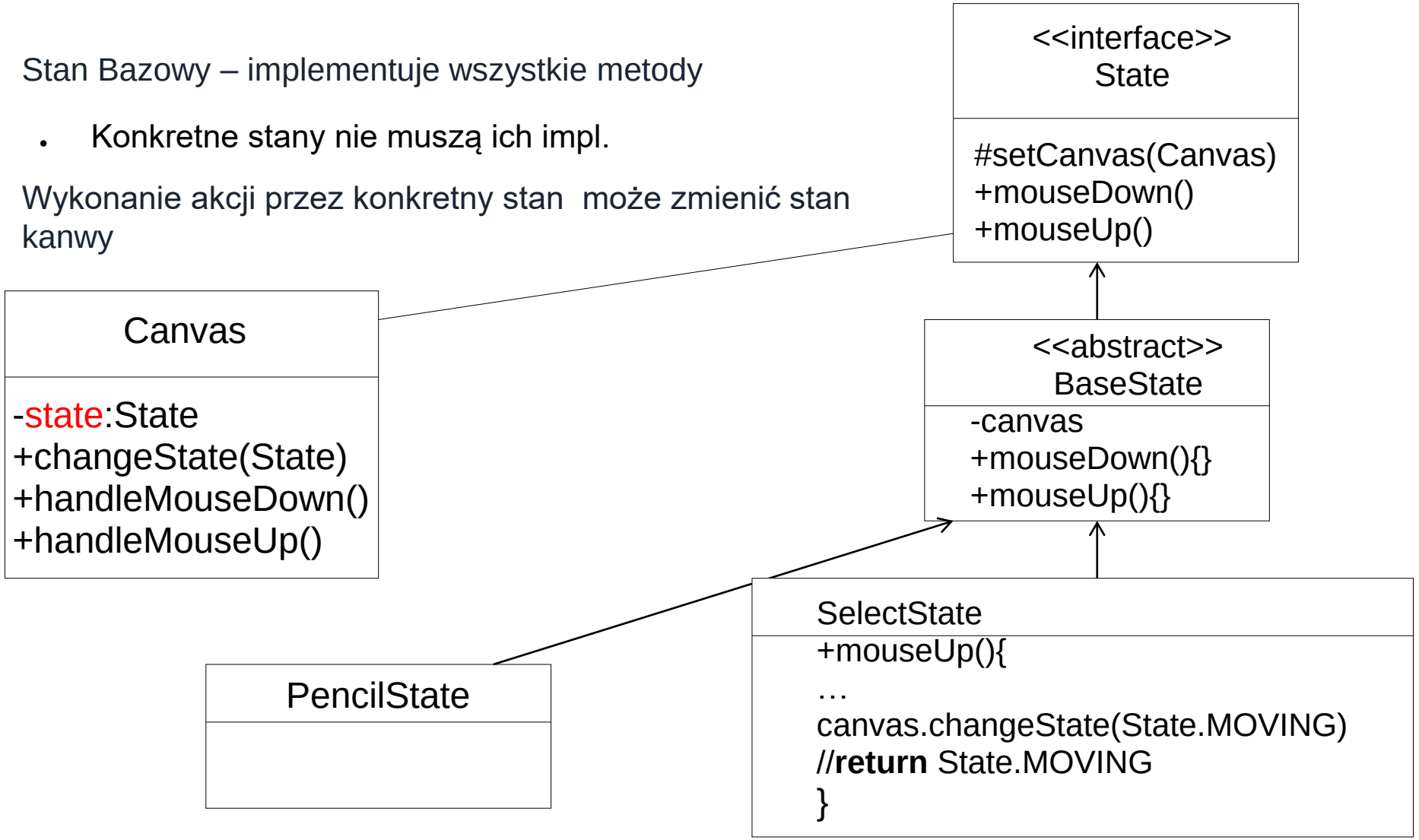
bottega
IT SOLUTIONS

State



State – przykład techniczny

- Stan Bazowy – implementuje wszystkie metody
  - Konkretne stany nie muszą ich impl.
- Wykonanie akcji przez konkretny stan może zmienić stan kanwy



State – przykład biznesowy

- Zamówienie może znajdować się w kilku stanach
  - draft, tymczasowe, zatwierdzone, w realizacji, zrealizowane,..
- Zamówienie potrafi wykonać następujące polecenia biznesowe:
  - dodać i usunąć produkt
  - zmienić adres dostawy
  - zatwierdzić się

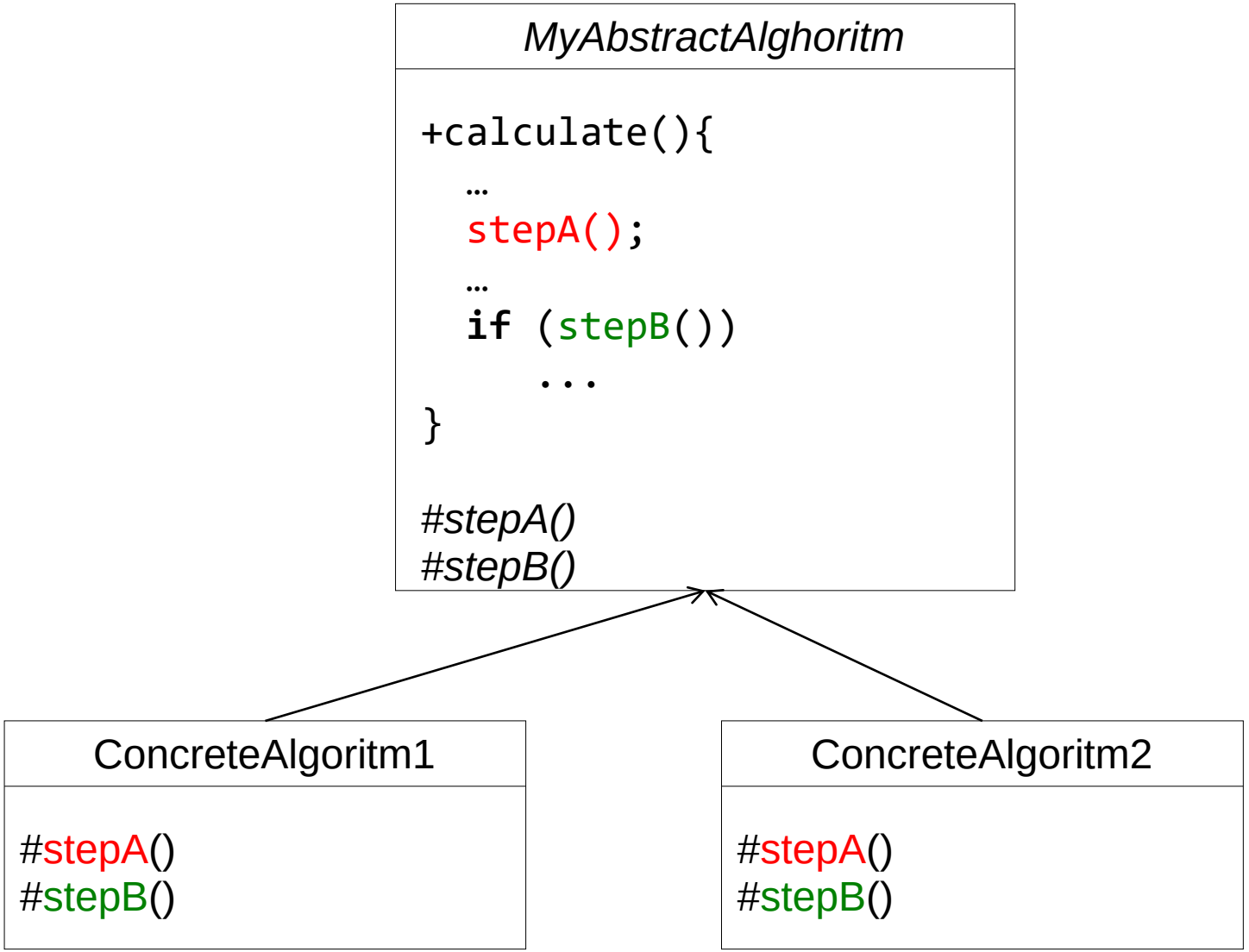
Zadanie

bottega IT SOLUTIONS

Template Method

- Wzorzec behawioralny
- Definiuje szkielet algorytmu
  - pewne kroki są abstrakcyjne
- Klasy potomne definiują abstrakcyjne zachowanie
- Widoczność metod
  - metoda szablonowa – publiczna
  - metody abstrakcyjne - chroniony

Template Method



bottega IT SOLUTIONS

Template Method

- Wzorzec TM ma ograniczone zastosowanie
- Problem pojawia się gdy pewne konkretne klasy współdzielą zachowanie
  - Prowadzi do eksplozji kombinatorycznej
    - konieczność tworzenia klas per każda możliwa kombinacja
- Wówczas należy użyć strategii
  - refaktoryzacja powinna być nieinwazyjna jeżeli zastosujemy interfejs

Zadanie

bottega IT SOLUTIONS

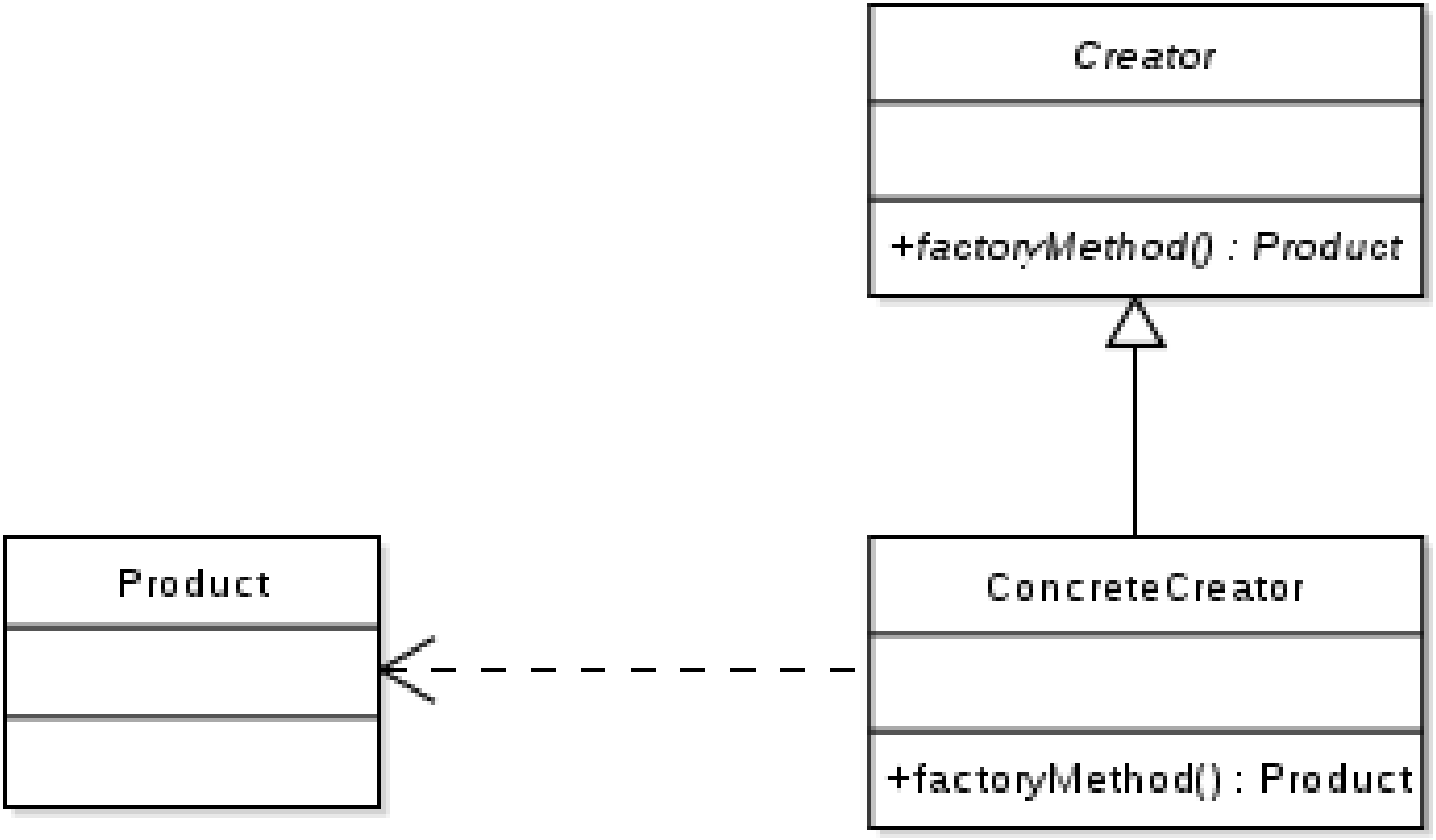
Factory Method

- Odmiana metody szablonowej
  - Wzorzec kreacyjny
- Metoda szablonowa, która tworzy obiekt

```
public class Phone
{
    public static Phone FromString(string s)
    {
      ///.....
      return new Phone(…);
    }
}
```

bottega
IT SOLUTIONS

Factory Method



bottega IT SOLUTIONS

Adapter

- Wzorzec strukturalny
- Umożliwia współpracę dwóm klasom o niekompatybilnych interfejsach
  - przekształca jeden interfejs na drugi
    - różne metody
    - różna ilość metod
    - różne parametry
    - różne wyjątki
- Modyfikacja – Wrapper
  - opakowanie istniejącego interfejsu w nowy

bottega IT SOLUTIONS

Adapter

- Jeden ze sposobów obliczania rabatu korzysta z istniejącej biblioteki klienta
  - w szczególności może to być WebService etc
- Chcemy zachować spójność core biznesowego integrując się z tą biblioteką

bottega
IT SOLUTIONS

Value Objects

bottega IT SOLUTIONS

Value Objects

- Brak rozróżnienia – brak identyfikacji
  - VO są takie same gdy ich atrybuty są takie same
- Zwykle są *immutable* – ponieważ nie mają tożsamości
  - zatem można ich reużywać – nie dbamy o konkretną instancję, brak efektów ubocznych

bottega IT SOLUTIONS

Value Objects

- Wyrażają jakieś znaczenie – wchodzą do słownictwa
  - zwiększają siłę wyrazu koncepcji i czytelności kodu
  - więcej znaczenia niż zwykły String, Integer itp.
- Zawierają użyteczne metody (zamiast Utils)
  - walidacja (np. w konstruktorze)
- Przykłady: kolor, punkt, address*, numer tel, money

bottega IT SOLUTIONS

Przykład

```
public class DocumentNumber : ValueObject
{
  public const int MaxLen = 128;
  private string _value;

  public DocumentNumber(string value)
  {
    _value = value;
  }

  public static implicit operator DocumentNumber(string number)
  {
    return new DocumentNumber(number);
  }

  public static implicit operator string(DocumentNumber number)
  {
    if (number != null) {      return number._value;    }
    return null;
  }
  public override string ToString()
  {
    return _value;
  }
}
```

VO -Zapis do bazy danych

```
public OrderMap()
{
  Lazy(false);

  Id(x => x.Id);
  //...
  Property("_created",
            m => m.Type(new ValueTypeAsStringType<Date>()));
}
```

bottega IT SOLUTIONS

Typowi kandydaci do VO

- String z ograniczeniami formatowania
  - zip code
  - nazwa
- Liczby z ograniczeniami
  - procent (jako ułamek? jako liczba całkowita?), ilość w jednostkach
- Złożone struktury
  - money (również waluta, data), adres (również czas obowiązywania), przedział czasu (operacje: przecięcia, czas trwania,...)
- Obiekty zwracane przez metody innych klass

bottega IT SOLUTIONS

VO - przykłady

- Money
  - hermetyzuje implementację (BigDecimal, Integer z przesuniętym przecinkiem)
  - hermetyzuje reprezentację – bazowa waluta
  - zawiera wygodne metody: przeliczanie
- PhoneNumber
  - hermetyzuje reprezentację
  - zawiera wygodne metody: ekstrakcja numeru kierunkowego, walidacja, etc

bottega
IT SOLUTIONS

Przykład Value Object

bottega IT SOLUTIONS

Zastosowania VO

- Wrappery typów „technicznych” nadające znaczenie biznesowe
- Parametry – sposób na wymianę danych między złożonymi obiektami
  - hermetyzacja wewnętrznej struktury
- Clean Code, redukcja „code smell”:
  - Primitive Obsession, Data Clumps, Long Parameter List

bottega IT SOLUTIONS

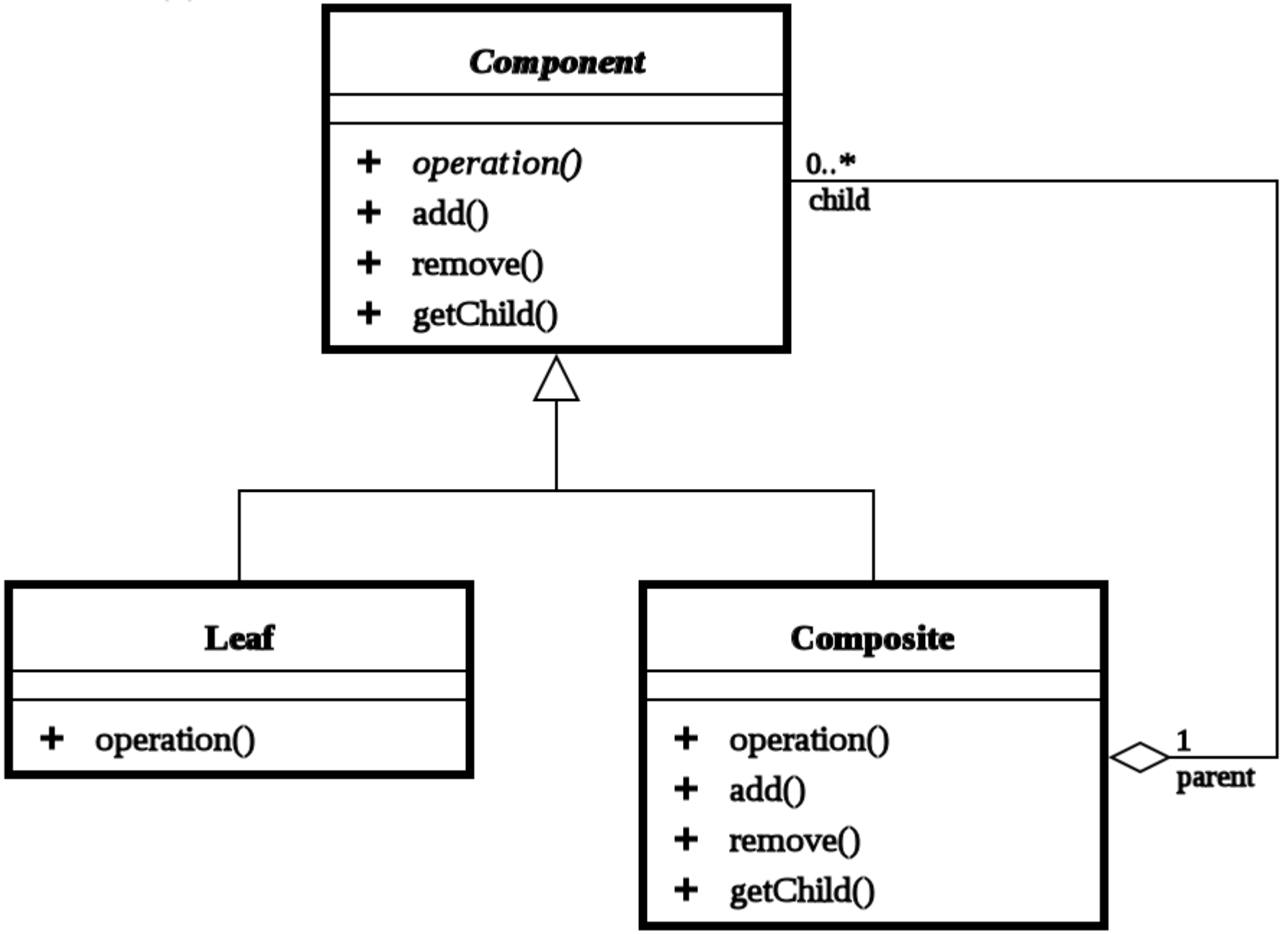
Jak można zrefaktorowac ten kod ?

```
xml.append("<orders>");
for (int i = 0; i < orders.getOrderCount(); i++) {
  Order order = orders.getOrder(i);
  xml.append("<order");
  xml.append(" id='");
  xml.append(order.getOrderId());
  xml.append("'>");
  for (int j=0; j < order.getProductCount(); j++) {
    Product product = order.getProduct(j);
    xml.append("<product");
    xml.append(" id='");
    xml.append(product.getID());
    xml.append("'");
    xml.append(" color='");
    xml.append(getColorFor(product));
    xml.append("'");
    if (product.getSize() !=
      ProductSize.NOT_APPLICABLE) {
      xml.append(" size='");
      xml.append(getSizeFor(product));
      xml.append("'");
    }
    xml.append(">");
    xml.append("<price");
    xml.append(" currency='");
    xml.append(getCurrencyFor(product));
    xml.append("'>");
    xml.append(product.getPrice());
    xml.append("</price>");
    xml.append(product.getName());
    xml.append("</product>");
  }
  xml.append("</order>");
}
xml.append("</orders>");
```

bottega IT SOLUTIONS

Composite

- Wzorzec strukturalny
- Reprezentacja struktury drzewiastej
  - bez rozróżniania węzła od liścia
  - jeżeli węzły i liście posiadają **wspólne operacje**
- Klient może wykonywać polecenia na korzeniu
  - delegacja polecenia w głąb struktury drzewa
- Przykład: wszelkie biblioteki komponentów graficznych (panele/kontenery + kontrolki)


Component
+ operation()
+ add()
+ remove()
+ getChild()
0..*
child
Leaf
+ operation()
Composite
+ operation()
+ add()
+ remove()
+ getChild()
1
parent

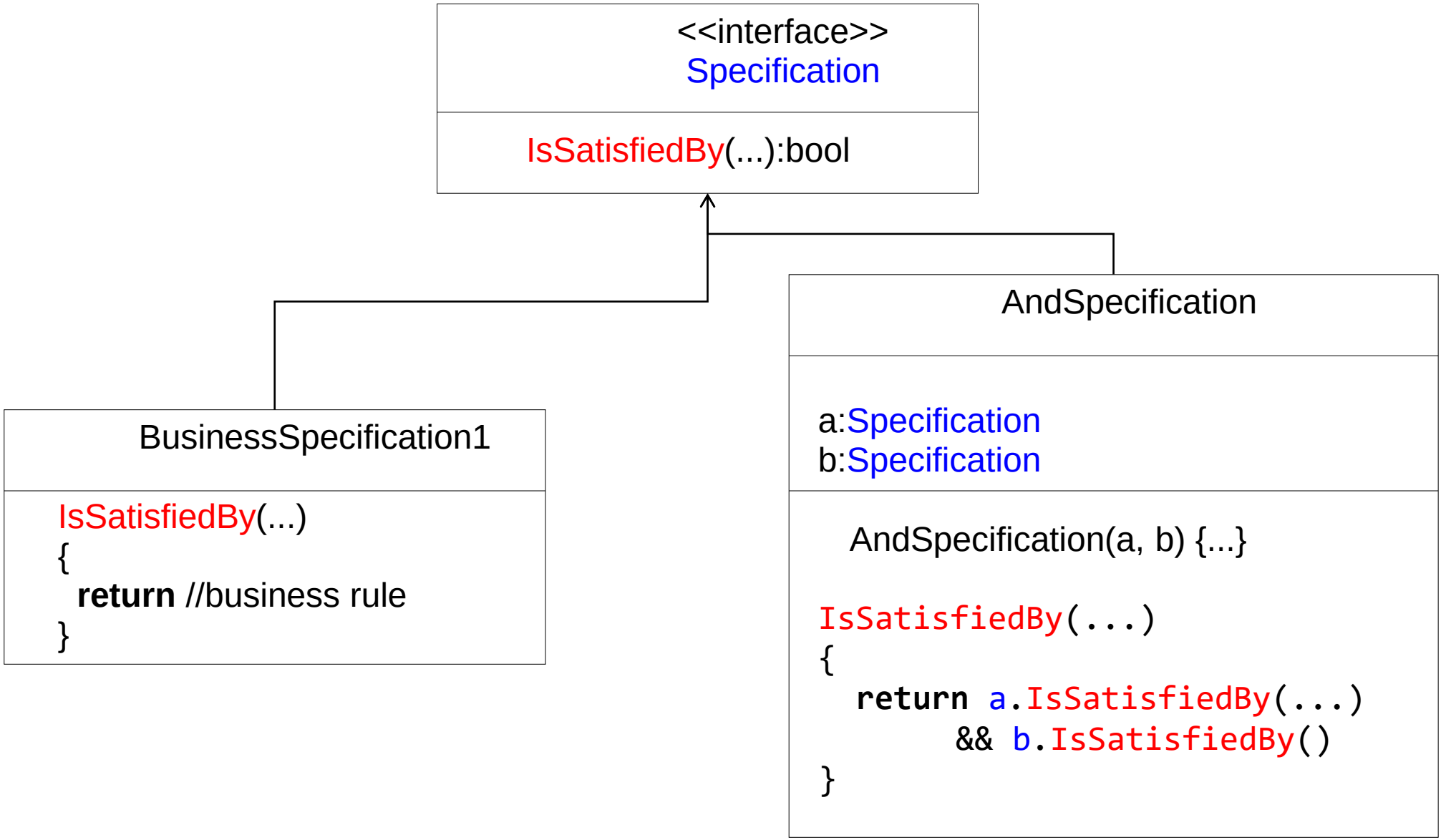
bottega
IT SOLUTIONS

Visitor

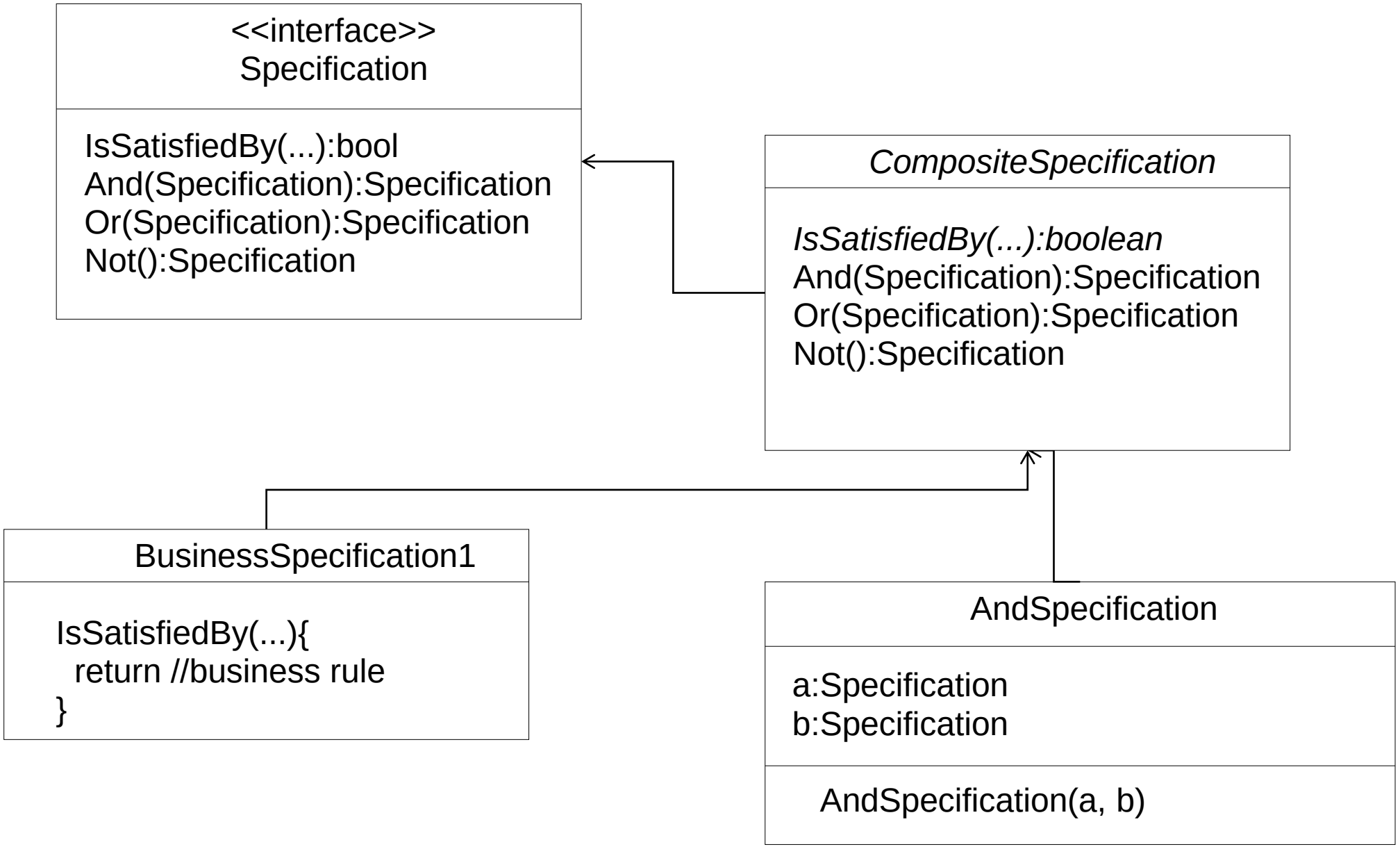
bottega IT SOLUTIONS

Specification

- Wzorzec behawioralny
- Pozwala na składanie logiki biznesowej z mniejszych elementów
  - logika polega na udzieleniu odpowiedzi czy pewien obiekt spełnia specyfikację
- Pomiędzy elementami mogą zachodzić relacje logiczne
- Jeden z Building Block w Domain Driven Design


<<interface>>
Specification
IsSatisfiedBy(...):bool
BusinessSpecification1
IsSatisfiedBy(...)
{
return //business rule
}
AndSpecification
a:Specification
b:Specification
AndSpecification(a, b) {...}
IsSatisfiedBy(...)
{
return a.IsSatisfiedBy(...)
&& b.IsSatisfiedBy()
}

Specification - dsl



Specification - przykład

```
public interface ISpecification<T>
{
  bool IsSatisfiedBy(T candidate);
  ISpecification<T> And(ISpecification<T> other);
  ISpecification<T> Or(ISpecification<T> other);
  ISpecification<T> Not();
}
```

```
ISpecification<User> grandpaRockersSpecification =
   new GenderSpecification(Gender.Male)
    .And(new AgeSpecification(60, 100))
    .And(new BikeSpecification(MotorbikeType.Suzuki).Not())
    .And(
      new BikeSpecification(MotorbikeType.HarleyDavidson)
      .Or(new BikeSpecification(MotorbikeType.Honda)));
```

Specification – przykład biznesowy

- System decyduje o udzieleniu kredytu
- Decyzja jest podejmowana na podstawie zestawu wielu kryteriów
  - zestawy kryteriów różnią się u różnych klientów (w różnych wdrożeniach)
    - ale część z nich jest wspólna
  - można zauważyć, że istnieje pewna pula kilkudziesięciu kryteriów
    - w danym wdrożeniu używa się kilkunastu kryteriów z puli

Zadanie

bottega IT SOLUTIONS

Podsumowanie

- Singleton
- Fabryka
- Strategia
- Dekorator
- Chain of Responsibility
- Builder
- Assembler / AssertObject
- State
- Template Method
- Factory Method
- Adapter
- ValueObject
- Composite
- Specification