Wybrane aspekty projektowania - budowa wielowarstwowego modelu implementacji, zastosowanie wzorców projektowych
Wykład 7– część 2

Zofia Kruczkiewicz

Literatura

1. Roger S. Pressman, Praktyczne podejście do oprogramowania, WNT, 2004
2. Stephen H. Kan, Metryki i modele w inżynierii jakości oprogramowania, Mikom, 2006
3. Jacobson, Booch, Rumbaung, The Unified Software Development Process,Addison Wesley, 1999
4. Shalloway A.,Trott James R.,Projektowanie zorientowane obiektowo. Wzorce projektowe. Gliwice, Helion, 2005
5. Robert C. Martin, Micah Martin, Agile Programowanie zwinne. Zasady, wzorce i praktyki zwinnego wytwarzania oprogramowania w C#, Helion 2008
6. D.Alur, J.Crupi, D. Malks, Core J2EE. Wzorce projektowe
7. https://docs.oracle.com/javaee/7/JEETT.pdf

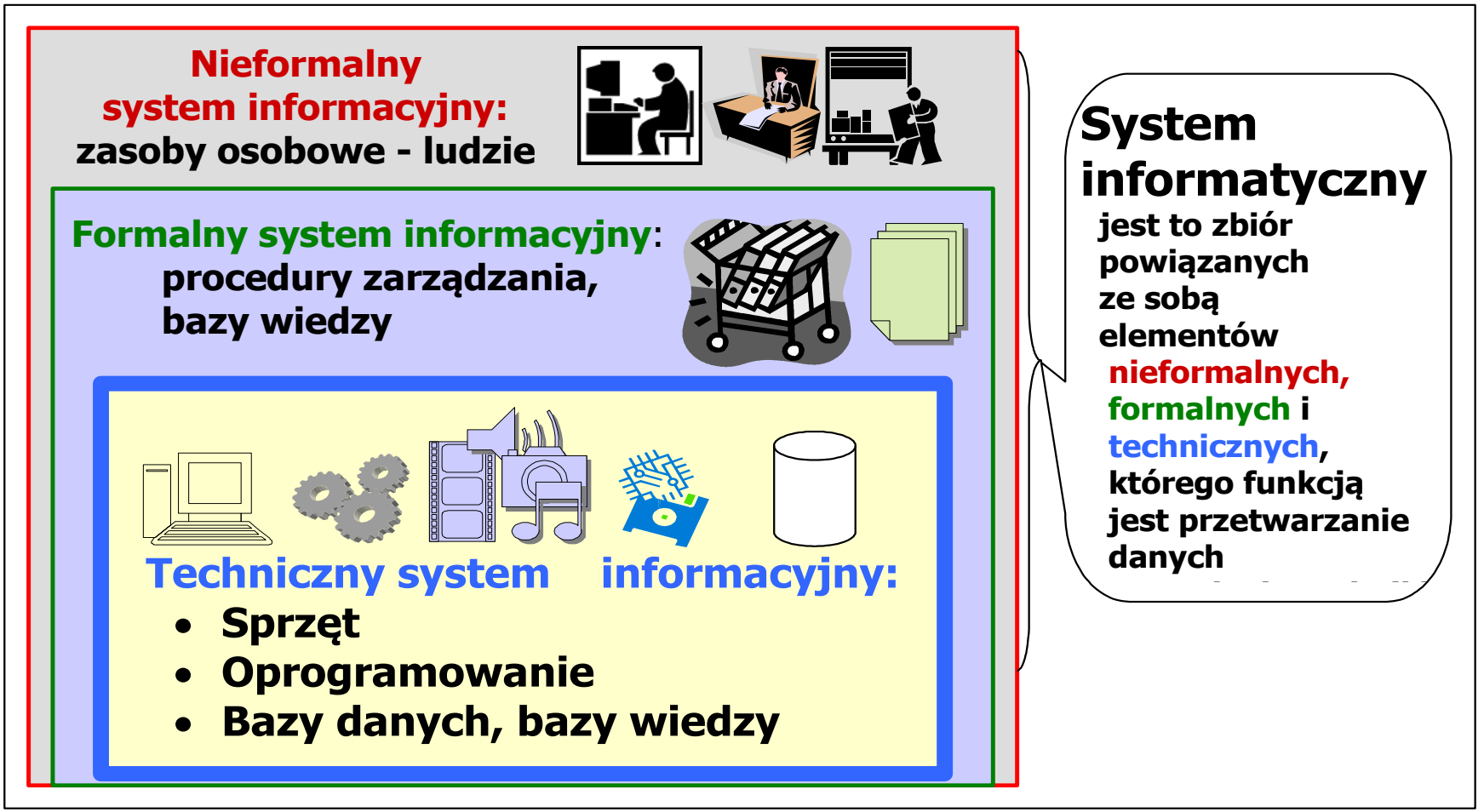
Zagadnienia

1. Wielowarstwowa architektura systemu informatycznego
2. Ocena i poprawa (refaktoryzacja) architektury wielowarstwowej systemu informatycznego
3. Wzorce projektowe stosowane przy budowie wielowarstwowej aplikacji internetowej
4. Przykład modelowania i projektowania części warstwy biznesowej z obiektami typu POJO. Wykonanie aplikacji dwuwarstwowej dla jednego użytkownika.
5. Przykłady architektury wielowarstwowej aplikacji internetowej typu EE. Wykonanie aplikacji typu EE. Warstwa biznesowa: komponenty typu EJB + obiekty POJO
6. Warstwa zasobów (EIS) - baza danych w systemie baz danych Derby
7. Utworzenie obiektowego modelu danych do utrwalania ORM
8. Warstwa integracji. Zastosowanie wzorca projektowego typu *Domain Store* w technologii JPA (Java Persistence) na platformie Java EE
9. Warstwa prezentacji - JSF
10. Dodatek

Zagadnienia

1. **Wielowarstwowa architektura systemu informatycznego**

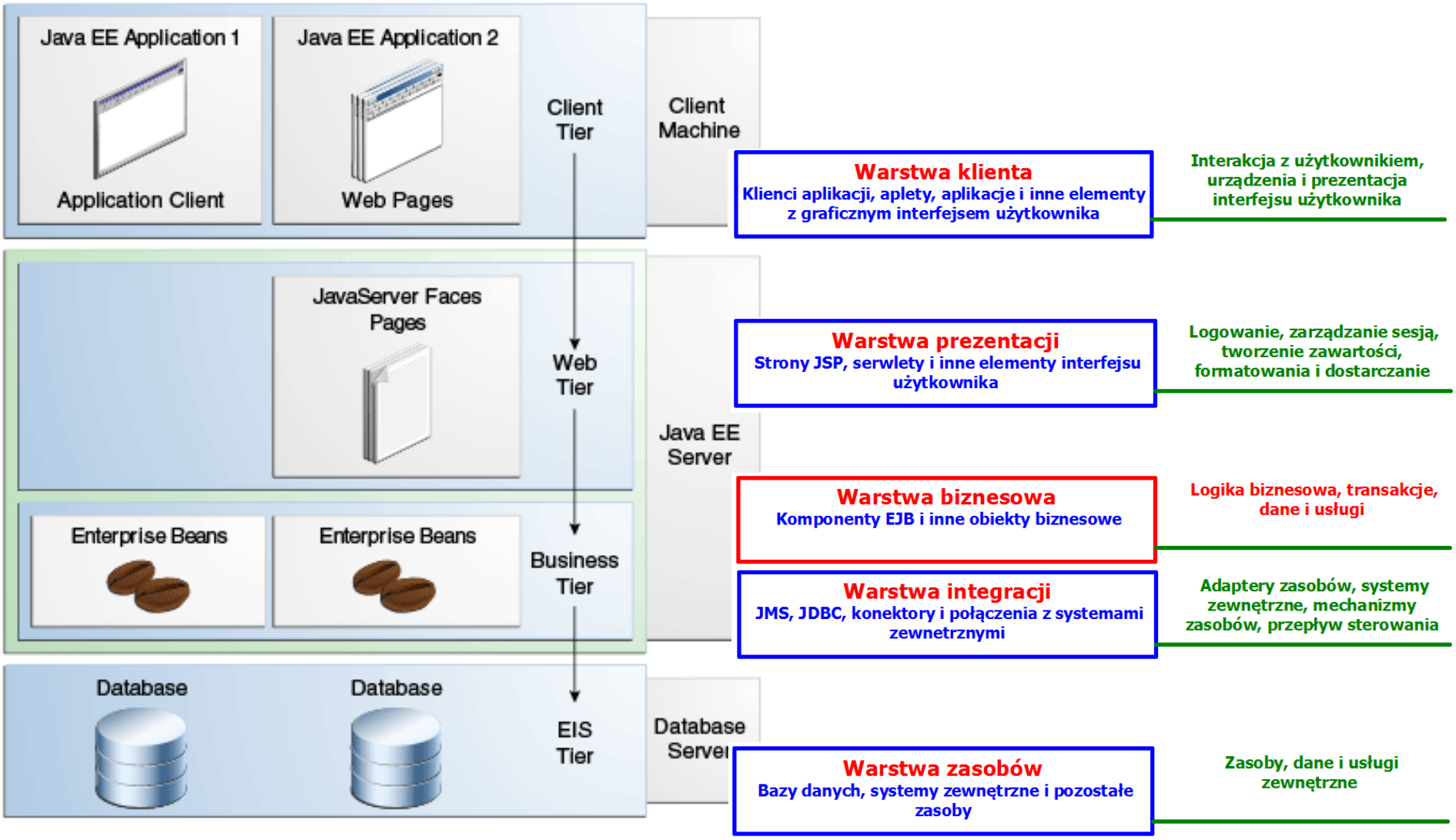
Definicja systemu informatycznego (wykład 1)



Techniczny system informacyjny

- zorganizowany zespół środków technicznych (komputerów, oprogramowania, urządzeń teletransmisyjnych itp.)
- służący do gromadzenia, przetwarzania i przesyłania informacji

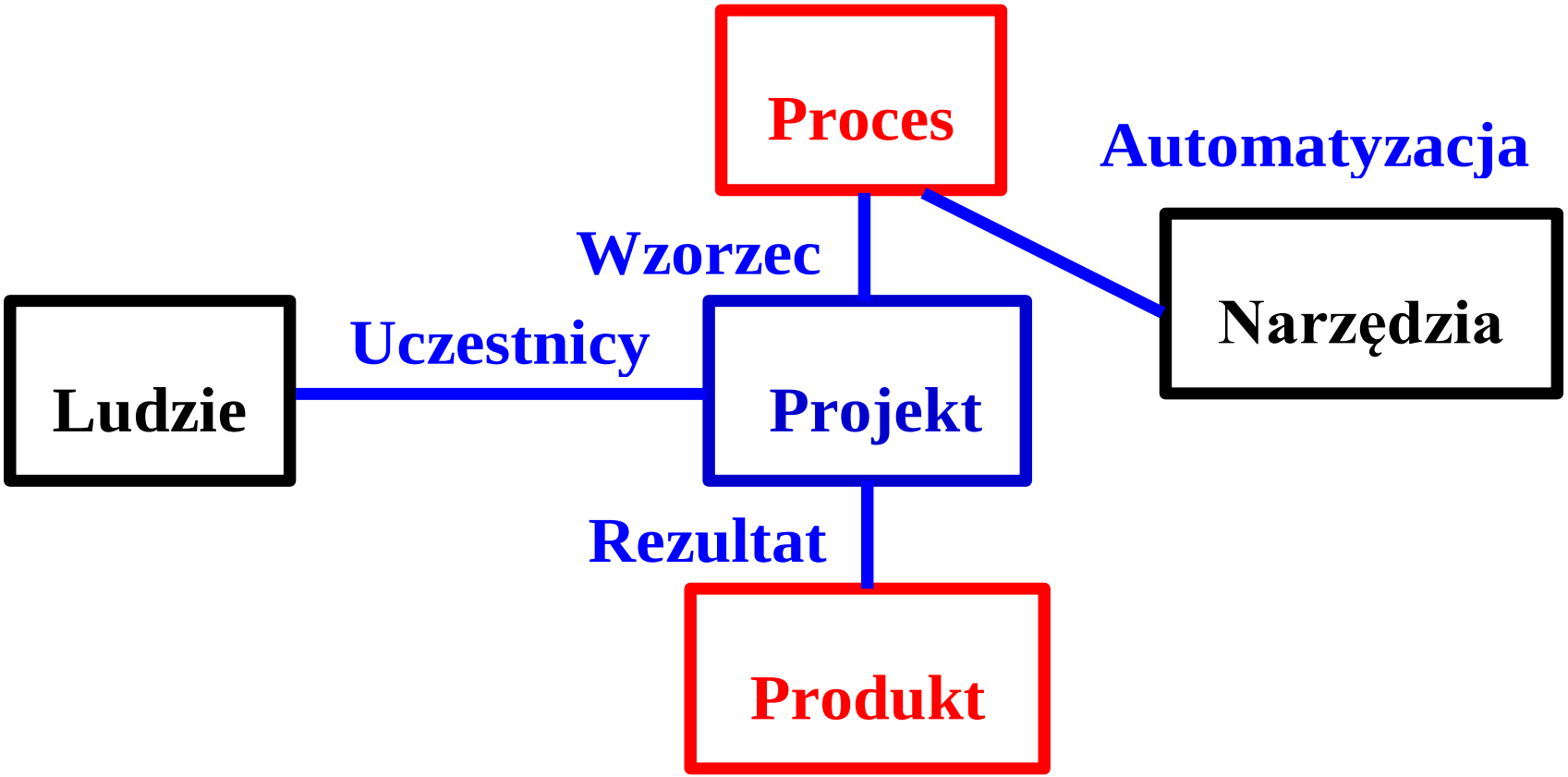
Java EE 7: https://docs.oracle.com/javaee/7/tutorial
Pięciowarstwowy model logicznego rozdzielania zadań [6]



Komponent – produkt do budowy warstw

- skompilowany moduł programowy,
- funkcjonalność dostarczana za pomocą interfejsu,
- zdolny do współdziałania z innymi komponentami oraz innymi częściami systemu informatycznego.

Elementy tworzenia oprogramowania – struktura (wykład 1) – dopasowanie procesu wytwarzania do typu produktu



Zagadnienia

1. **Wielowarstwowa architektura systemu informatycznego**
2. **Ocena i poprawa (refaktoryzacja) architektury wielowarstwowej systemu informatycznego [6]**

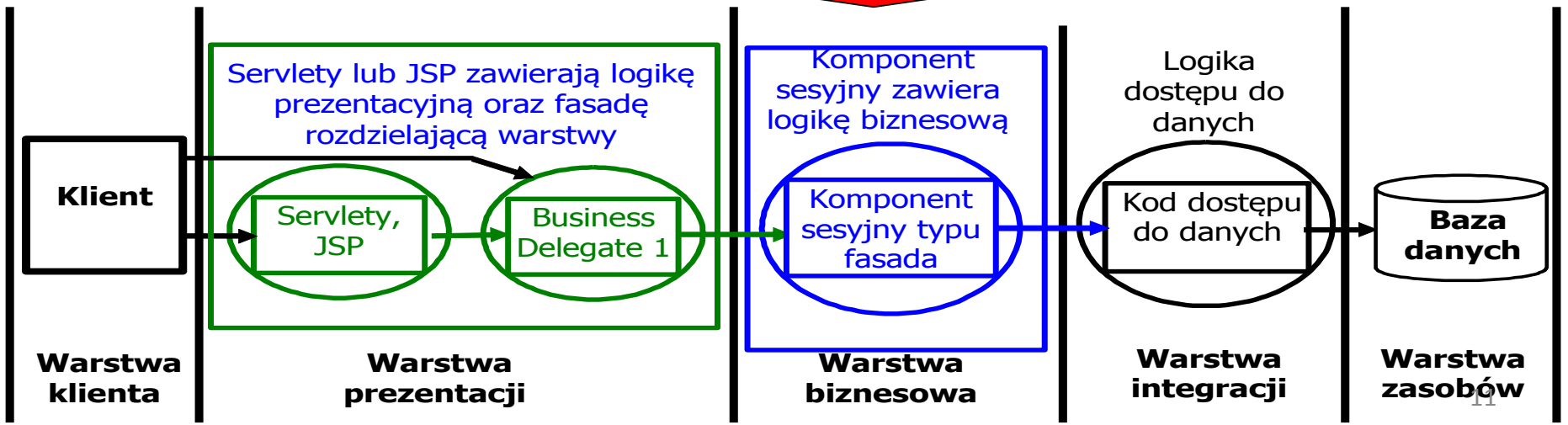
Ocena oprogramowania – metryki [2]

Refaktoryzacja to poprawa struktury oprogramowania bez utraty funkcjonalności – w celu poprawy jego metryk:

- **wydajności**
- **funkcjonalności**
- **kosztu**
- **jakości oprogramowania:**
  - **Testowalności**
  - **Pielęgnowalności**
  - **Wielożywalności**
  - **Zrozumiałości**
  - **Stopnia osiągniętej abstrakcji**

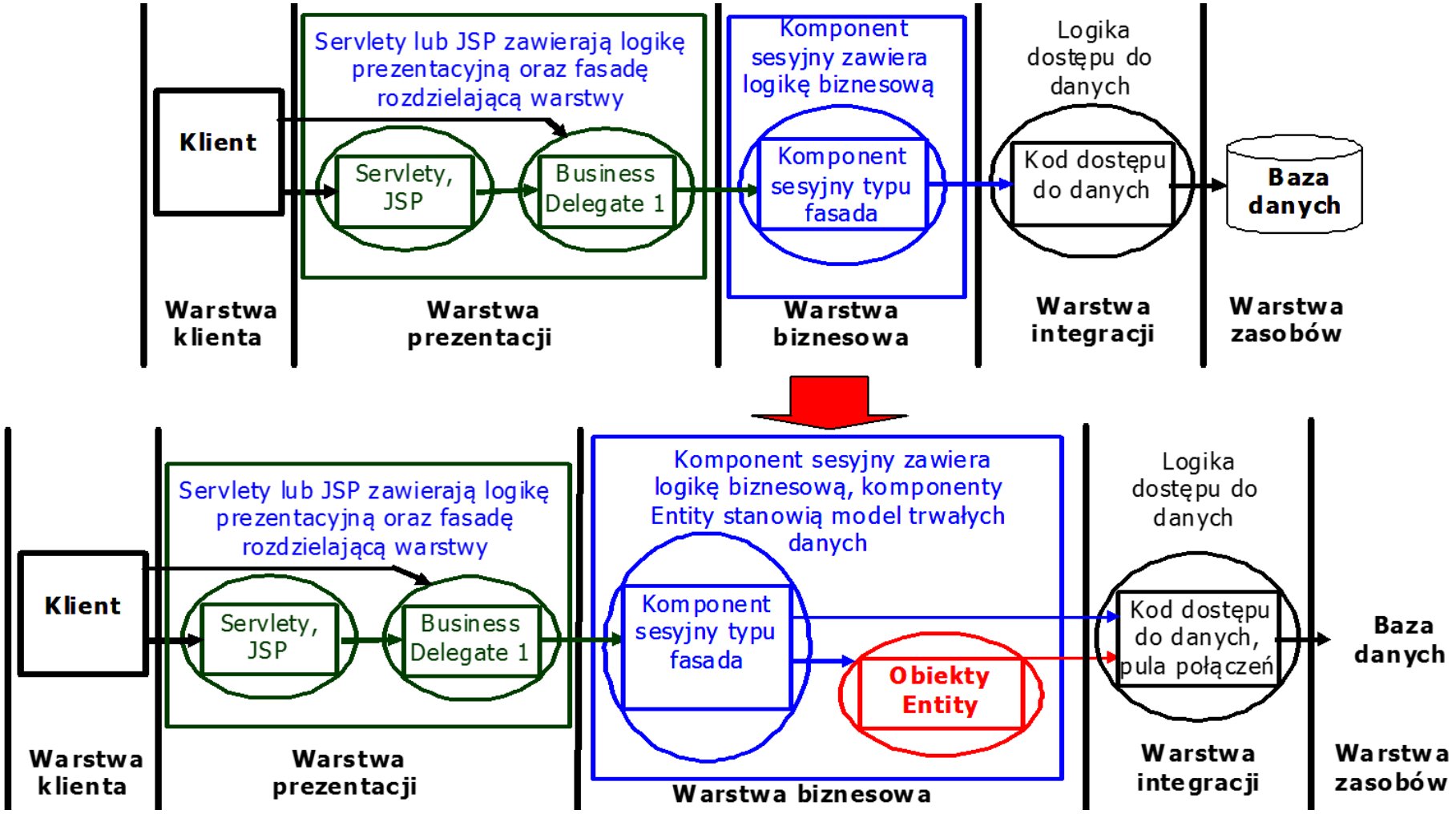
Refaktoryzacja architektury wielowarstwowej - część 1

1. Przeniesienie kodu dostępu do danych logicznie lub fizycznie bliżej rzeczywistego źródła danych ➡ warstwa integracji
2. Przeniesienie kodu logiki przetwarzania z klienta i warstwy prezentacji ➡ warstwy biznesowej zawierającej fasadowe komponenty sesyjne typu „Control”. Komponenty Business Delegate typu „Control” hermetyzują dostęp do warstwy biznesowej z warstwy prezentacji.



Refaktoryzacja architektury wielowarstwowej - część 2

3. Należy zdefiniować obiektowy model danych, który zbudowany jest z obiektów danych typu „Entity”



Zagadnienia

1. **Wielowarstwowa architektura systemu informatycznego**
2. **Ocena i poprawa (refaktoryzacja) architektury wielowarstwowej systemu informatycznego**
3. **Wzorce projektowe stosowane przy budowie wielowarstwowej aplikacji internetowej [6]**

Identyfikacja wzorców projektowych

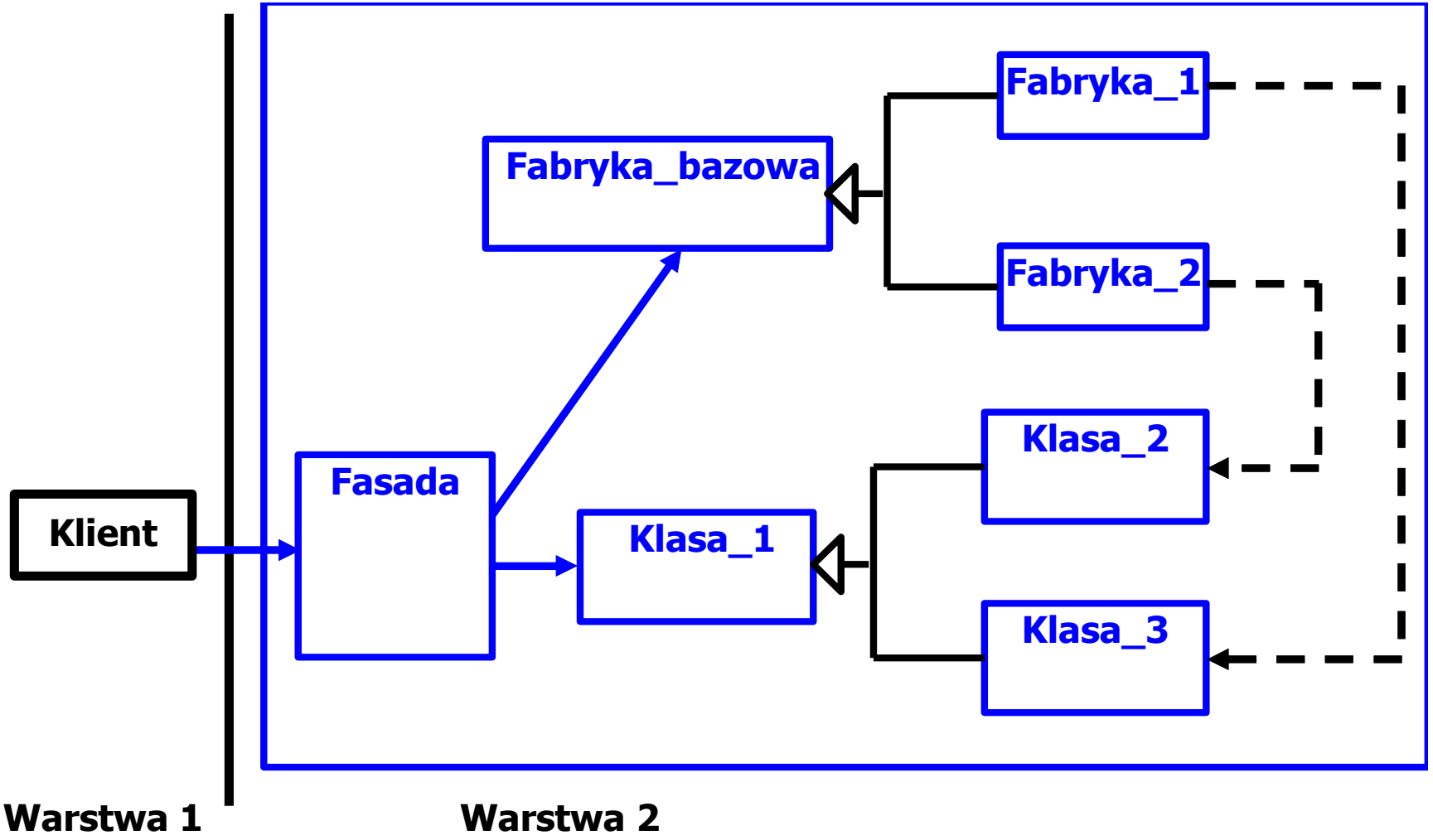
- Dobrze zbudowany system obiektowy jest pełen wzorców obiektowych
- Wzorzec to zwyczajowo przyjęte rozwiązanie typowego problemu w danym kontekście
- Strukturę wzorca przedstawia się w postaci diagramu klas
- Zachowanie się wzorca przedstawia się za pomocą diagramu sekwencji
- **Wzorce projektowe: Wzorzec reprezentuje powiązanie problemu z rozwiązaniem**
  (wg Booch G., Rumbaugh J., Jacobson I., UML przewodnik użytkownika)

- Każdy wzorzec składa się z trzech części, które wyrażają związek między konkretnym kontekstem, problemem i rozwiązaniem **(Christopher Aleksander)**
- Każdy wzorzec to trzyczęściowa reguła, która wyraża związek między konkretnym kontekstem, rozkładem sił powtarzającym się w tym kontekście i konfiguracją oprogramowania pozwalająca na wzajemne zrównoważenie się tych sił w celu rozwiązania zadania. **(Richar Gabriel)**
- **Wzorzec to pomysł, który okazał się użyteczny w jednym rzeczywistym kontekście i prawdopodobnie będzie użyteczny w innym.** **(Martin Fowler)**

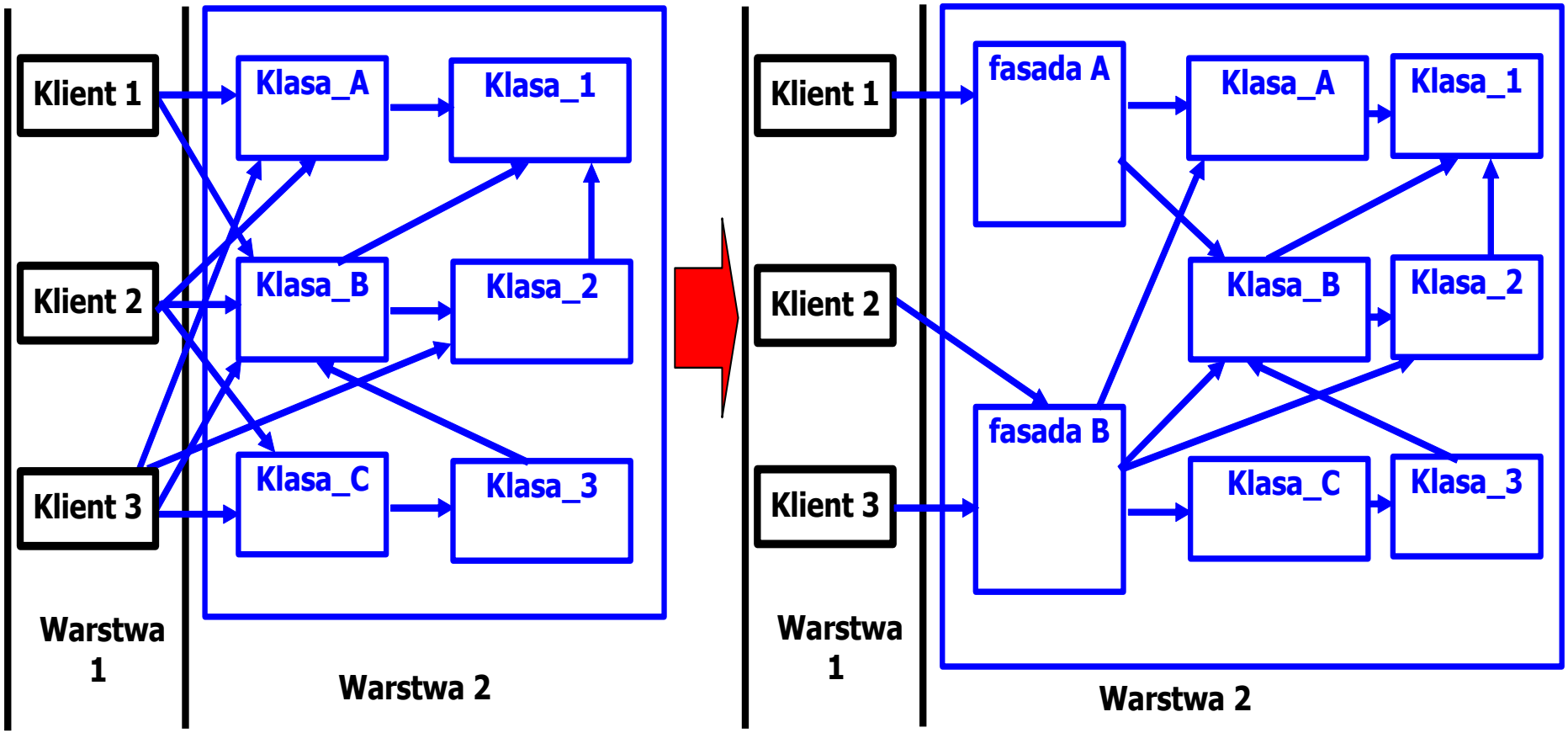
3.1. Wzorzec uniwersalny kreacyjny stosowany w każdej z warstw:

Fabryka obiektów –

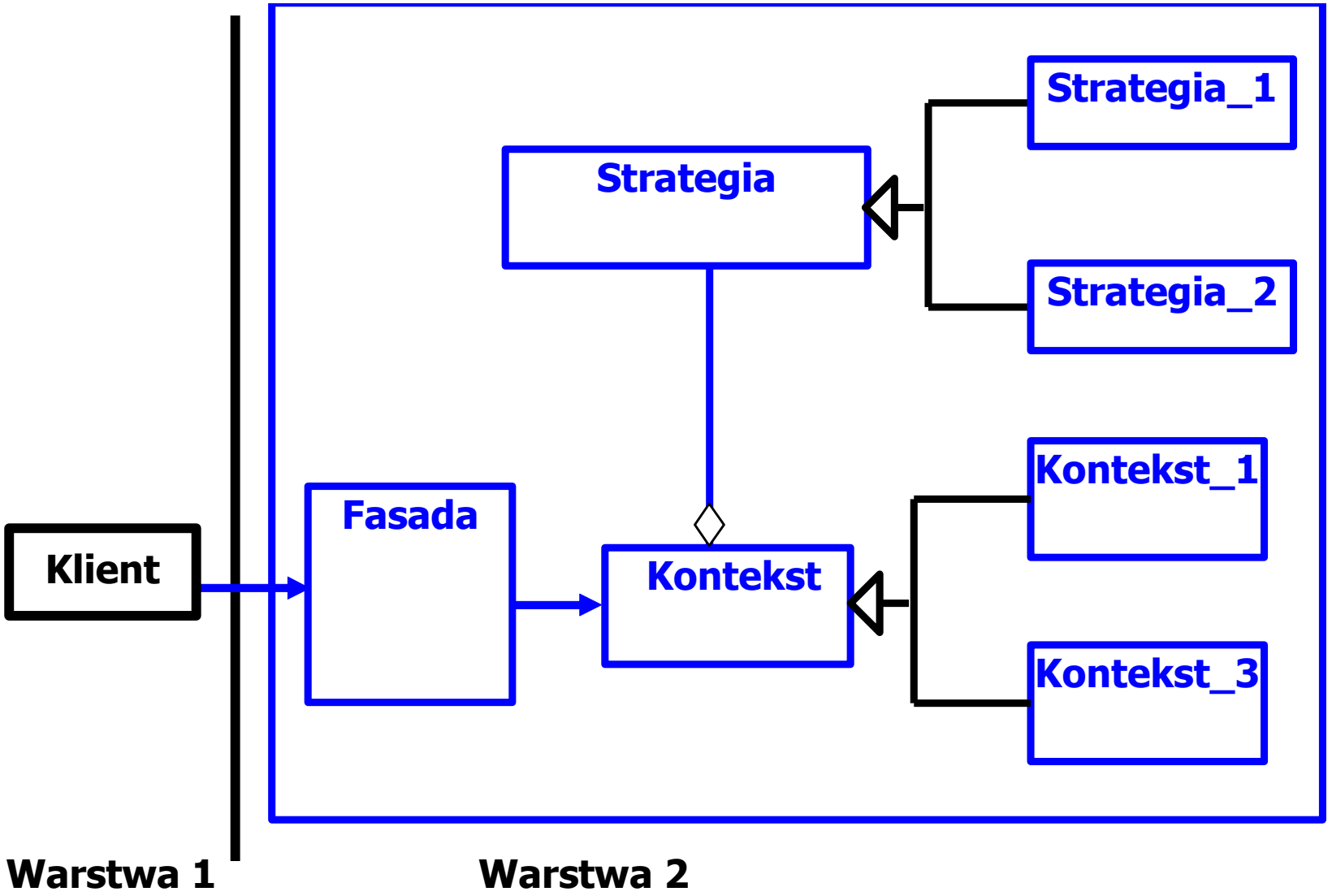
oddzielenie tworzenia obiektów od zarządzania nimi i używania ich



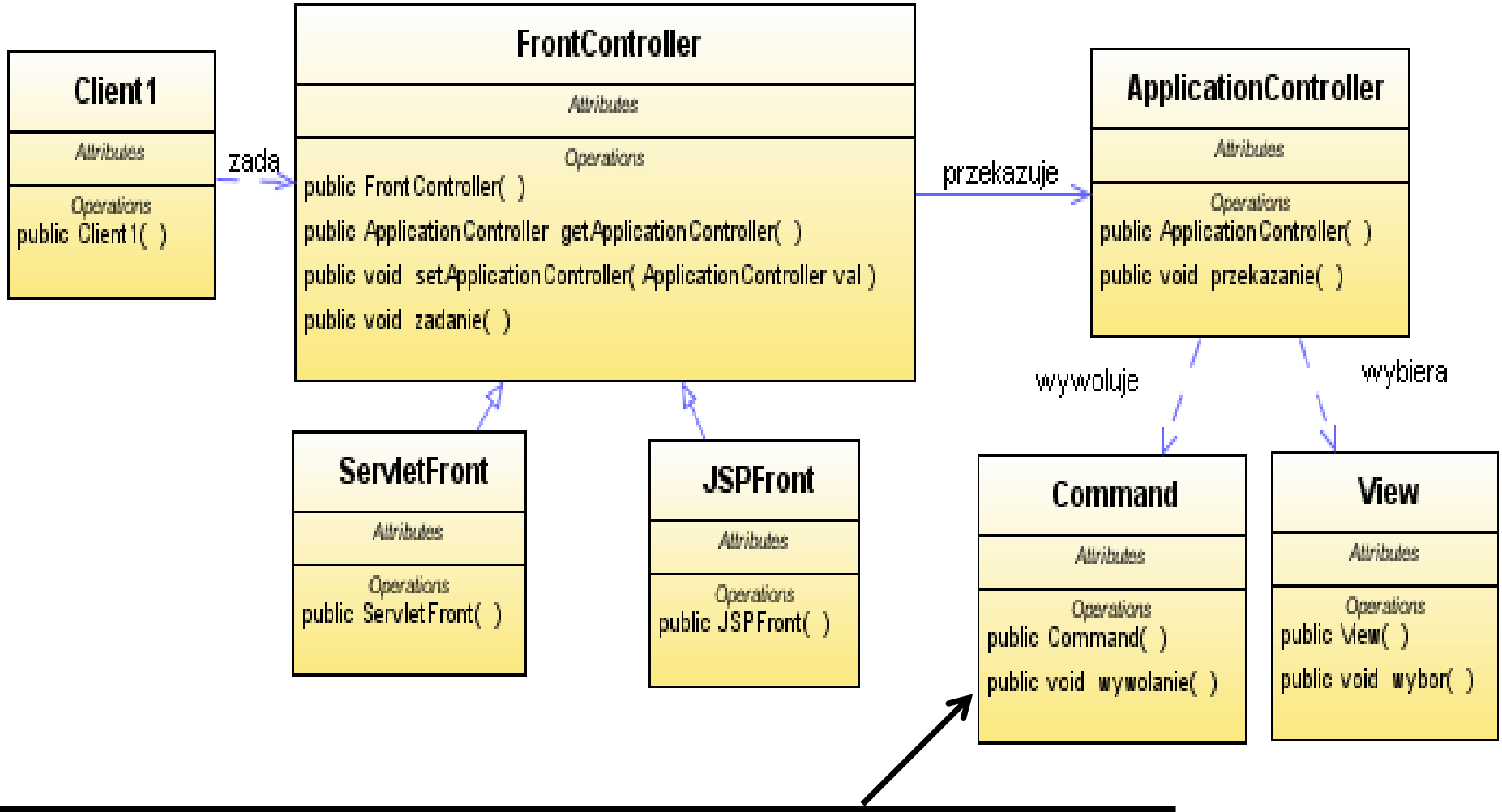
3.2. Wzorzec uniwersalny strukturalny: Fasada – hermetyzacja logiki biznesowej



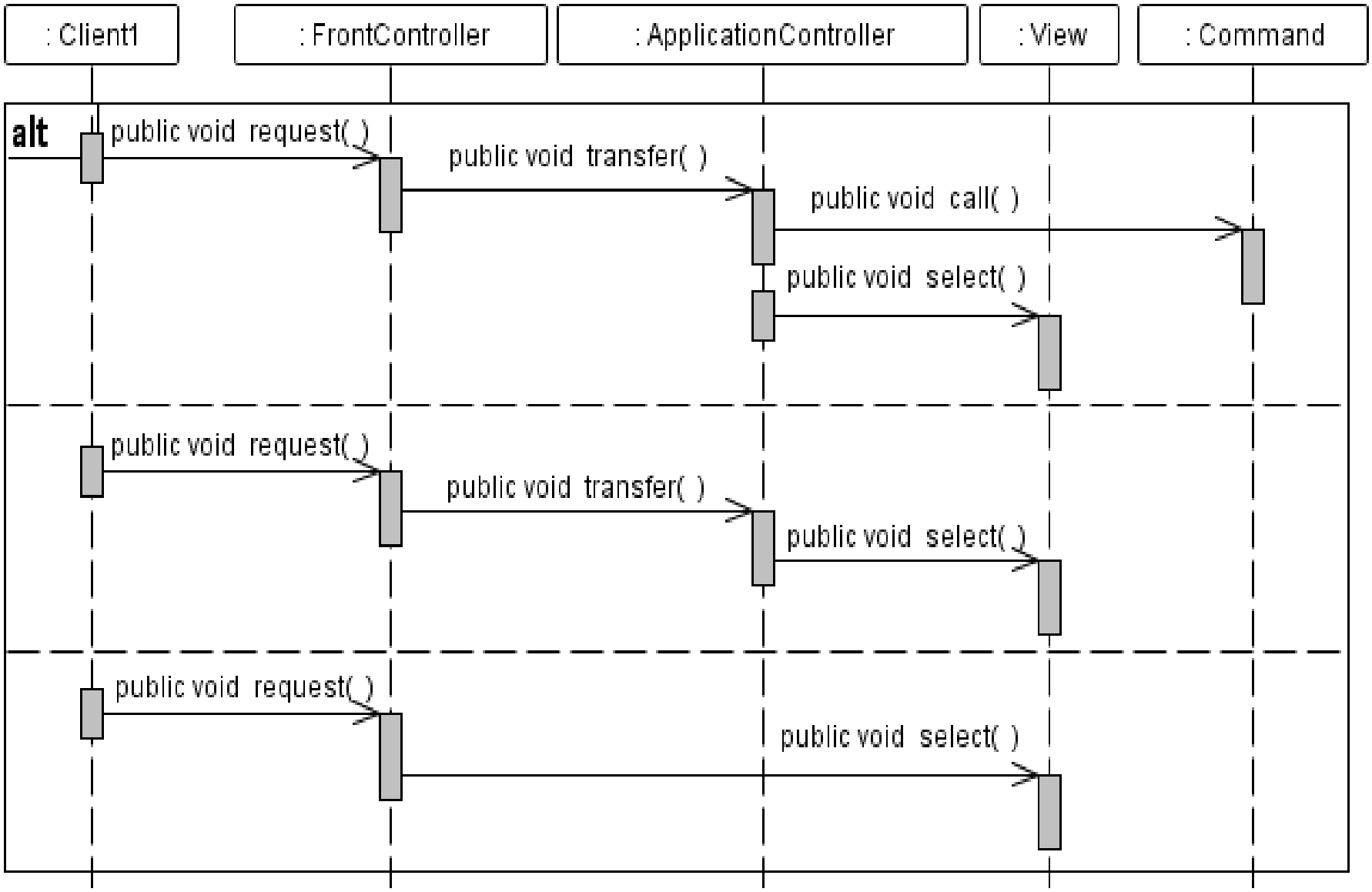
3.3. Wzorzec uniwersalny czynnościowy kreacyjny stosowany w każdej z warstw: : Strategia – zastosowanie polimorfizmu do wyboru algorytmu



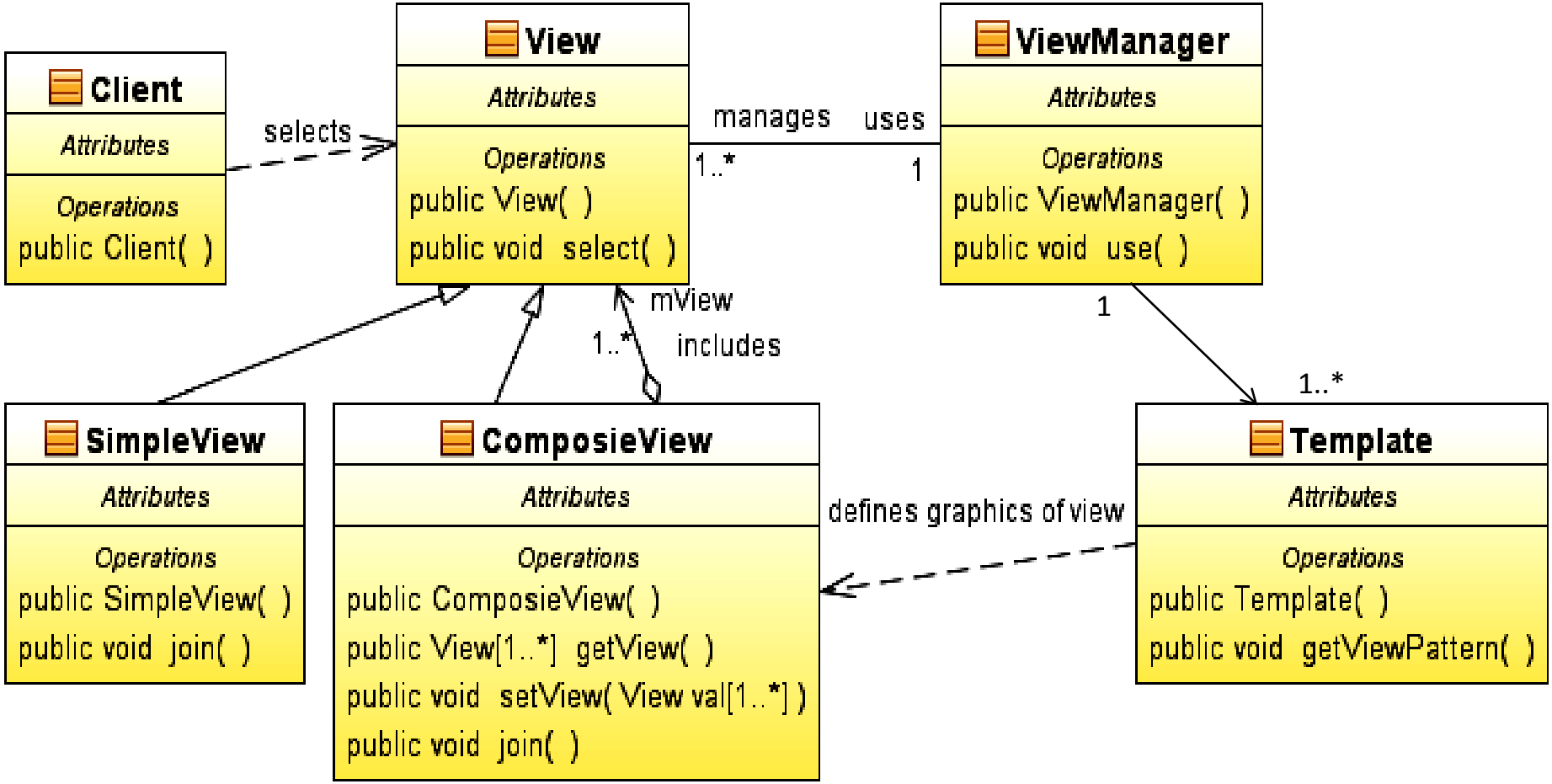
3.4. Wzorzec EE warstwy internetowej: *FrontController* – scentralizowany punkt dostępowy do obsługi żądań w warstwie internetowej



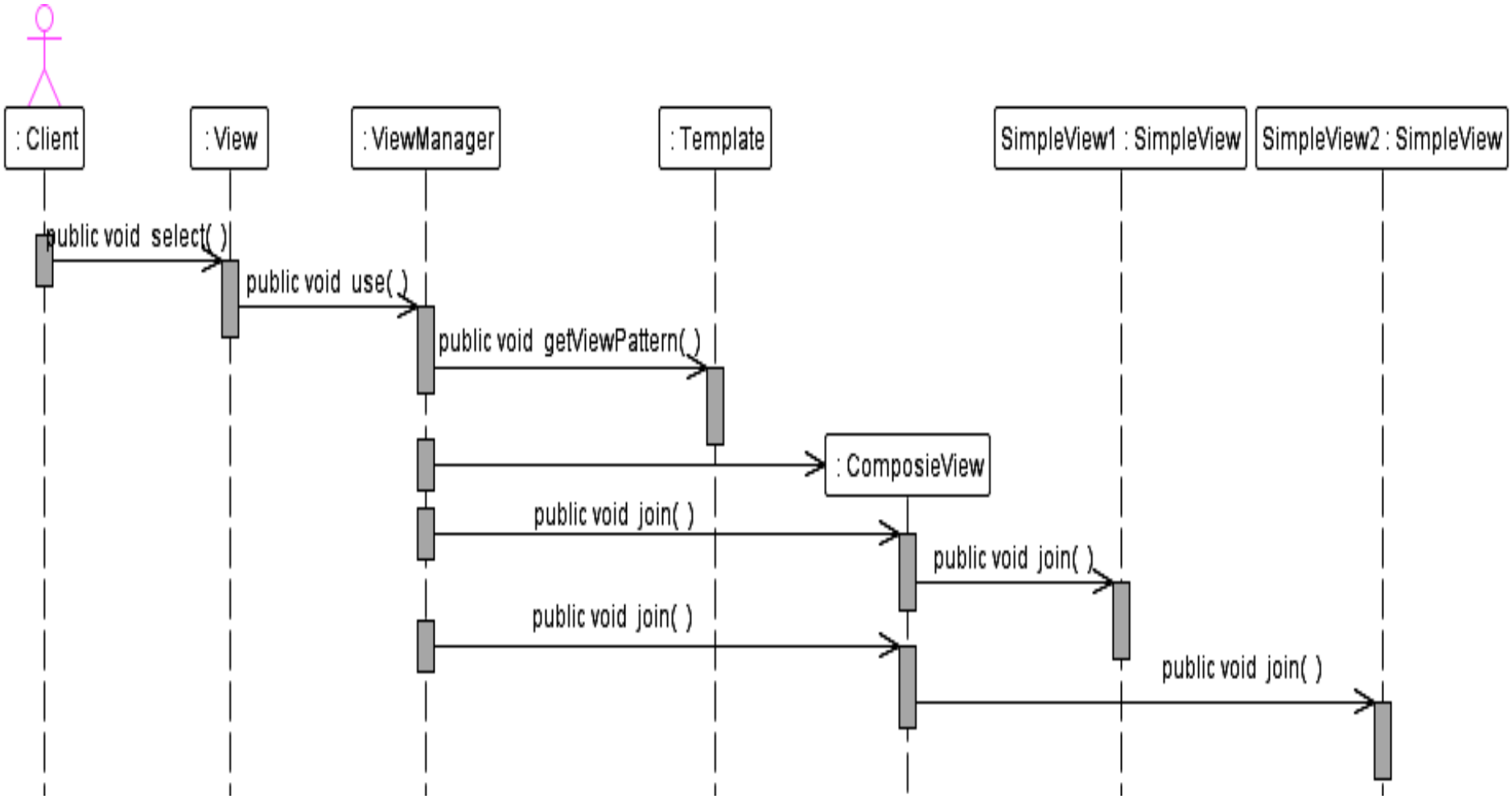
Wywołanie metod z warstwy biznesowej np. za pośrednictwem wzorca ApplicationService


: Client1
: FrontController
: ApplicationController
: View
: Command
alt
public void request( )
public void transfer( )
public void call( )
public void select( )
public void request( )
public void transfer( )
public void select( )
public void request( )
public void select( )

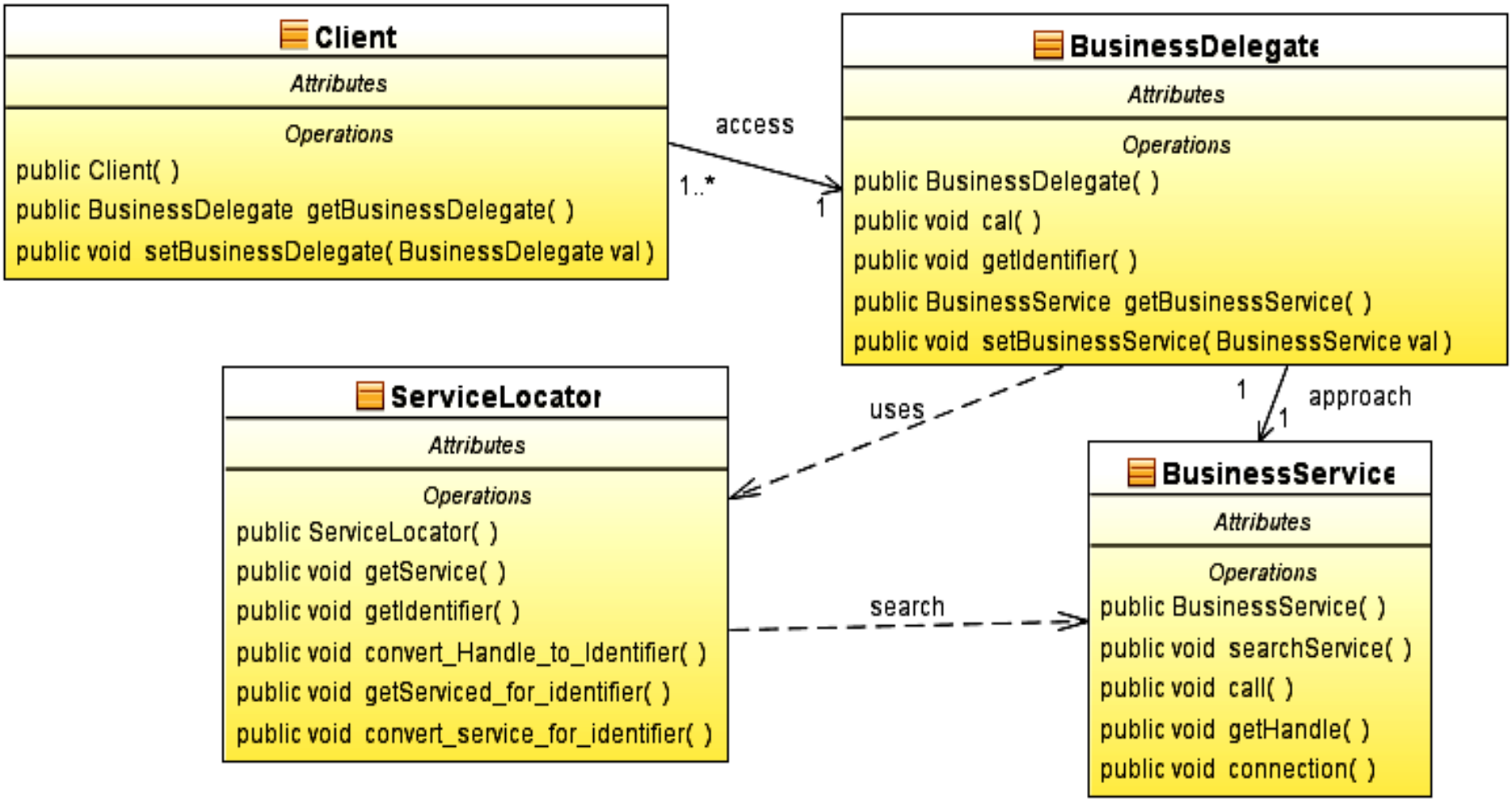
3.5. Wzorzec EE warstwy internetowej: *Composite View* - widok kompozytowy powinien mieć strukturę modułową, zbudowaną z komponentów prostych, które razem tworzą złożoną stronę są zarządzane niezależnie.



Tworzenie złożonego widoku wykonanego z wieloużywalnych komponentów, oparty na szablonie widoku

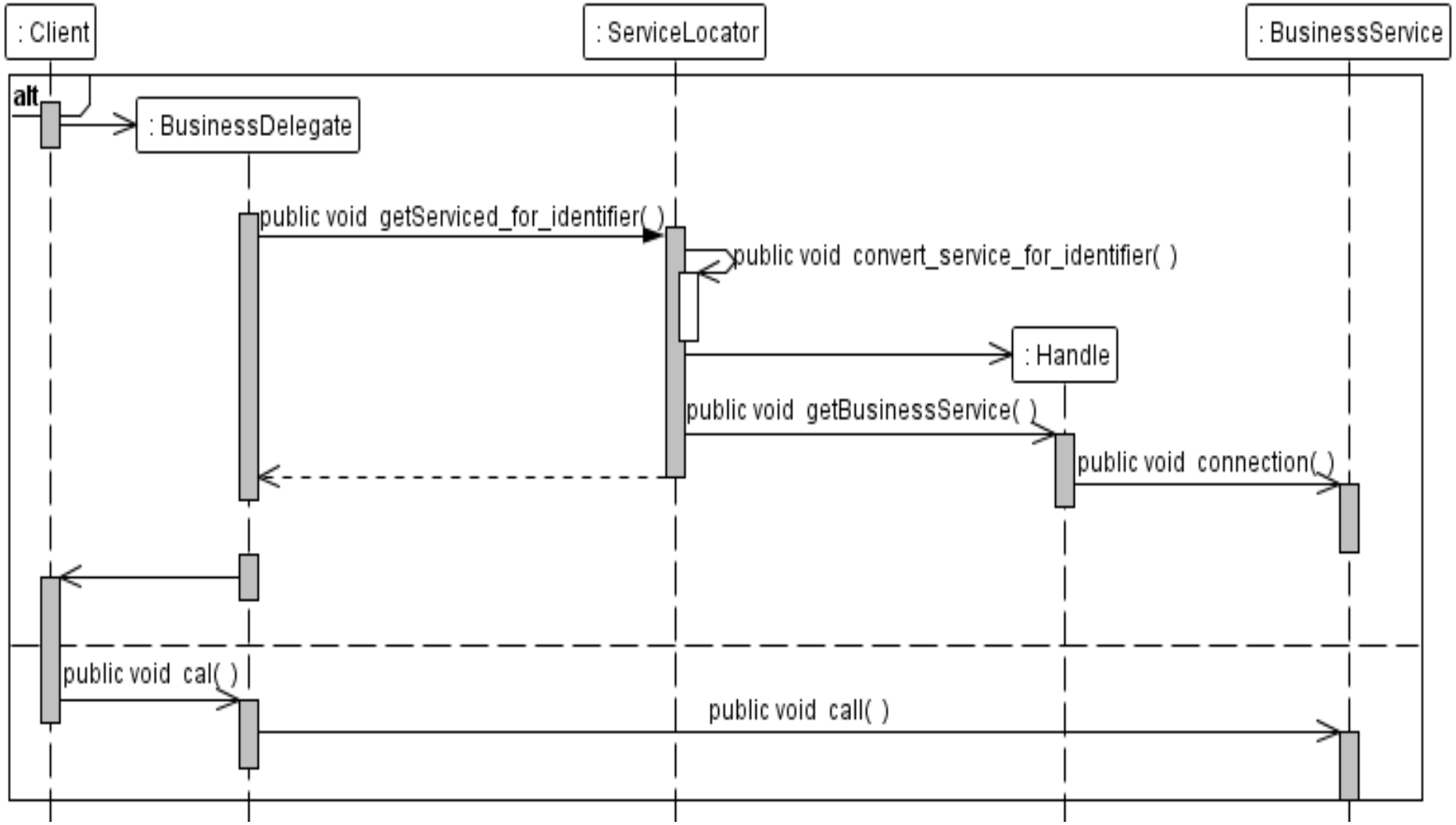


3.6. Wzorzec EE warstwy klienta: do zdalnego wywołania usług z warstwy klienta w celu ukrycia złożoności zdalnej komunikacji z komponentem usług biznesowych - *BusinessDelegate*

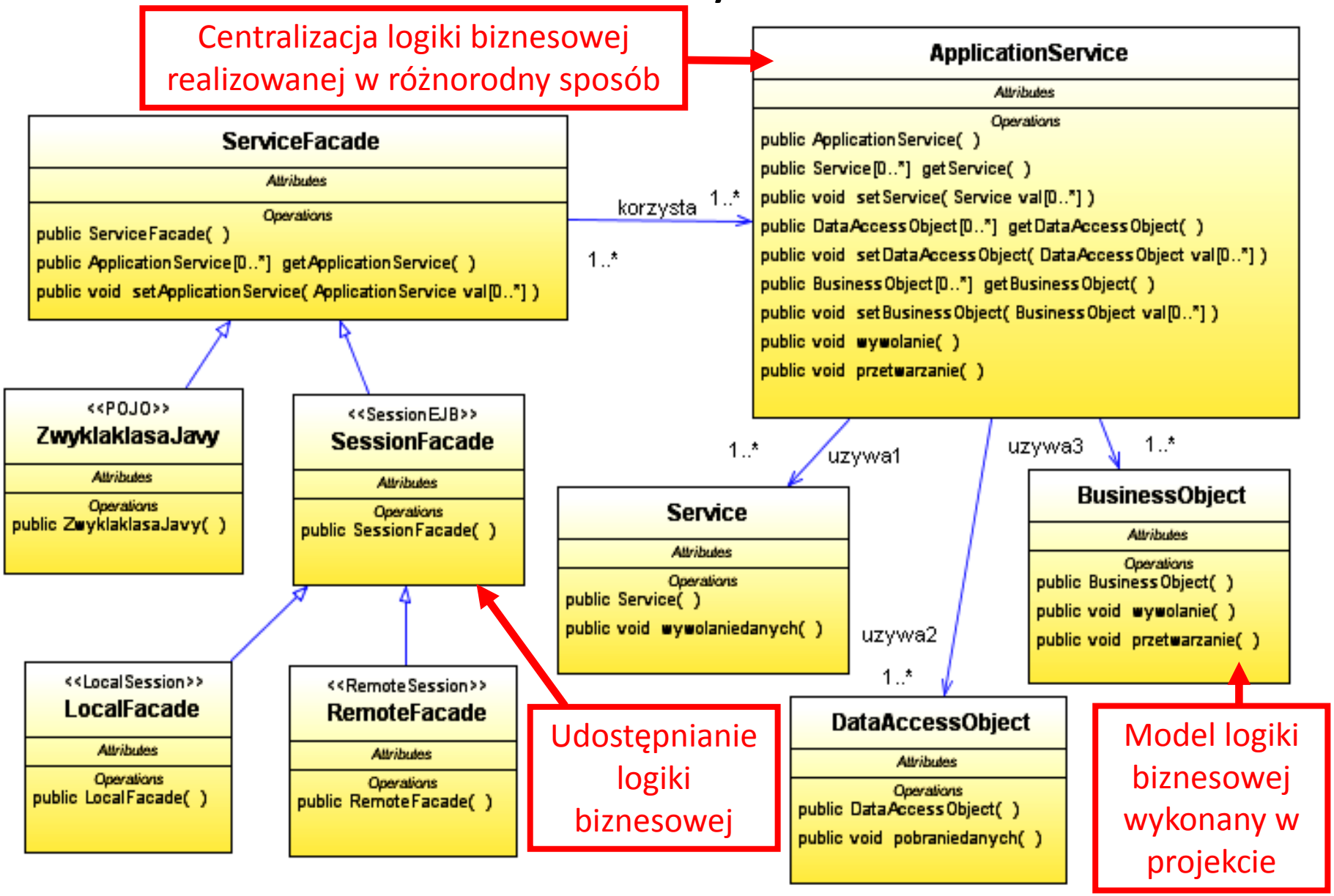


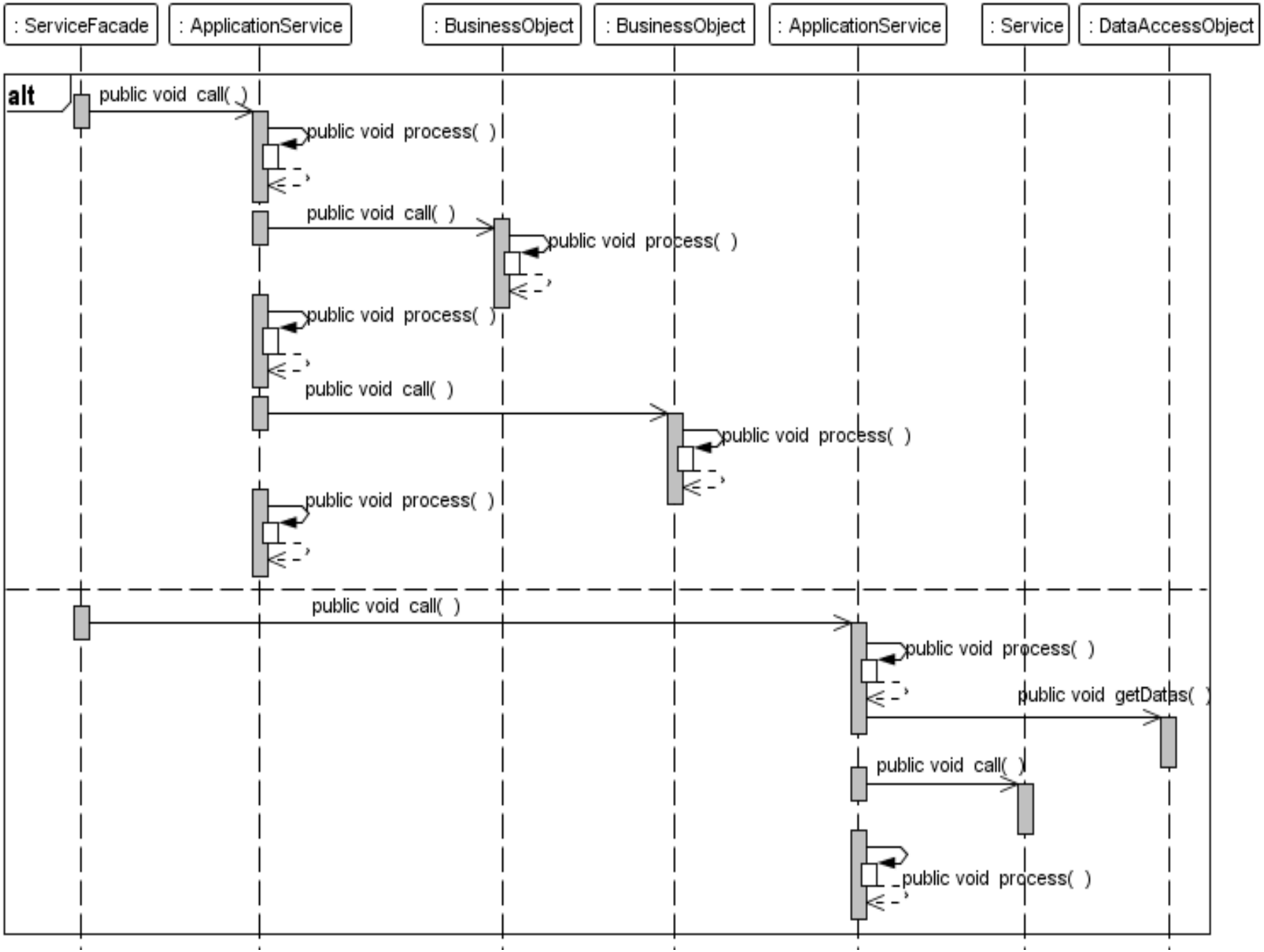
Komponenty EJB, *JMS* , lub część wzorca *SessionFacade*

Wywołanie usługi biznesowej

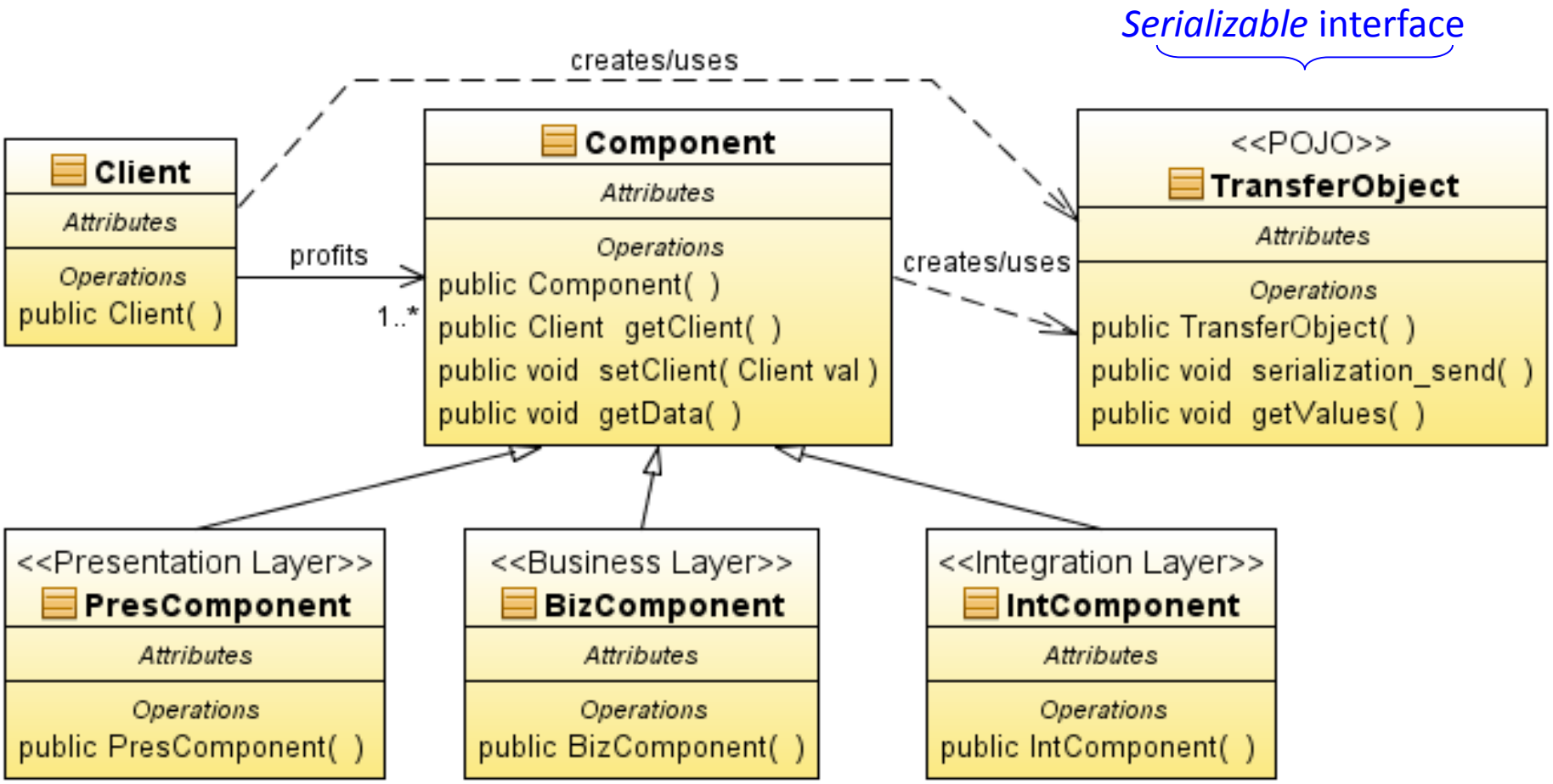


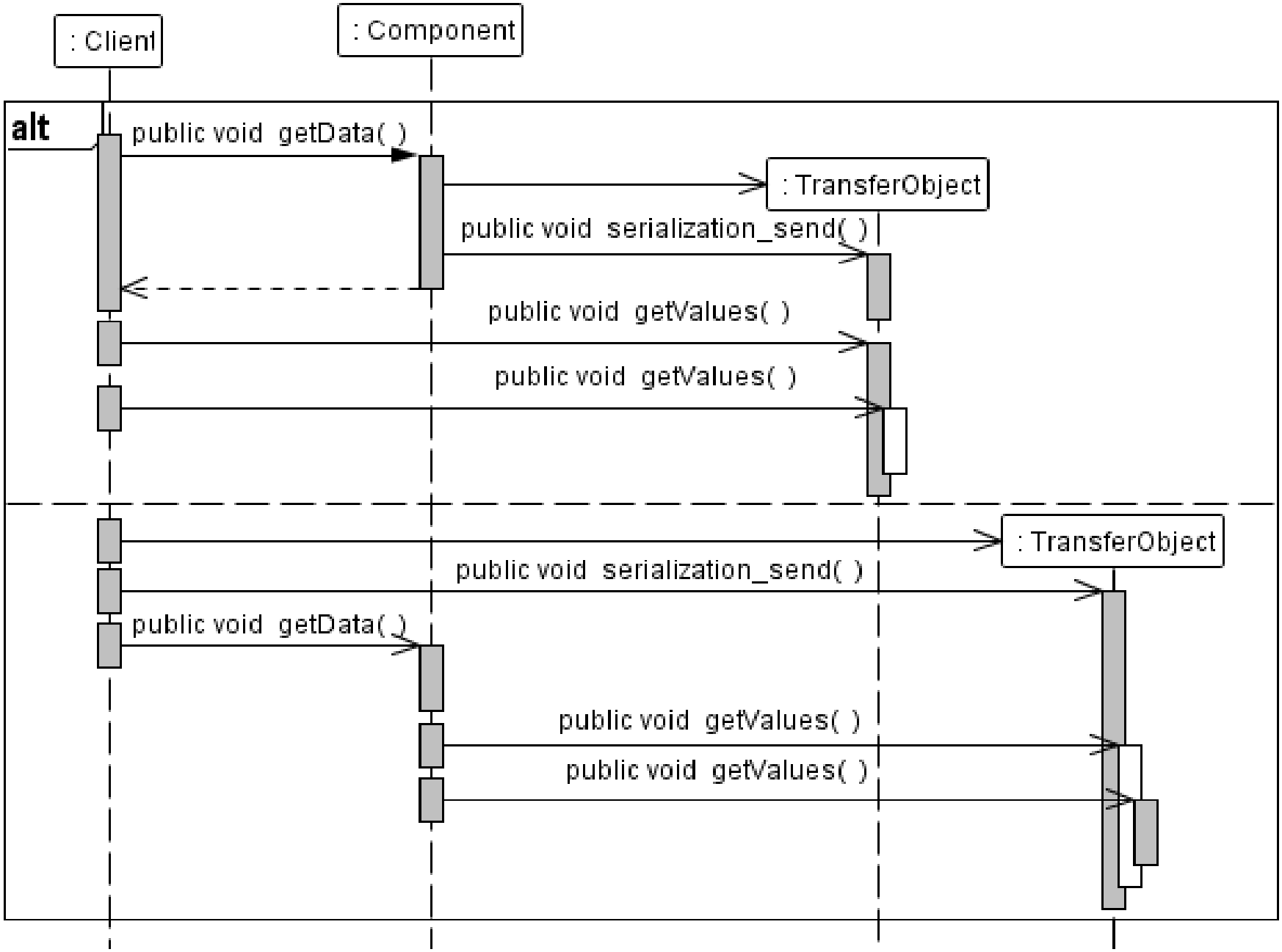
3.7. Wzorce EE warstwy biznesowej: *SessionFacade, ApplicationService* – udostępnianie i centralizacja logiki biznesowej kilku komponentów i usług biznesowych



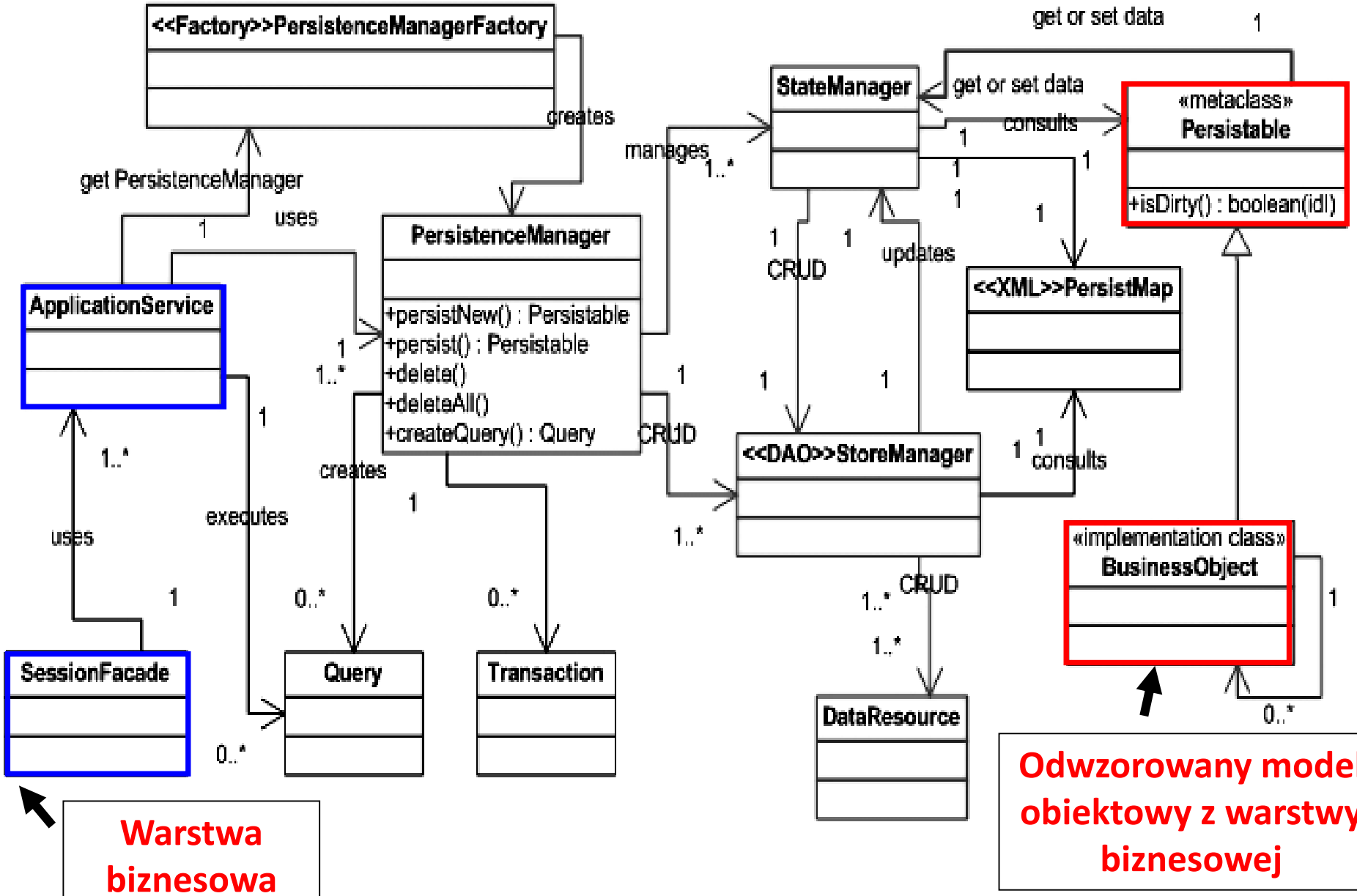

: ServiceFacade
: ApplicationService
: BusinessObject
: BusinessObject
: ApplicationService
: Service
: DataAccessObject
alt
public void call( )
public void process( )
public void call( )
public void process( )
public void process( )
public void call( )
public void process( )
public void process( )
public void call( )
public void process( )
public void getDatas( )
public void call( )
public void process( )

3.8. Wzorce EE warstwy biznesowej: *TransferObject* - przesyłanie danych między warstwami aplikacji (zmniejszanie ruchu w sieci poprzez zmniejszanie liczby połączeń zdalnych lub zwiększanie wydajności)

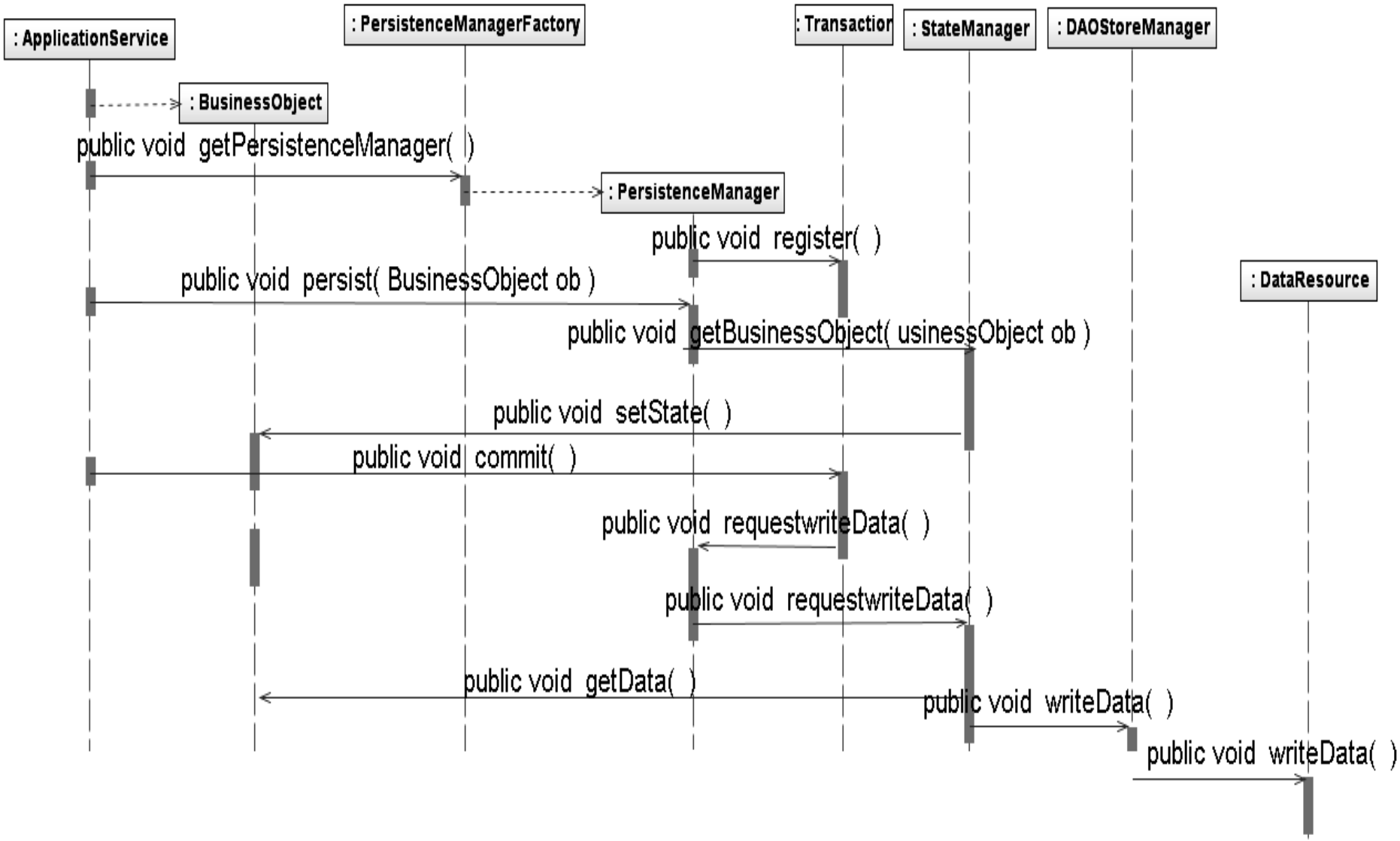



: Client
: Component
alt
public void getData( )
: TransferObject
public void serialization_send( )
public void getValues( )
public void getValues( )
: TransferObject
public void serialization_send( )
public void getData( )
public void getValues( )
public void getValues( )

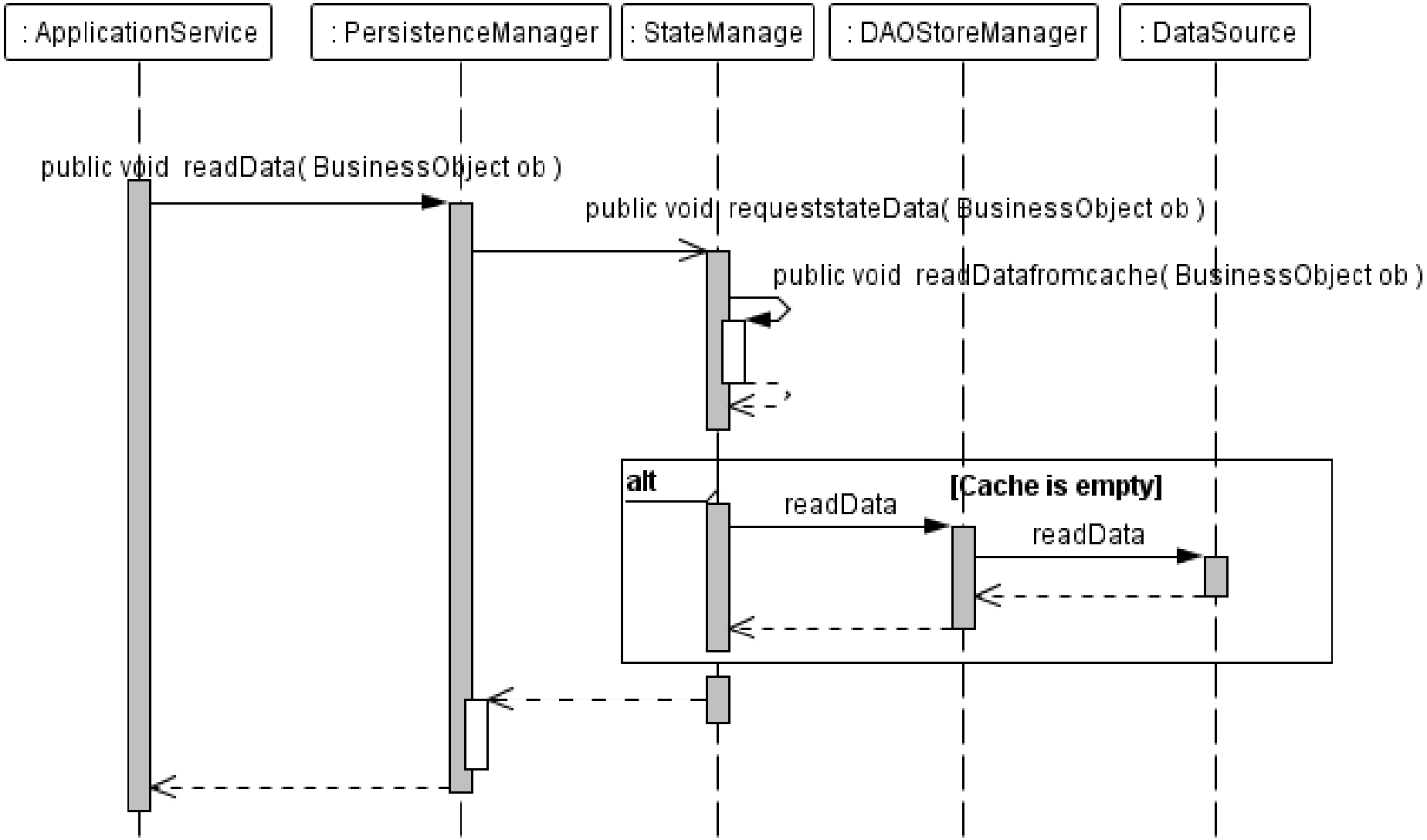
3.9. Wzorzec EE warstwy integracji: DomainStore (ORM) – oddzielenie mechanizmów trwałości od modelu obiektowego



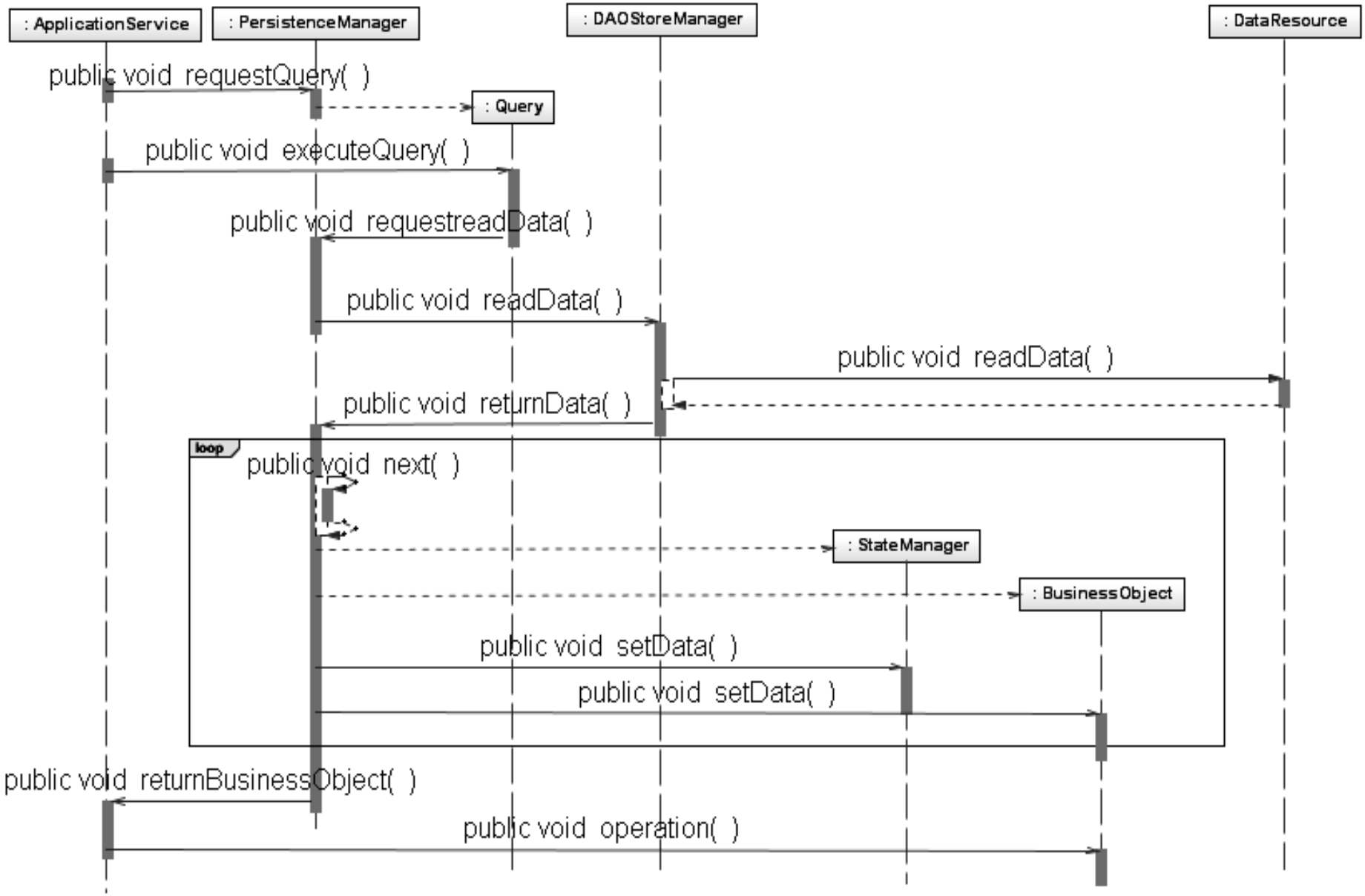
Tworzenie i utrwalanie obiektów (*Business Object*)



Zapełnianie buforów



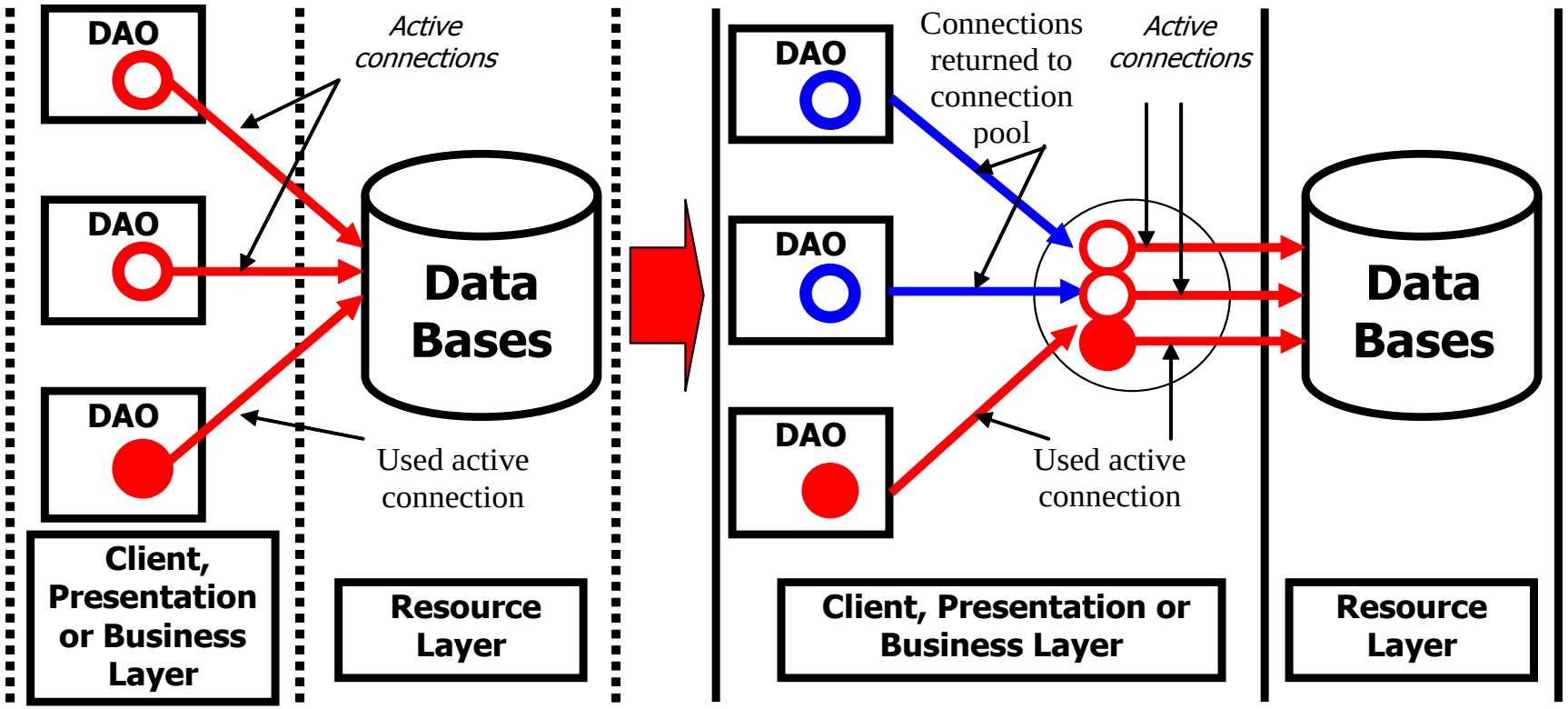
Tworzenie i wykonanie zapytanie (*Query*)



Zarządzanie połączeniami do bazy danych – pula połączeń

Połączenia z bazą danych nie są udostępniane, co zmniejsza wydajność i skalowalność.

Pula wspólnych połączeń poprawia wydajność i skalowalność aplikacji



Wyniki eksperymentów dostępu do baz danych z wykorzystaniem wzorców DAO (JDBC), DAO (JDBC) i puli połączeń oraz wzorców projektowych *Domains Store*

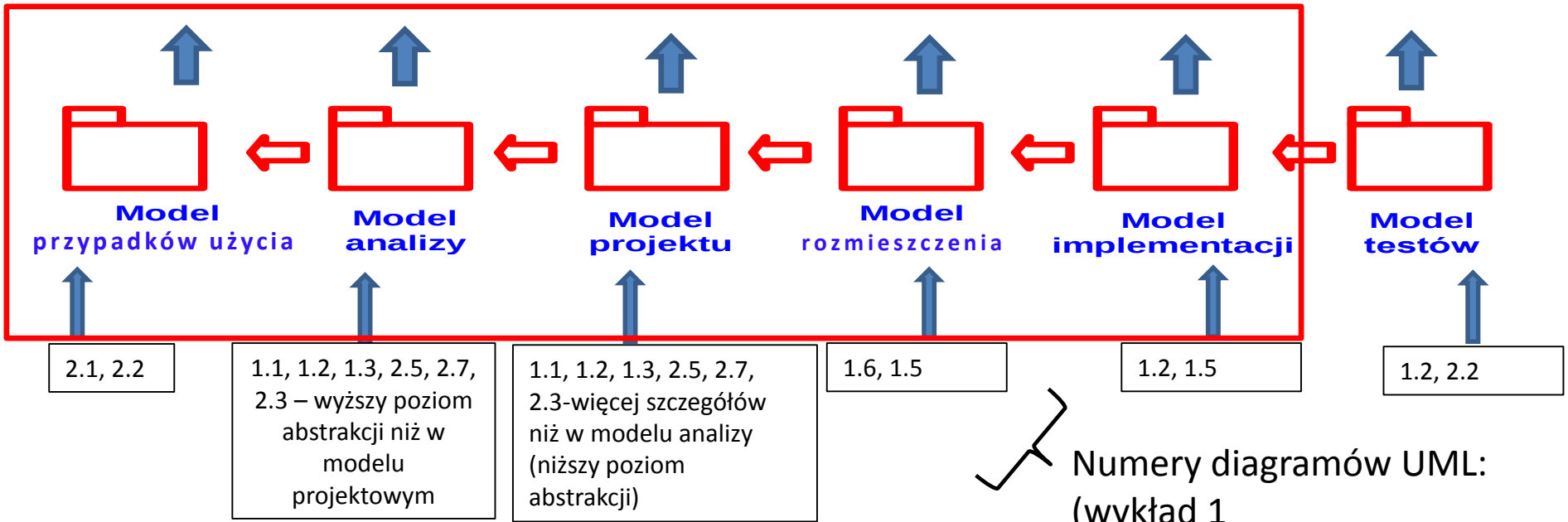
Requests no.	Jdbc[ms] (DAO)	Jndi[ms] (DAO and pool of connections)	Orm[ms] (Domain Store)
			lazy
20	8 431	5 136	2 627
100	49 356	19 311	2 445

Zagadnienia

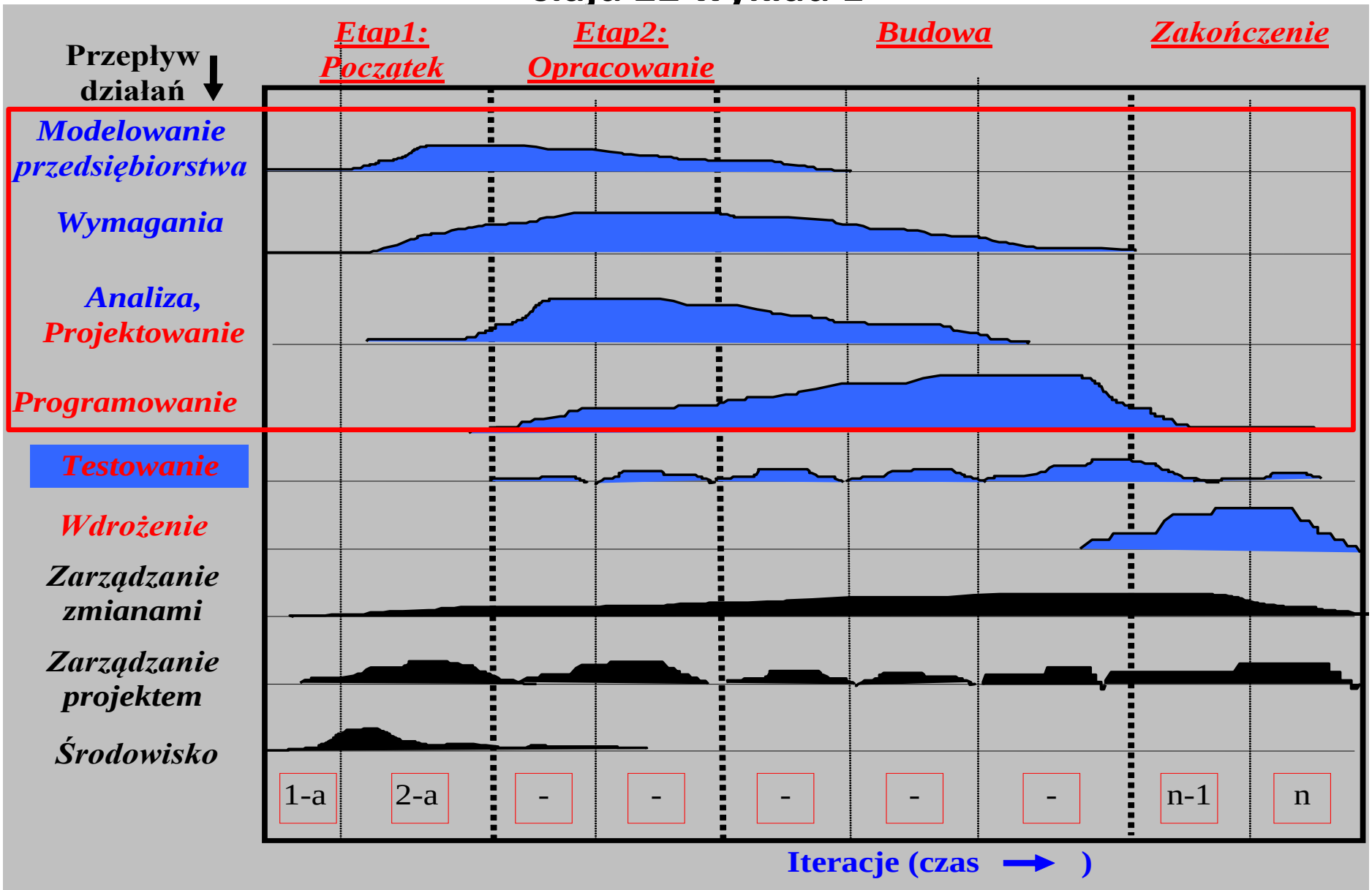
1. Wielowarstwowa architektura systemu informatycznego
2. Ocena i poprawa (refaktoryzacja) architektury wielowarstwowej systemu informatycznego
3. Wzorce projektowe stosowane przy budowie wielowarstwowej aplikacji internetowej
4. Przykład modelowania i projektowania części warstwy biznesowej z obiektami typu POJO. Wykonanie aplikacji dwuwarstwowej dla jednego użytkownika.

Produkt - diagramy UML – modele, proces (wykład 1)

Modelowanie struktury i dynamiki systemu	Implementacja systemu,	struktury i dynamiki generowanie kodu
Perspektywa koncepcji *co należy wykonać?*	Perspektywa specyfikacji *jak należy używać?*	Perspektywa implementacji *jak należy wykonać?*
• model problemu np. przedsiębiorstwa • wymagania • analiza (model konceptualny: diagram przypadków użycia, diagram klas, diagramy sekwencji,) • testy modelu	• projektowanie (model projektowy: architektura sprzętu i oprogramowania; dostęp użytkownika; przechowywanie danych) • testy projektu	• programowanie, wdrażanie (specyfikacja programu : deklaracje, definicje; dodatkowe struktury danych: struktury „pojemnikowe”, pliki, bazy danych) • testy oprogramowania • wdrażanie • testy wdrażania



Proces - zunifikowany iteracyjno- przyrostowy proces tworzenia oprogramowania – kiedy należy wykonać? [3LU]
- slajd 22 wyklad 1



System informacyjny „Biblioteka”

I. Opis biznesowy „świata rzeczywistego”
II. Sformułowanie wymagań funkcjonalnych i niefunkcjonalnych aplikacji
III. Model analizy aplikacji oparty na diagramie przypadków użycia
IV. Model projektowy i implementacja warstwy biznesowej warstwy biznesowej oparty na diagramie klas i diagramie sekwencji tworzony metodą **iteracyjno-rozwojową,** sterowany realizacją przypadków użycia

I. Opis biznesowy „świata rzeczywistego” w języku klienta

1. Opis zasobów ludzkich

Co robią pracownicy?

2. Przepisy i strategia firmy

Co ogranicza działalność firmy?

3. Dane techniczne

Dane ilościowe:
ilu pracowników,
ile danych,
jak często,
Dane o lokalizacji firmy
Dane o klientach firmy
Dane o używanym sprzęcie i oprogramowaniu

Opis biznesowy „świata rzeczywistego” biblioteki.

1. Opis zasobów ludzkich

Bibliotekarz może dodawać do katalogu tytułów nowe tytuły. Każdy tytuł jest reprezentowany przez dane: tytuł, autor, wydawnictwo, ISBN oraz informacje o liczbie egzemplarzy i miejscu ich przechowywania i występuje w bibliotece jako pojedyncza informacja dla każdego tytułu. Pewna grupa tytułów opisuje książki nagrane na wybranym nosniku, dlatego dodatkowo tytuł zawiera dane nagrania np nazwisko aktora. Każdy egzemplarz, niezależnie, czy jest książką czy kasetą, jest opisany odrębną informacją zawierajacą numer egzemplarza, który może się powtarzać dla różnych tytułów. Bibliotekarz może dodawać nowe tytuły i egzemplarze oraz je przeszukiwać, natomiast klient może jedynie przeszukiwać tytuły i sprawdzać egzemplarze wybranych tytułów.

W celu wypozyczenia książki klient musi ją **zarezerwować** podając dane rejestracji, dane książki oraz datę rezerwacji. Klient musi **wypożyczyć** zarezerwowaną książkę w terminie jej rezerwacji podając dane rejestracji i rezerwacji, wyszukać rezerwację i następnie ją usunąć. Musi wykonać dane wypozyczenia zawierające: dane rejestracji, dane zarezerwowanej książki oraz datę zwrotu. Rezerwacje można usunąć bez konieczności jej wypożyczenia – w terminie rezerwacji.

W celu **zwrotu książki** należy podać dane rejestracji oraz dane wypożyczonej książki. Zwrot musi nastąpić w okresie wypożyczenia podanym w danych wypożyczenia.

Opis biznesowy „świata rzeczywistego” biblioteki (cd)

2. **Przepisy**

   Pracownik ponosi odpowiedzialność za poprawność danych - odpowiada materialnie za niezgodność danych ze stanem wypożyczalni.

3. **Dane techniczne**

   Klient może przeglądać dane wypożyczalni za pośrednictwem strony internetowej lub bezpośrednio za pomocą specjalnego programu. Zakłada się, że klientów jednocześnie przeglądających dane wypożyczalni może być ponad 1000 oraz wypożyczalnia może zawierać kilkadziesiąt tysięcy tytułów oraz przynajmniej dwukrotnie więcej egzemplarzy. Biblioteka składa się z kilku ośrodków w różnych miastach na terenie kraju (lista miast jest dołączona do umowy). Zaleca się stosowanie technologii Java.

II. Sformułowanie wymagań funkcjonalnych i niefunkcjonalnych

Lista wymagań funkcjonalnych

1. System zawiera katalog tytułów
2. System zawiera dwa typy egzemplarzy do wypożyczenia: książki i kasety z nagraniami dźwiękowymi książek.
3. Każdy egzemplarz zawiera tytuł, nazwisko autora, ISBN, wydawnictwo, jeśli jest to książka oraz dodatkowo nazwisko aktora, jeżeli jest to nagranie dźwiękowe.
   4. Może wystąpić wiele egzemplarzy książek oraz kaset z tymi samymi tytułami. Każdy egzemplarz, zarówno książka i kaseta, posiadają numer niepowtarzający się w ramach pozostałych identycznych danych (ISBN lub ISBN i nazwisko aktora).
4. W celu znalezienia tytułu należy podać ISBN lub ISBN i nazwisko aktora, jeżeli należy odszukać tytuł nagranej książki.
5. W celu wybrania właściwego egzemplarza należy podać ISBN, jeśli jest to książka oraz dodatkowo nazwisko aktora, jeśli jest to kaseta oraz numer egzemplarza.
6. Zarówno egzemplarze typu książka lub kaseta, mogą być przeznaczane do wypożyczenia na okres umowny oraz na okres ściśle określony.

Lista wymagań niefunkcjonalnych

1. Wstawianie danych o tytułach i egzemplarzach może odbywać się tylko przez uprawnione osoby
2. Wyszukiwanie informacji powinno odbywać się samodzielnie przez klienta
3. Operacje zarządzania i wyszukiwania informacji mogą być dokonane przez Internet przez aplikację uruchamianą przez przeglądarkę lub bez jej pośrednictwa

Diagram wymagań funkcjonalnych

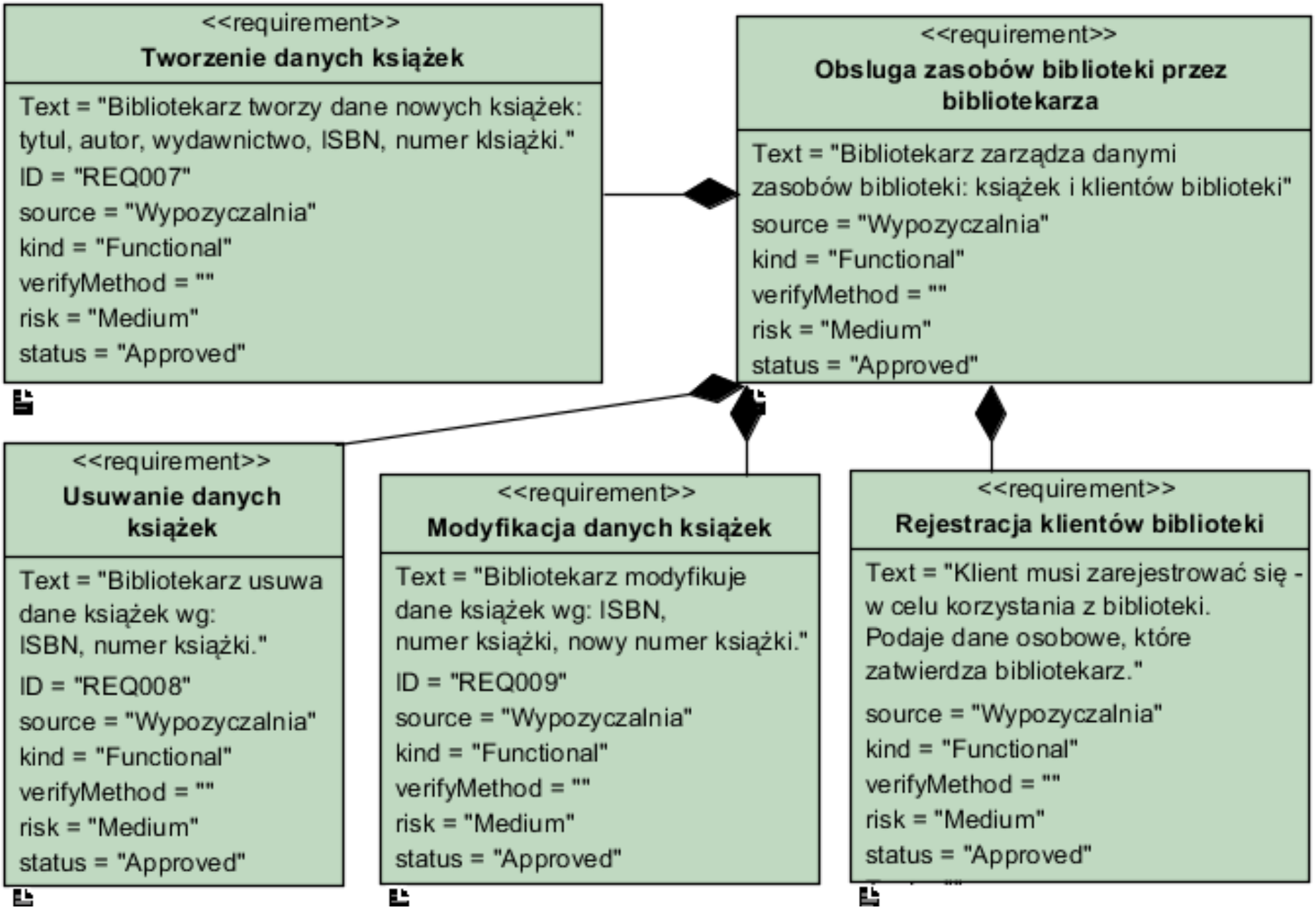


Diagram wymagań funkcjonalnych (cd)

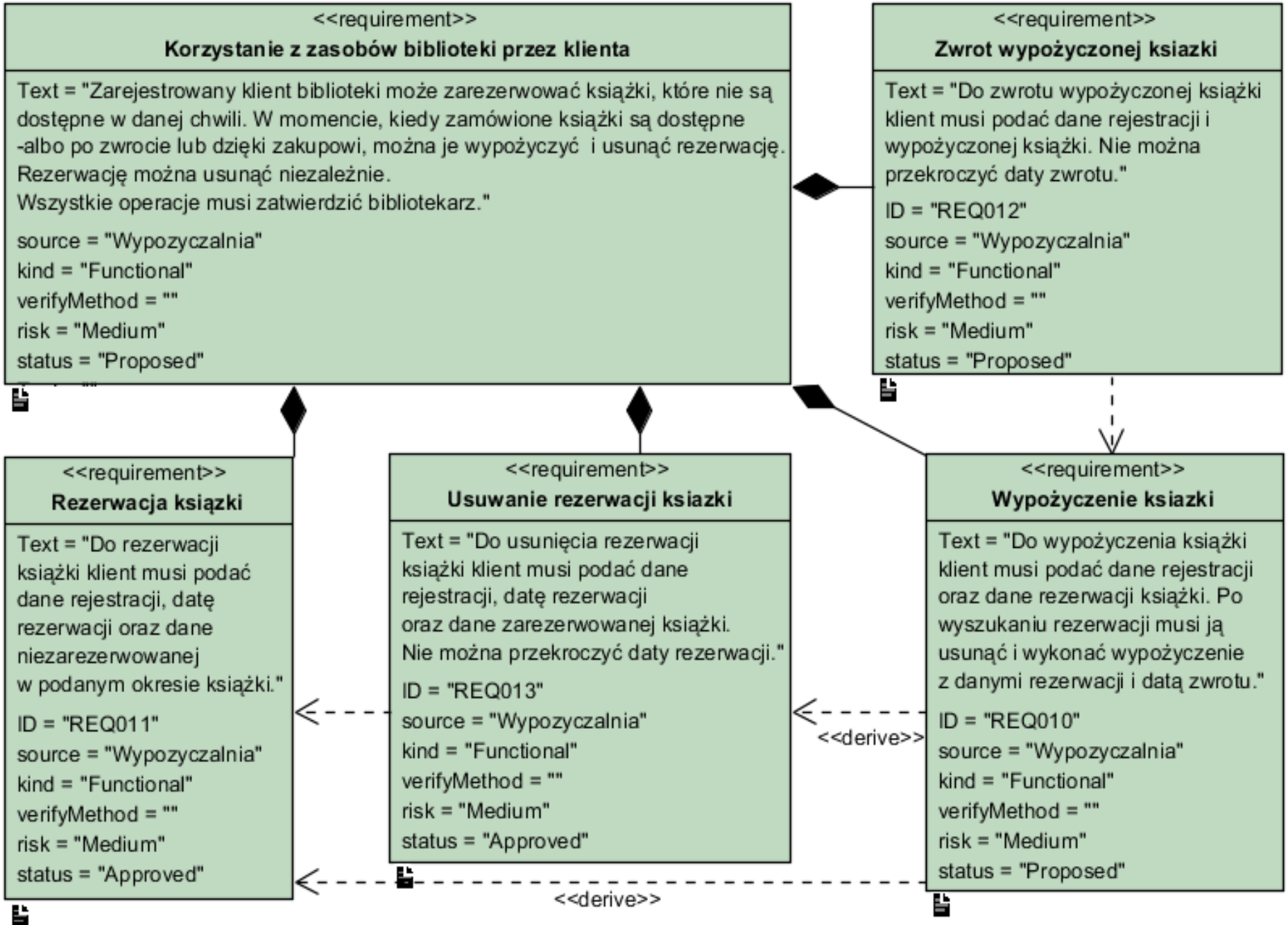
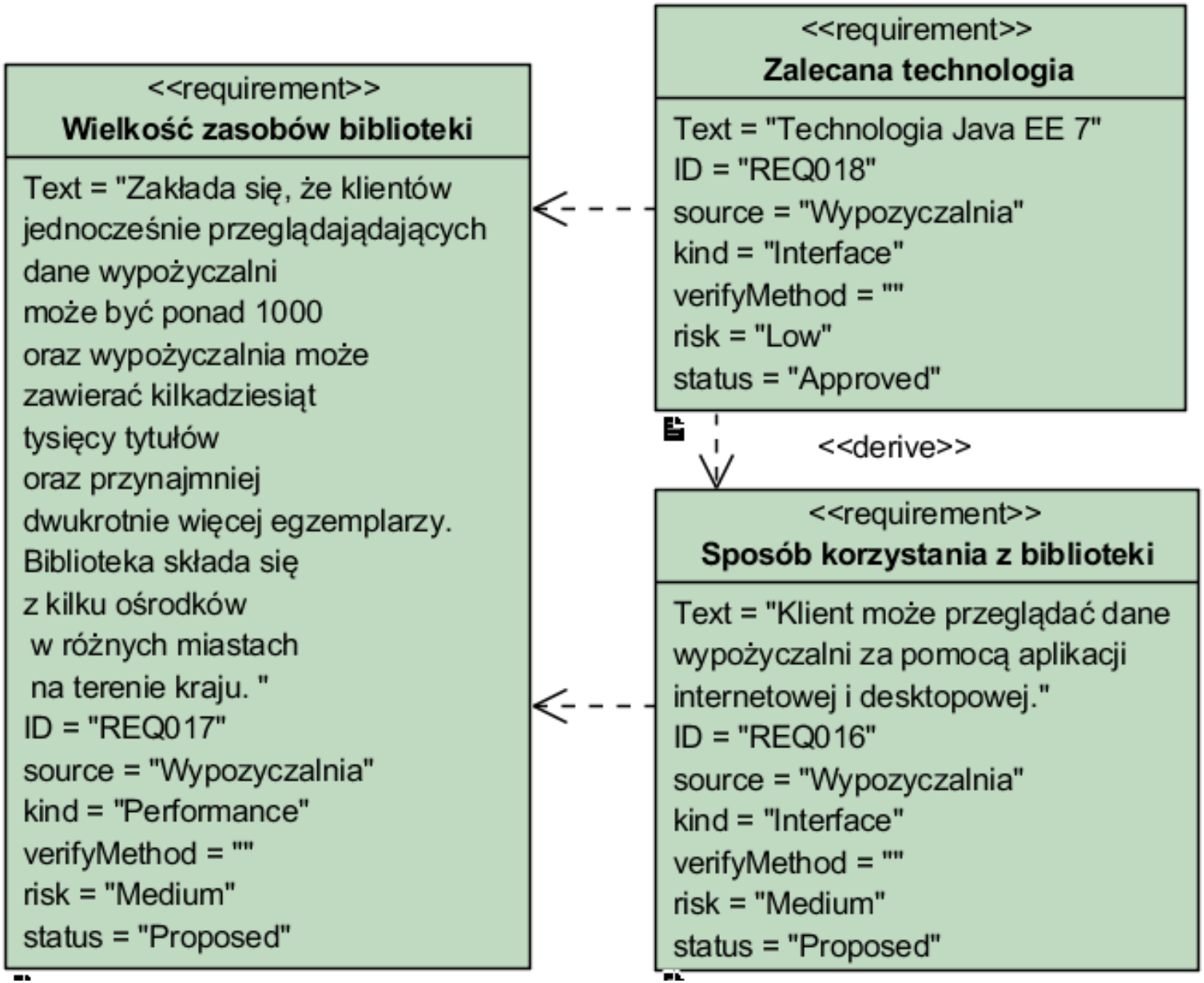


Diagram wymagań niefunkcjonalnych (cd)



III. Model analizy aplikacji oparty na diagramie przypadków użycia

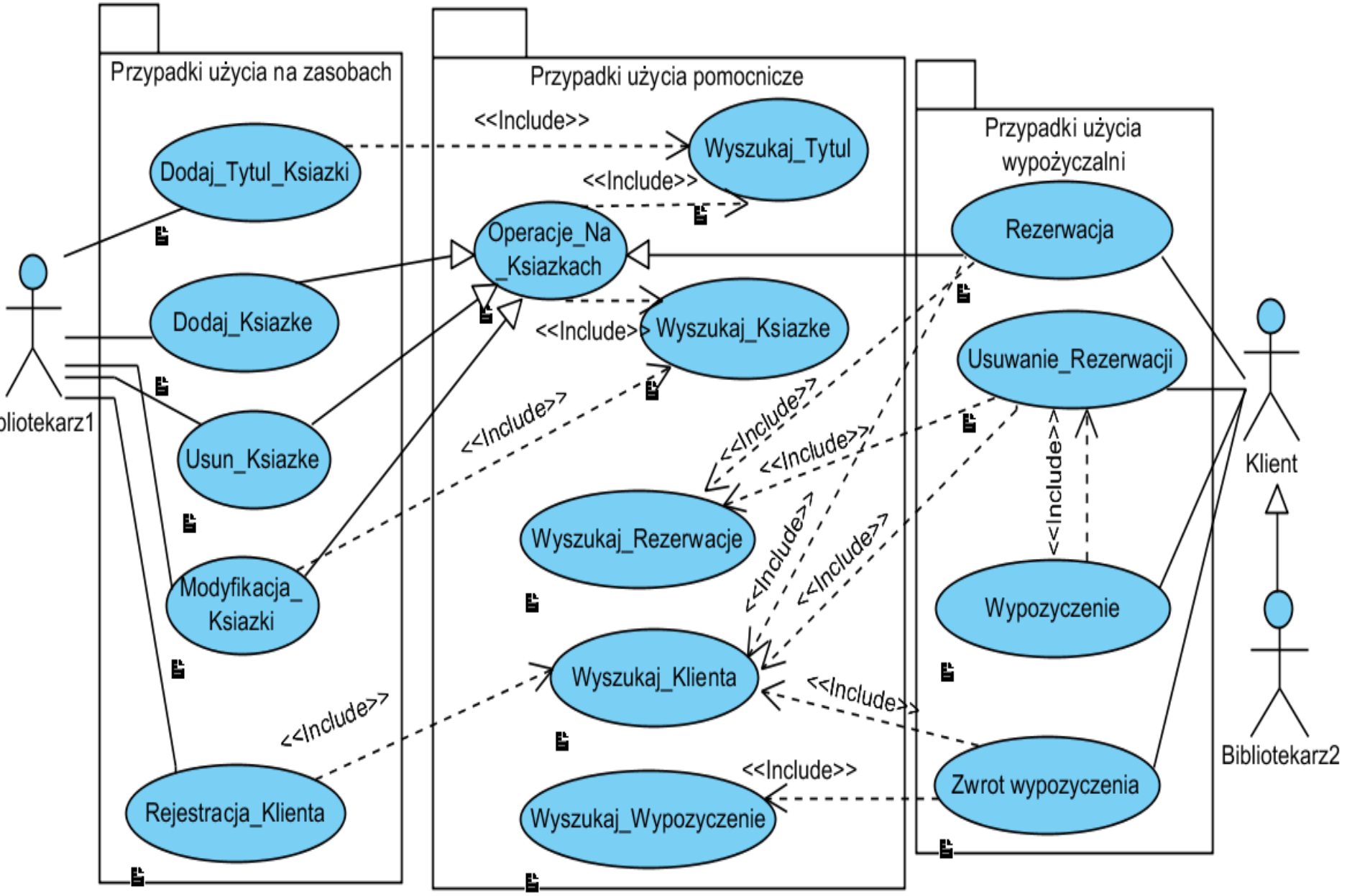
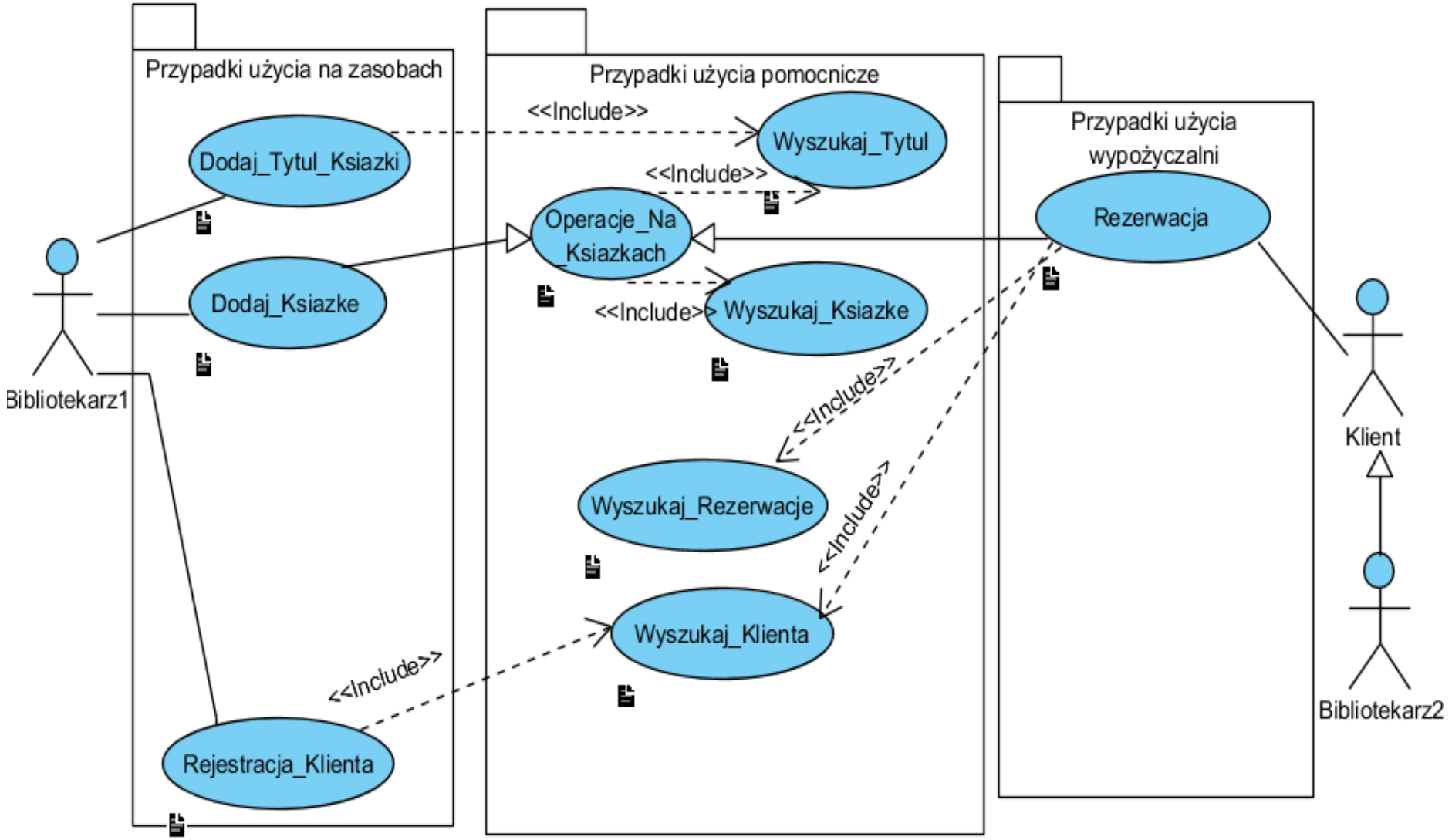


Diagram przypadków użycia – wybrany fragment



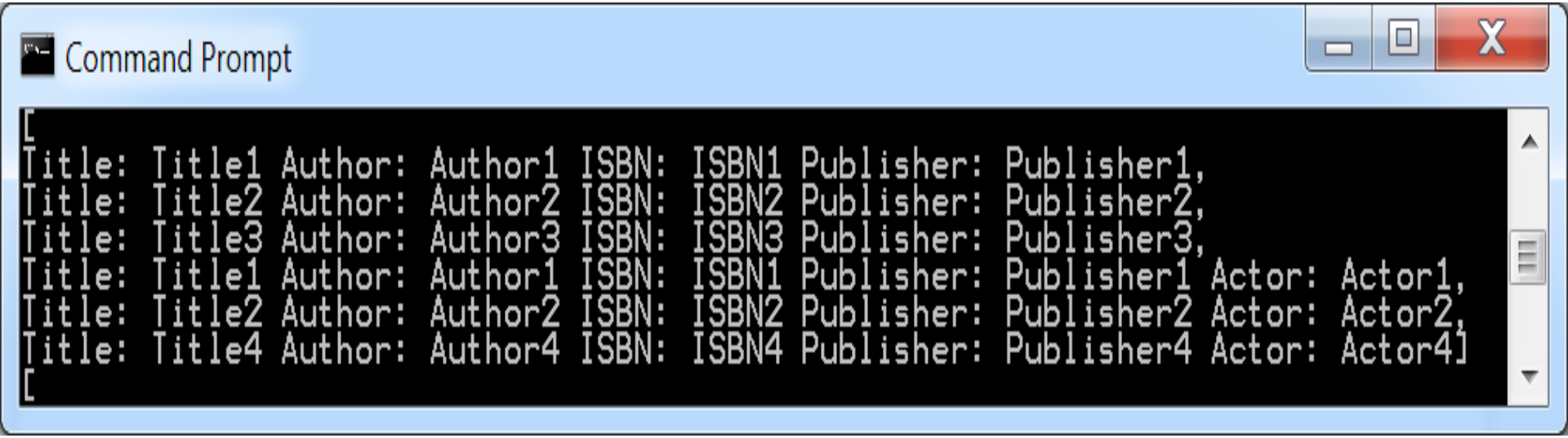
IV. Model projektowy i implementacja warstwy biznesowej warstwy biznesowej oparty na diagramie klas i diagramie sekwencji tworzony metodą iteracyjno-rozwojową, sterowany realizacją przypadków użycia

Iteracja 1

Projekt przypadku użycia

„ **Dodaj_Tytul_Ksiazki**”

za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

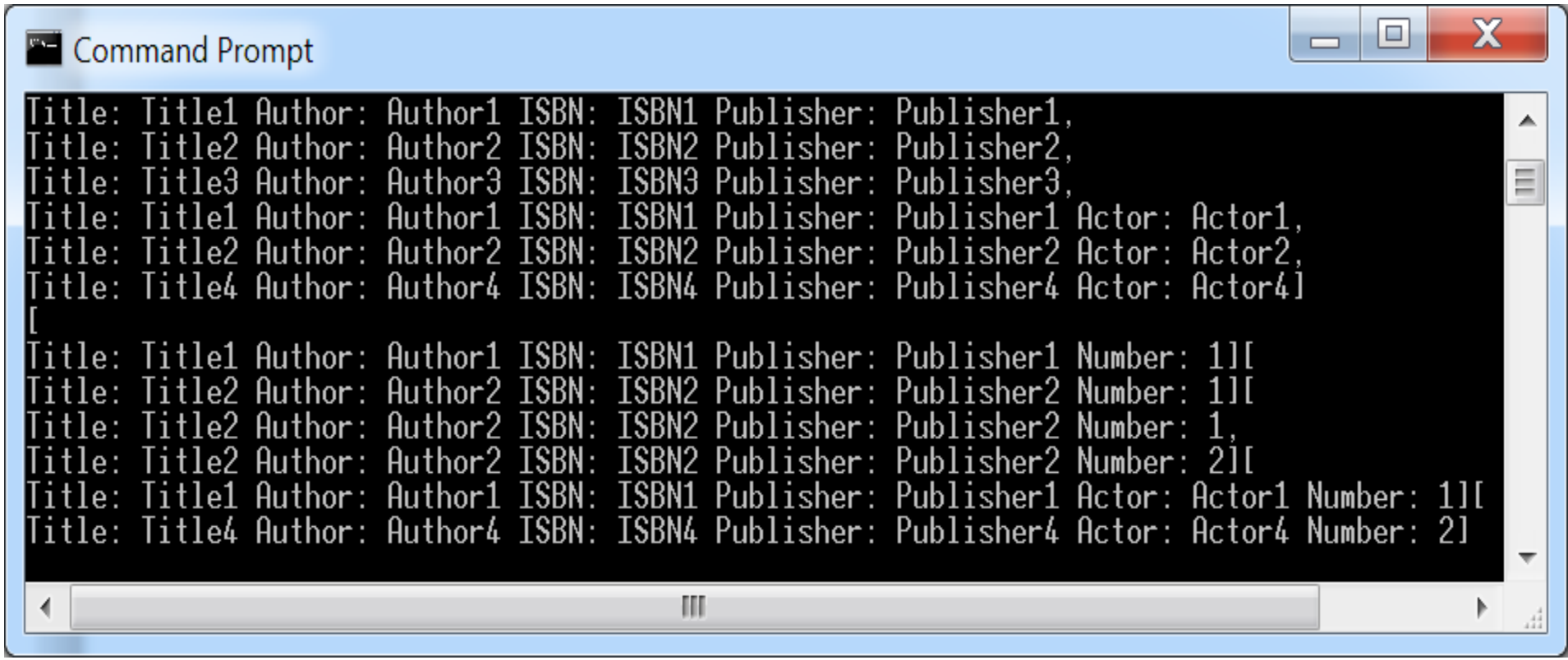

Command Prompt
[
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1,
Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publisher2,
Title: Title3 Author: Author3 ISBN: ISBN3 Publisher: Publisher3,
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1,
Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publisher2 Actor: Actor2,
Title: Title4 Author: Author4 ISBN: ISBN4 Publisher: Publisher4 Actor: Actor4]
[

Iteracja 2

Projekt przypadku użycia

„**Dodaj_Ksiazke**”

za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.


Command Prompt
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1,
Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publisher2,
Title: Title3 Author: Author3 ISBN: ISBN3 Publisher: Publisher3,
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1,
Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publisher2 Actor: Actor2,
Title: Title4 Author: Author4 ISBN: ISBN4 Publisher: Publisher4 Actor: Actor4]
[
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Number: 1][
Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publisher2 Number: 1][
Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publisher2 Number: 1,
Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publisher2 Number: 2][
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 1][
Title: Title4 Author: Author4 ISBN: ISBN4 Publisher: Publisher4 Actor: Actor4 Number: 2]

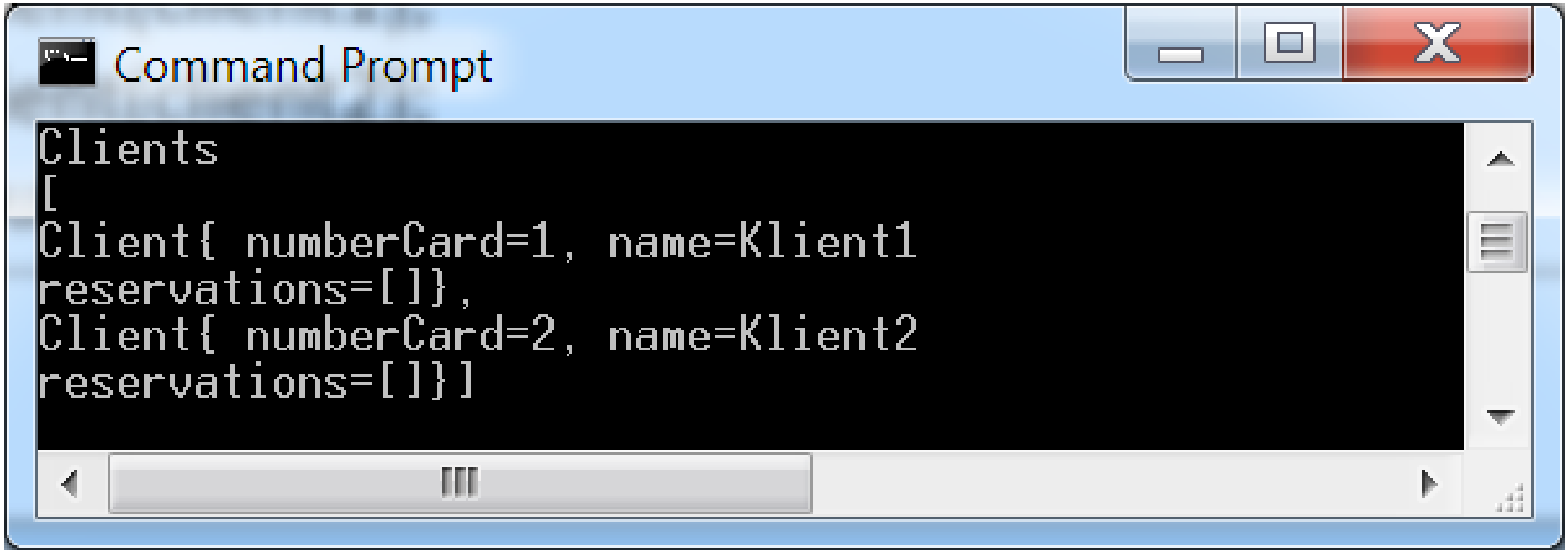
Iteracja 3

Projekt przypadku użycia

„ **Rejestracja_Klienta**”

za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

```
//dodawanie klientów
        System.out.println("\nClients");
String[] client1 = {"1", "Klient1", "1"}, dclient1 = {"0", "1"};
String[] client2 = {"1", "Klient2", "2"}, dclient2 = {"0", "2"}, dclient3 = {"0", "3"};
        ap.addClient(client1);
        ap.addClient(client1);
        ap.addClient(client2);
        System.out.println(ap.clients);
```



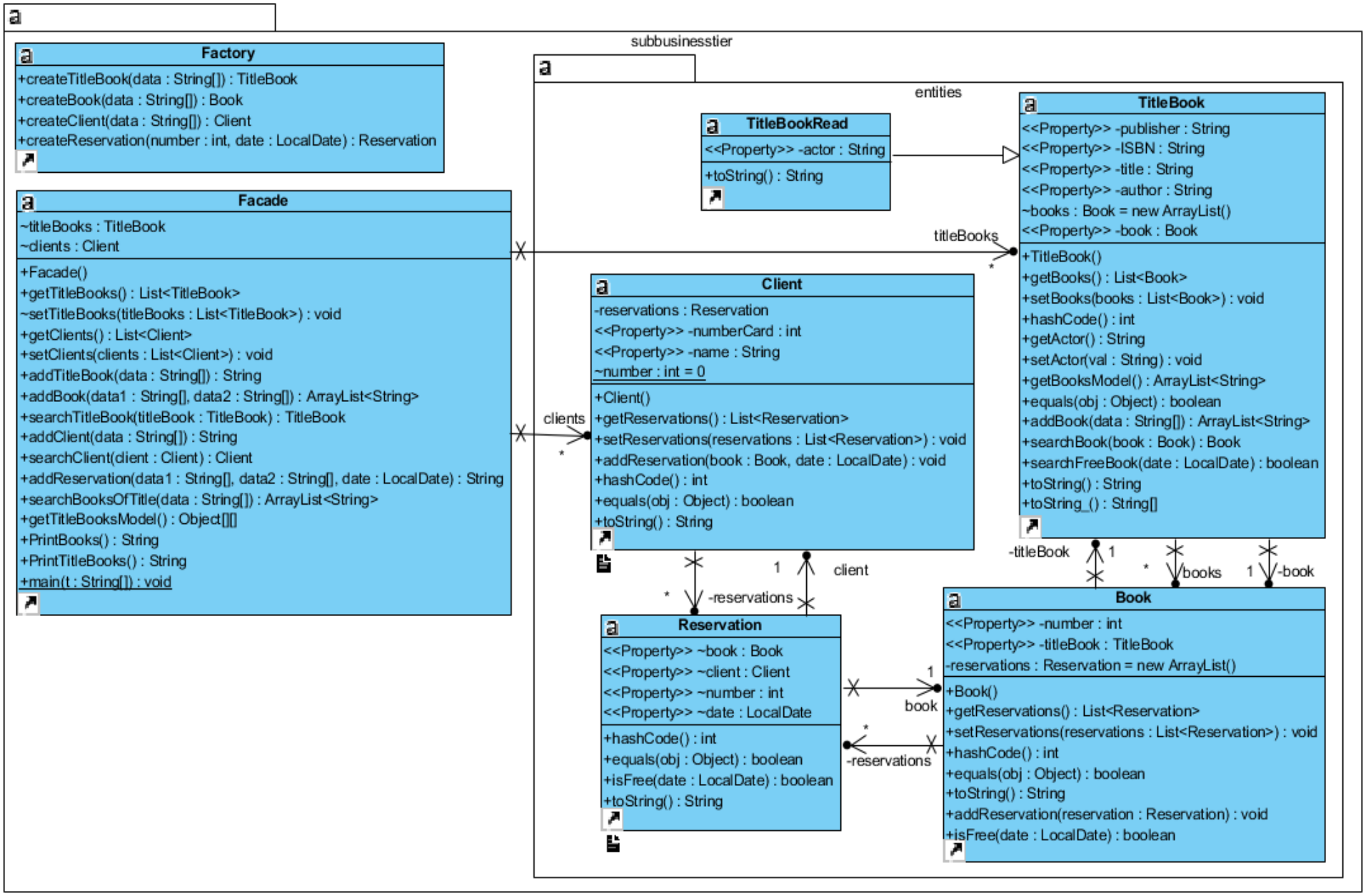
Iteracja 4

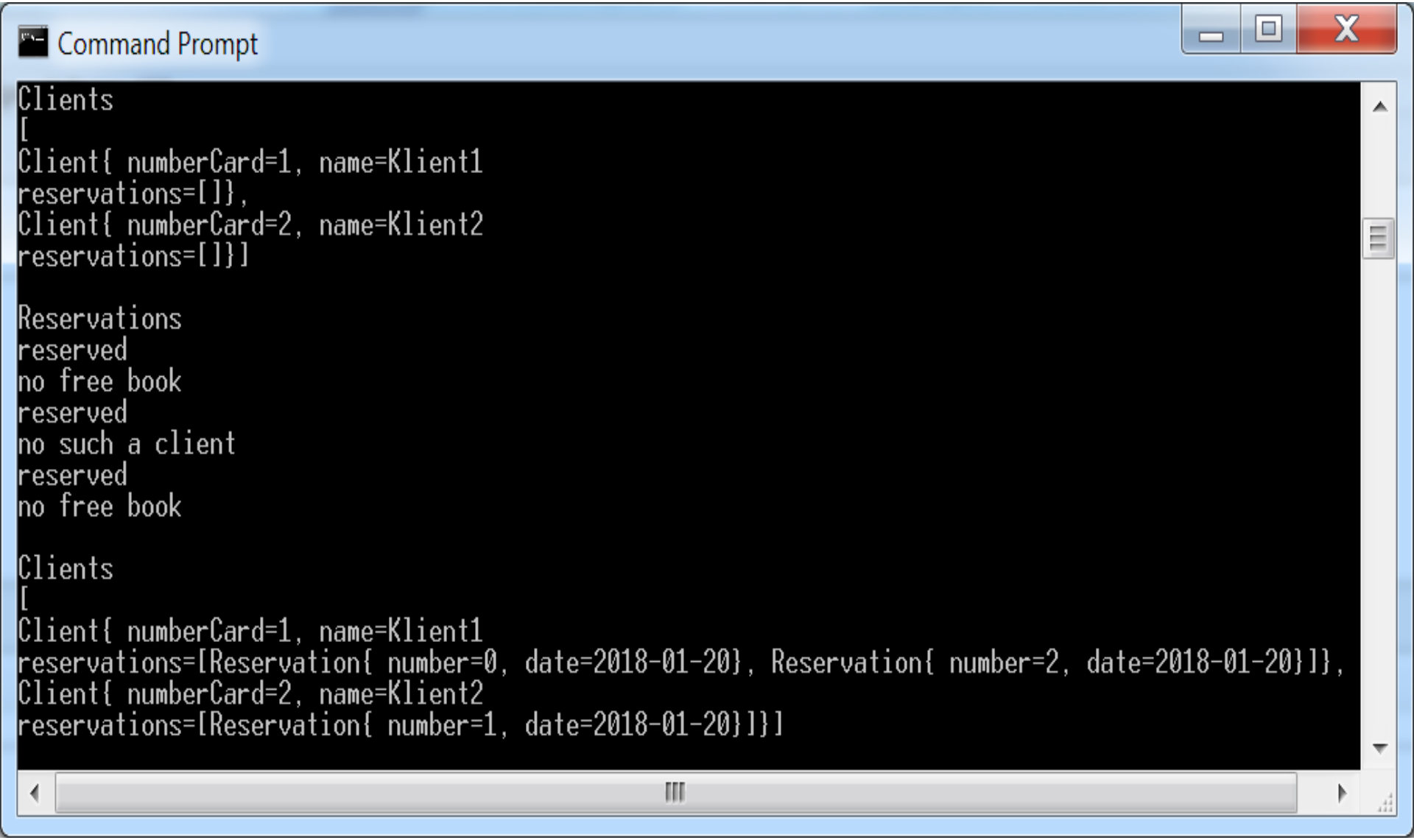
Projekt przypadku użycia

„**Rezerwacja**”

za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

Rezultat – diagram klas uzyskany w procesie projektowania (przebieg pokazany w dodatku do wykładu 5)




Command Prompt
Clients
[
Client{ numberCard=1, name=Klient1
reservations=[]},
Client{ numberCard=2, name=Klient2
reservations=[]}]
Reservations
reserved
no free book
reserved
no such a client
reserved
no free book
Clients
[
Client{ numberCard=1, name=Klient1
reservations=[Reservation{ number=0, date=2018-01-20}, Reservation{ number=2, date=2018-01-20}]},
Client{ numberCard=2, name=Klient2
reservations=[Reservation{ number=1, date=2018-01-20}]}]

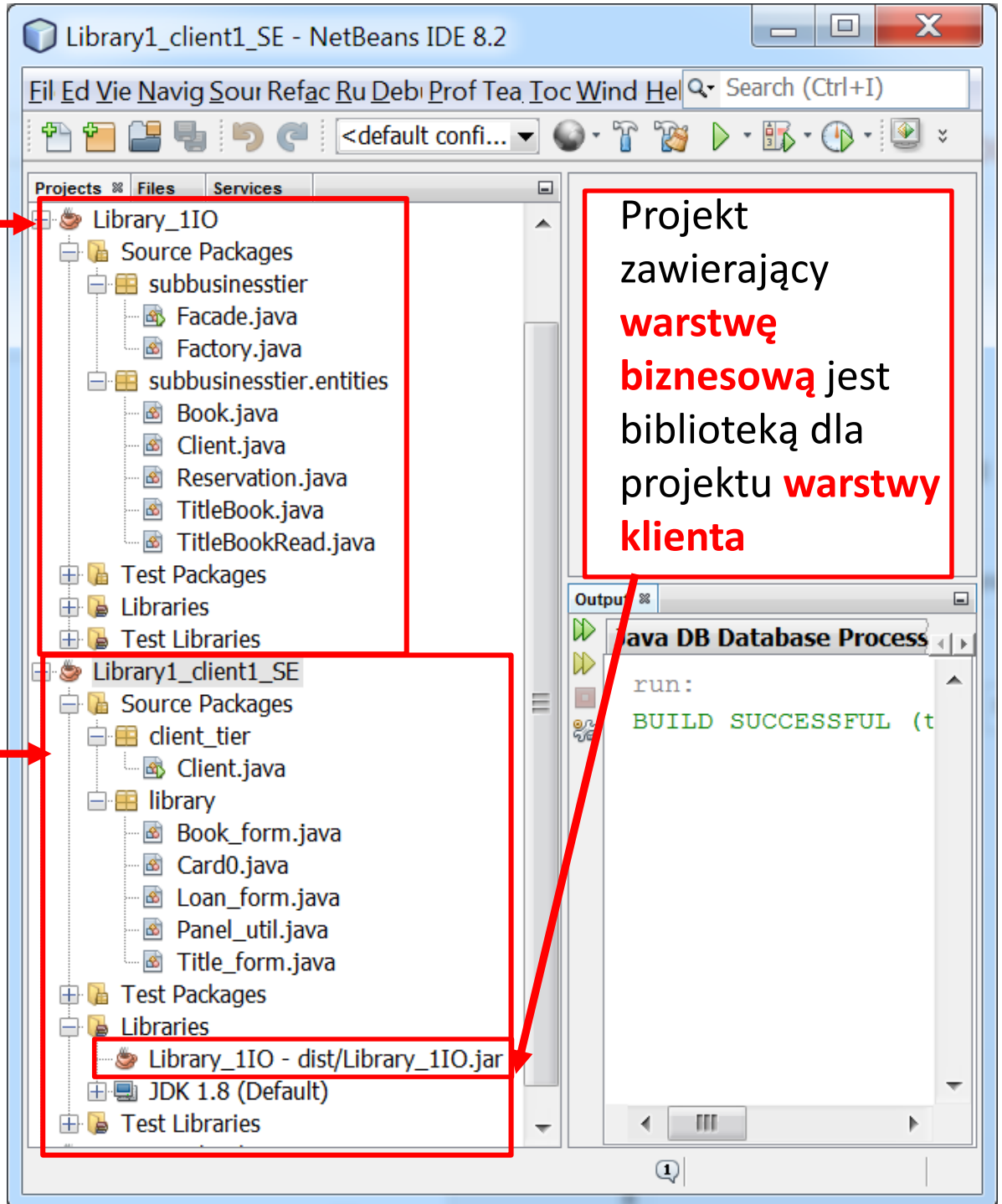
Wykonanie aplikacji dwuwarstwowej

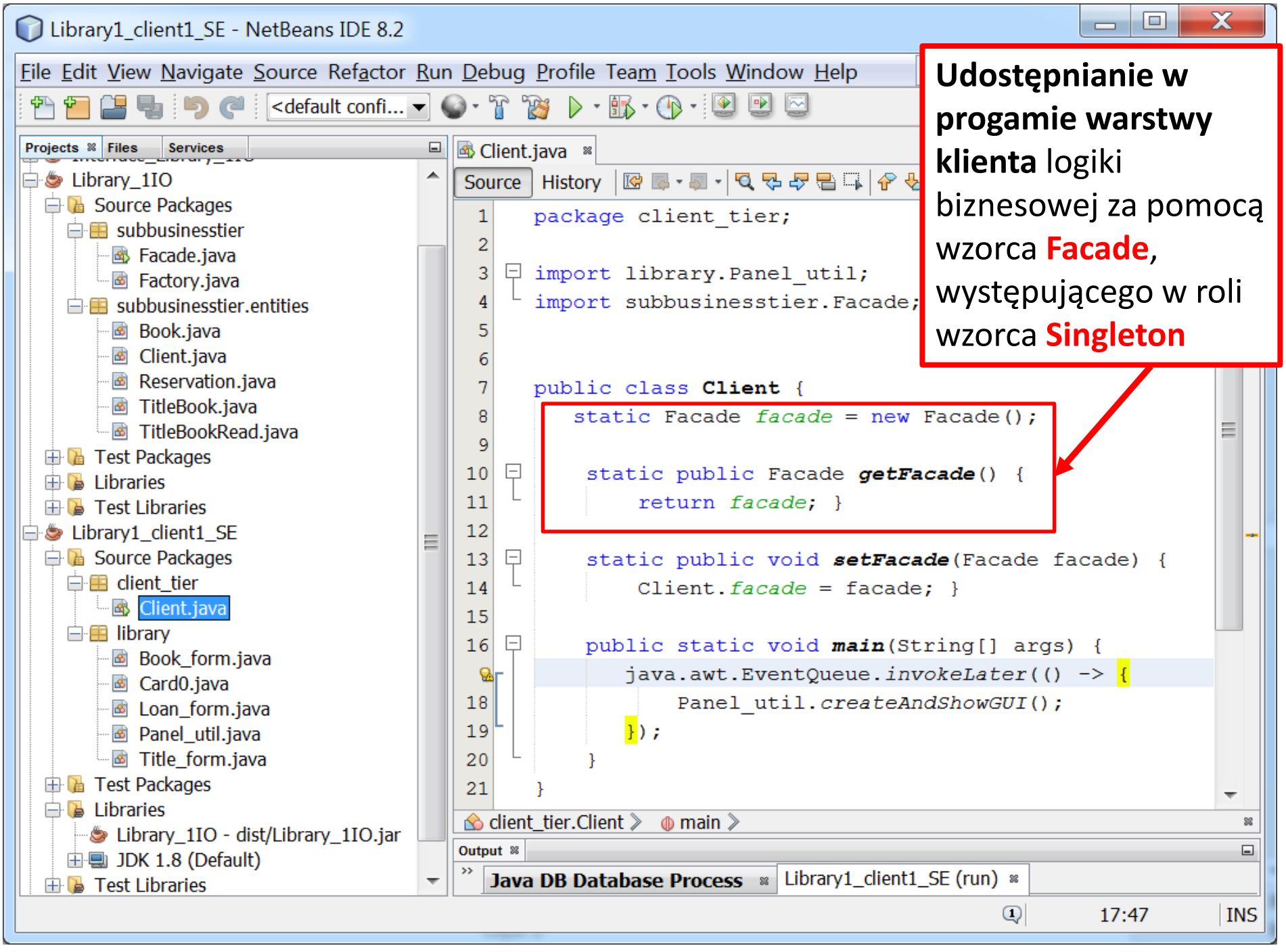
Warstwa klienta
-aplikacja desktopowa (GUI)

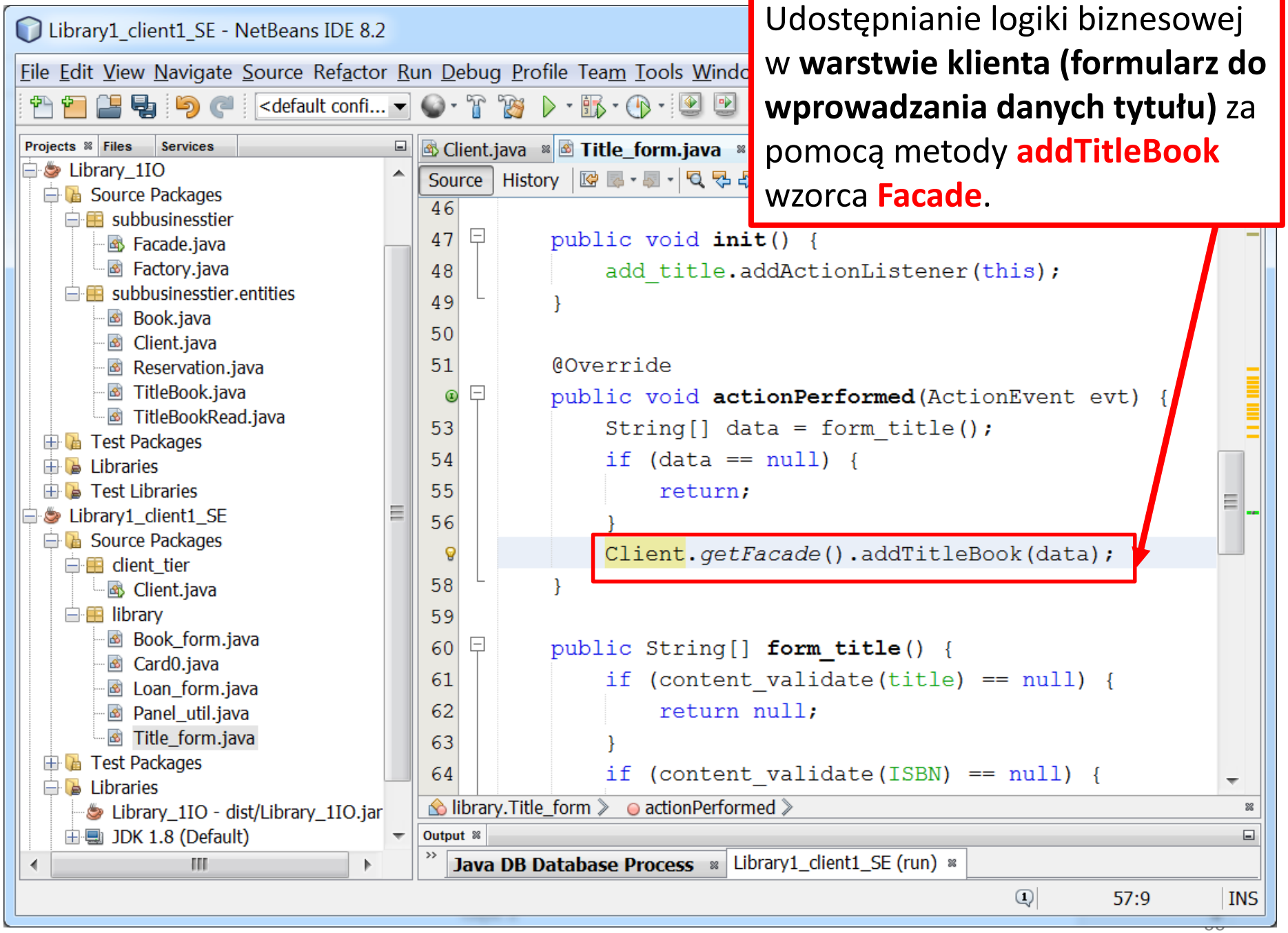
Warstwa biznesowa – przetwarzanie danych (projekt typu Java Class Library, używany jako biblioteka logiki biznesowej)

Projekt typu Java Class Library zawierający kod warstwy biznesowej wykonany podczas 4 iteracji

Projekt typu Java Application zawierający kod warstwy klienta desktopowego z interfejsem graficznym użytkowania (GUI) do kodu 1-ej i 2-ej iteracji




Library1_client1_SE - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default confi...
Projects Files Services
Library_1IO
Source Packages
subbusinesstier
Facade.java
Factory.java
subbusinesstier.entities
Book.java
Client.java
Reservation.java
TitleBook.java
TitleBookRead.java
Test Packages
Libraries
Test Libraries
Library1_client1_SE
Source Packages
client_tier
Client.java
library
Book_form.java
Card0.java
Loan_form.java
Panel_util.java
Title_form.java
Test Packages
Libraries
Library_1IO - dist/Library_1IO.jar
JDK 1.8 (Default)
Test Libraries
Client.java
Source History
1 package client_tier;
2
3 import library.Panel_util;
4 import subbusinesstier.Facade;
5
6
7 public class Client {
8 static Facade facade = new Facade();
9
10 static public Facade getFacade() {
11 return facade; }
12
13 static public void setFacade(Facade facade) {
14 Client.facade = facade; }
15
16 public static void main(String[] args) {
java.awt.EventQueue.invokeLater(() -> {
18 Panel_util.createAndShowGUI();
19 });
20 }
21 }
client_tier.Client
main
Output
Java DB Database Process
Library1_client1_SE (run)
17:47
INS
Udostępnianie w progamie warstwy klienta logiki biznesowej za pomocą wzorca Facade, występującego w roli wzorca Singleton


Udostępnianie logiki biznesowej w **warstwie klienta (formularz do wprowadzania danych tytułu)** za pomocą metody **addTitleBook** wzorca **Facade**.
Library1_client1_SE - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Windo
<default confi...
Projects Files Services
Library_1IO
Source Packages
subbusinesstier
Facade.java
Factory.java
subbusinesstier.entities
Book.java
Client.java
Reservation.java
TitleBook.java
TitleBookRead.java
Test Packages
Libraries
Test Libraries
Library1_client1_SE
Source Packages
client_tier
Client.java
library
Book_form.java
Card0.java
Loan_form.java
Panel_util.java
Title_form.java
Test Packages
Libraries
Library_1IO - dist/Library_1IO.jar
JDK 1.8 (Default)
Client.java
Title_form.java
Source History
46
47 public void init() {
48 add_title.addActionListener(this);
49 }
50
51 @Override
public void actionPerformed(ActionEvent evt) {
53 String[] data = form_title();
54 if (data == null) {
55 return;
56 }
Client.getFacade().addTitleBook(data);
58 }
59
60 public String[] form_title() {
61 if (content_validate(title) == null) {
62 return null;
63 }
64 if (content_validate(ISBN) == null) {
library.Title_form
actionPerformed
Output
Java DB Database Process
Library1_client1_SE (run)
57:9
INS

Client.java | Title_form.java | Book_form.java

Source | History

```
98
99      void table_content() {
100         Object[][] titles = Client.getFacade().getTitleBooksModel();
101         model.setData(titles);
102     }
103
104     @Override
        public void actionPerformed(ActionEvent event) {
106         if (!table.getSelectionModel().isSelectionEmpty()) {
107             String what_book_type;
108             if (number.getText().equals("")) {
109                 JOptionPane.showMessageDialog(this, "required value");
110                 return;
111             }
112             what_book_type = "0";
113             String data2[] = {what_book_type, (String) number.getText()};
114             ArrayList<String> help3 = Client.getFacade().addBook(title(), data2);
115             if (help3 != null) {
116                 list_content(help3, books);
117             }
118         }
119
```

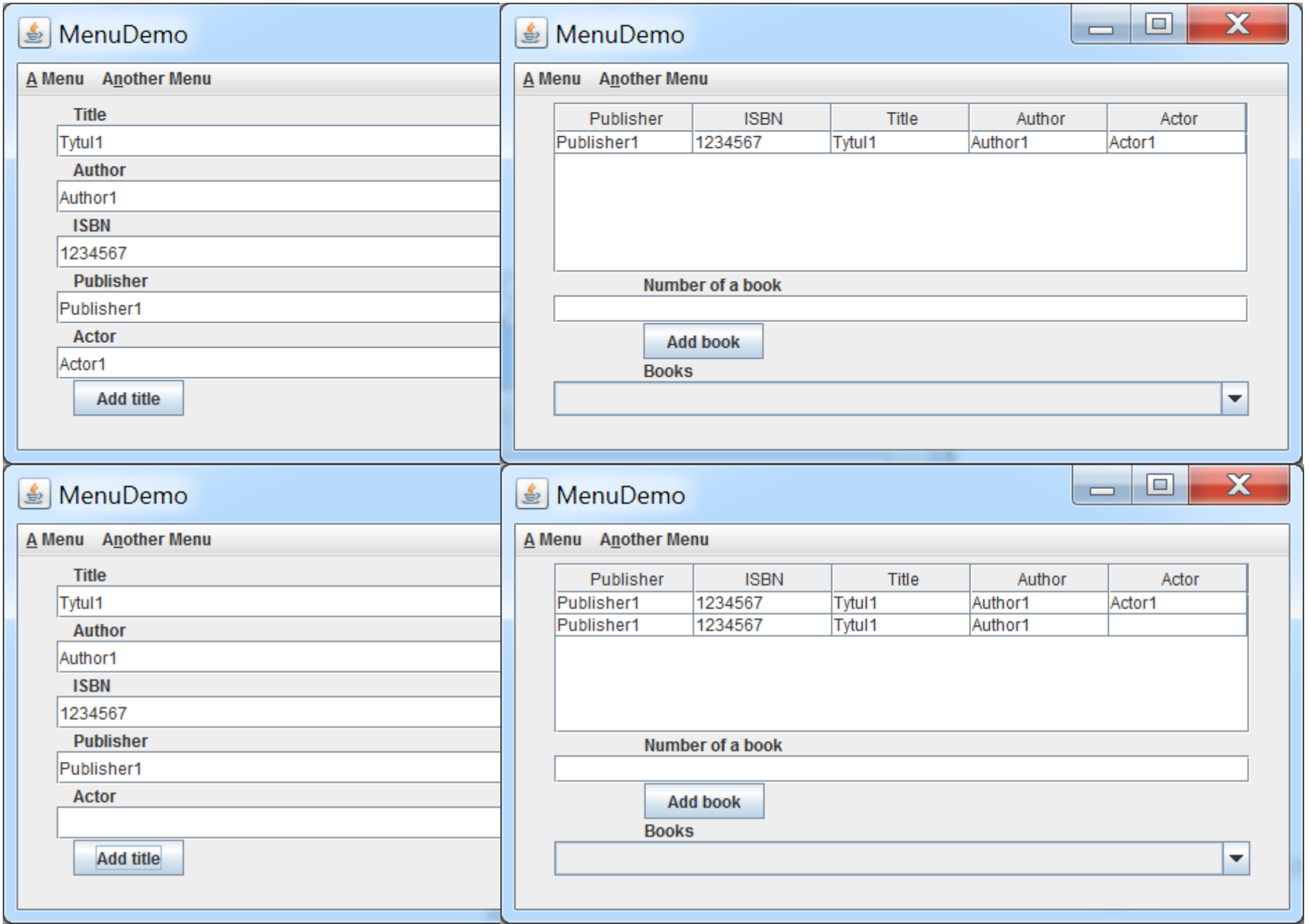
Wynik metody **getTitkeBookModel** jest wykorzystany jako **model widoku JTable**

Udostępnianie logiki biznesowej w **warstwie klienta (formularz do wprowadzania danych książki)** za pomocą metody **addBook** wzorca **Facade**,

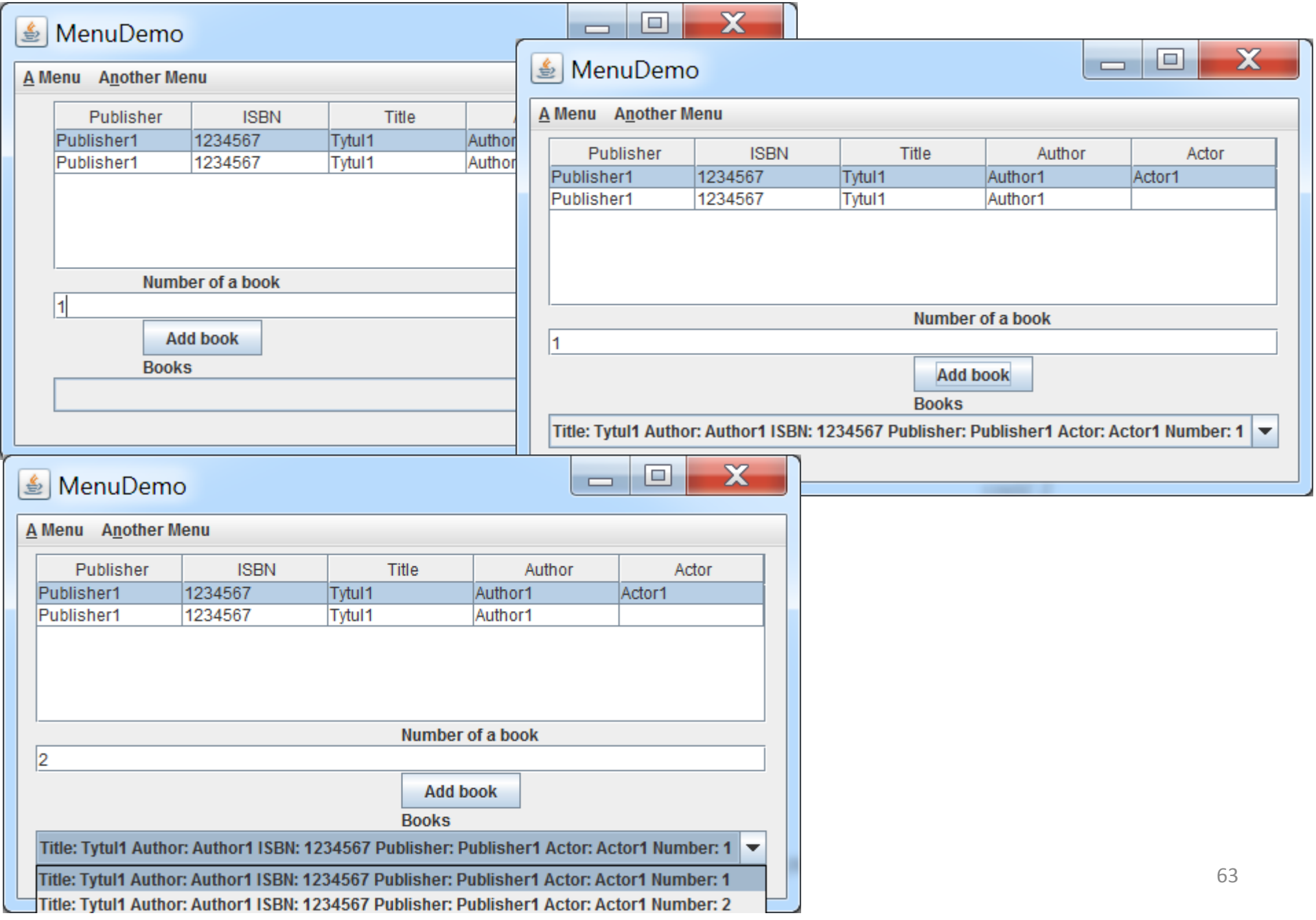
Wynik **help3** metody **addBook** jest wykorzystany jako **model widoku JComboBox**

```
private void list_content(ArrayList<String> col, JComboBox list)
{    String s;
     list.removeAllItems();
     Iterator<String> iterator = col.iterator();
     while (iterator.hasNext()) {
        s = iterator.next();
        list.addItem(s);  }
  }
```

Formularze do wprowadzanie danych tytułów (z lewej) i książek (z prawej)



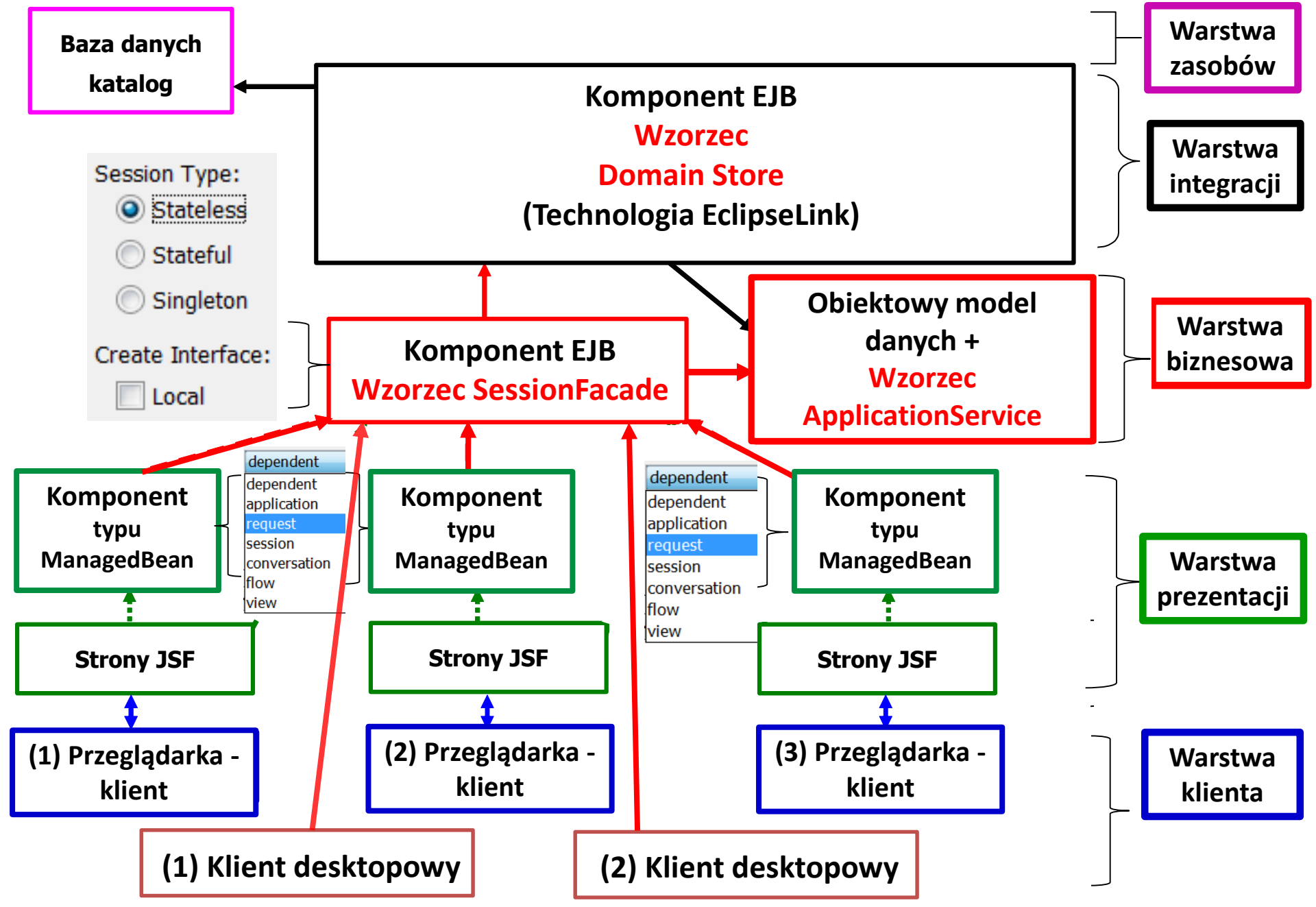
Wprowadzanie danych książek papierowych i nagranych



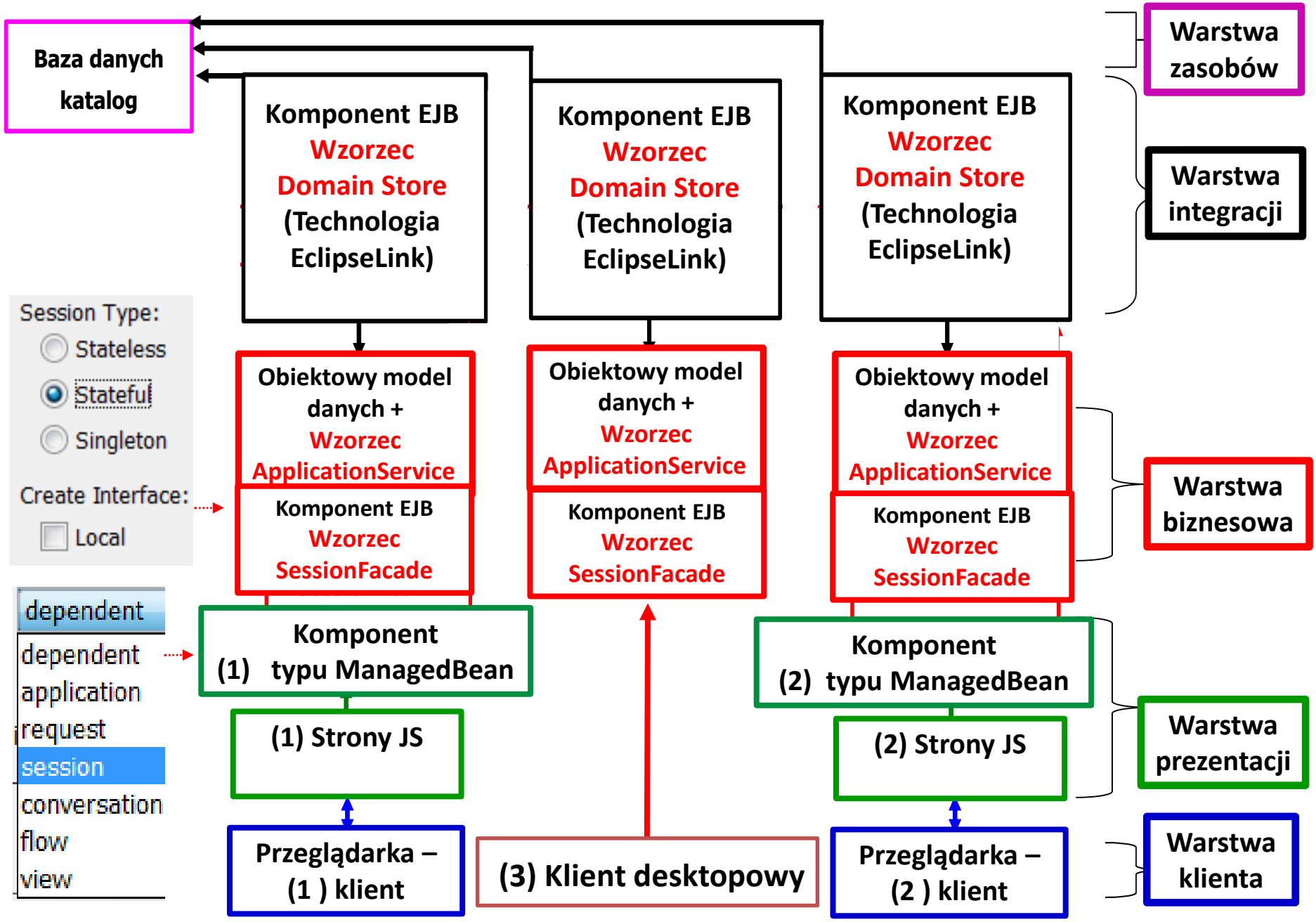
Zagadnienia

1. Wielowarstwowa architektura systemu informatycznego
2. Ocena i poprawa (refaktoryzacja) architektury wielowarstwowej systemu informatycznego
3. Wzorce projektowe stosowane przy budowie wielowarstwowej aplikacji internetowej
4. Przykład modelowania i projektowania części warstwy biznesowej z obiektami typu POJO. Wykonanie aplikacji dwuwarstwowej dla jednego użytkownika.
5. Przykłady architektury wielowarstwowej aplikacji internetowej typu EE. Wykonanie aplikacji typu EE. Warstwa biznesowa: komponenty typu EJB + obiekty POJO

Architektura aplikacji pięciowarstwowej -Java EE 7.0 JavaServer Faces

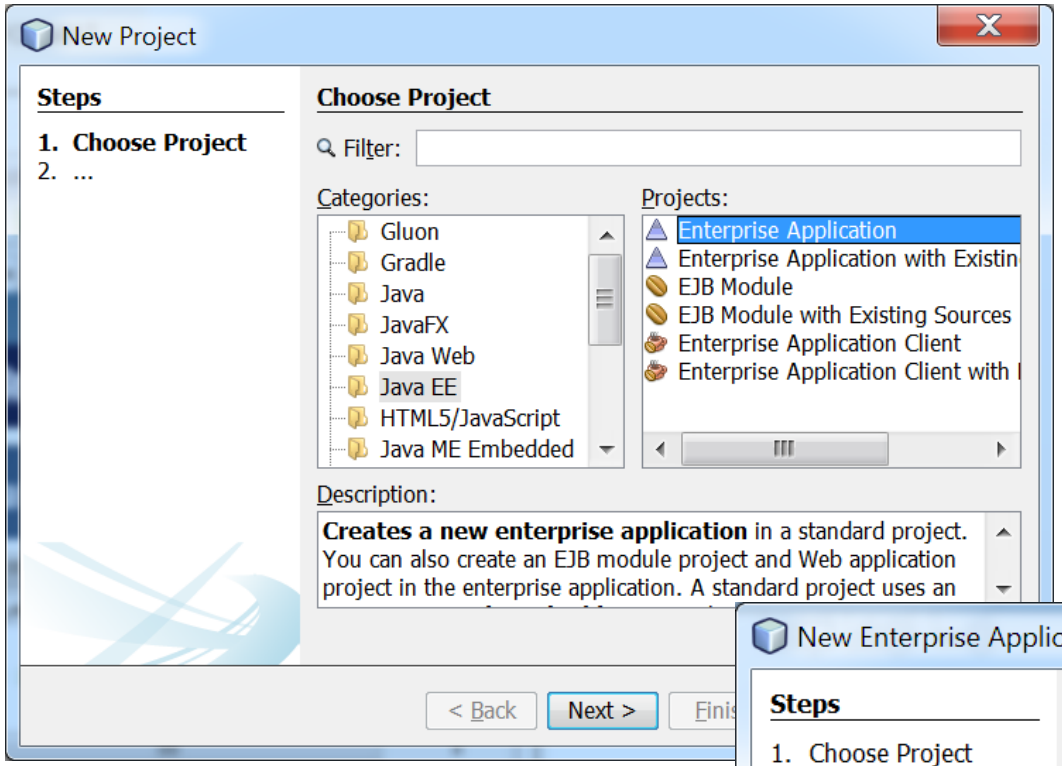


Architektura aplikacji pięciowarstwowej – Java EE 7.0 JavaServer Faces

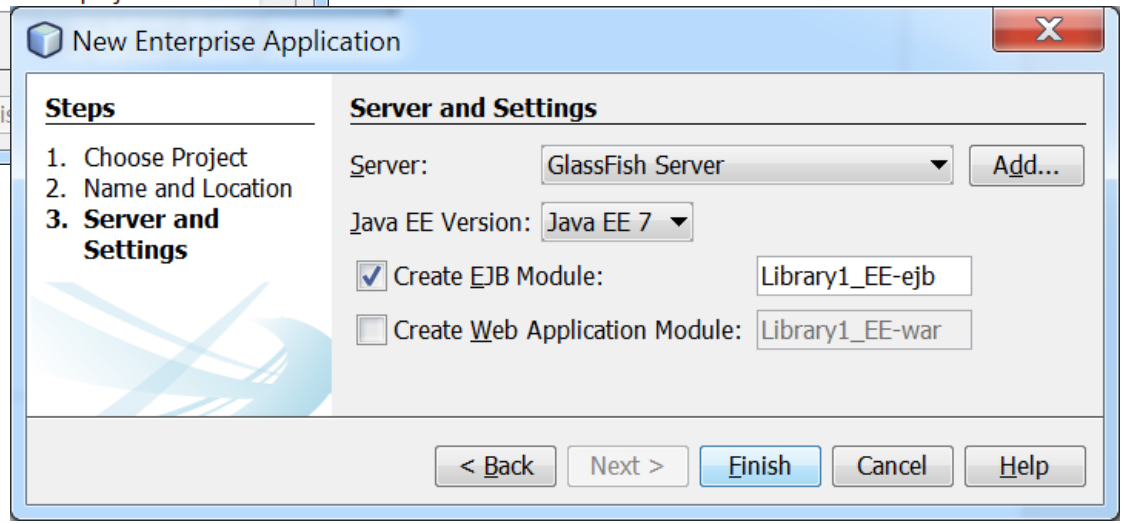


Wykonanie aplikacji typu Java EE z modułem EJB.

W tym module należy umieścić komponent typu EJB w celu umożliwienia zdalnego dostępu do metod obiektu typu Facade przez wiele aplikacji reprezentujących warstwę klienta: desktopowych i internetowych

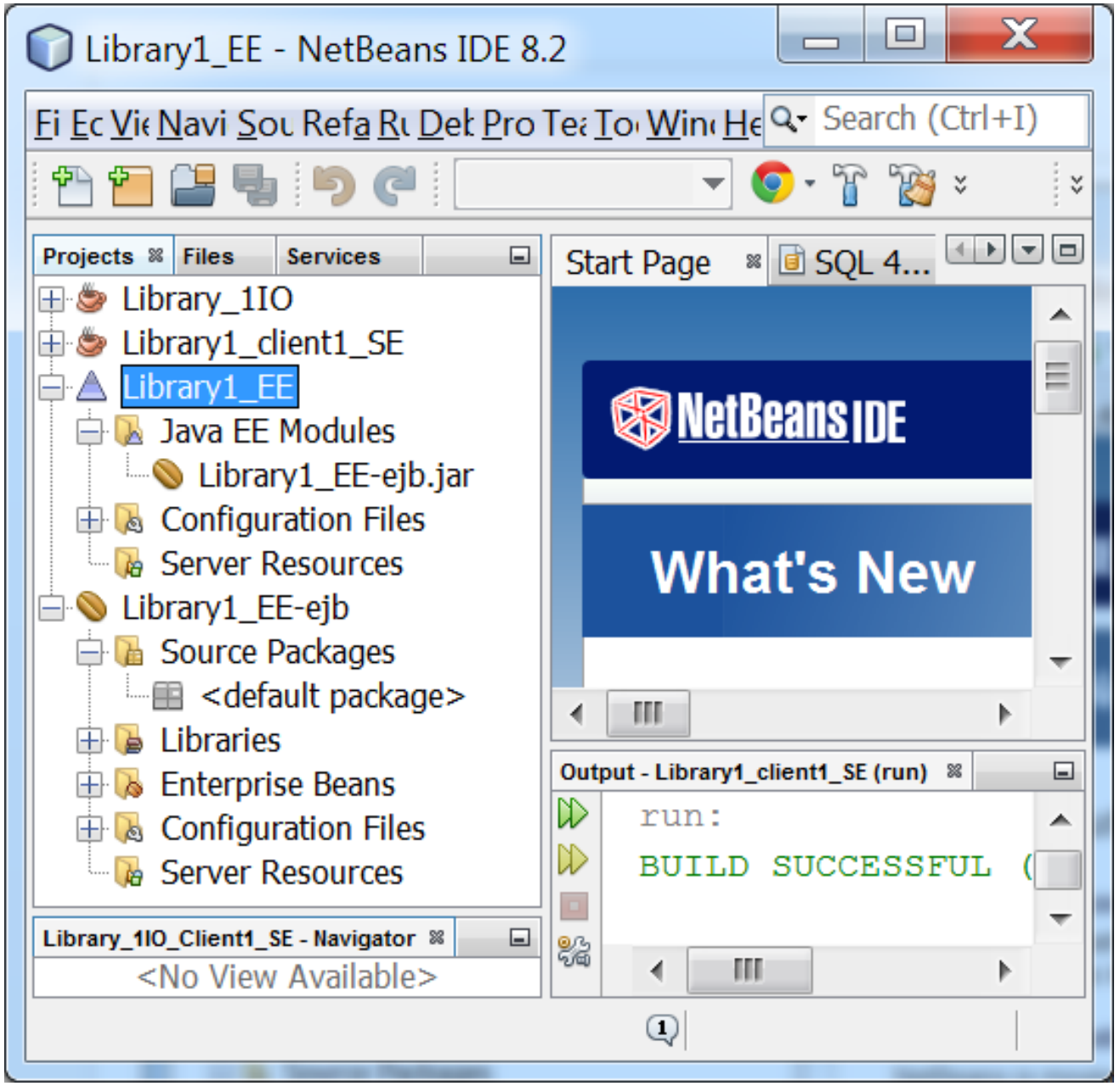


1. Tworzenie aplikacji typu Enterprise Application

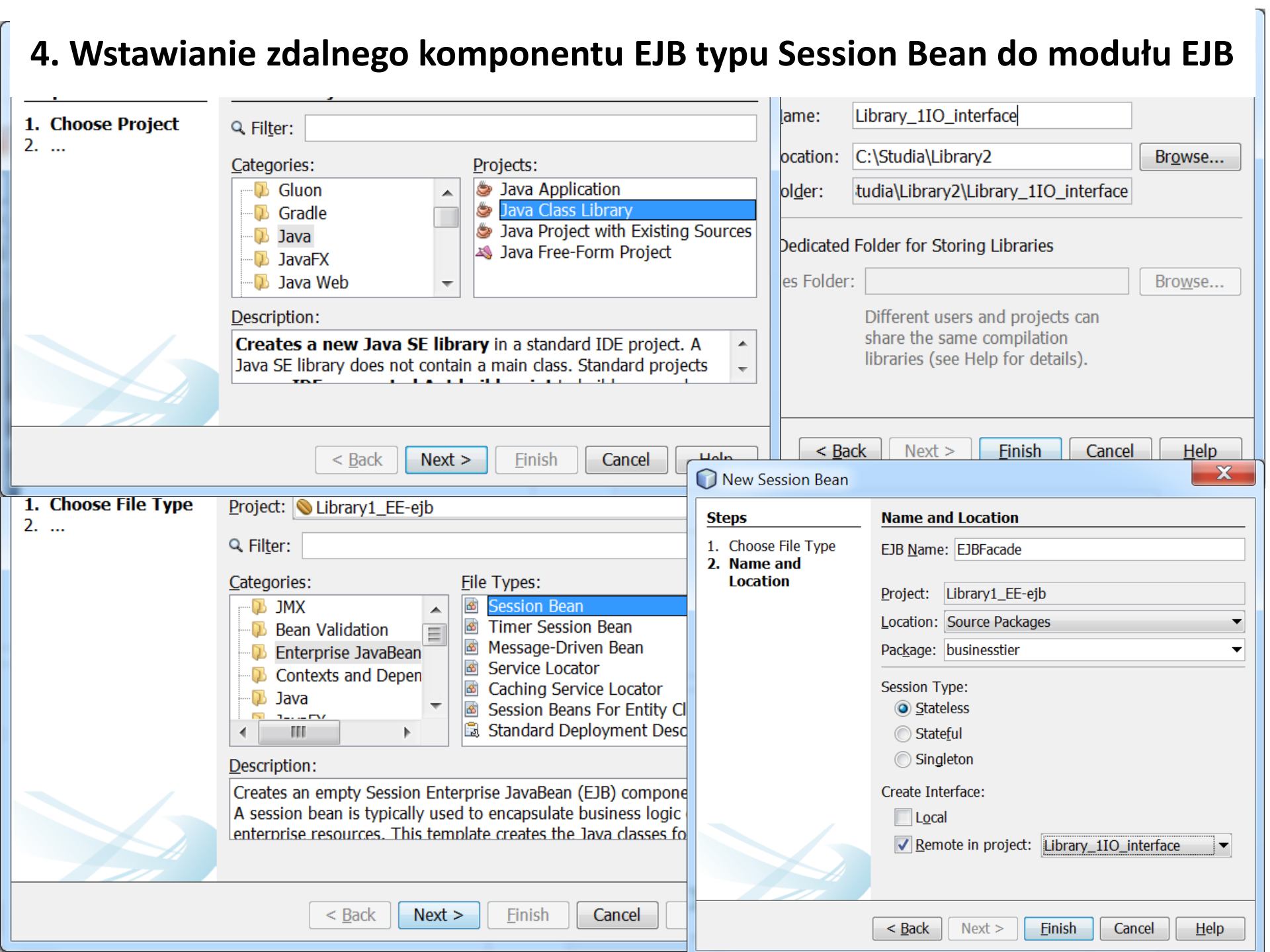


2. Dodanie modułu EJB w aplikacji Enterprise Application

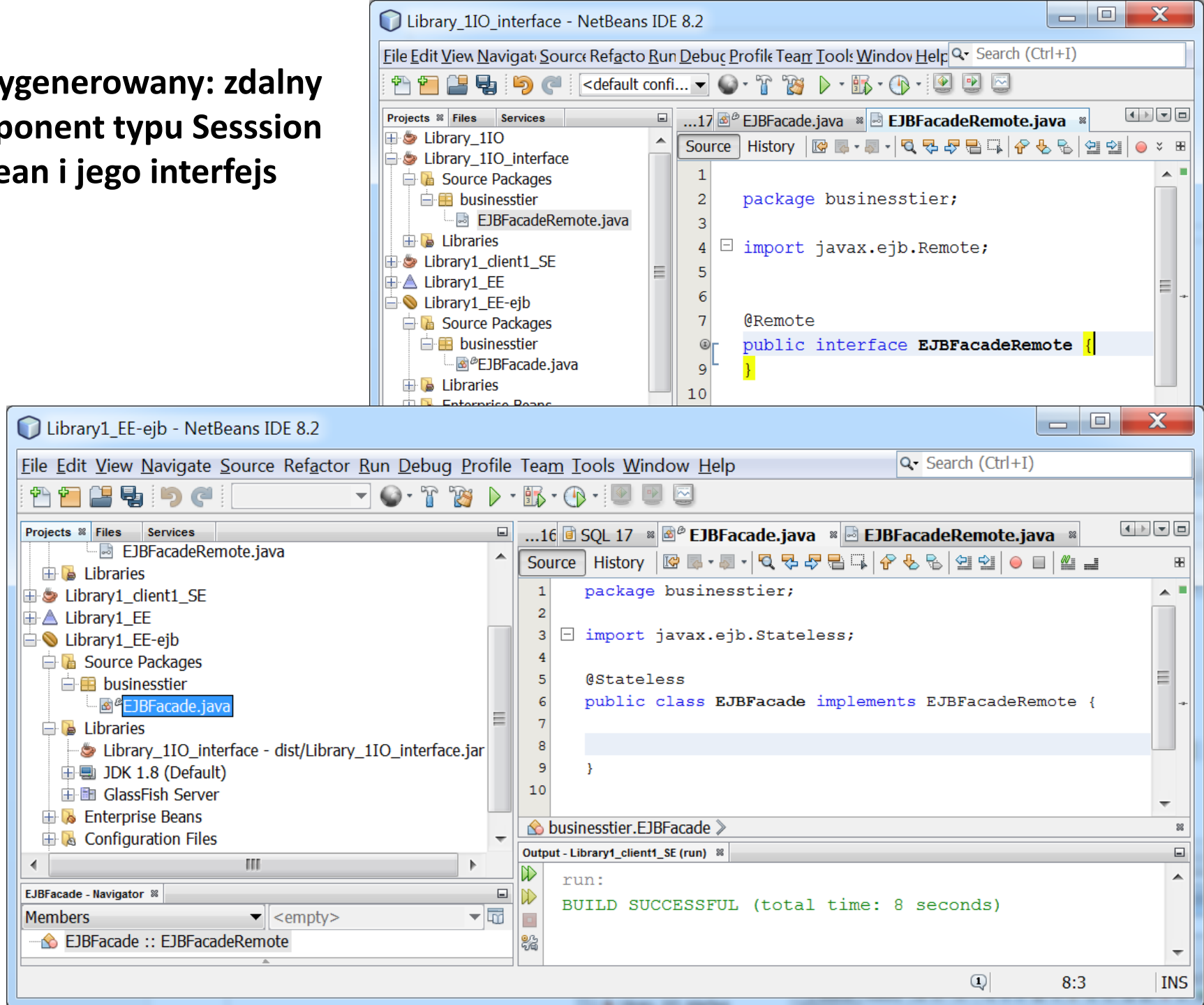
3. Projekt typu Enterprise Application



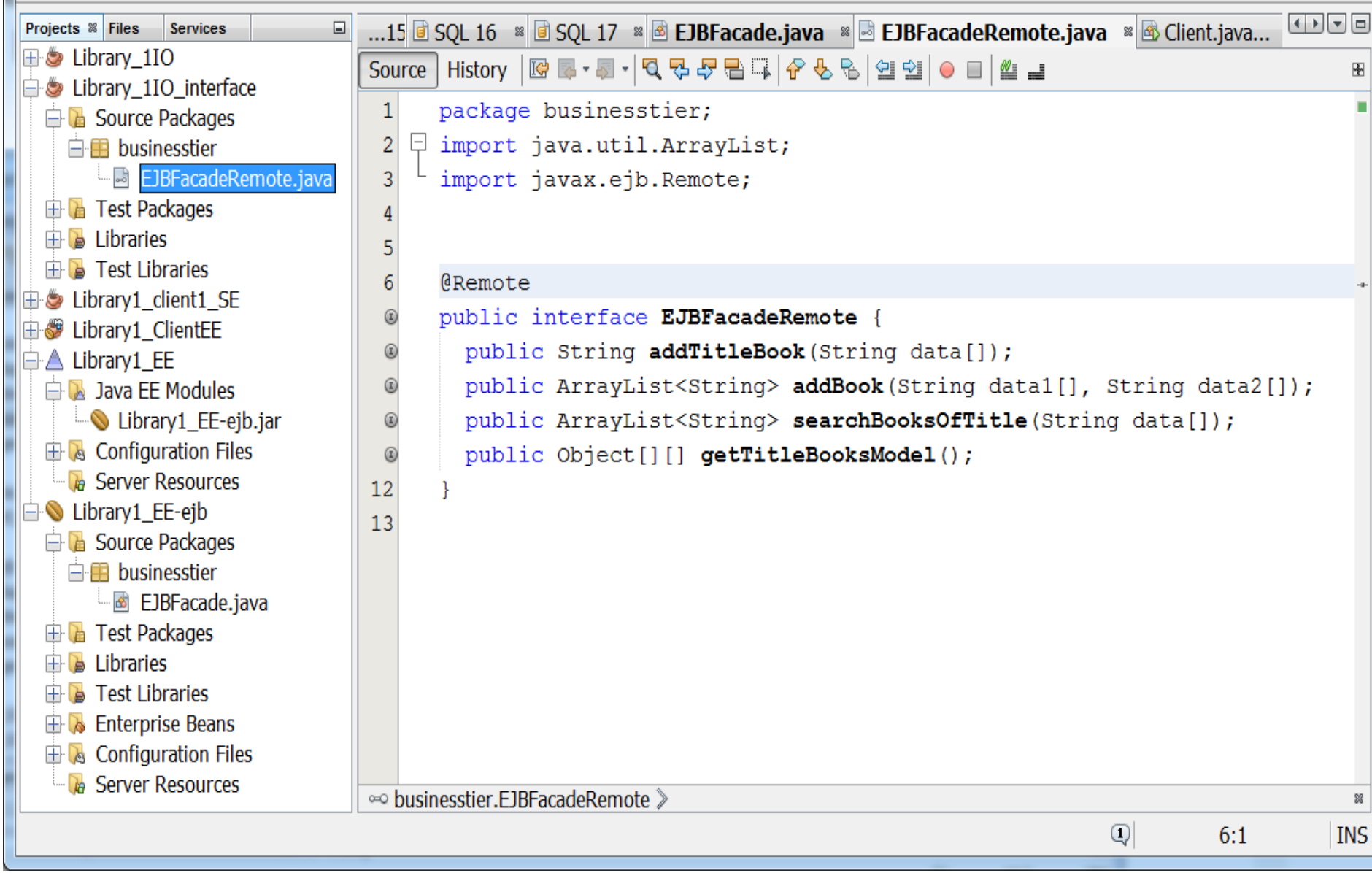
4. Wstawianie zdalnego komponentu EJB typu Session Bean do modułu EJB



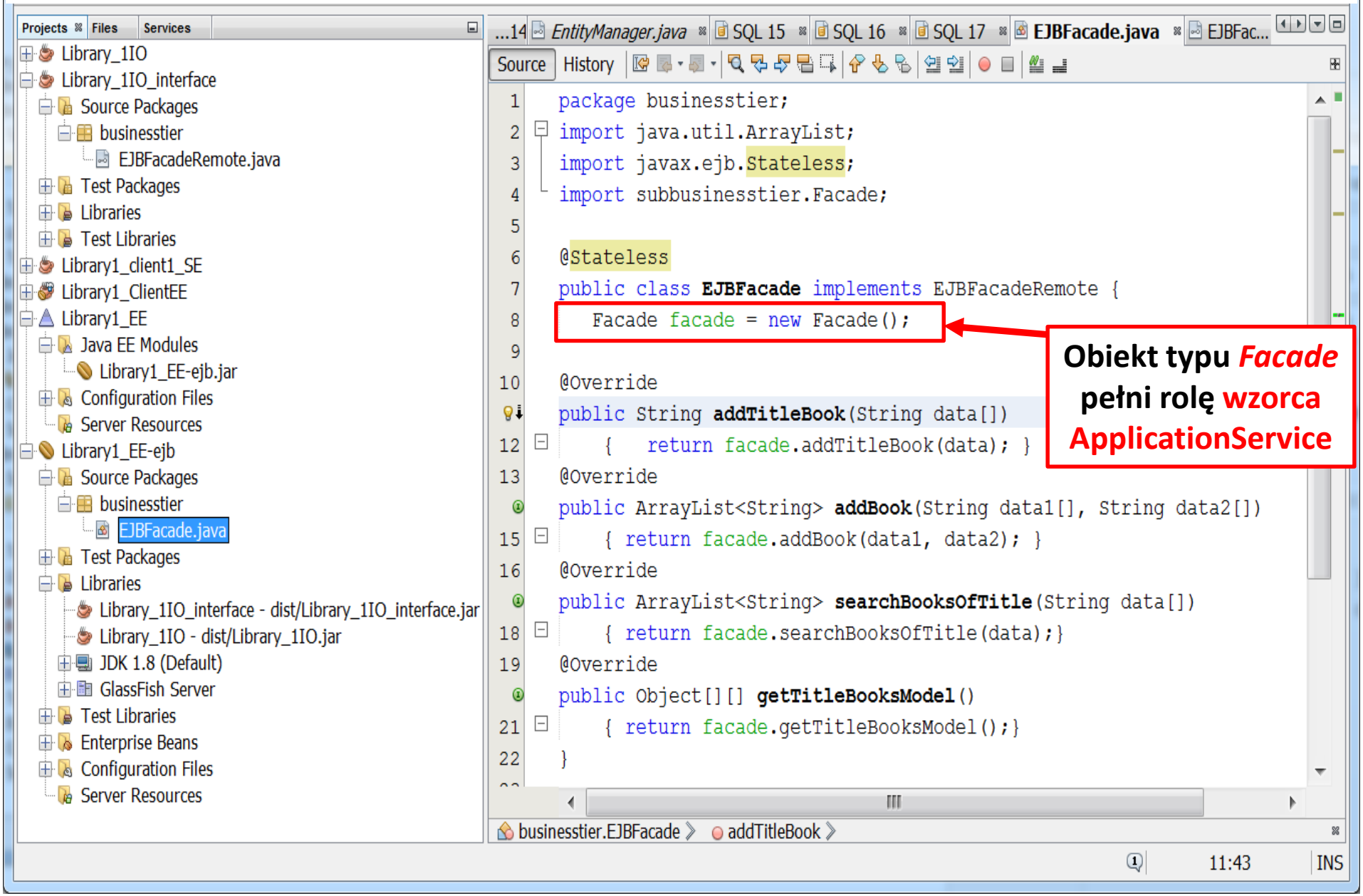
5. Wygenerowany: zdalny komponent typu Sesssion Bean i jego interfejs

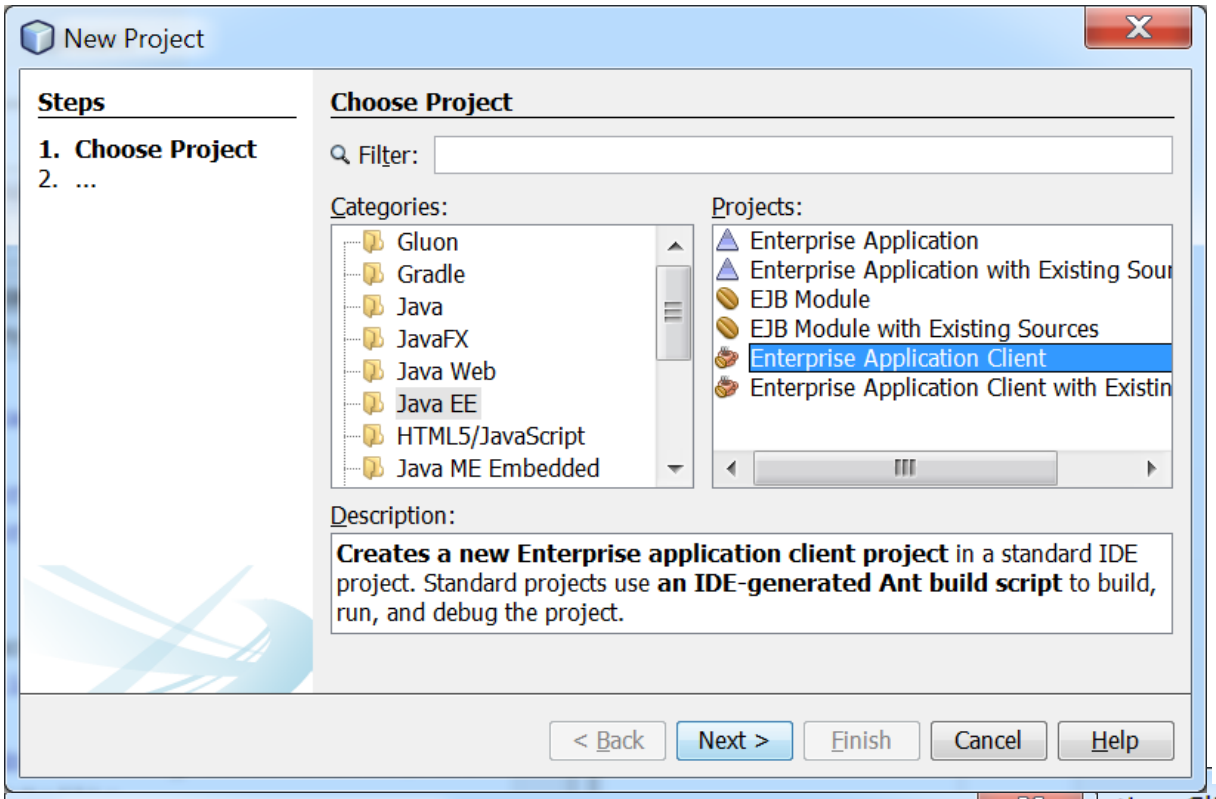


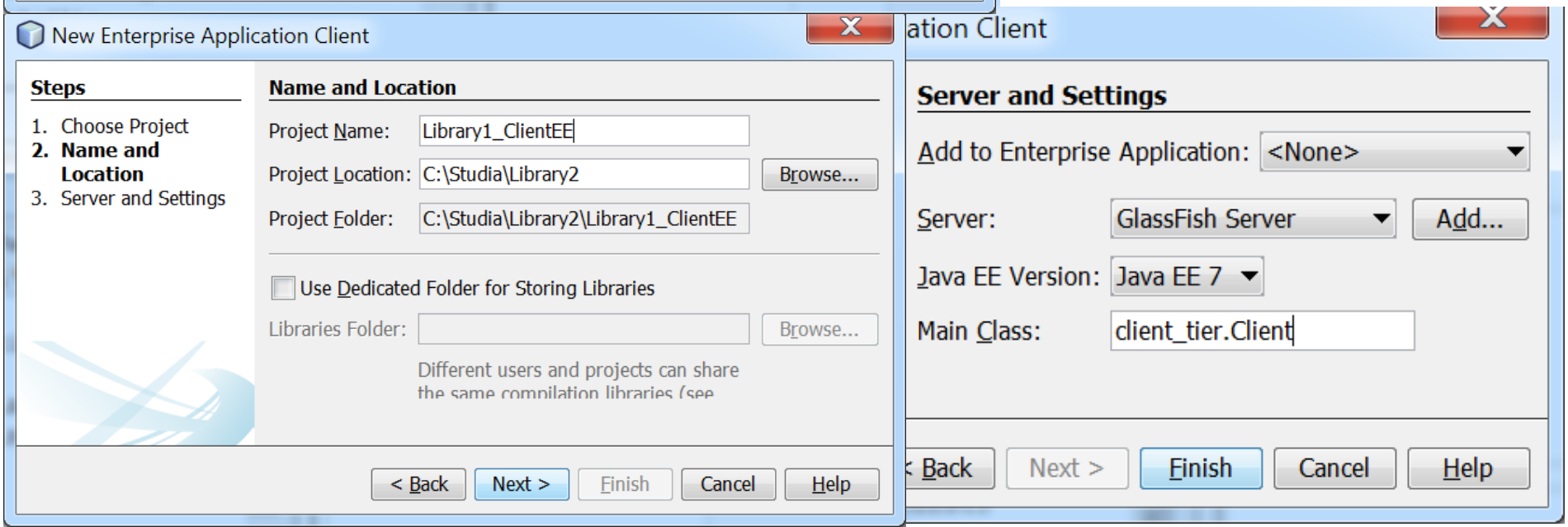
6. Deklaracja metod o zdalnym dostępie do metod logiki biznesowej klasy Facade w interfejsie komponentu typu Session Bean



7. Definicja metod o zdalnym dostępie do metod logiki biznesowej klasy Facade (wzorzec ApplicationService) w komponencie Session Bean (wzorzec SessionFacade)



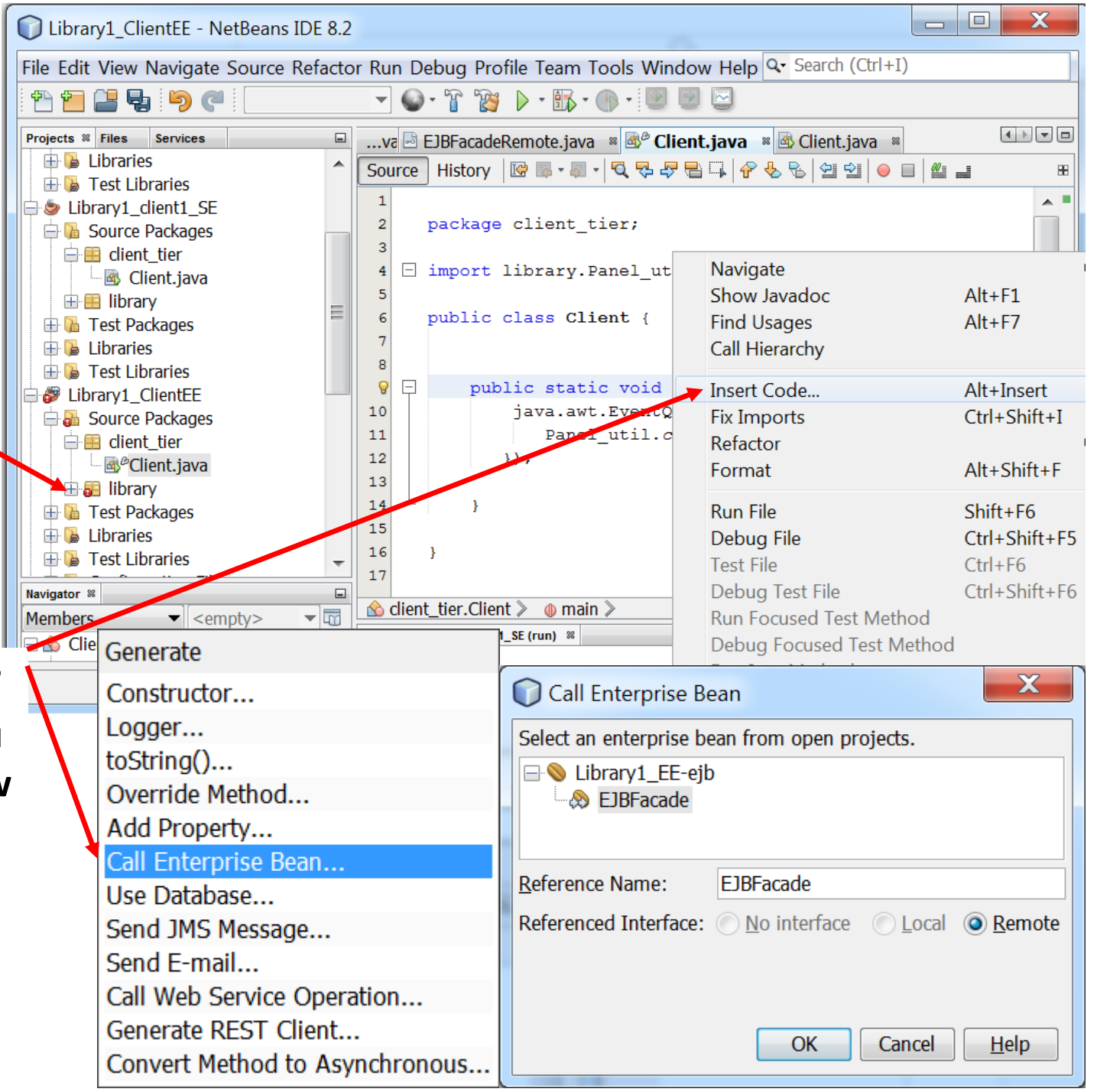




8. Wykonanie projektu typu Enterprise Application Client. Kod tego projektu będzie wykonany w oparciu o pakiet library z projektu warstwy klienta (aplikacja dwuwarstwowa). Zdefiniowano domyślną klasę Client w pakiecie client_tier w celu uniknięcia modyfikacji kodu klas w pakiecie library.

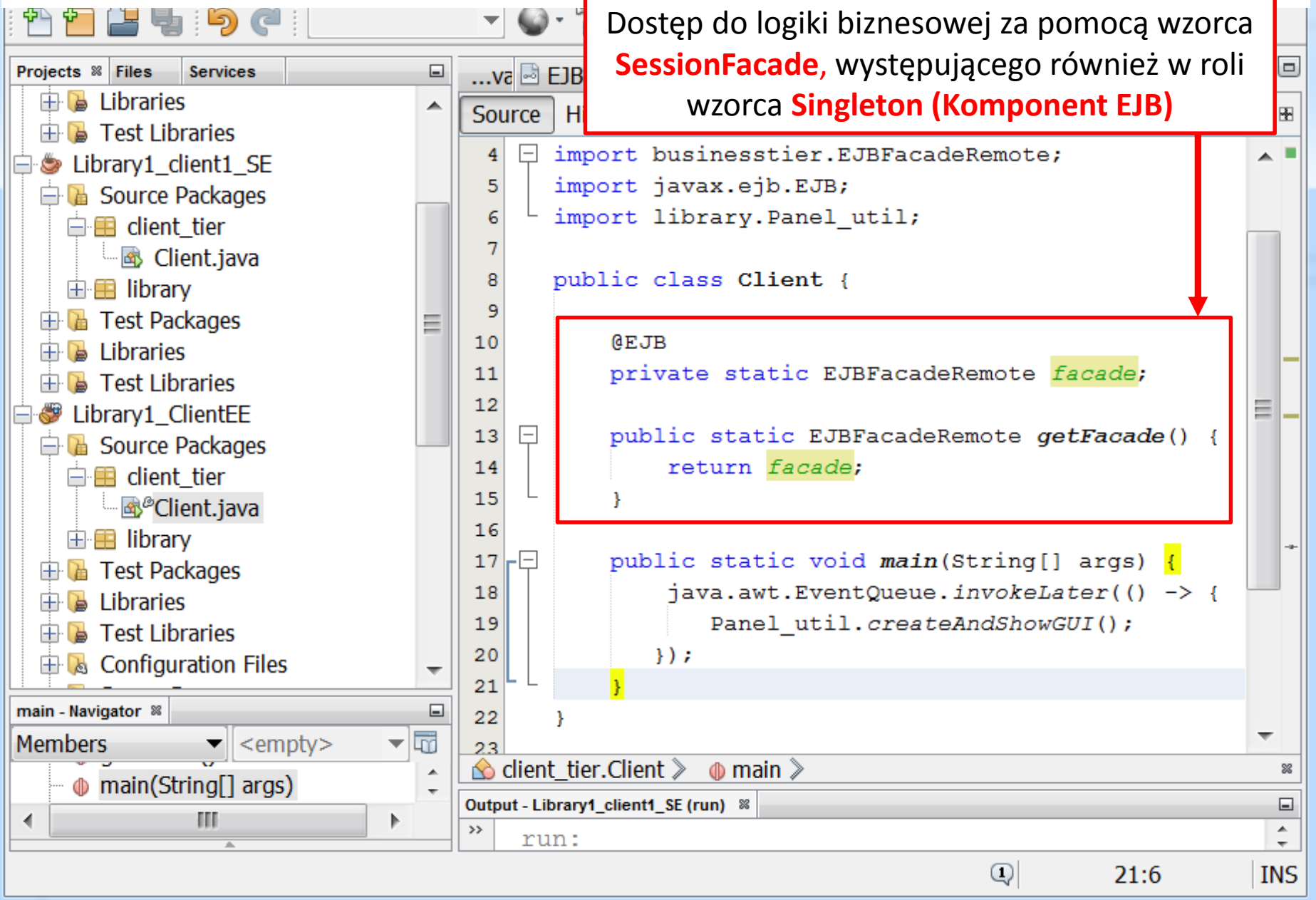
9. Skopiowanie pakietu library z aplikacji reprezentującej aplikację warstwy klienta z p.5.

10. „Wstrzyknięcie” dostępu do obiektu typu Session Bean w projekcie w typu Enterprise Application Client (w kodzie klasy Client)

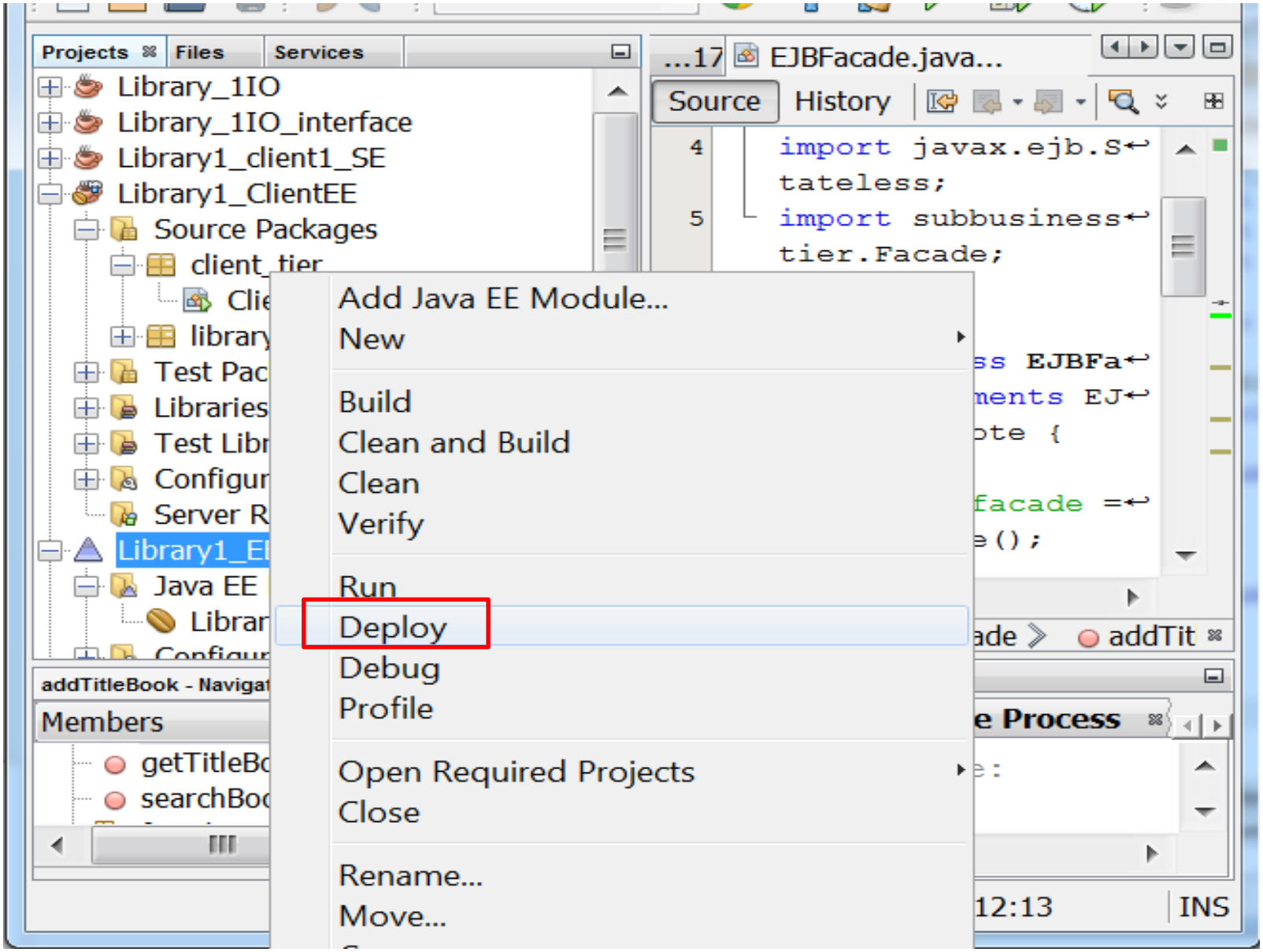


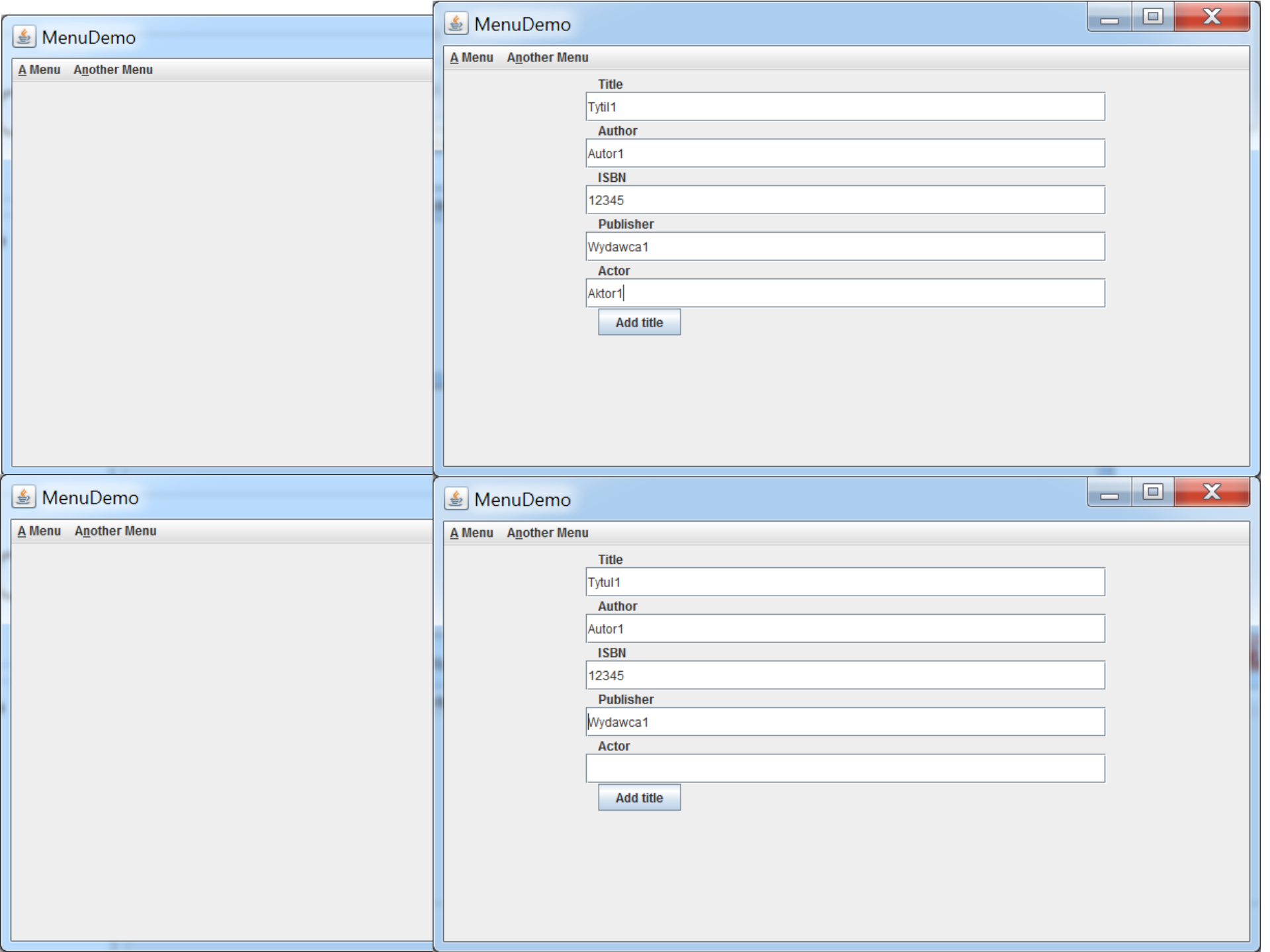
11. Wykonana aplikacja reprezentująca warstwę klienta EE

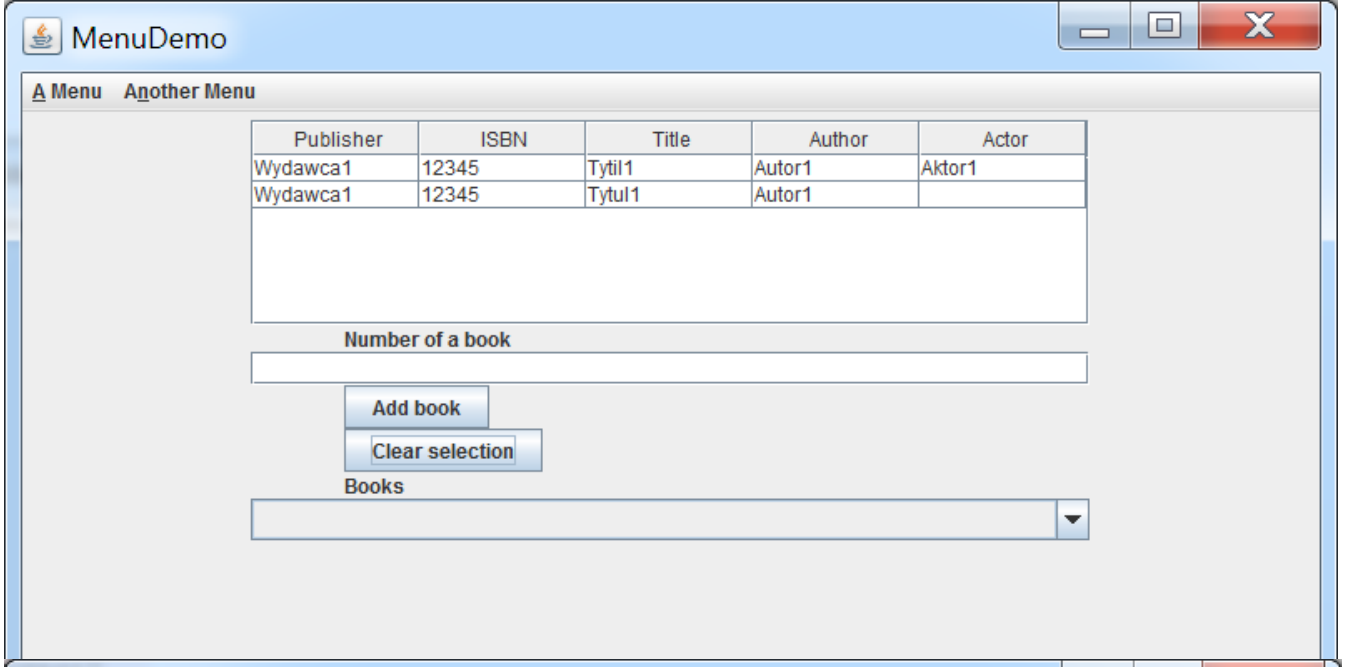
Dostęp do logiki biznesowej za pomocą wzorca **SessionFacade,** występującego również w roli wzorca **Singleton (Komponent EJB)**

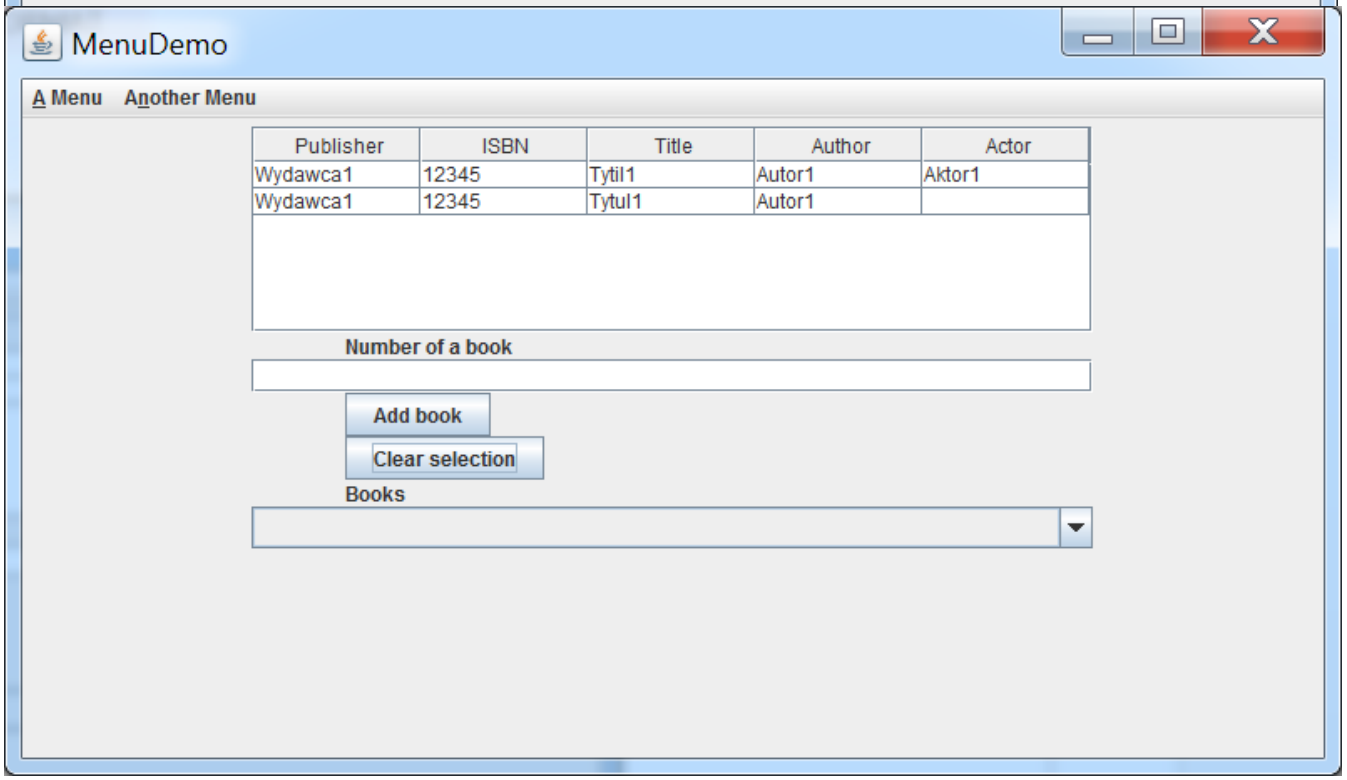


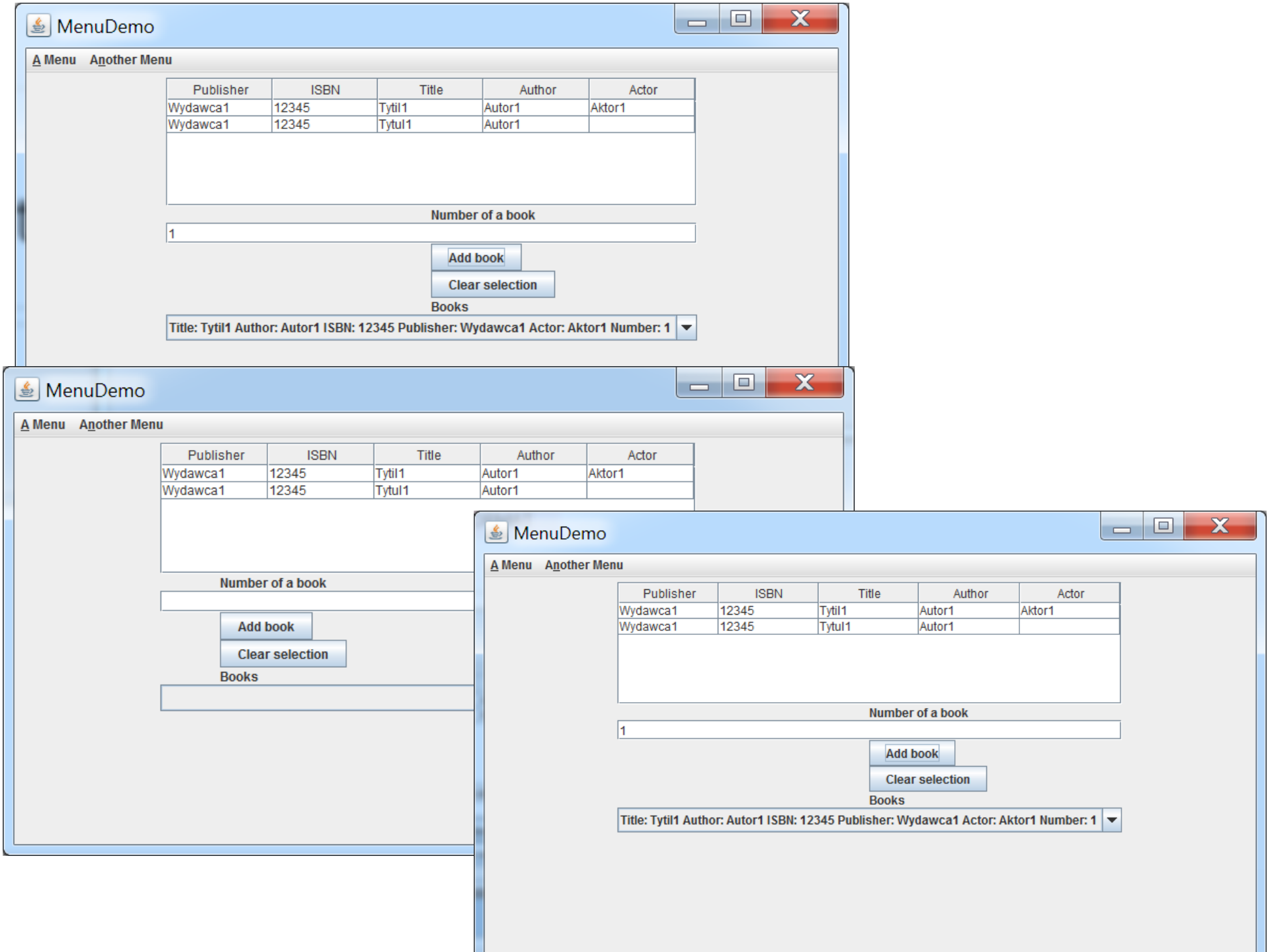
12. Uruchomienie desktopowej aplikacji wielowarstwowej typu EE: Deploy na aplikacji typu Enterprise Application, a potem Run na projekcie typu Enterprise Application Client




MenuDemo
A Menu
Another Menu
MenuDemo
A Menu
Another Menu
Title
Tytil1
Author
Autor1
ISBN
12345
Publisher
Wydawca1
Actor
Aktor1
Add title
MenuDemo
A Menu
Another Menu
MenuDemo
A Menu
Another Menu
Title
Tytul1
Author
Autor1
ISBN
12345
Publisher
Wydawca1
Actor
Add title


MenuDemo
A Menu
Another Menu
Publisher
ISBN
Title
Author
Actor
Wydawca1
12345
Tytil1
Autor1
Aktor1
Wydawca1
12345
Tytul1
Autor1
Number of a book
Add book
Clear selection
Books

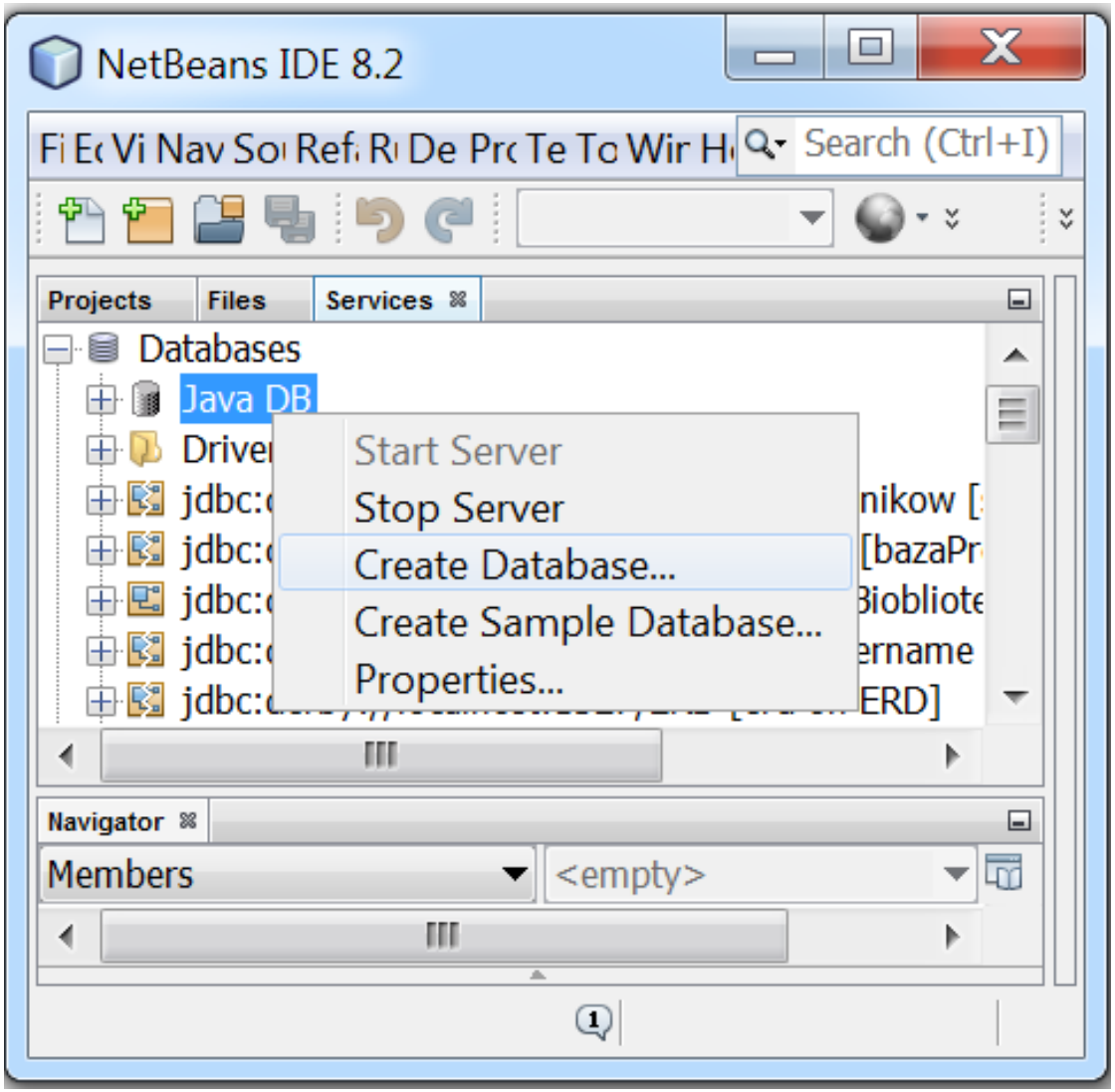

MenuDemo
A Menu
Another Menu
Publisher
ISBN
Title
Author
Actor
Wydawca1
12345
Tytil1
Autor1
Aktor1
Wydawca1
12345
Tytul1
Autor1
Number of a book
Add book
Clear selection
Books


MenuDemo
A Menu
Another Menu
Publisher
ISBN
Title
Author
Actor
Wydawca1
12345
Tytil1
Autor1
Aktor1
Wydawca1
12345
Tytul1
Autor1
Number of a book
1
Add book
Clear selection
Books
Title: Tytil1 Author: Autor1 ISBN: 12345 Publisher: Wydawca1 Actor: Aktor1 Number: 1
MenuDemo
A Menu
Another Menu
Publisher
ISBN
Title
Author
Actor
Wydawca1
12345
Tytil1
Autor1
Aktor1
Wydawca1
12345
Tytul1
Autor1
Number of a book
Add book
Clear selection
Books
MenuDemo
A Menu
Another Menu
Publisher
ISBN
Title
Author
Actor
Wydawca1
12345
Tytil1
Autor1
Aktor1
Wydawca1
12345
Tytul1
Autor1
Number of a book
1
Add book
Clear selection
Books
Title: Tytil1 Author: Autor1 ISBN: 12345 Publisher: Wydawca1 Actor: Aktor1 Number: 1

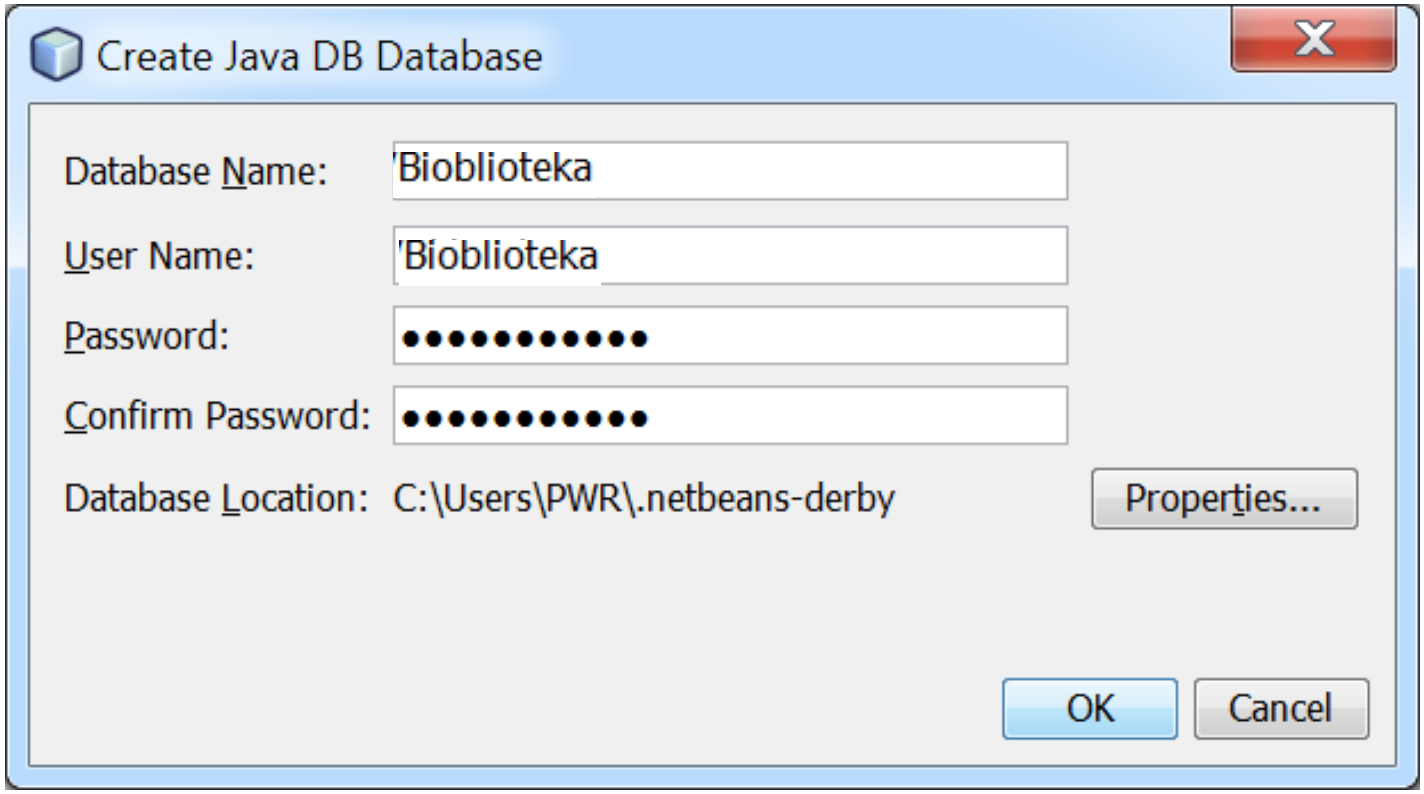
Zagadnienia

1. Wielowarstwowa architektura systemu informatycznego
2. Ocena i poprawa (refaktoryzacja) architektury wielowarstwowej systemu informatycznego
3. Wzorce projektowe stosowane przy budowie wielowarstwowej aplikacji internetowej
4. Przykład modelowania i projektowania części warstwy biznesowej z obiektami typu POJO. Wykonanie aplikacji dwuwarstwowej dla jednego użytkownika.
5. Przykłady architektury wielowarstwowej aplikacji internetowej typu EE. Wykonanie aplikacji typu EE.
   Warstwa biznesowa: komponenty typu EJB + obiekty POJO
6. Warstwa zasobów (EIS)- baza danych w systemie baz danych Derby

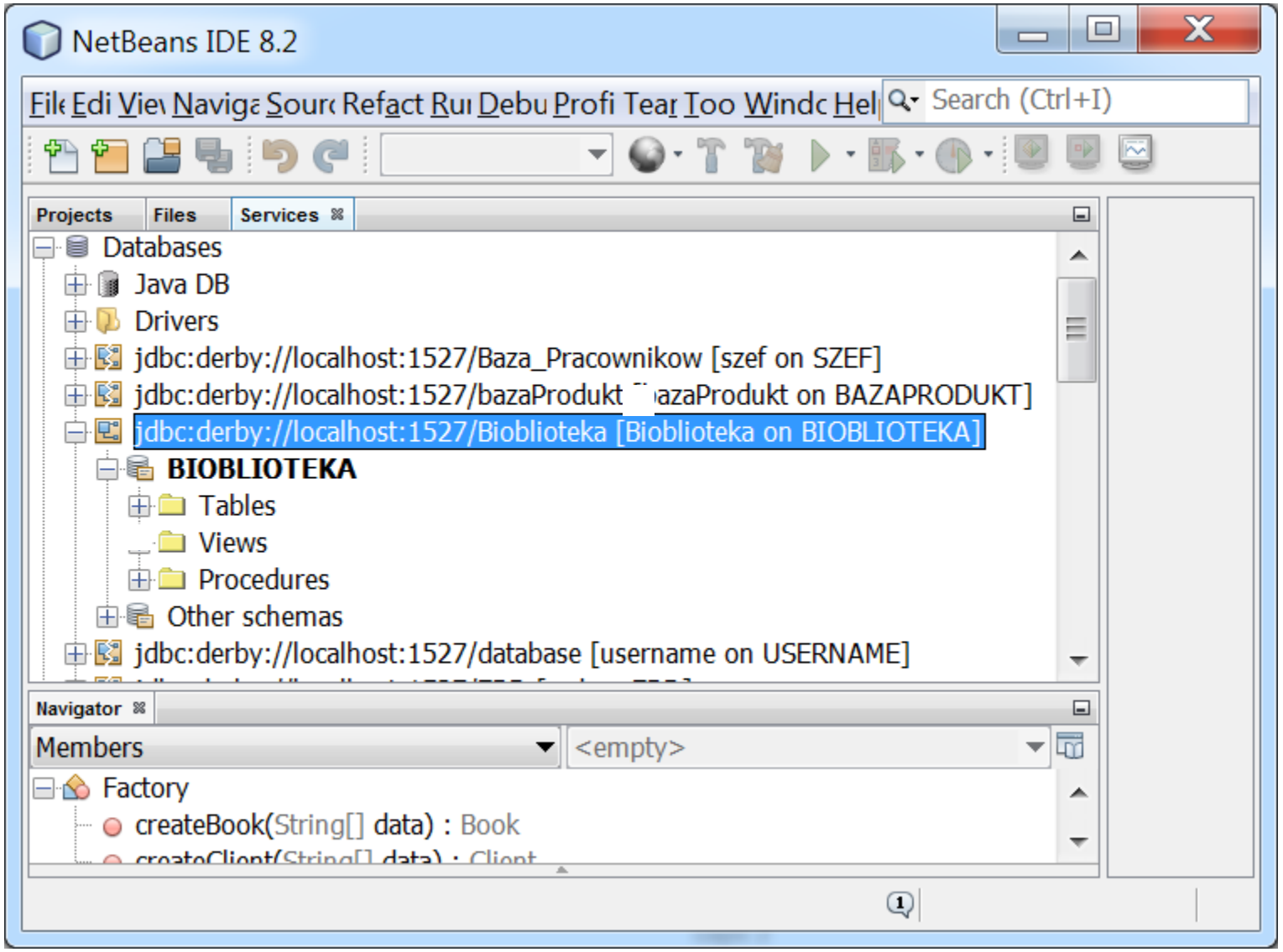
1. Zakładanie pustej bazy danych dla systemu baz danych Derby



2. Zakładanie pustej bazy danych Bioblioteka w systemie baz danych Derby



3. Utworzona pusta baza danych



Zagadnienia

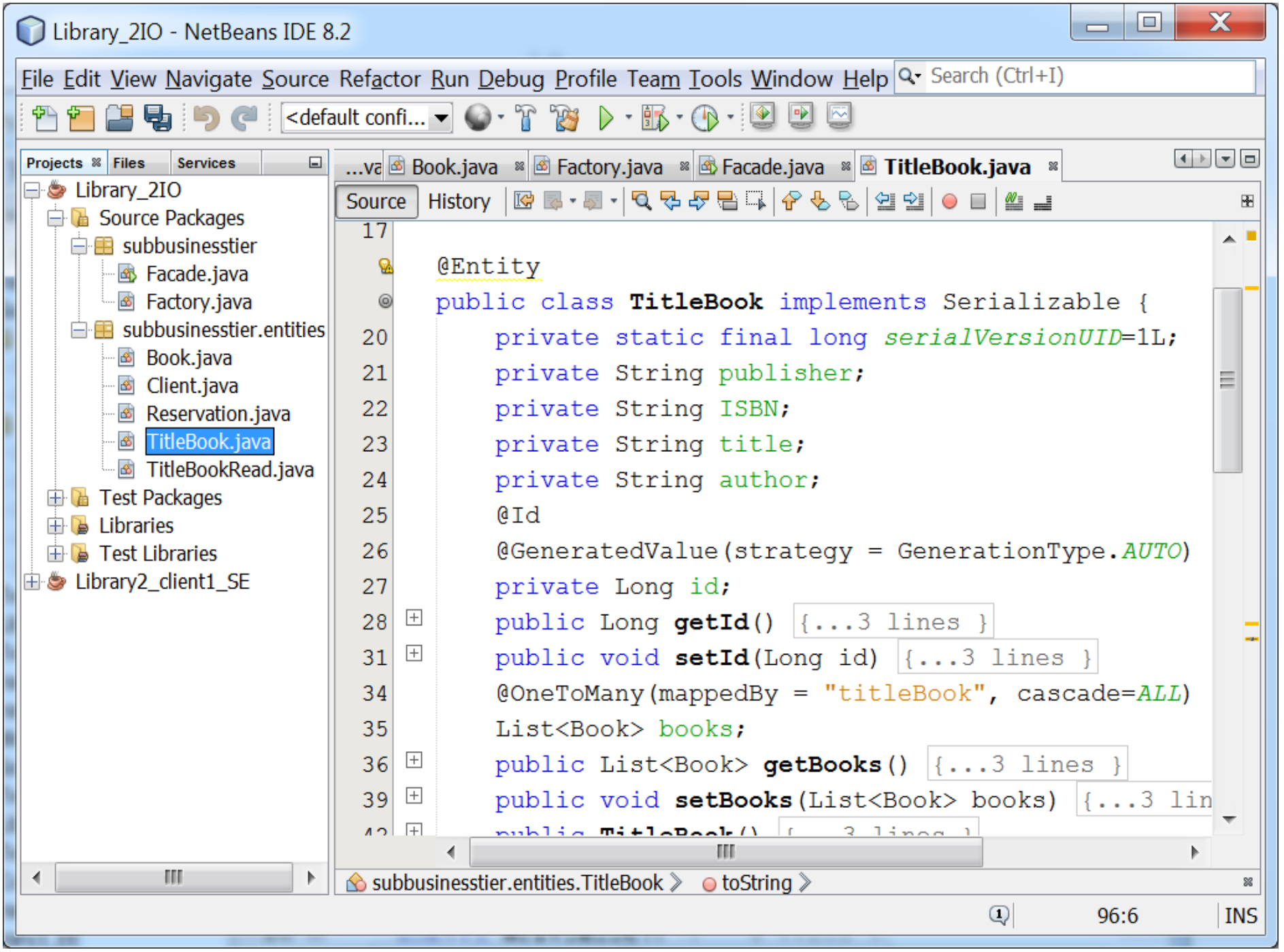
1. Wielowarstwowa architektura systemu informatycznego
2. Ocena i poprawa (refaktoryzacja) architektury wielowarstwowej systemu informatycznego
3. Wzorce projektowe stosowane przy budowie wielowarstwowej aplikacji internetowej
4. Przykład modelowania i projektowania części warstwy biznesowej z obiektami typu POJO. Wykonanie aplikacji dwuwarstwowej dla jednego użytkownika.
5. Przykłady architektury wielowarstwowej aplikacji internetowej typu EE. Wykonanie aplikacji typu EE. Warstwa biznesowa: komponenty typu EJB + obiekty POJO
6. Warstwa zasobów (EIS)- baza danych w systemie baz danych Derby
7. Utworzenie obiektowego modelu danych do utrwalania ORM

```java
@Entity
public class TitleBook implements Serializable {
    private static final long serialVersionUID=1L;
    private String publisher;
    private String ISBN;
    private String title;
    private String author;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    public Long getId() {...3 lines }
    public void setId(Long id) {...3 lines }
    @OneToMany(mappedBy = "titleBook", cascade=ALL)
    List<Book> books;
    public List<Book> getBooks() {...3 lines }
    public void setBooks(List<Book> books) {...3 lines }
    public TitleBook() {...3 lines }
    public String getPublisher() {...3 lines }
    public void setPublisher(String publisher) {...3 lines }
    public String getISBN() {...3 lines }
    public void setISBN(String ISBN) {...3 lines }
    public String getTitle() {...3 lines }
    public void setTitle(String title) {...3 lines }
    public String getAuthor() {...3 lines }
    public void setAuthor(String author) {...3 lines }
    public String getActor() {...3 lines }
    public void setActor(String val) {...2 lines }
```

Zmiana typu klas danych na typ @Entity w projekcie warstwy biznesowej – dodano adnotacje, nowy atrybut Id. Należy zestandaryzować nazwy metod dostępu do składowych klasy typu Entity.

```
package subbusinesstier.entities;

import ...

@Entity
public class TitleBookRead extends TitleBook {
private static final long serialVersionUID=1L;
    private String actor;

    @Override
    public String getActor() {...3 lines }

    @Override
    public void setActor(String actor) {...3 lines }

    @Override
    public String toString() {...5 lines }

}
```

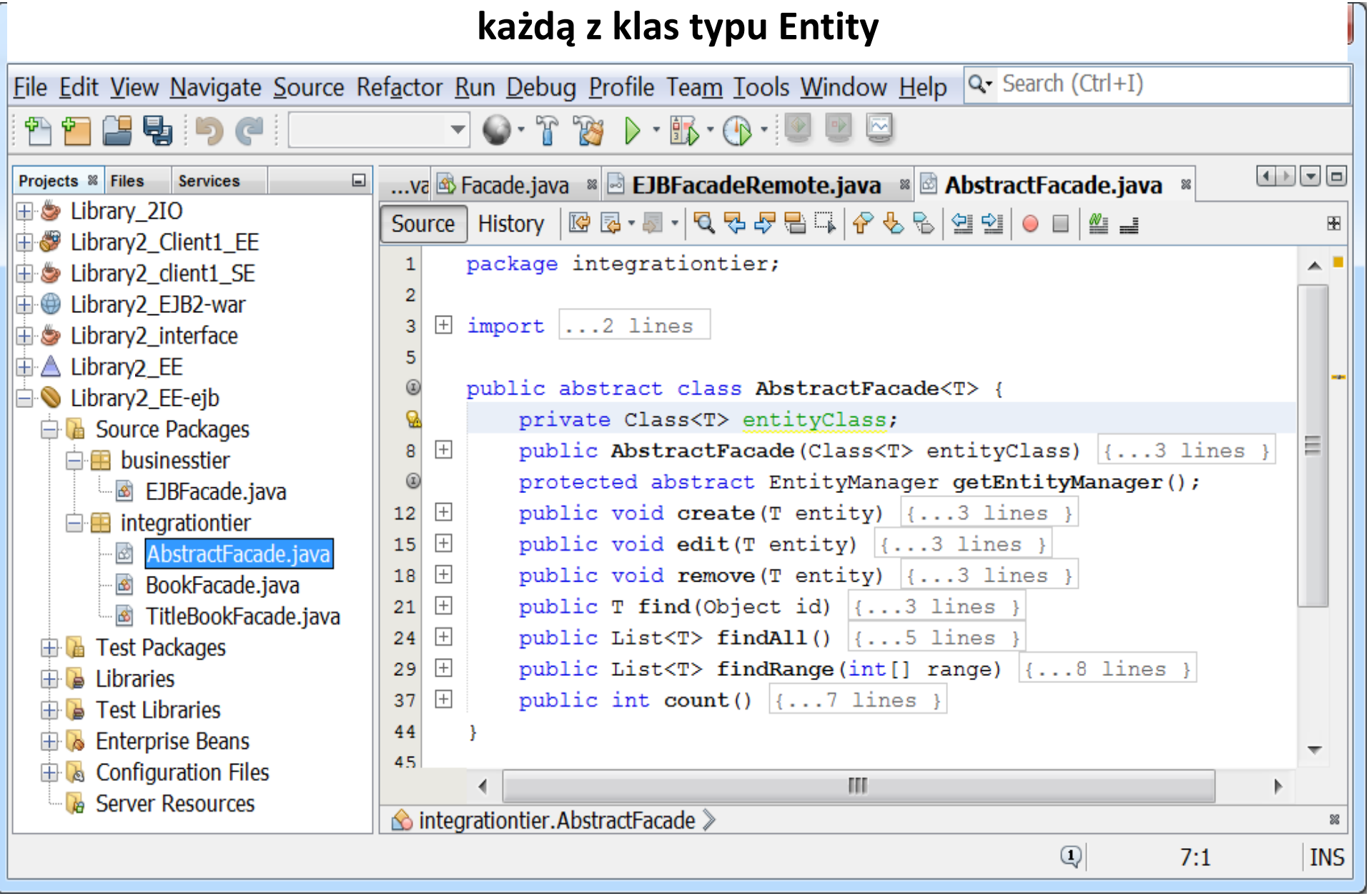
```
@Entity
public class Book implements Serializable {
private static final long serialVersionUID=1L;
    private int number;
    @ManyToOne
    private TitleBook titleBook;
   // @OneToMany (mappedBy="book")
    @Transient
    private List<Reservation> reservations;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    public Long getId() {...3 lines }
    public void setId(Long id) {...3 lines }

    public List<Reservation> getReservations() {...3 lines }
    public void setReservations(List<Reservation> reservations) {...3 lines }
```


Library_2IO - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default confi...
Projects
Files
Services
Library_2IO
Source Packages
subbusinesstier
Facade.java
Factory.java
subbusinesstier.entities
Book.java
Client.java
Reservation.java
TitleBook.java
TitleBookRead.java
Test Packages
Libraries
Test Libraries
Library2_client1_SE
Book.java
Factory.java
Facade.java
TitleBook.java
Source
History
17
@Entity
public class TitleBook implements Serializable {
20 private static final long serialVersionUID=1L;
21 private String publisher;
22 private String ISBN;
23 private String title;
24 private String author;
25 @Id
26 @GeneratedValue(strategy = GenerationType.AUTO)
27 private Long id;
28 public Long getId() {...3 lines }
31 public void setId(Long id) {...3 lines }
34 @OneToMany(mappedBy = "titleBook", cascade=ALL)
35 List<Book> books;
36 public List<Book> getBooks() {...3 lines }
39 public void setBooks(List<Book> books) {...3 lin
subbusinesstier.entities.TitleBook
toString
96:6
INS

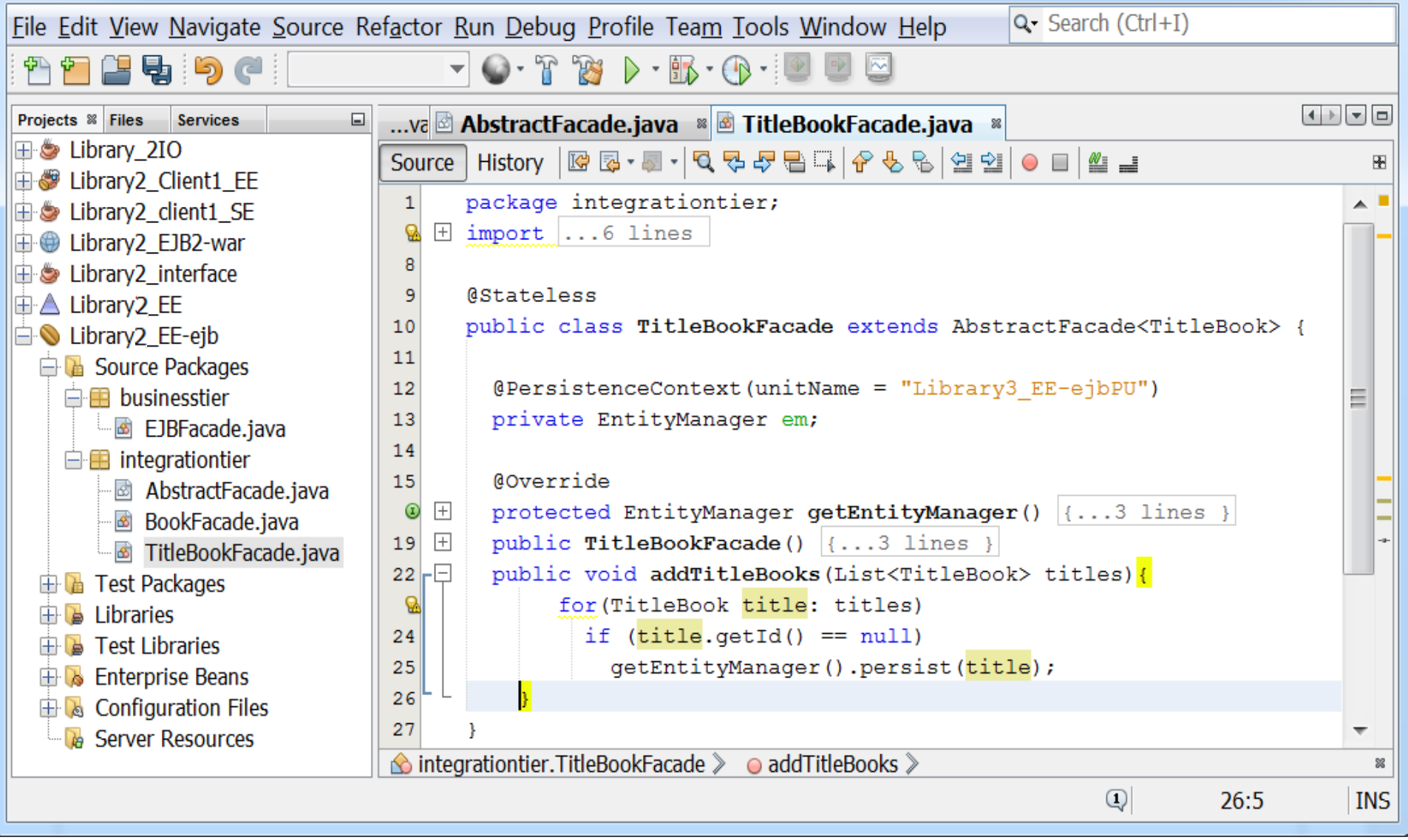
Zagadnienia

1. Wielowarstwowa architektura systemu informatycznego
2. Ocena i poprawa (refaktoryzacja) architektury wielowarstwowej systemu informatycznego
3. Wzorce projektowe stosowane przy budowie wielowarstwowej aplikacji internetowej
4. Przykład modelowania i projektowania części warstwy biznesowej z obiektami typu POJO. Wykonanie aplikacji dwuwarstwowej dla jednego użytkownika.
5. Przykłady architektury wielowarstwowej aplikacji internetowej typu EE. Wykonanie aplikacji typu EE. Warstwa biznesowa: komponenty typu EJB + obiekty POJO
6. Warstwa zasobów (EIS)- baza danych w systemie baz danych Derby
7. Utworzenie obiektowego modelu danych do utrwalania ORM
8. Warstwa integracji. Zastosowanie wzorca projektowego typu *Domain Store* w technologii JPA (Java Persistence) na platformie Java EE

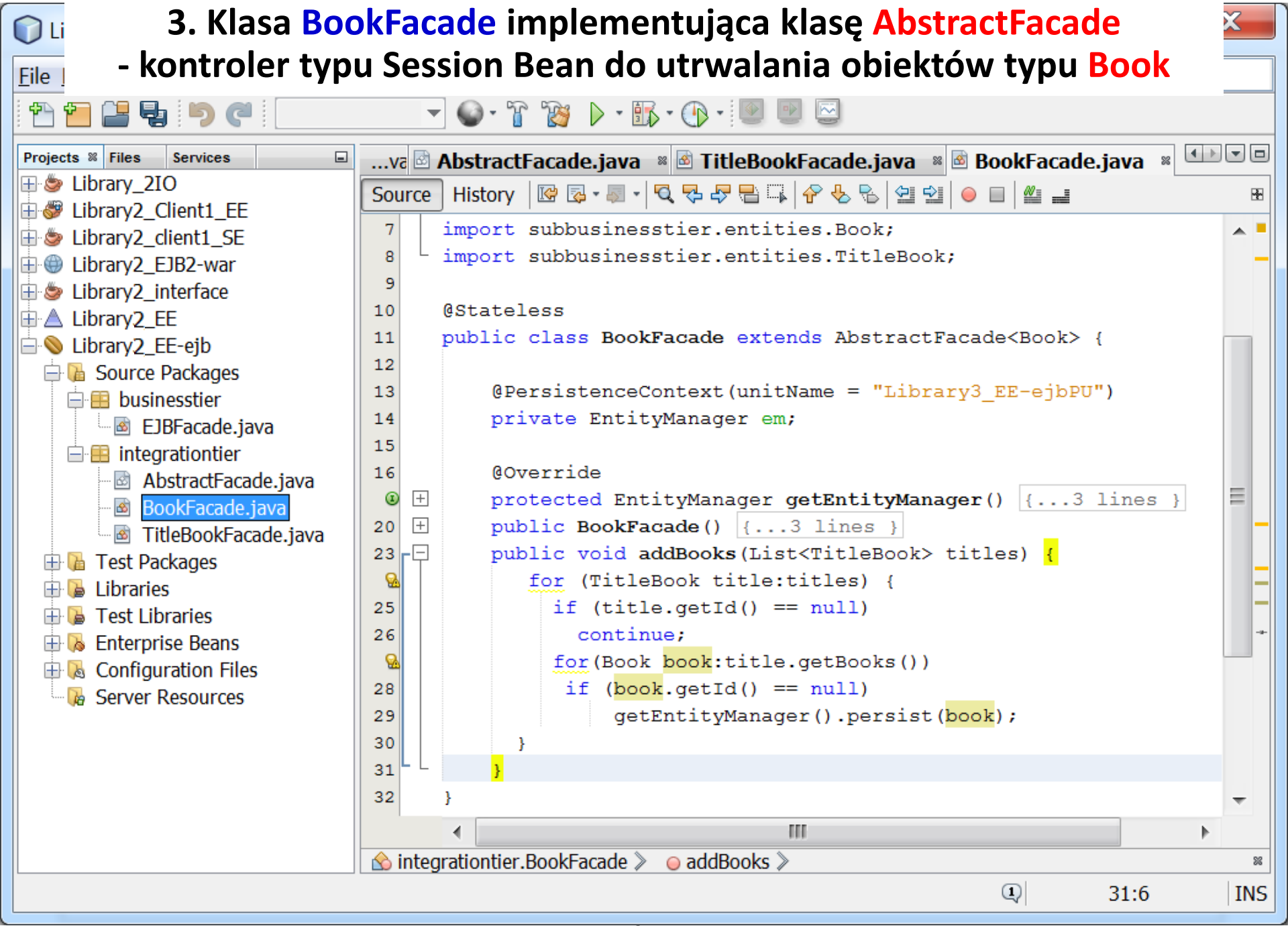
1. Klasa abstrakcyjna generyczna AbstractFacade z definicją uniwersalnych metod obsługujących transakcje JPA - parametr T może być zastąpiony każdą z klas typu Entity



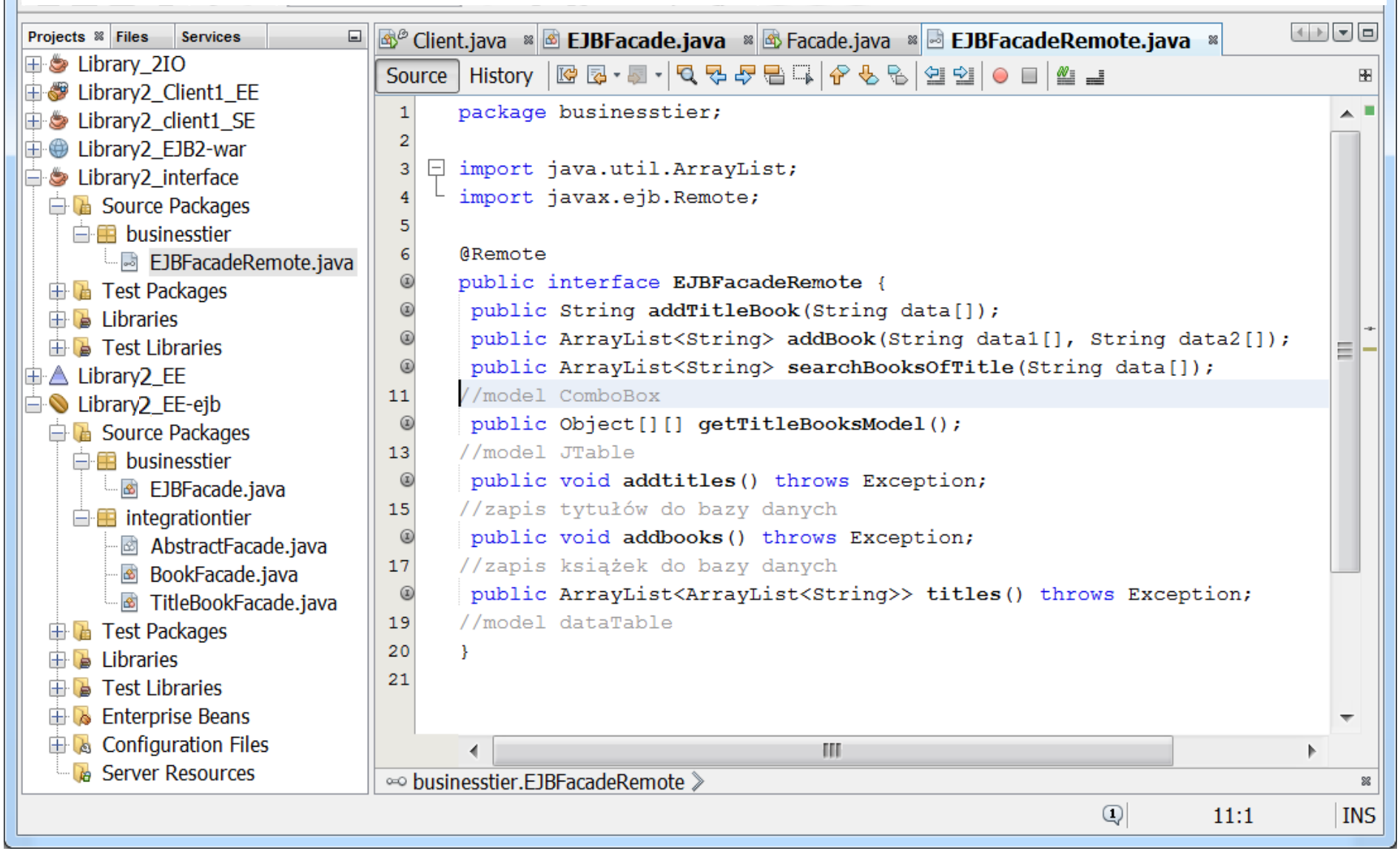
2. Klasa TitleBookFacade implementująca klasę AbstractFacade - kontroler typu Session Bean do utrwalania obiektów typu TitleBook i TitleBookRead

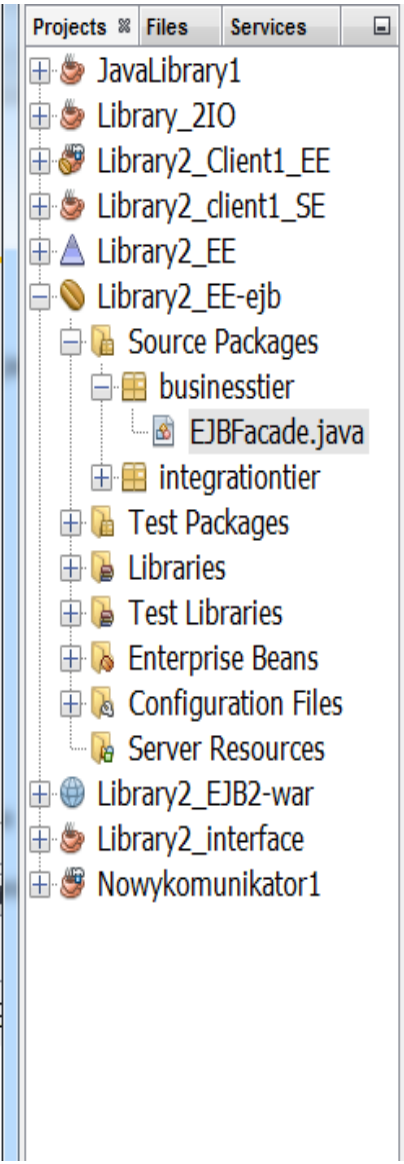


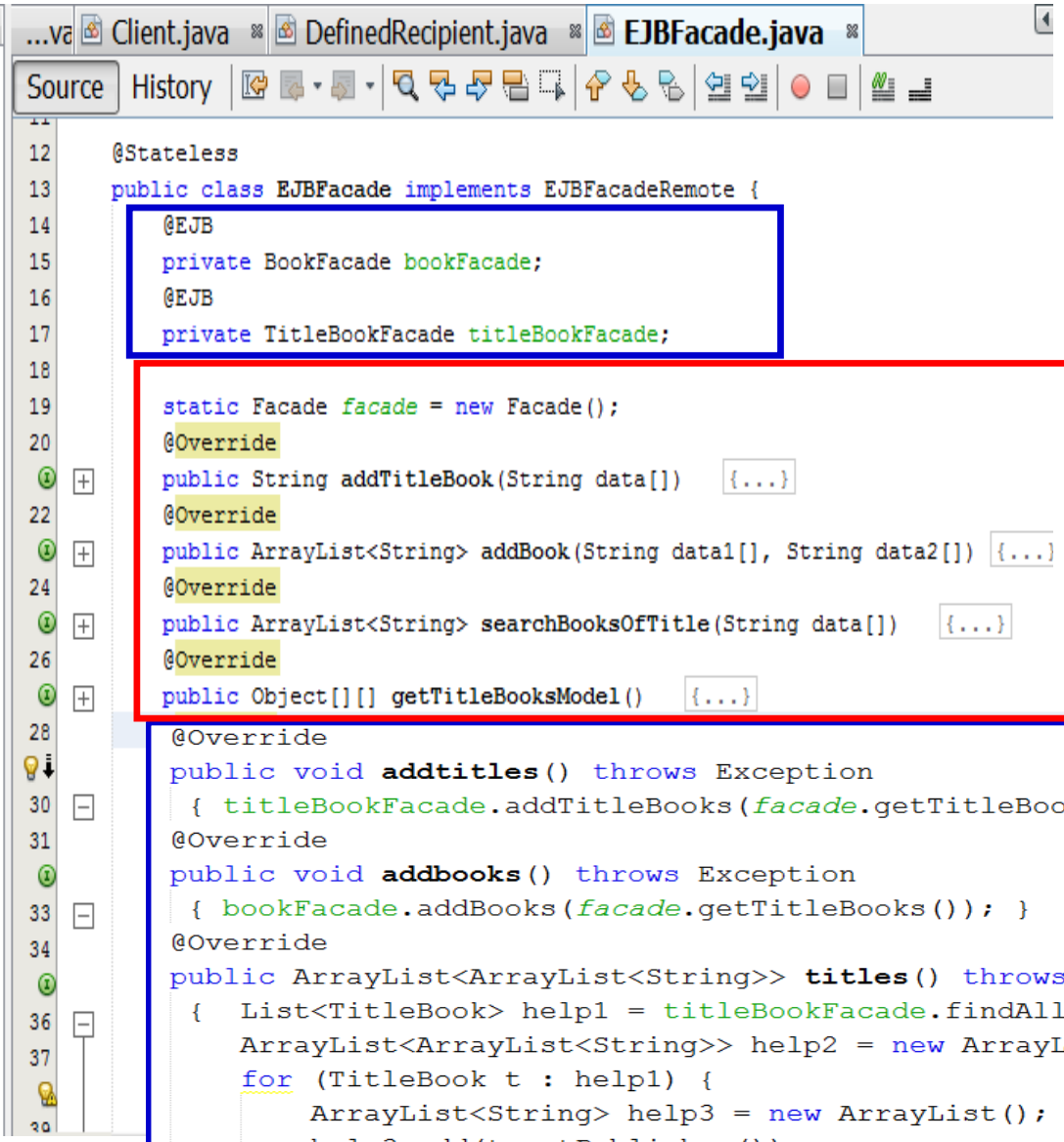
3. Klasa BookFacade implementująca klasę AbstractFacade - kontroler typu Session Bean do utrwalania obiektów typu Book



5. Uzupełnienie deklaracji metod o zdalnym dostępie do wybranych metod komponentów do utrwalania obiektów typu Entity w interfejsie komponentu typu Session Bean



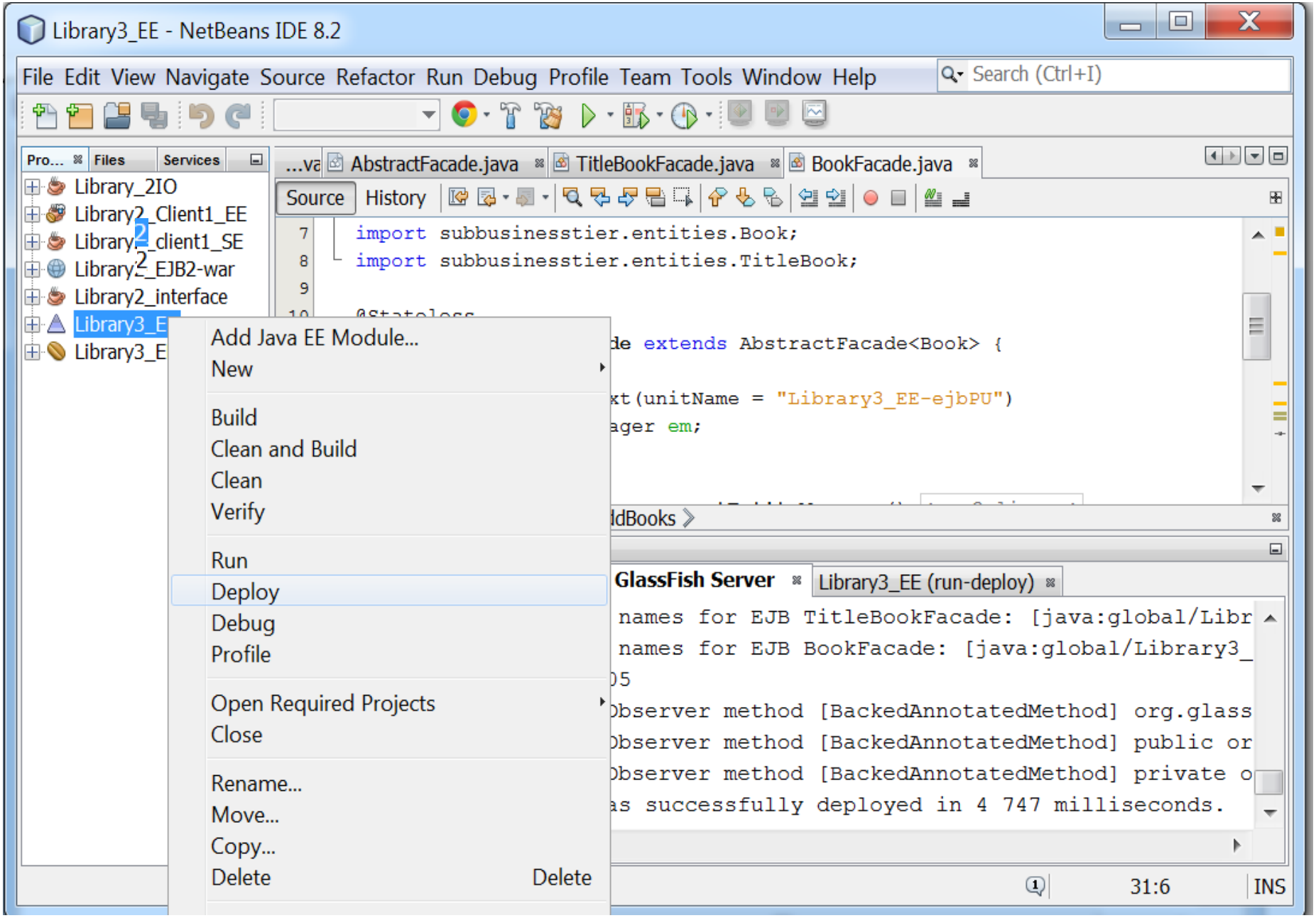




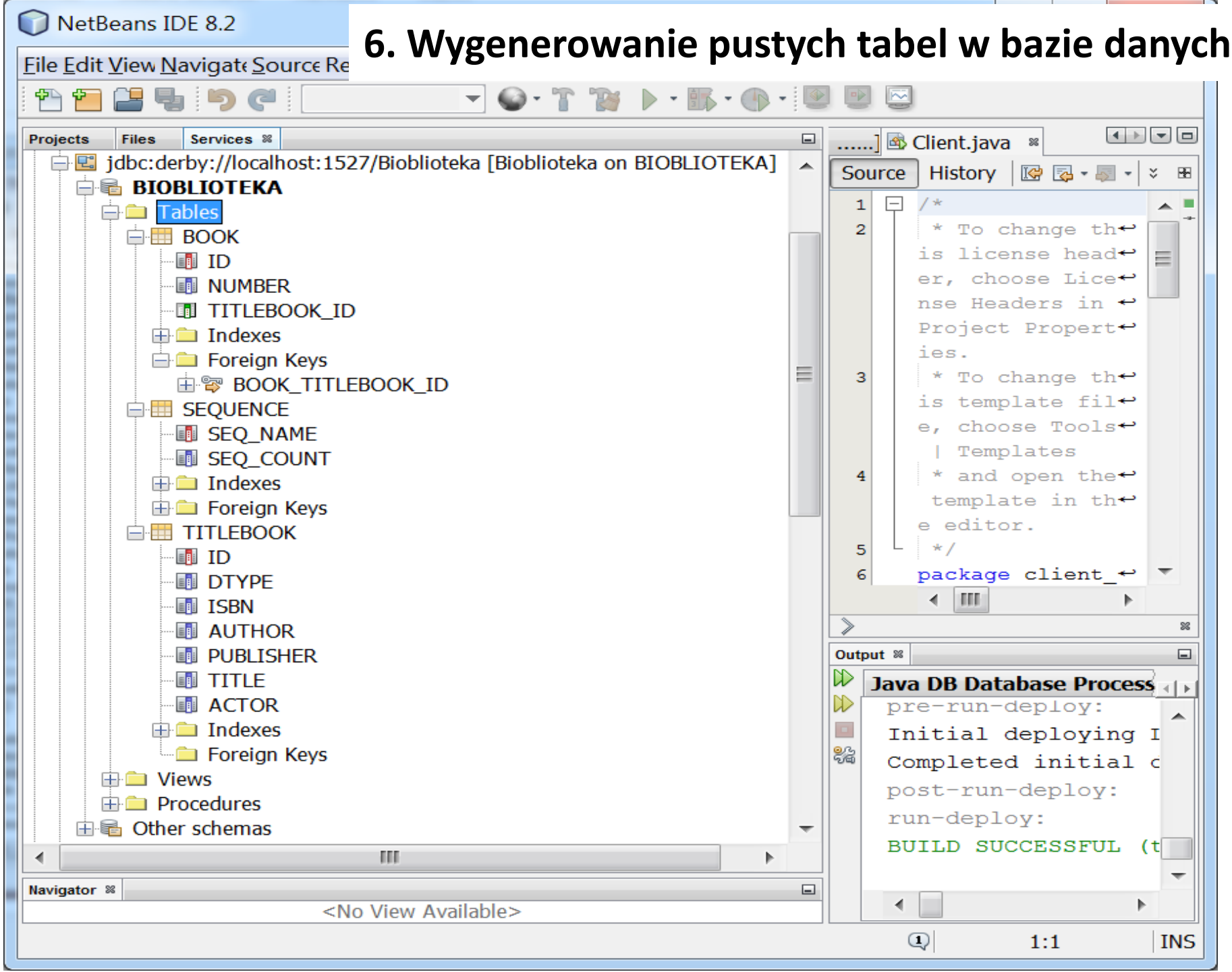

```
12  @Stateless
13  public class EJBFacade implements EJBFacadeRemote {
14      @EJB
15      private BookFacade bookFacade;
16      @EJB
17      private TitleBookFacade titleBookFacade;
18
19      static Facade facade = new Facade();
20      @Override
        public String addTitleBook(String data[])   {...}
22      @Override
        public ArrayList<String> addBook(String data1[], String data2[]) {...}
24      @Override
        public ArrayList<String> searchBooksOfTitle(String data[])   {...}
26      @Override
        public Object[][] getTitleBooksModel()   {...}
28      @Override
        public void addtitles() throws Exception
30       { titleBookFacade.addTitleBooks(facade.getTitleBooks()); }
31      @Override
        public void addbooks() throws Exception
33       { bookFacade.addBooks(facade.getTitleBooks()); }
34      @Override
        public ArrayList<ArrayList<String>> titles() throws Exception
36       {  List<TitleBook> help1 = titleBookFacade.findAll();
37          ArrayList<ArrayList<String>> help2 = new ArrayList();
            for (TitleBook t : help1) {
                ArrayList<String> help3 = new ArrayList();
                help3.add(t.getPublisher());
                help3.add(t.getISBN());
                help3.add(t.getTitle());
                help3.add(t.getAuthor());
                help3.add(t.getActor());
                help2.add(help3);   }
            return help2; }
```

4. Klasa EJBFacade - główny komponent warstwy biznesowej udostępnia metody logiki biznesowej i zarządza komponentami do utrwalania obiektów typu Entity.

5. Uruchomienie desktopowej aplikacji wielowarstwowej typu EE



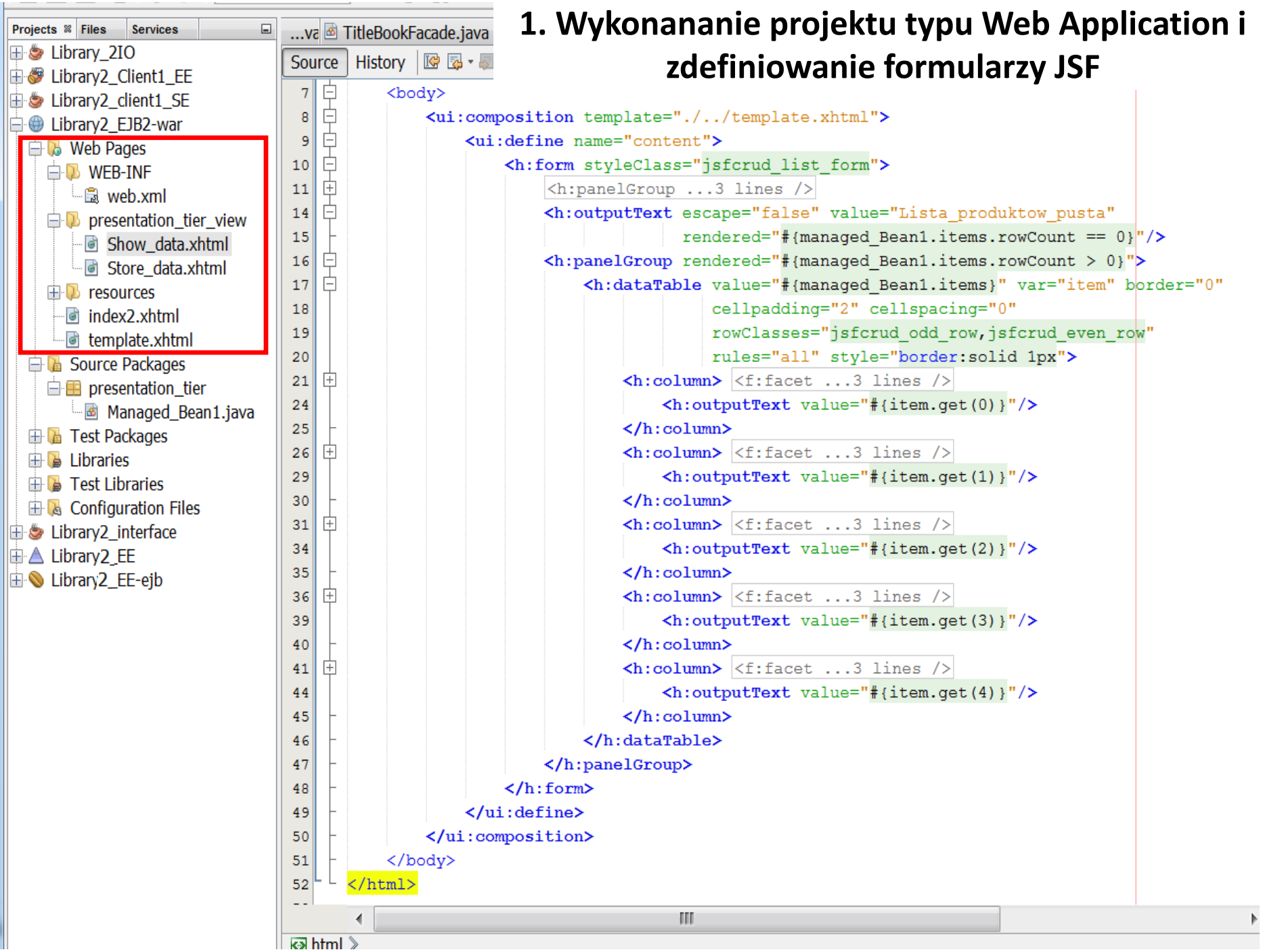
6. Wygenerowanie pustych tabel w bazie danych



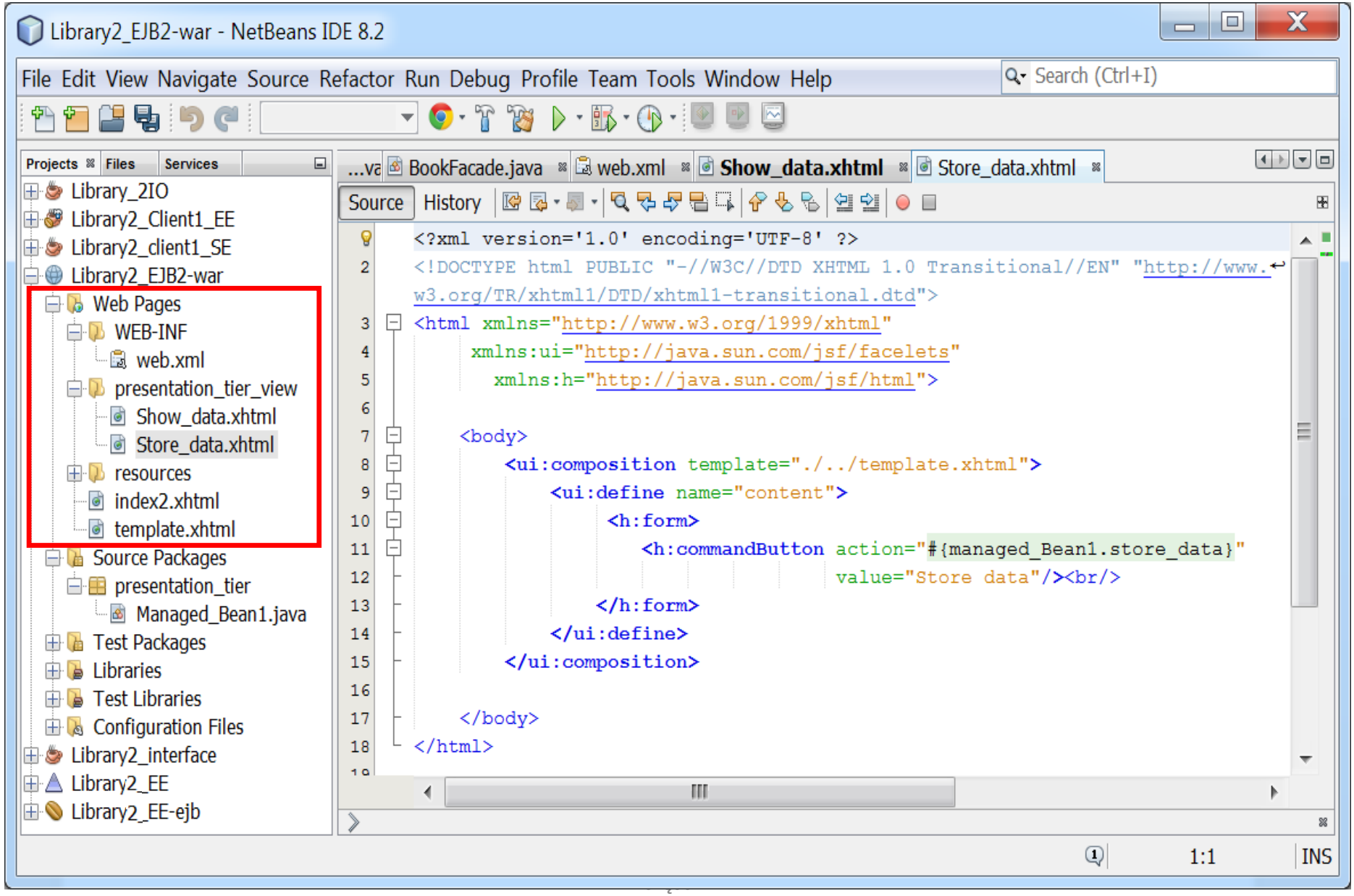
Zagadnienia

1. Wielowarstwowa architektura systemu informatycznego
2. Ocena i poprawa (refaktoryzacja) architektury wielowarstwowej systemu informatycznego
3. Wzorce projektowe stosowane przy budowie wielowarstwowej aplikacji internetowej
4. Przykład modelowania i projektowania części warstwy biznesowej z obiektami typu POJO. Wykonanie aplikacji dwuwarstwowej.
5. Przykłady architektury wielowarstwowej aplikacji internetowej typu EE. Wykonanie aplikacji typu EE. Warstwa biznesowa: komponenty typu EJB + obiekty POJO
6. Warstwa zasobów (EIS)- baza danych w systemie baz danych Derby
7. Utworzenie obiektowego modelu danych do utrwalania ORM
8. Warstwa integracji. Zastosowanie wzorca projektowego typu *Domain Store* w technologii JPA (Java Persistence) na platformie Java EE
9. Warstwa prezentacji - JSF

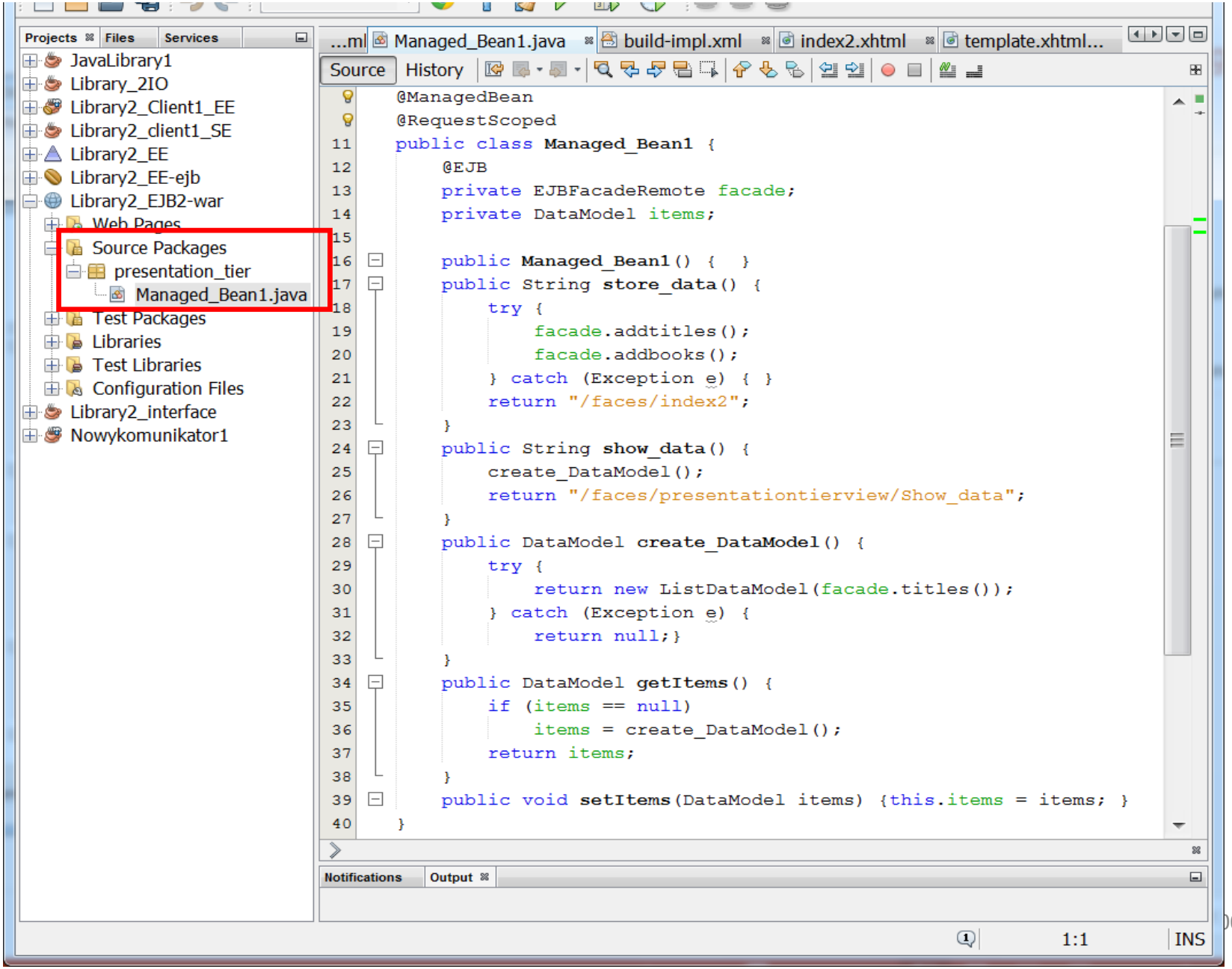
1. Wykonananie projektu typu Web Application i zdefiniowanie formularzy JSF

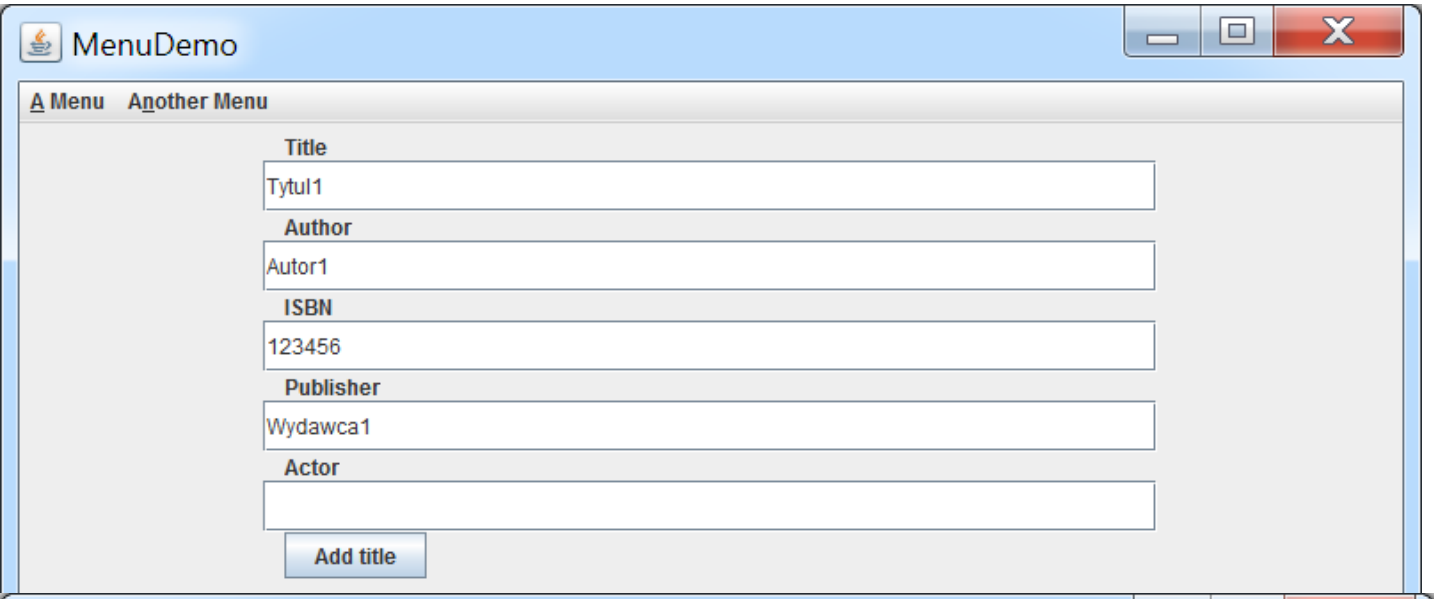


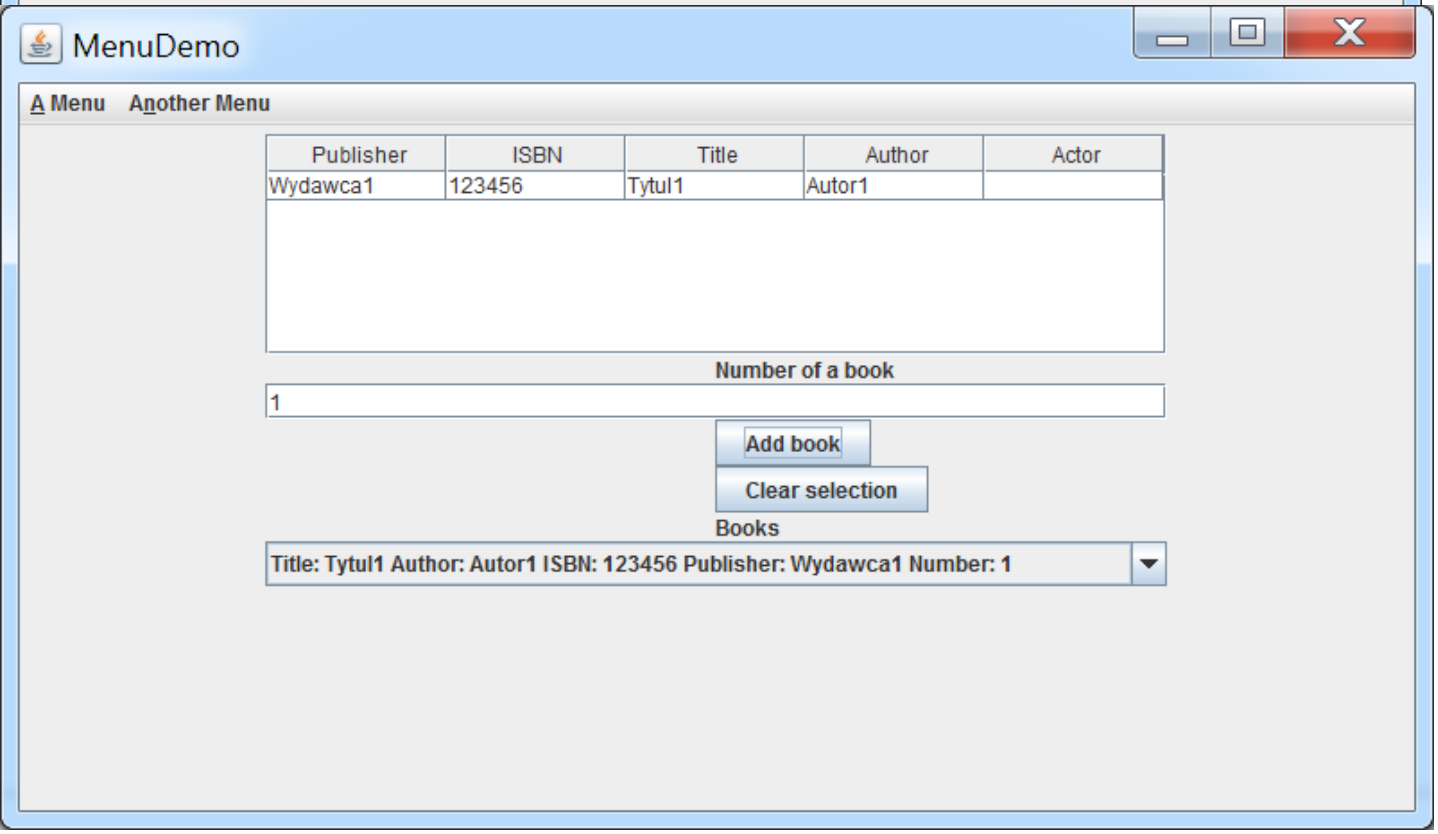
2. Definiowanie formularzy JSF (cd)



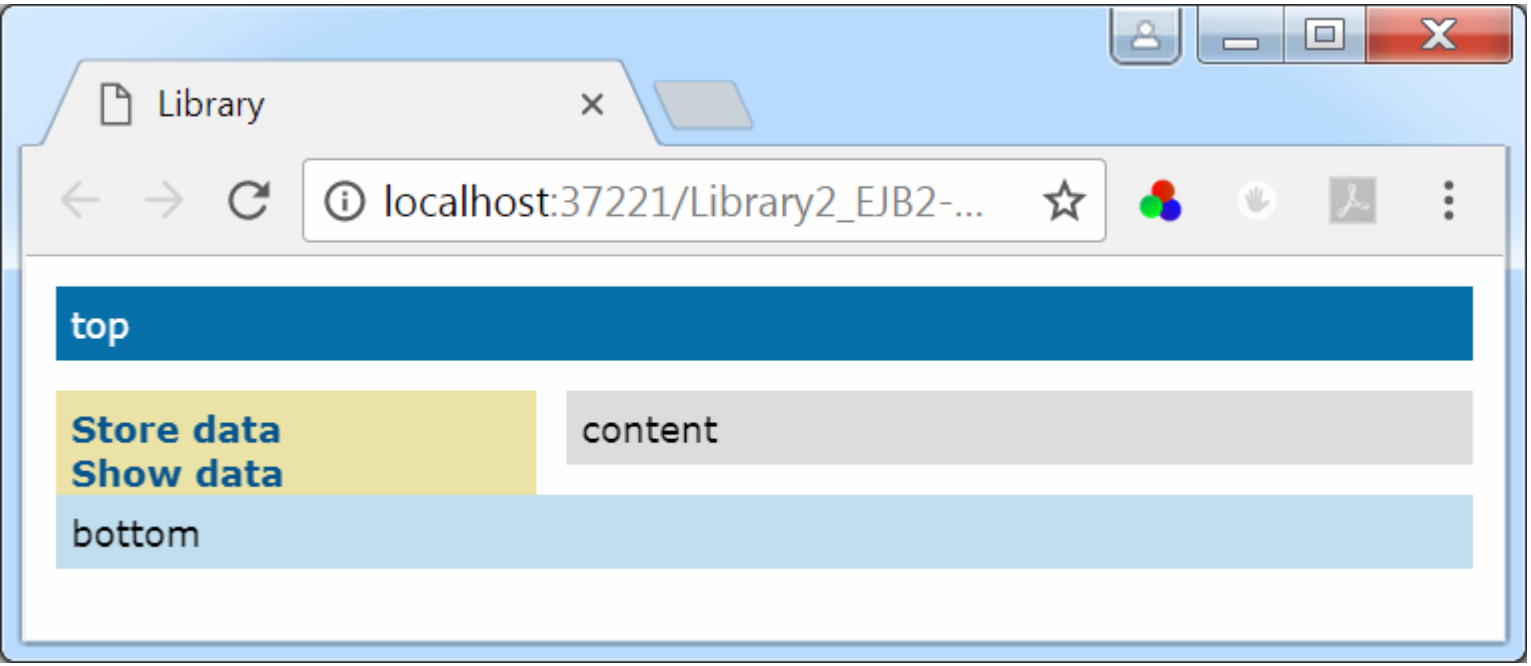
3. Definiowanicja komponentu typu Managed Bean – kontrolera widoków w JSF

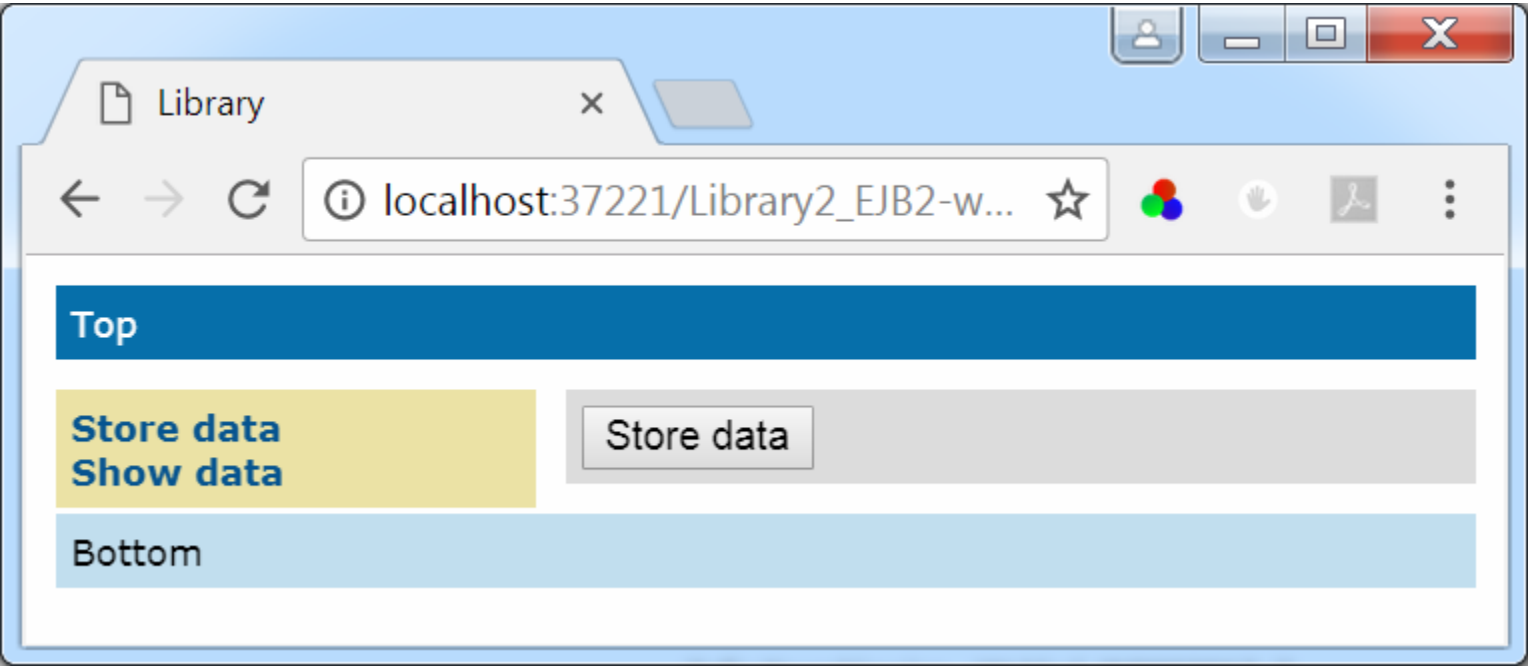






4. Uruchomienie bazodanowej aplikacji typu Enterprise na platformie JavaEE zawierającej dwa typy aplikacji klienckiej: desktopową i internetową



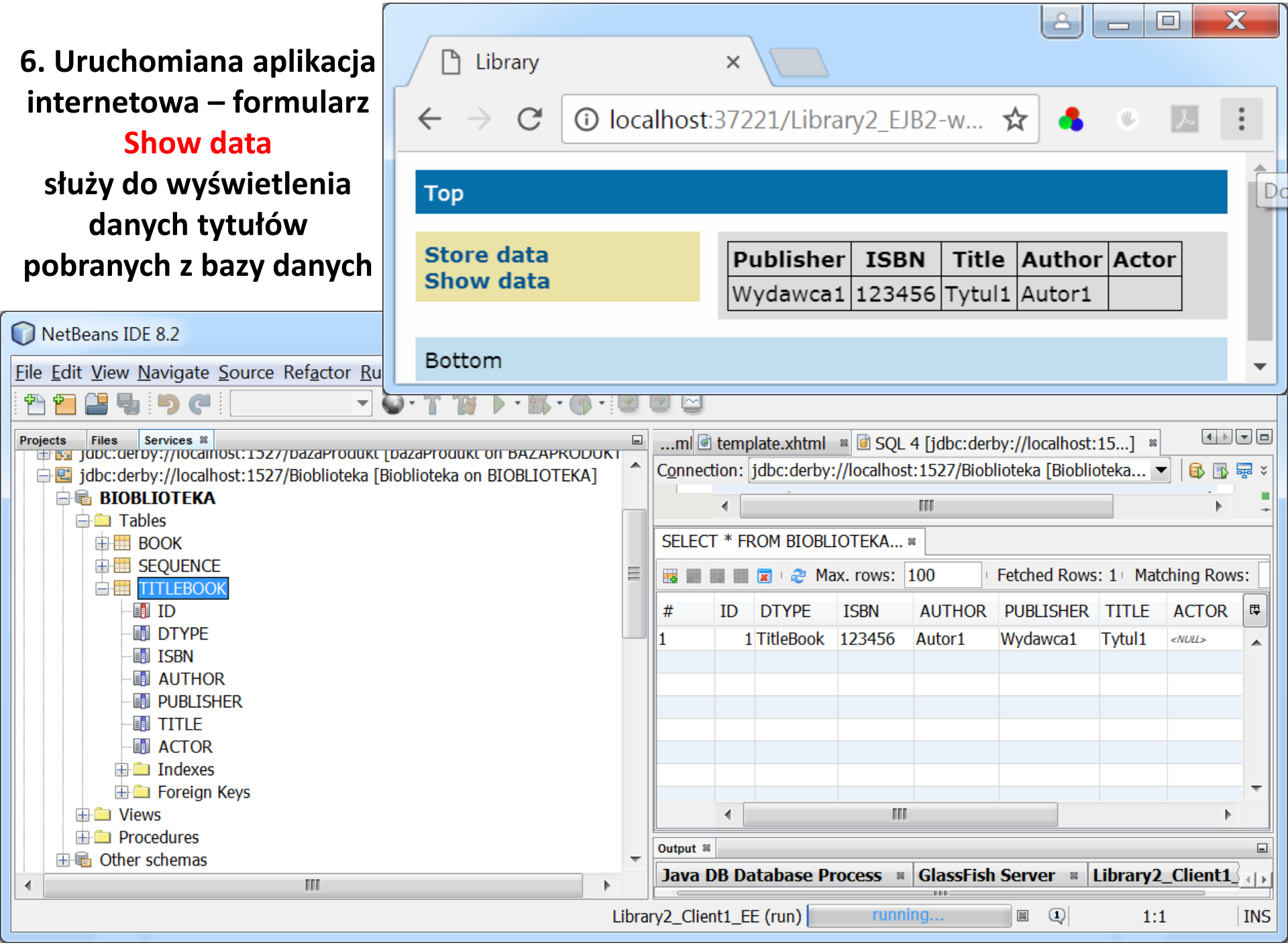


5. Uruchomiana aplikacja internetowa – formularz Store data służy do zapisania danych tytułów i książek do bazy danych

6. Uruchomiana aplikacja internetowa – formularz

Show data

służy do wyświetlenia danych tytułów pobranych z bazy danych



Zagadnienia

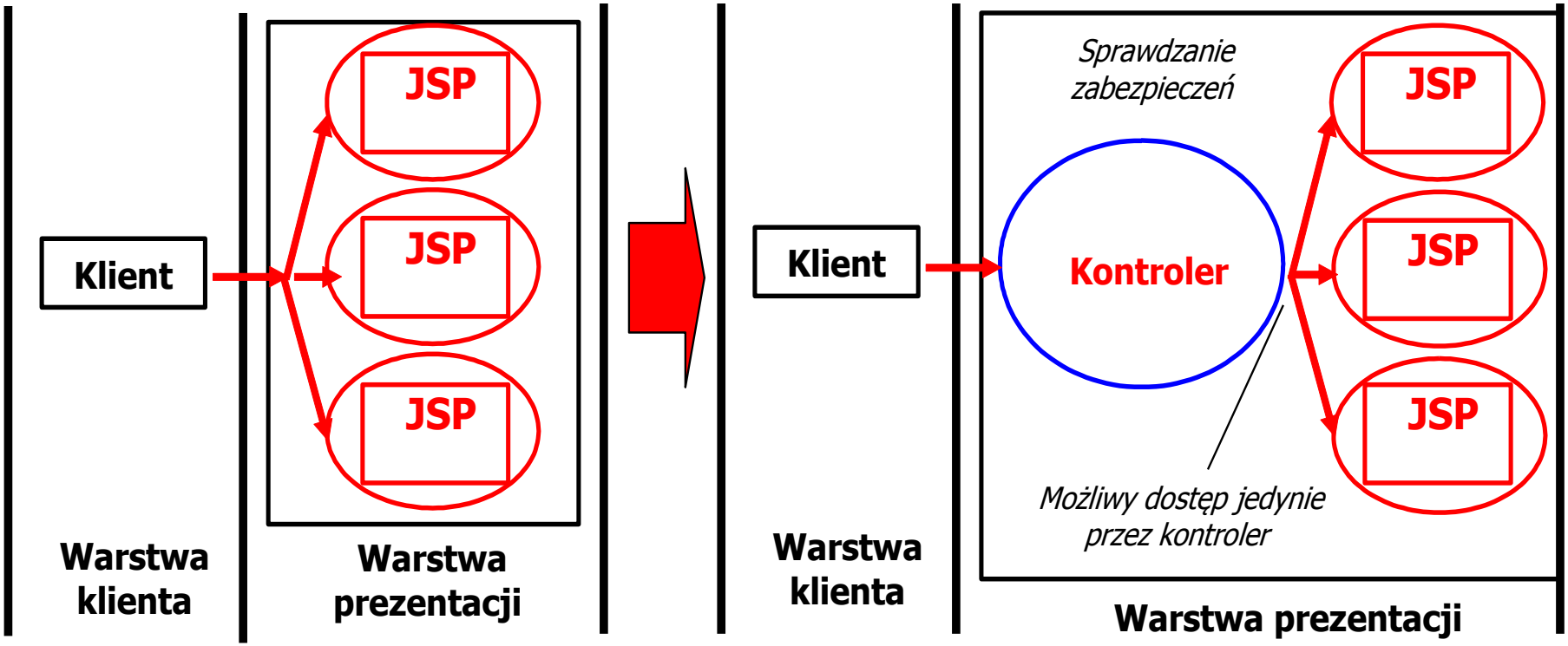
1. Wielowarstwowa architektura systemu informatycznego
2. Ocena i poprawa (refaktoryzacja) architektury wielowarstwowej systemu informatycznego
3. Wzorce projektowe stosowane przy budowie wielowarstwowej aplikacji internetowej
4. Przykład modelowania i projektowania części warstwy biznesowej z obiektami typu POJO. Wykonanie aplikacji dwuwarstwowej.
5. Przykłady architektury wielowarstwowej aplikacji internetowej typu EE. Wykonanie aplikacji typu EE. Warstwa biznesowa: komponenty typu EJB + obiekty POJO
6. Warstwa zasobów (EIS)- baza danych w systemie baz danych Derby
7. Utworzenie obiektowego modelu danych do utrwalania ORM
8. Warstwa integracji. Zastosowanie wzorca projektowego typu *Domain Store* w technologii JPA (Java Persistence) na platformie Java EE
9. Warstwa prezentacji - JSF
10. Dodatek

Dodatek

1. **Refaktoryzacja warstwy prezentacji**
2. **Refaktoryzacja warstwy biznesowej**
3. **Tworzenie warstwy integracji**

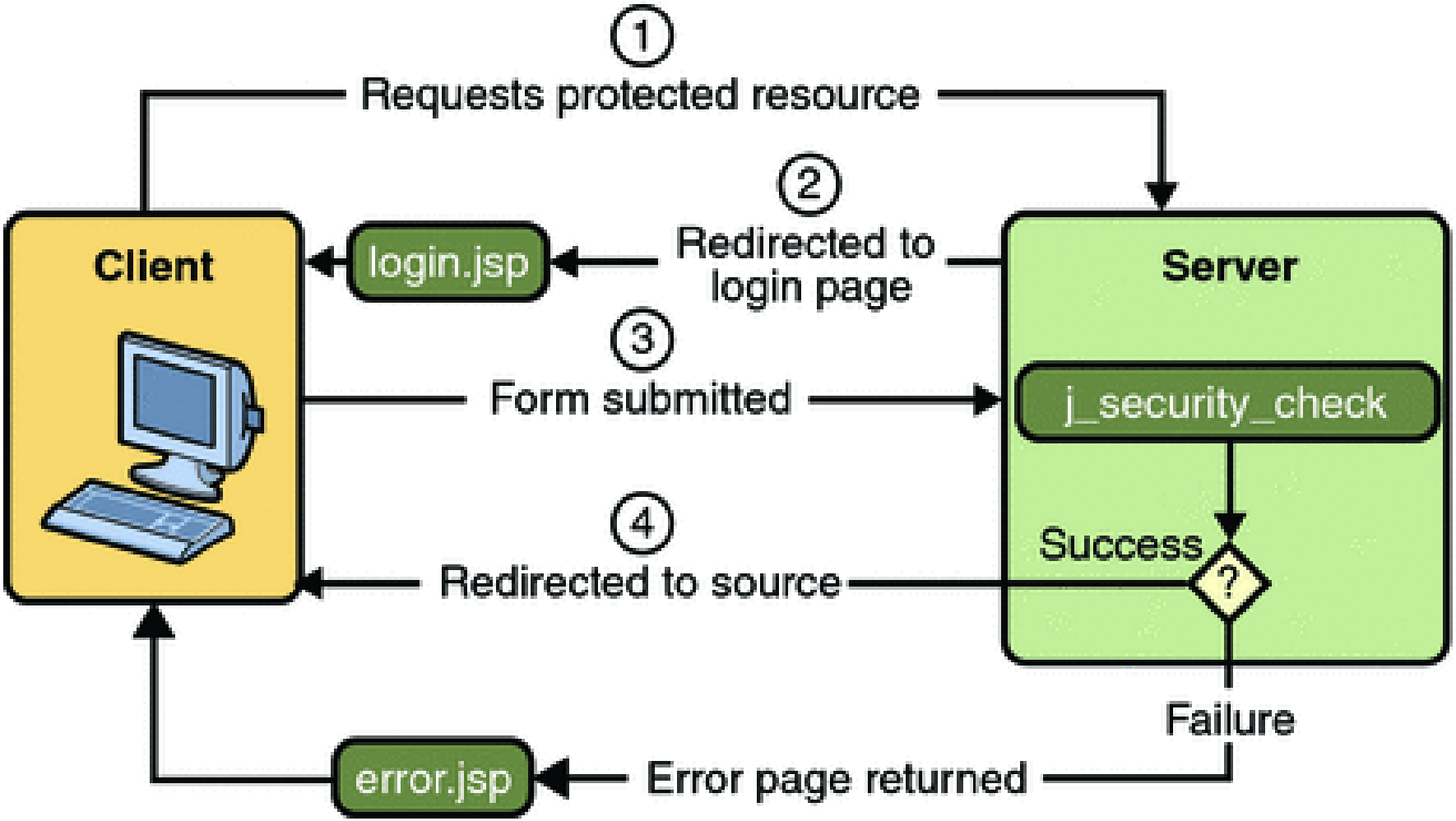
1. Refaktoryzacja warstwy prezentacji

Ukrywanie zasobów przed klientem za pomocą konfiguracji kontenera – uwierzytelnianie i autoryzacja



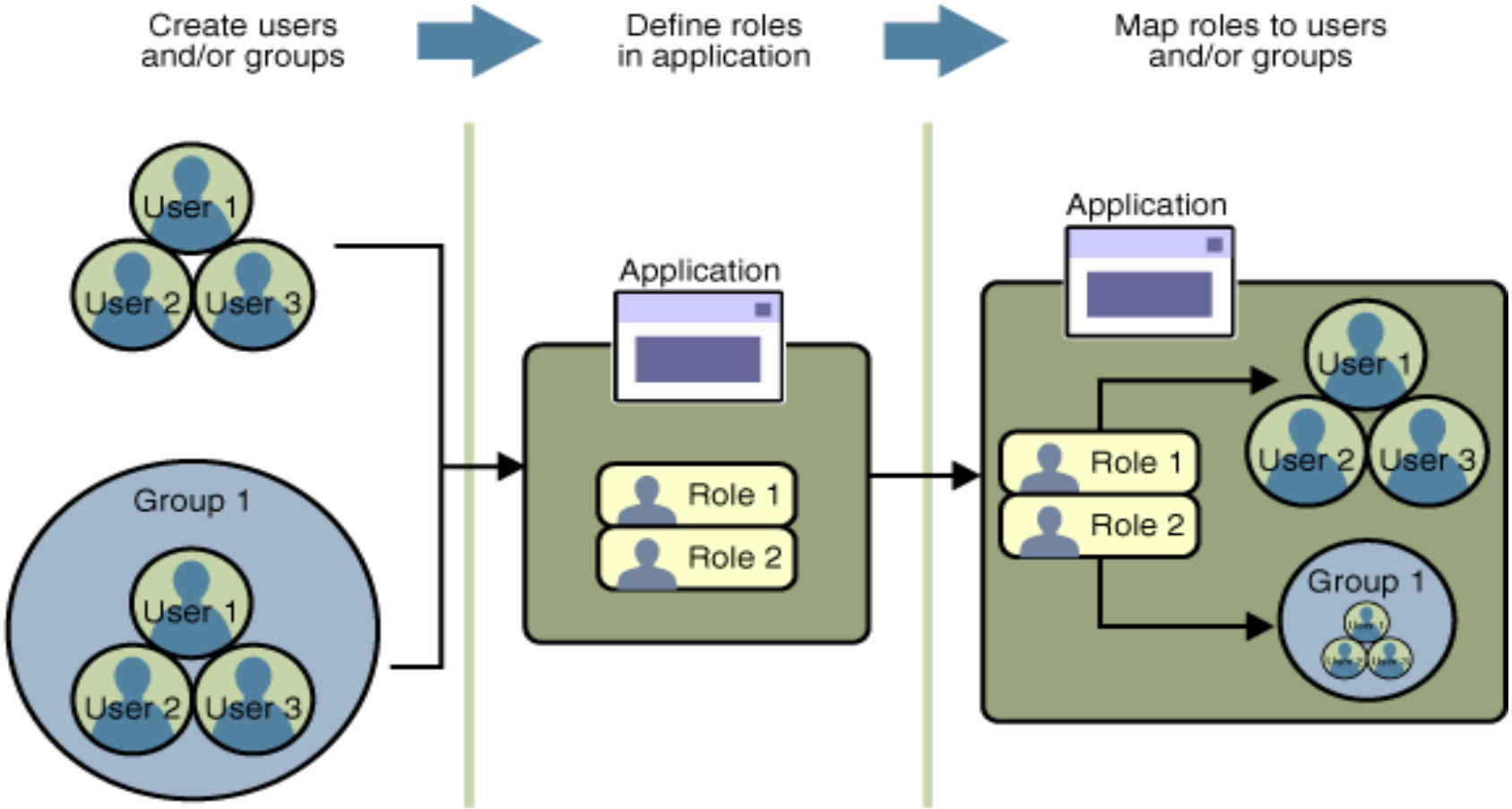
Przebieg uwierzytelniania (logowania)

http://download.oracle.com/javaee/5/tutorial/doc/bncbe.html



Bazy użytkowników i grup, Użytkownik, Grupa, Rola

http://download.oracle.com/javaee/5/tutorial/doc/bnbxj.html



Rodzaje mechanizmów bezpieczeństwa

w kontenerach (kod do zarządzania aplikacją przez serwer aplikacji m.in. Bezpieczeństwem aplikacji)

- **Deklaratywne mechanizmy bezpieczeństwa** – deklarowane za pomocą tzw. *„deployment descriptors" (deskryptory aplikacji np. **web.xml*** dla aplikacji typu ***web*)**. Deskryptory jako zewnętrzny element aplikacji zawierają informację specyfikującą role bezpieczeństwa i wymagania dostępu są mapowane w role specyficzne dla środowiska oraz użytkowników i polisy bezpieczeństwa.

- ***Programowe mechanizmy bezpieczeństwa*** - są osadzone w aplikacji i służą do podejmowanie decyzji o bezpieczeństwie. Uzupełniają deklaratywne mechanizmy bezpieczeństwa – lepiej wyrażają model bezpieczeństwa aplikacji. API mechanizmów programowych:
  - metody interfejsu EJBContext
  - metody interfejsu HttpServletRequest. Metody te pozwalają na podejmowanie decyzji biznesowych opartych na rolach bezpieczeństwa nadawcy lub zdalnego odbiorcy

- **Adnotacje lub metadane** są używane do specyfikowania informacji wewnątrz pliku z kodem klasy. Kiedy aplikacja jest uruchamiana, informacja ta jest używana lub pokrywana przez deskryptor aplikacji.

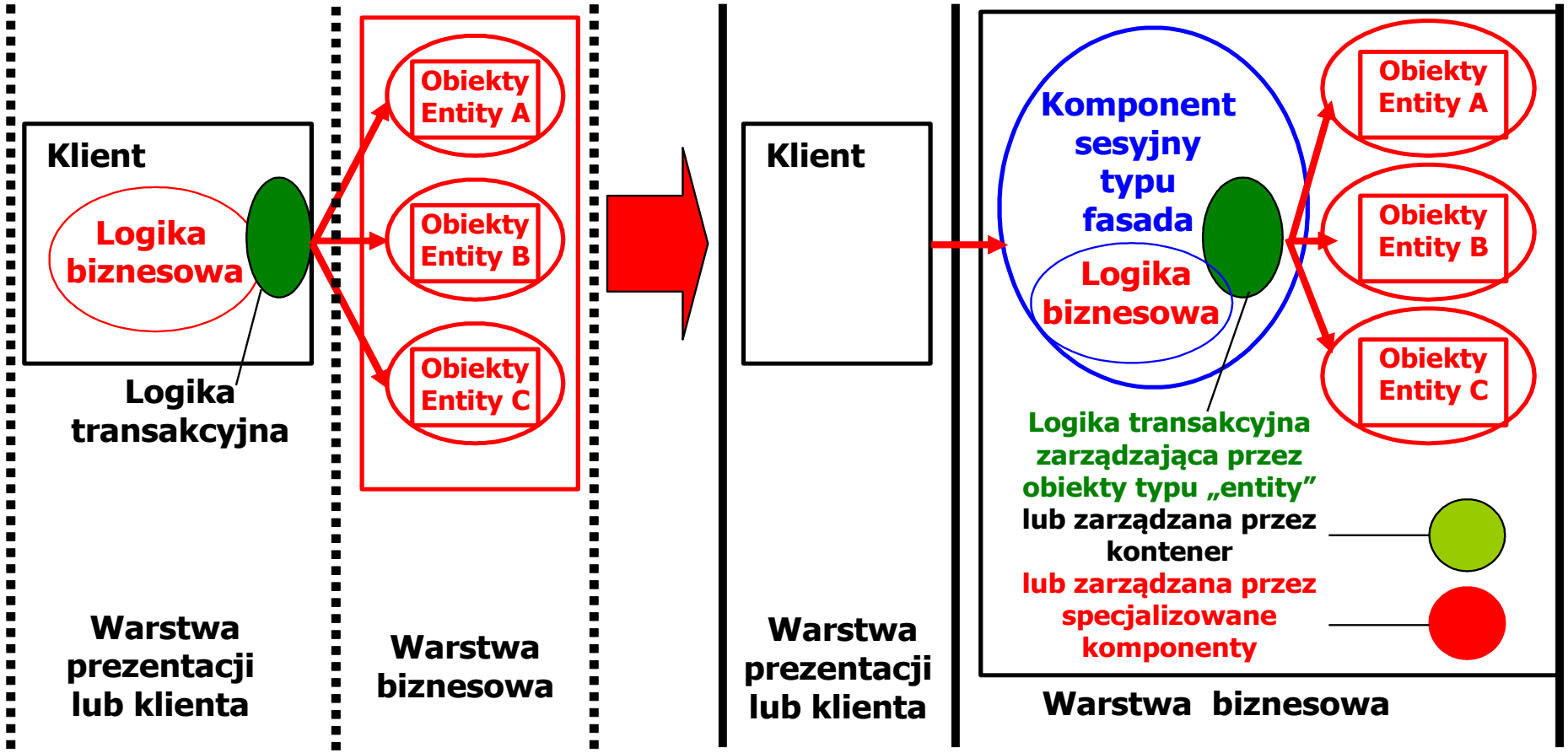
Np.

```
@DeclareRoles(„klient")
public class Page1 extends AbstractPageBean  {   //... }
```

2. Refaktoryzacja warstwy biznesowej

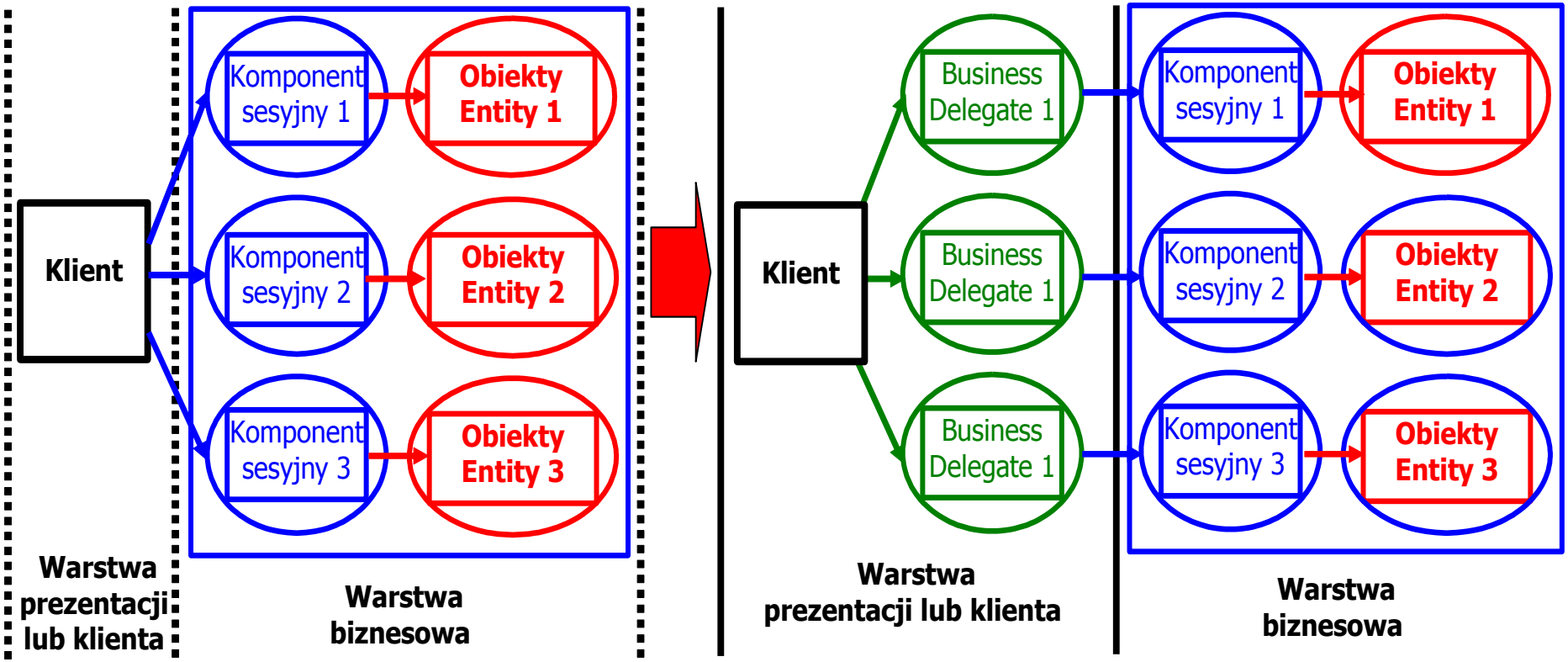
Refaktoryzacja warstwy biznesowej 1

Obiekty danych typu „Entity” (obiekty biznesowe) z warstwy biznesowej są udostępniane klientom w innych warstwach za pomocą **fasadowych komponentów sesyjnych typu „Control” (komponent typu fasada - hermetyzujący dostęp do usług biznesowych)**



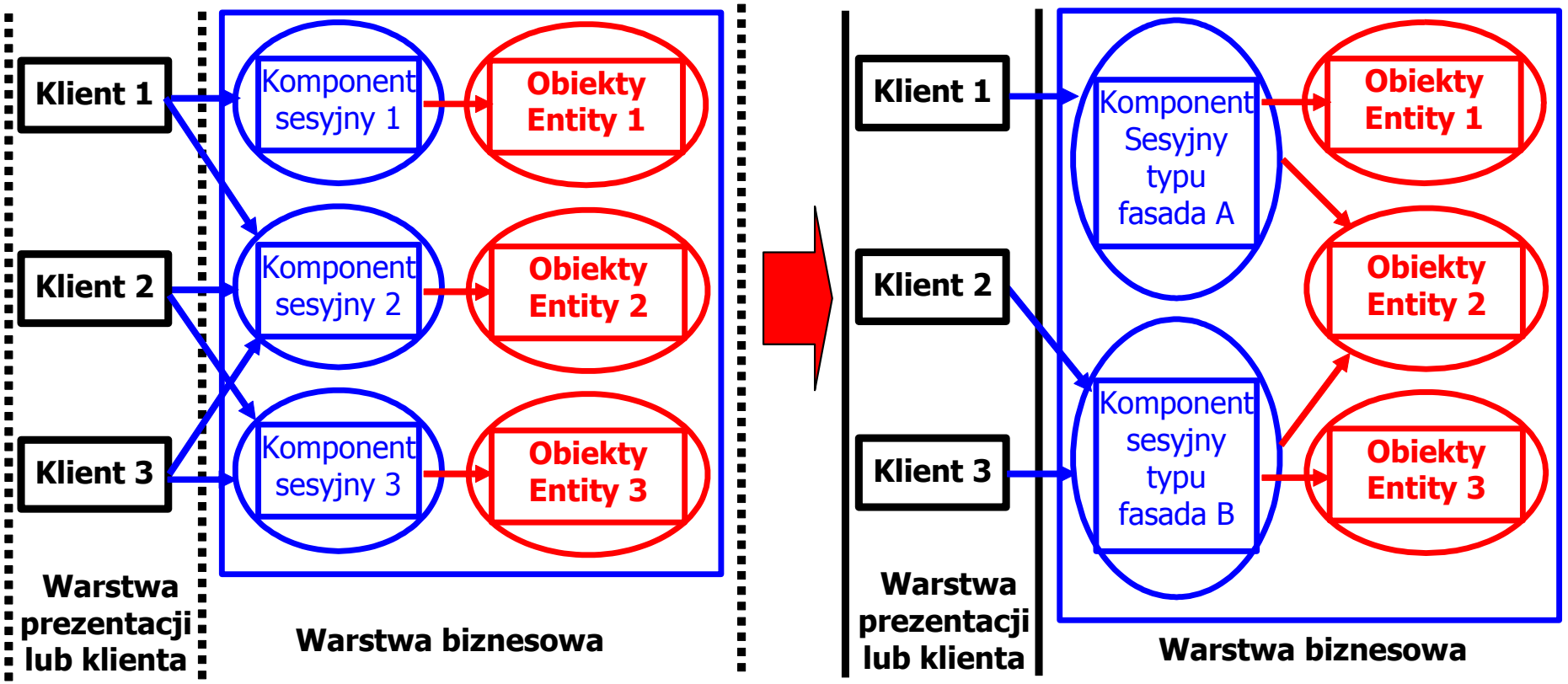
Refaktoryzacja warstwy biznesowej 2

Komponenty sesyjne typu „Control” (pośredniczące w dostępie do **obiektów danych typu „Entity”**) z warstwy biznesowej są udostępniane klientom w innych warstwach za pomocą **obiektów fasadowych typu „Control” (hermetyzujących dostęp do warstwy biznesowej- komponentów Business Delegate)**



Refaktoryzacja warstwy biznesowej 3

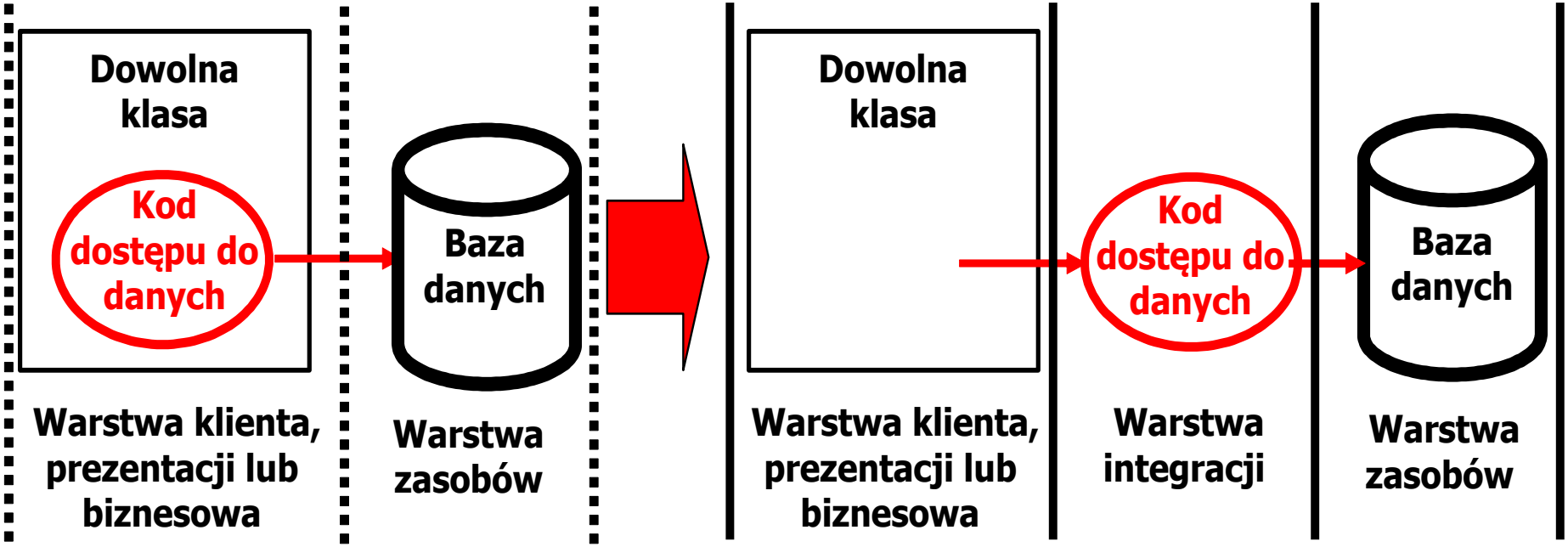
Sesyjne komponenty fasadowe typu „Control" (każdy komponent jako odrębna usługa biznesowa), hermetyzujące **obiekty danych typu „Entity"** z warstwy biznesowej są udostępniane klientom w innych warstwach. Zwykłe obiekty sesyjne są jedynie pośrednikami obiektów **„Entity",** natomiast nie hermetyzują całych usług, które wymagają odwołania do wielu zwykłych komponentów sesyjnych.



3. Tworzenie warstwy integracji

Wydzielanie kodu dostępu do danych

- **Kod dostępu do danych jest wydzielany z klas, które są używane do spełniania również innych celów**
- **Kod dostępu do danych powinno umieszczać się logicznie i fizycznie bliżej źródła danych**



Refaktoryzacja dostępu do danych – pula połączeń

- **Liczba połączeń kodu dostępu do danych (DAO) z bazą danych jest ograniczona**
- **Połączenia kodu dostępu do danych (DAO) nie zawsze są wykorzystywane, lecz są utrzymywane, ponieważ otwarcie połączenia z bazą danych zabiera i zasoby**
- **Pula połączeń kodu dostępu do danych (DAO) pozwala racjonalnie zarządzać połączeniami aplikacji z bazą danych**