Projekt INP002017
Instrukcja 1

Autor

Dr inż. Zofia Kruczkiewicz

I. Czynności wykonane zgodnie z harmonogramem grupy w tygodniach 1-6

1. Czynności (str. 3-12) wg instrukcji do lab3:

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/INP002017/Lab_INP002017_3.pdf

2. Czynności (str. 13-14) wg instrukcji do lab4

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/INP002017/Lab_INP002017_4.pdf

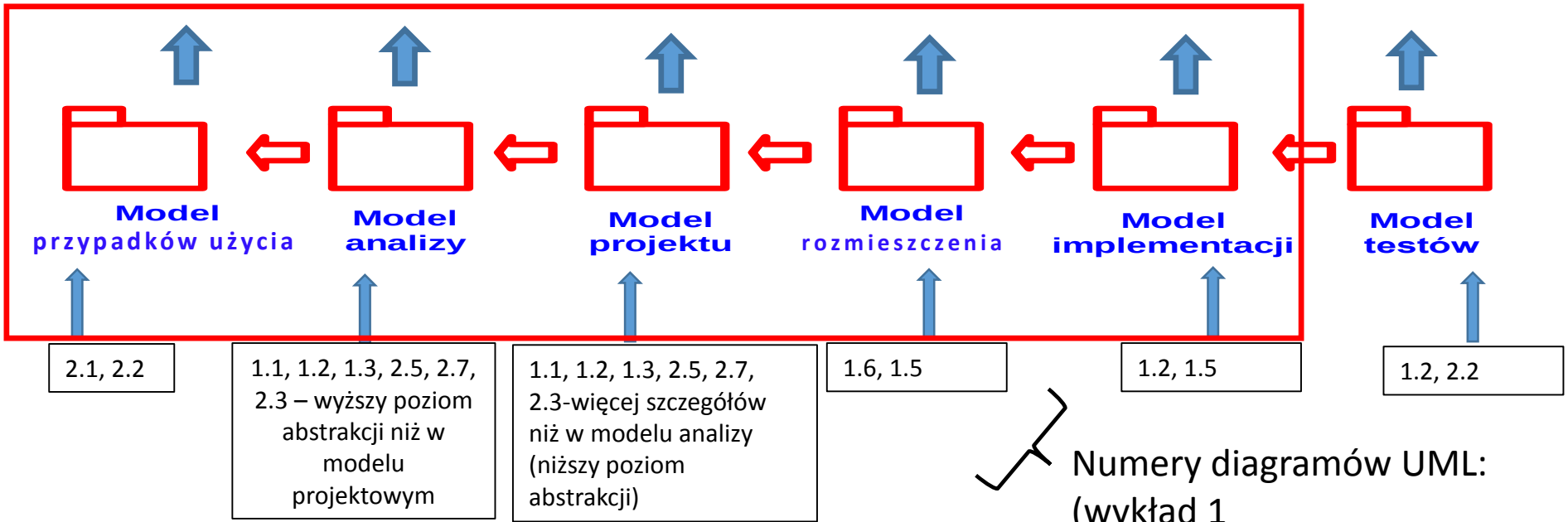
3. Czynności (str. 15-24) wg instrukcji dla lab5-7

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/INP002017/Lab_INP002017_5_1.pdf

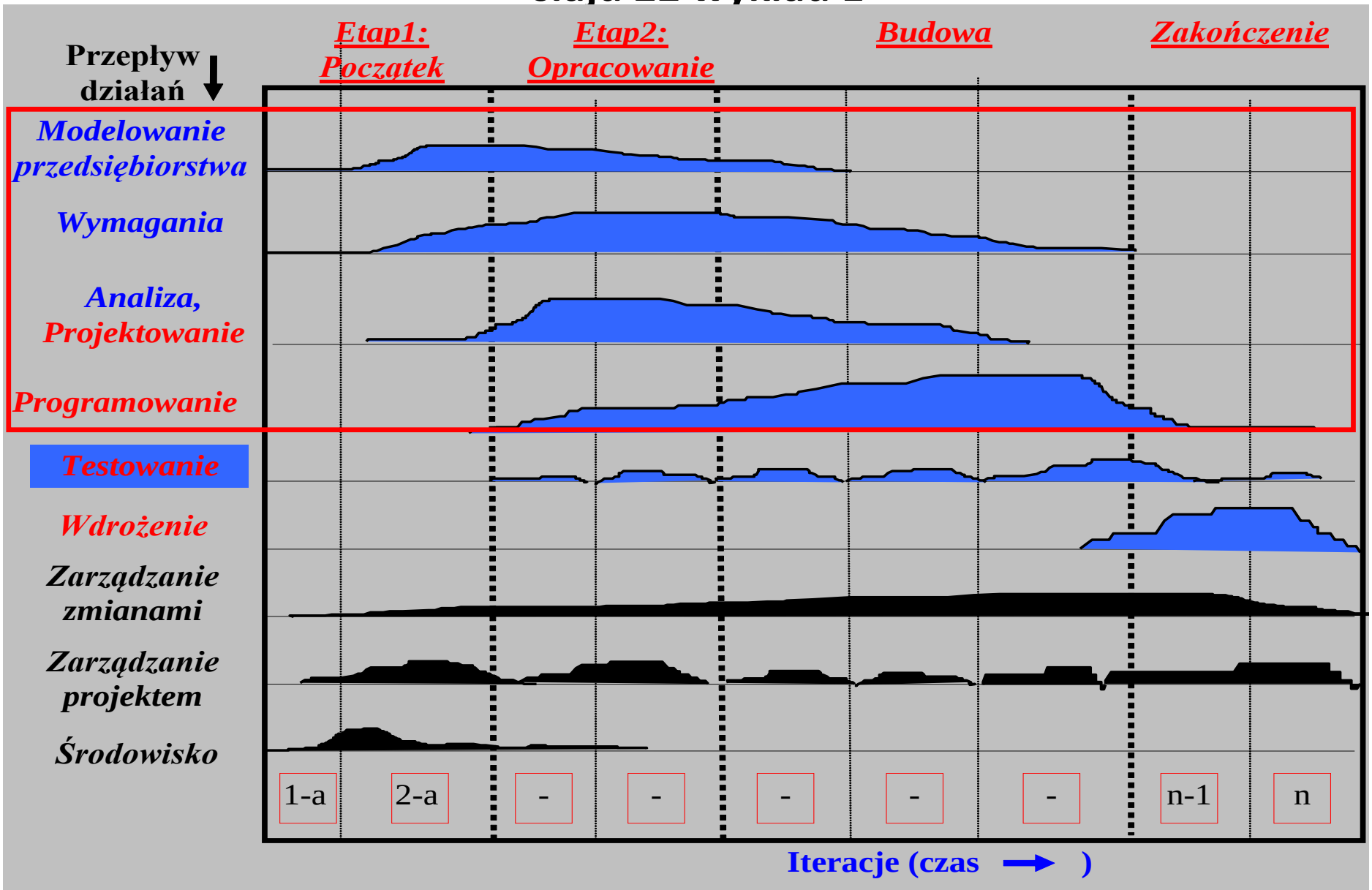
http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/INP002017/Lab_INP002017_5_2.pdf

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 przeglądającdających 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

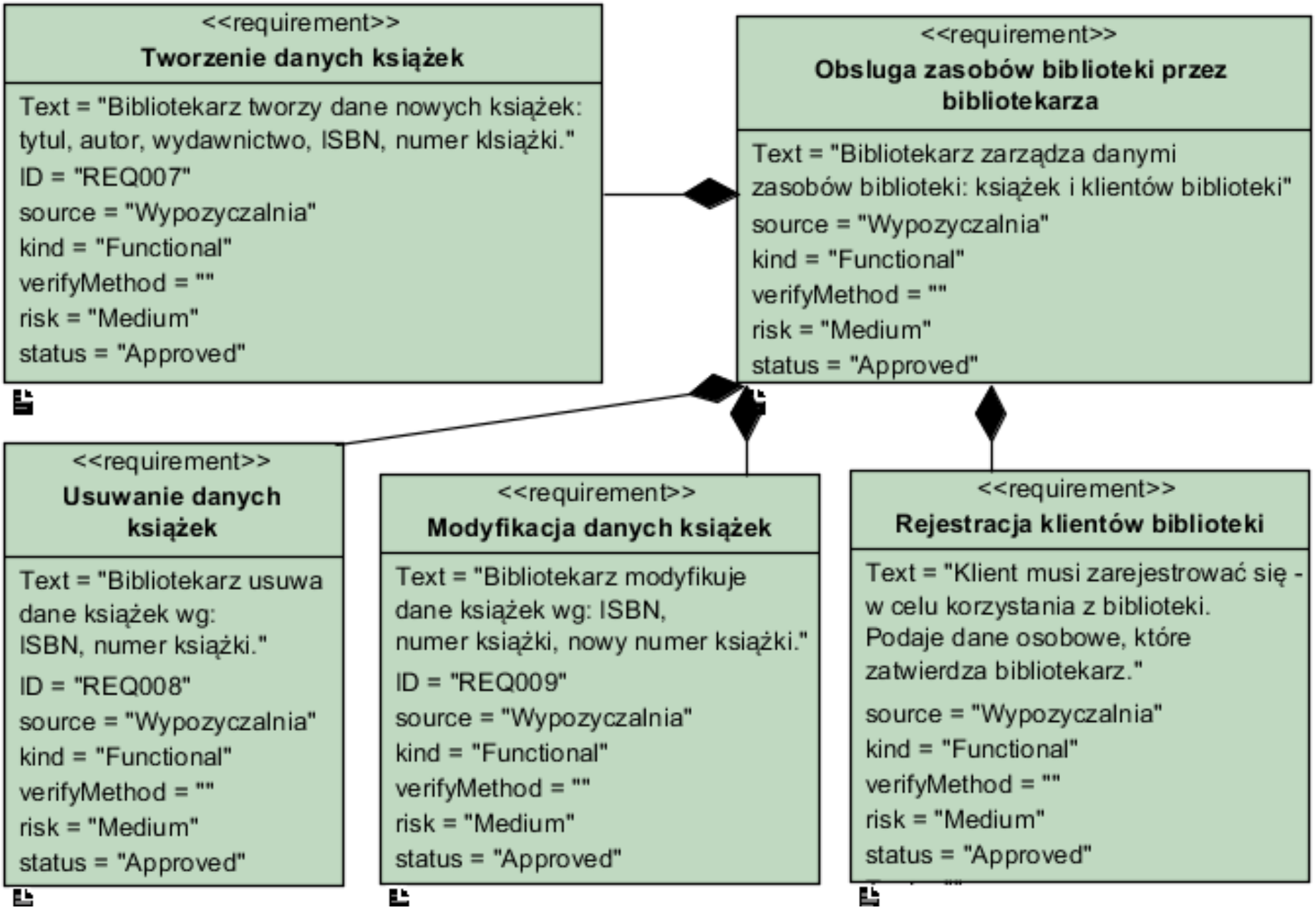


Diagram wymagań funkcjonalnych (cd)

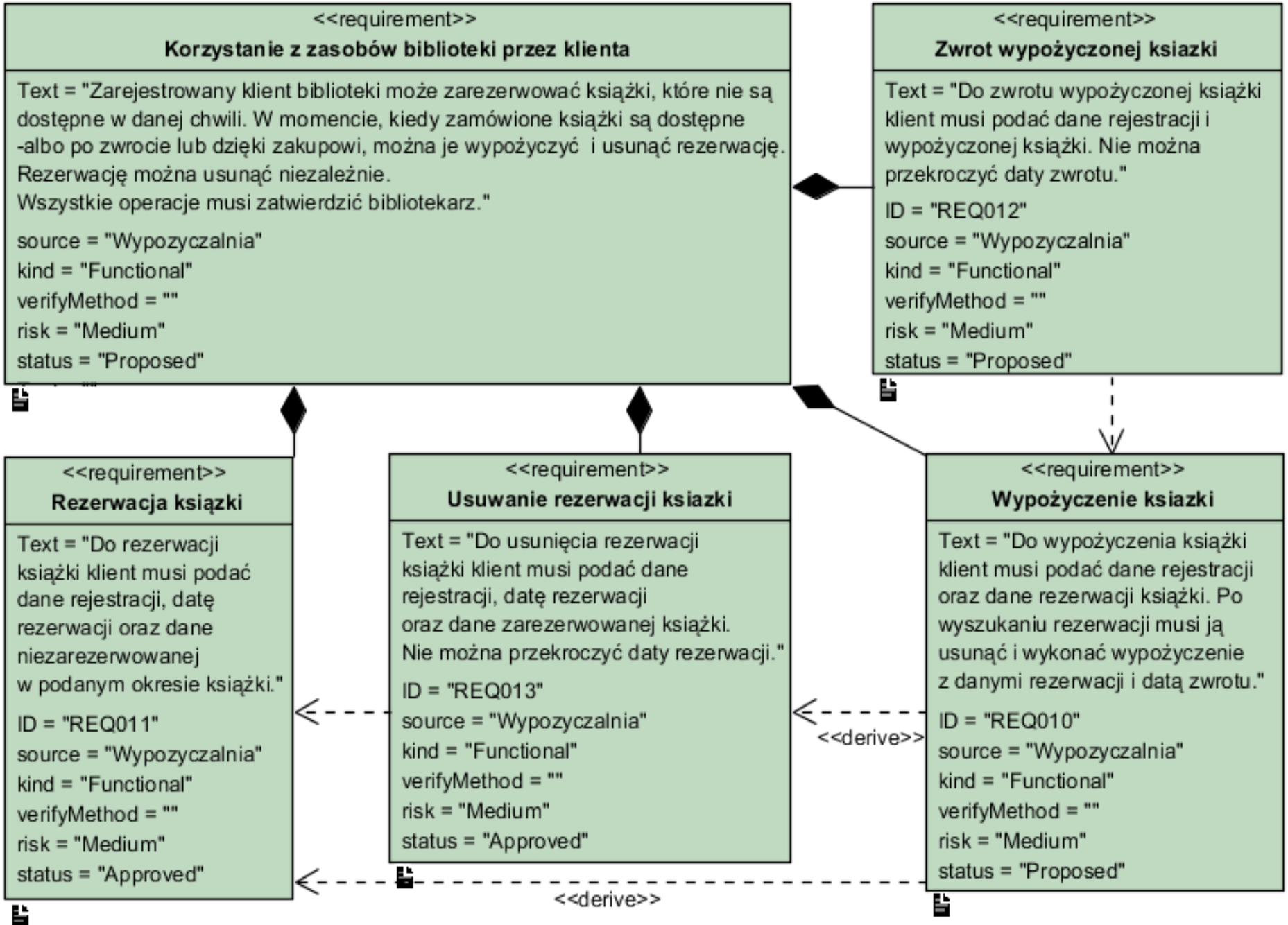
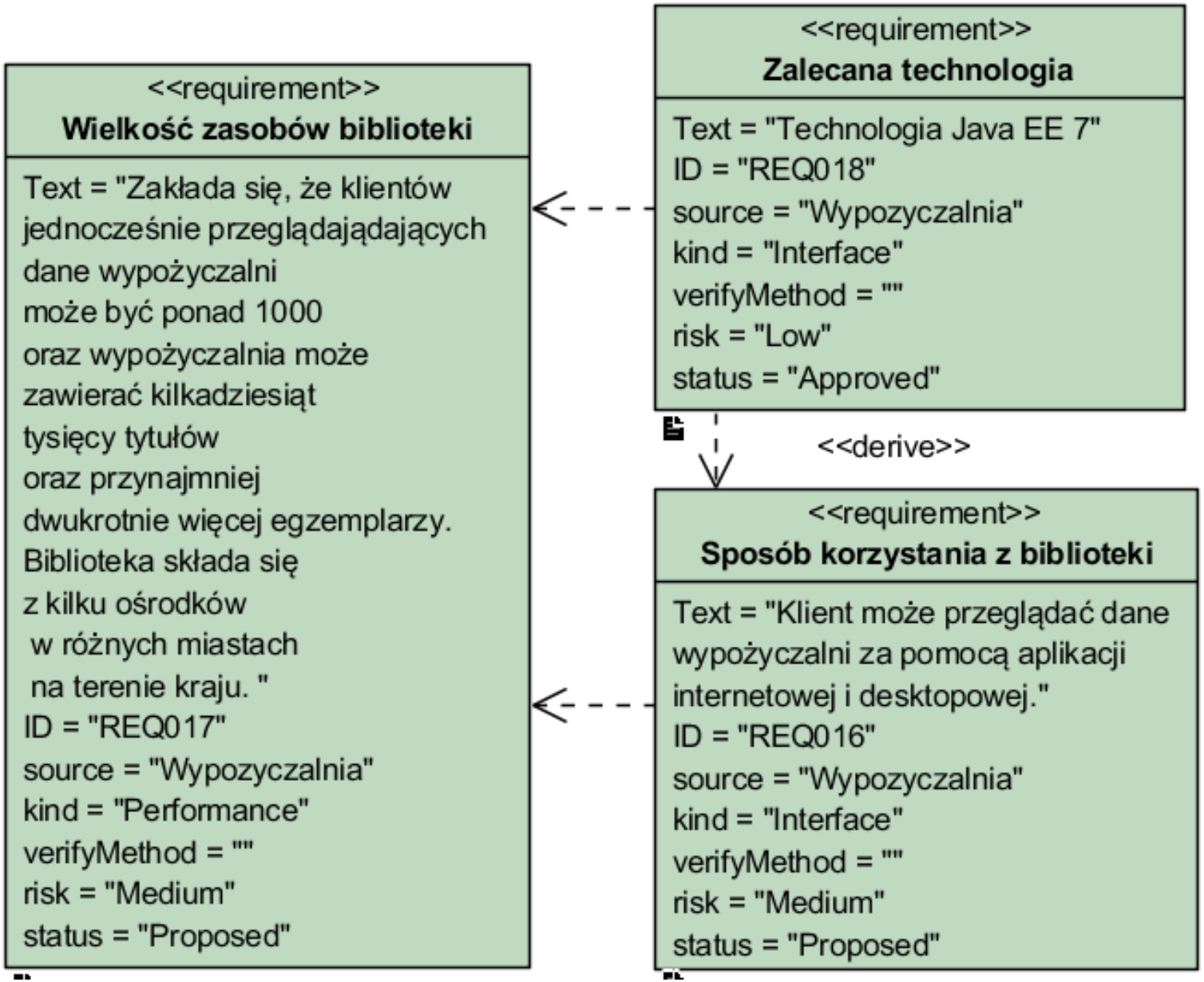


Diagram wymagań niefunkcjonalnych (cd)



III. Model analizy aplikacji oparty na diagramie przypadków użycia

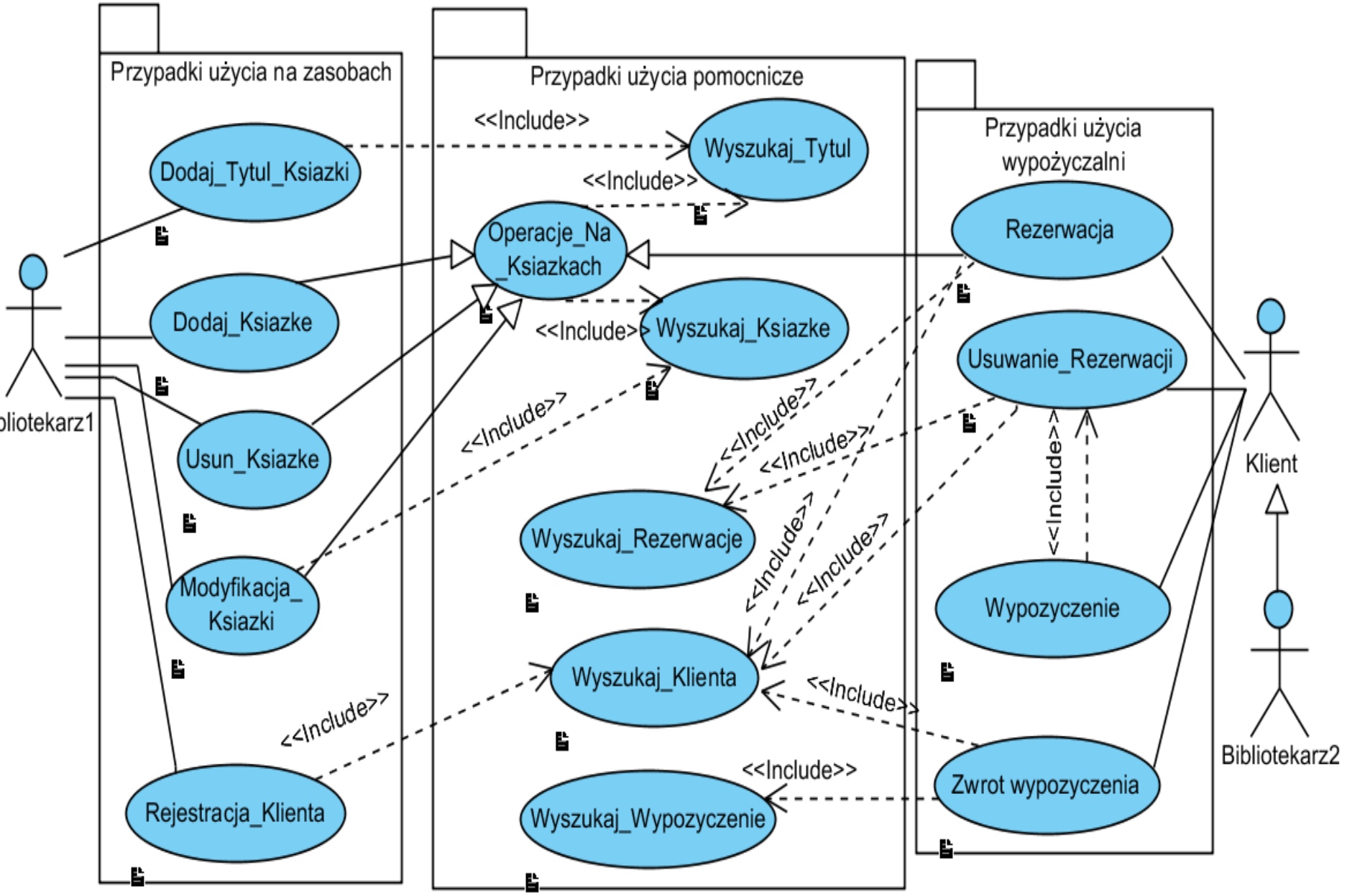
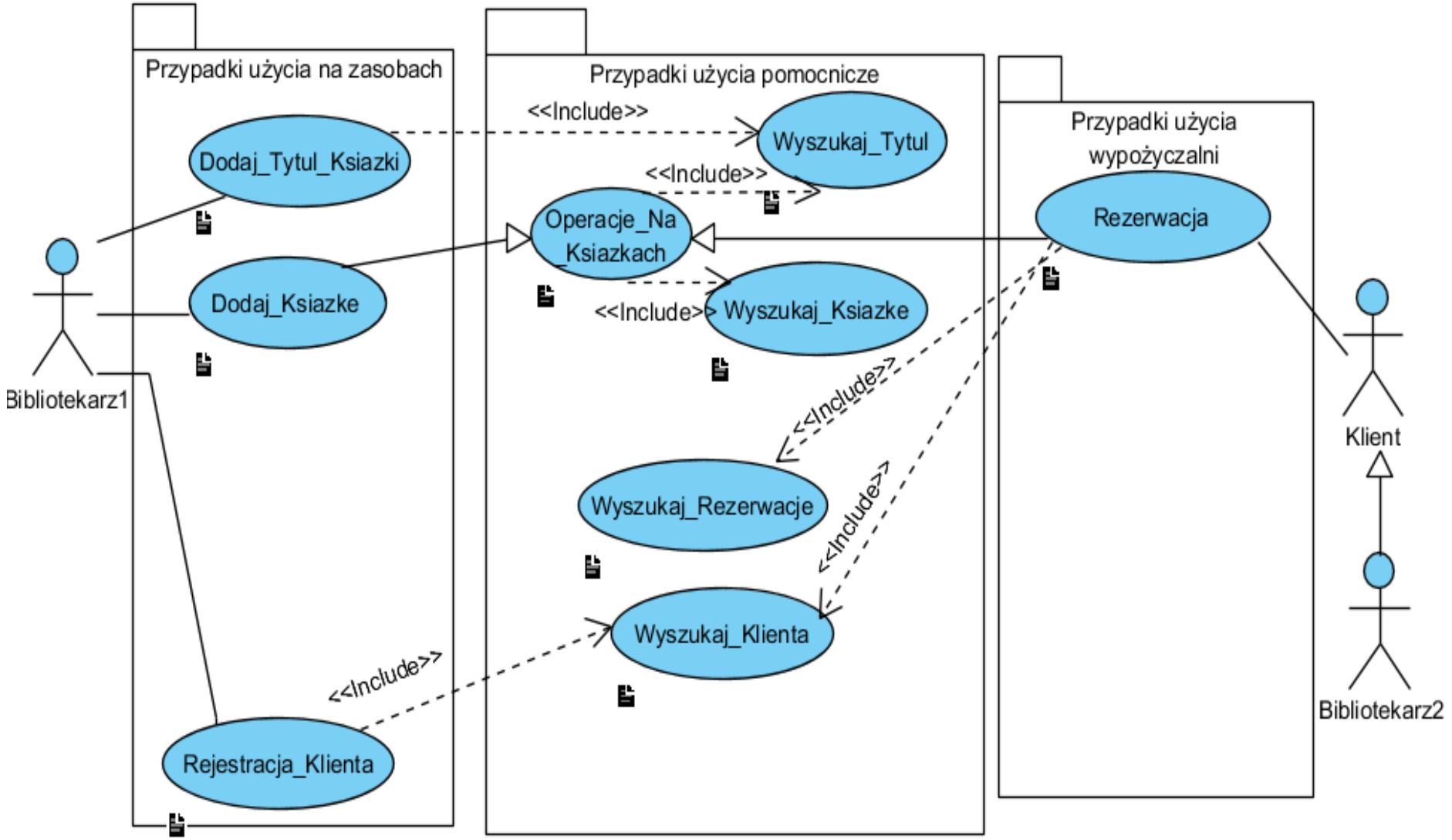


Diagram przypadków użycia – wybrany fragment



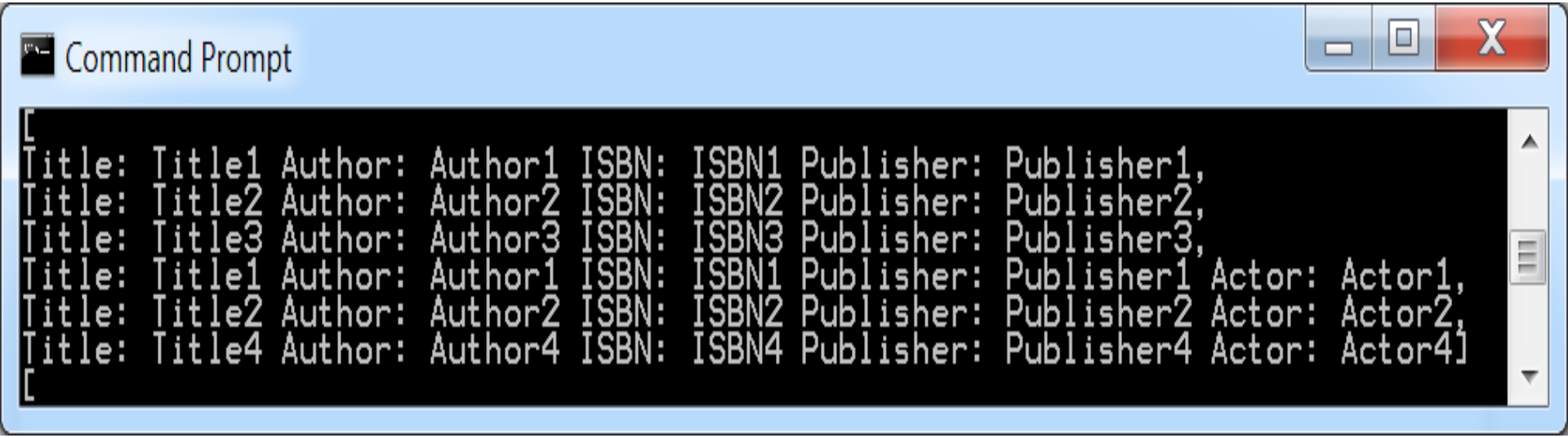
IV. Model projektowy i implementacja warstwy biznesowej warstwy biznesowej oparty na diagramie klas i diagramie sekwencji tworzony metodą iteracyjno-rozwojową, sterowany realizacją przypadków użycia

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.

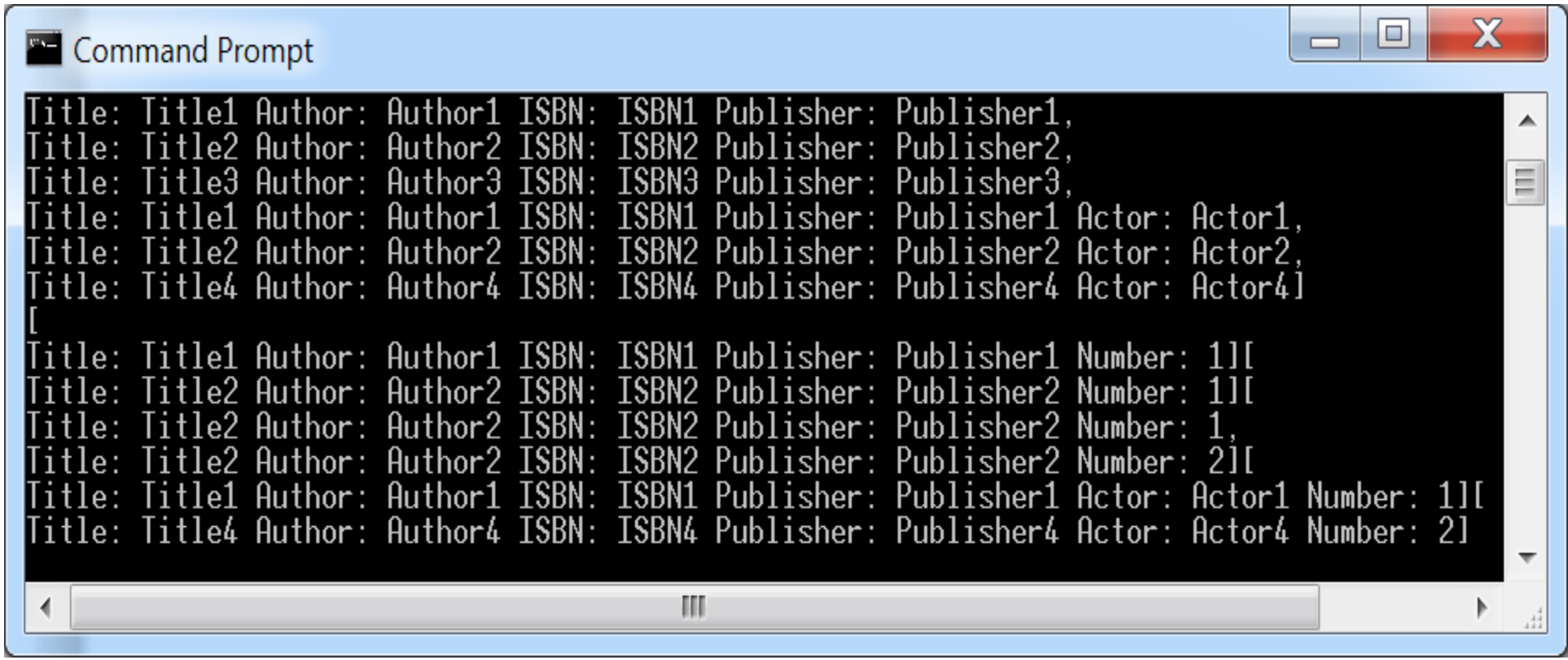

Command Prompt
[
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1,
Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publisher2,
Title: Title3 Author: Author3 ISBN: ISBN3 Publisher: Publisher3,
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1,
Title: Title2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

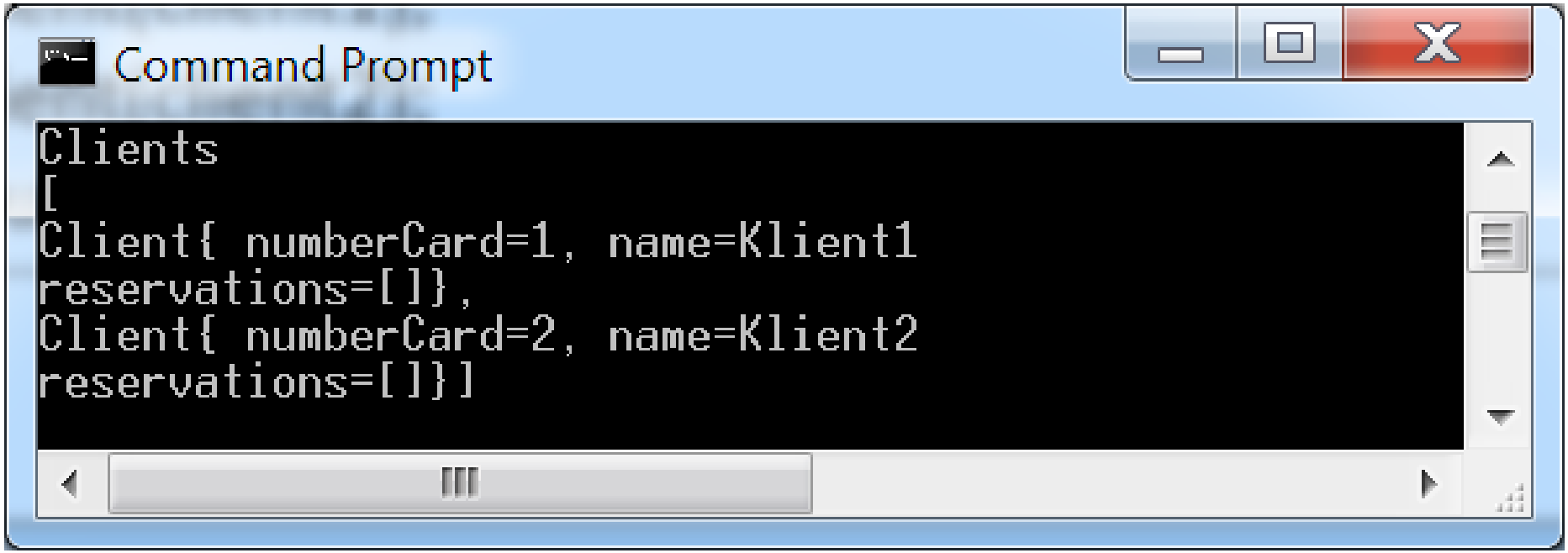
Iteracja 3

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);
```



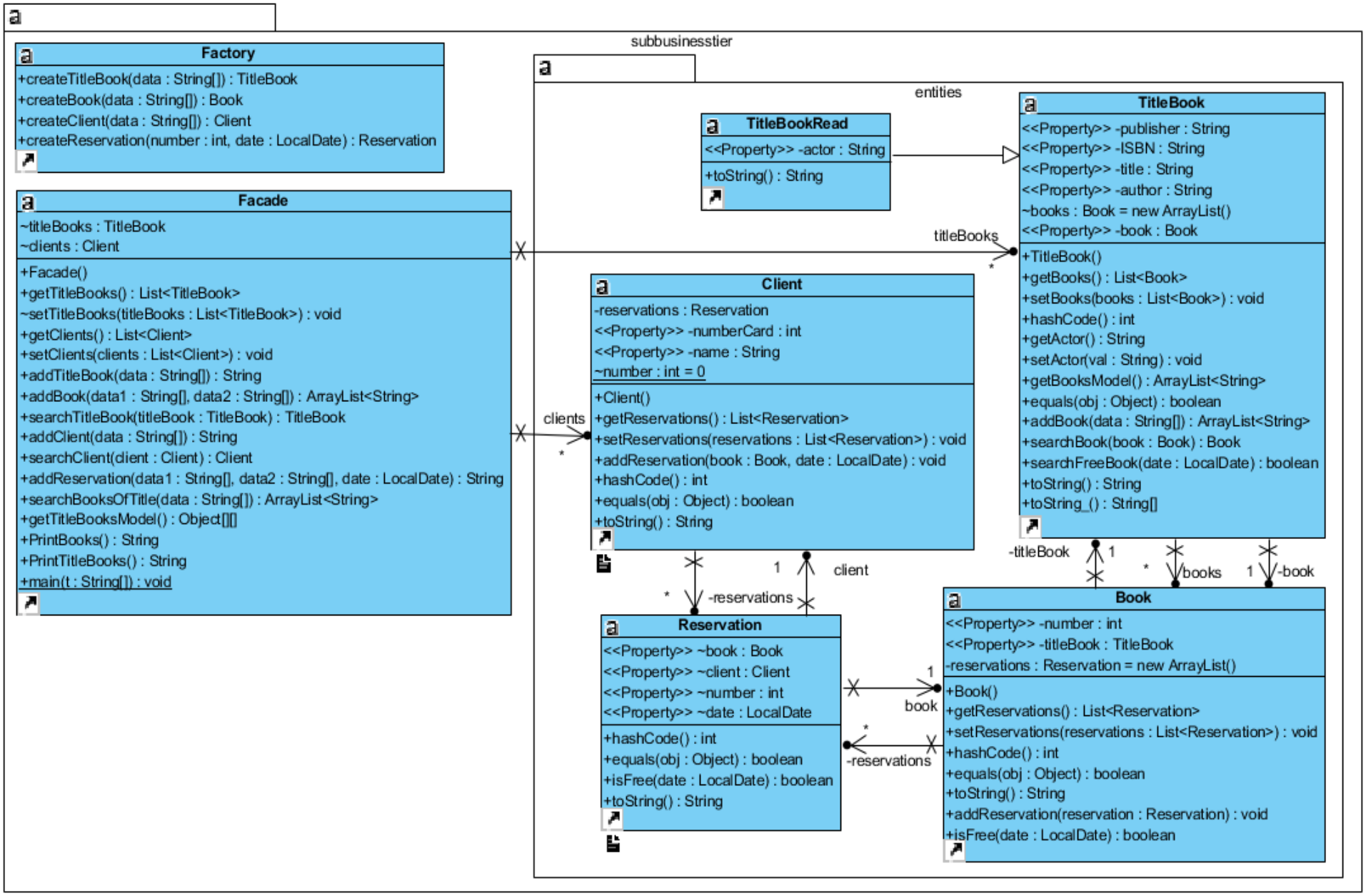
Iteracja 4

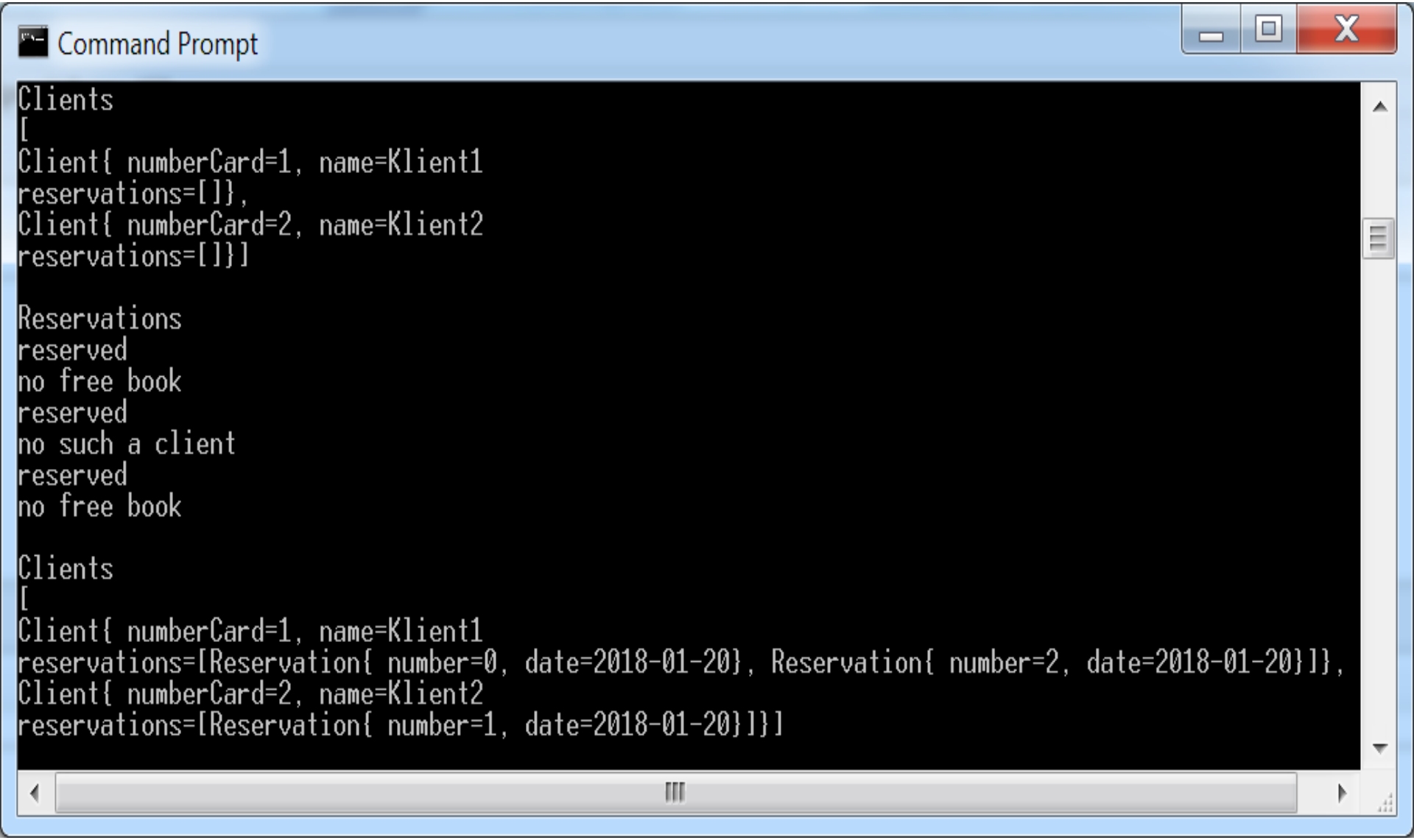
Projekt przypadku użycia

„**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)

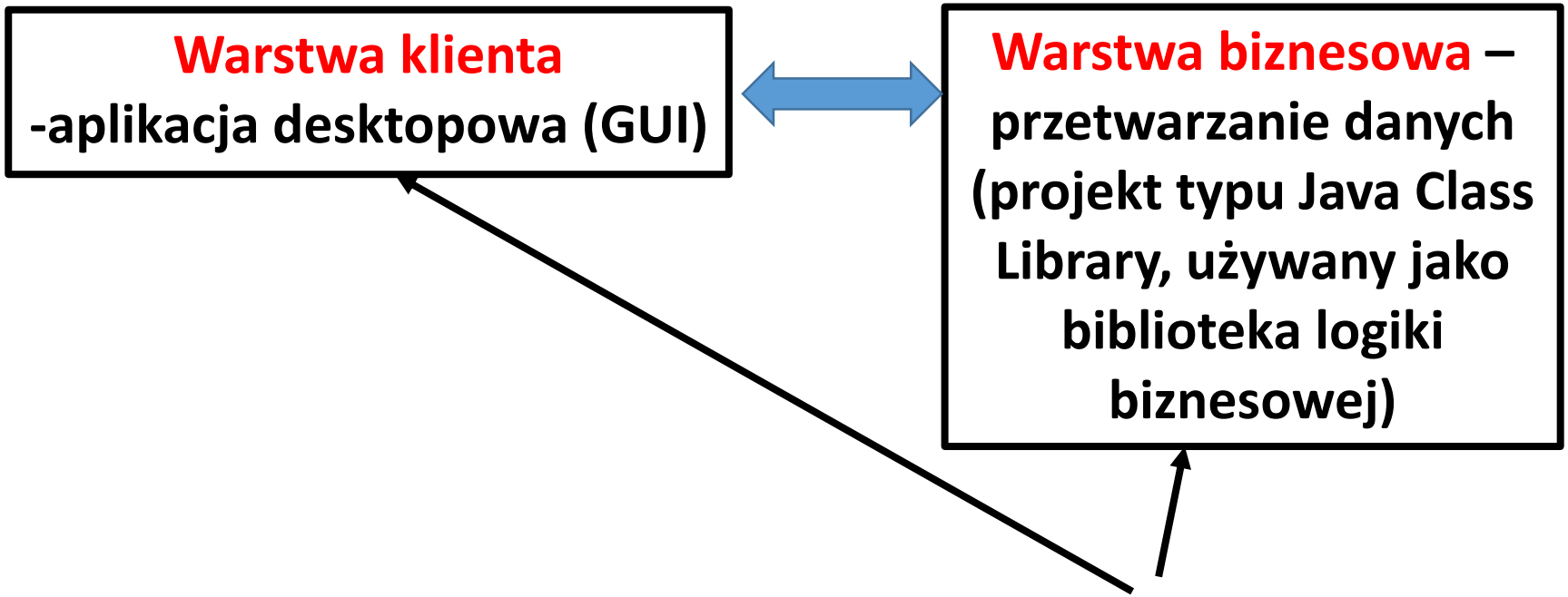



Command Prompt
Clients
[
Client{ numberCard=1, name=Klient1
reservations=[]},
Client{ numberCard=2, name=Klient2
reservations=[]}]
Reservations
reserved
no free book
reserved
no such a client
reserved
no free book
Clients
[
Client{ numberCard=1, name=Klient1
reservations=[Reservation{ number=0, date=2018-01-20},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

II. Czynności wykonane zgodnie z harmonogramem grupy w tygodniach 7-8
(wg materiałów do wykładu 7 Wyklad_INP002017_7_2.pdf - p.4-5)

1. Kontynuacja działań na programem z p.I (str. 26-32)- wykonanie aplikacji dwuwarstwowej obsługującej 1-go użytkownika na platformie Java SE.

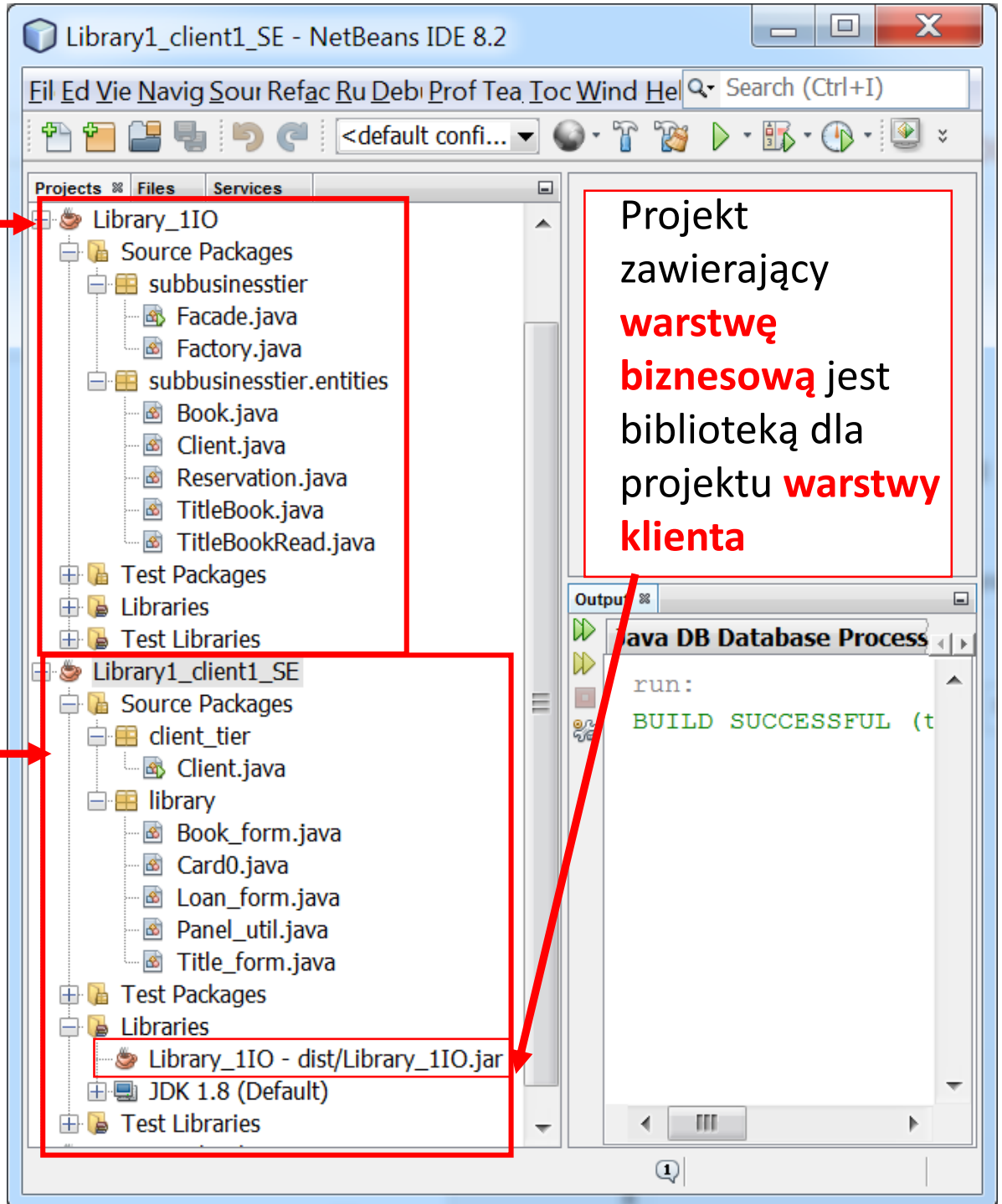
Wykonanie aplikacji dwuwarstwowej na platformie Java SE dla 1-go użytkownika, wykorzystująca kod z początkowych iteracji tworzenia programu bez GUI



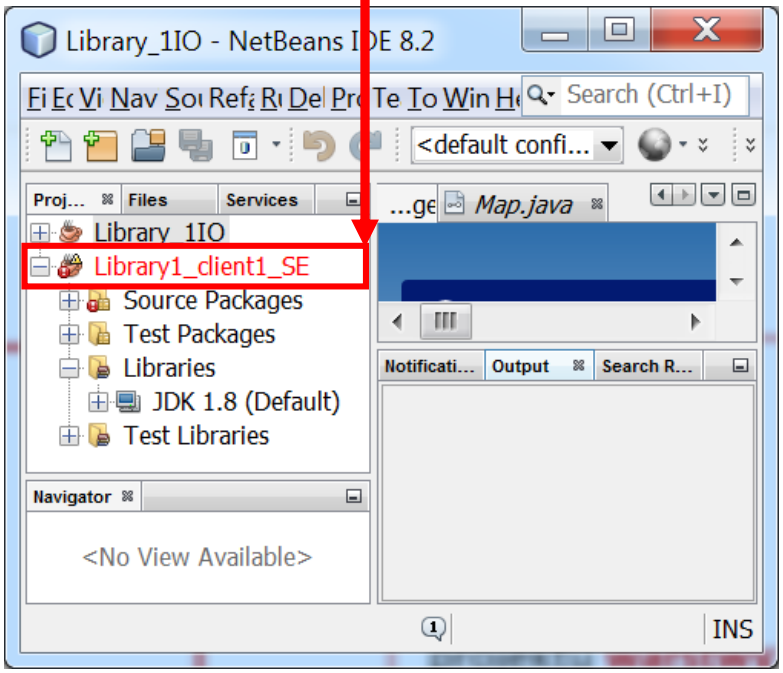
http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/INP002017/Library1SE.rar
Link do spakowanych dwóch projektów

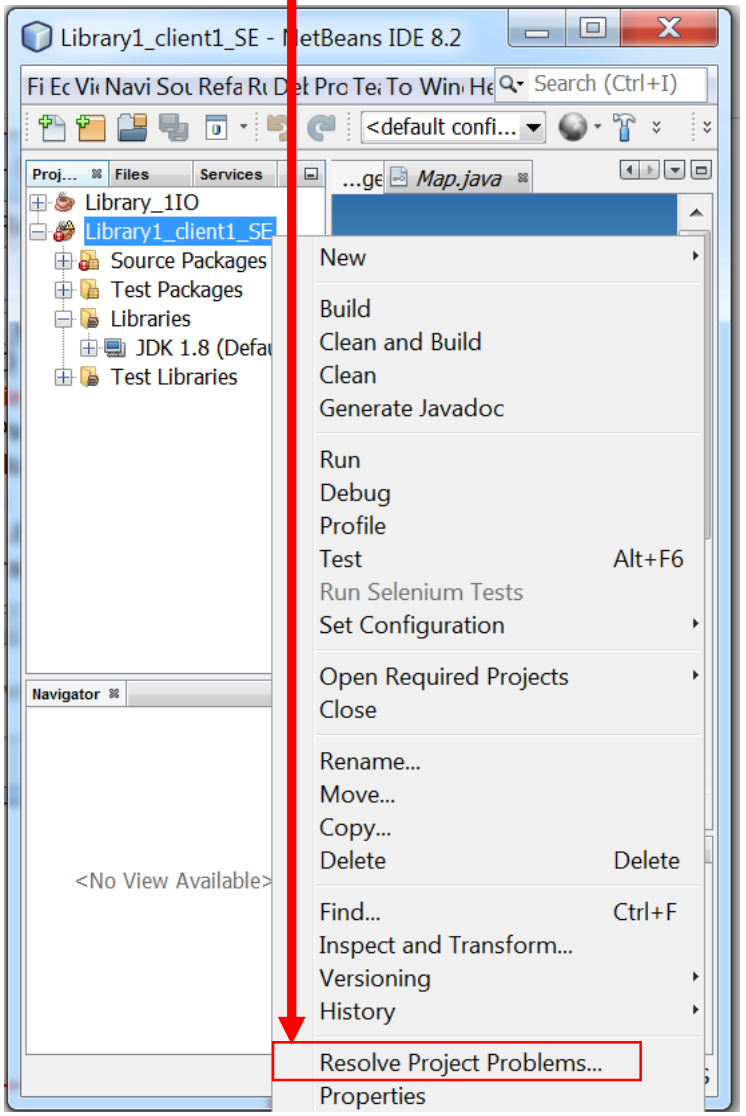
Projekt typu Java Class Library zawierający kod warstwy biznesowej wykonany podczas 4 iteracji

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

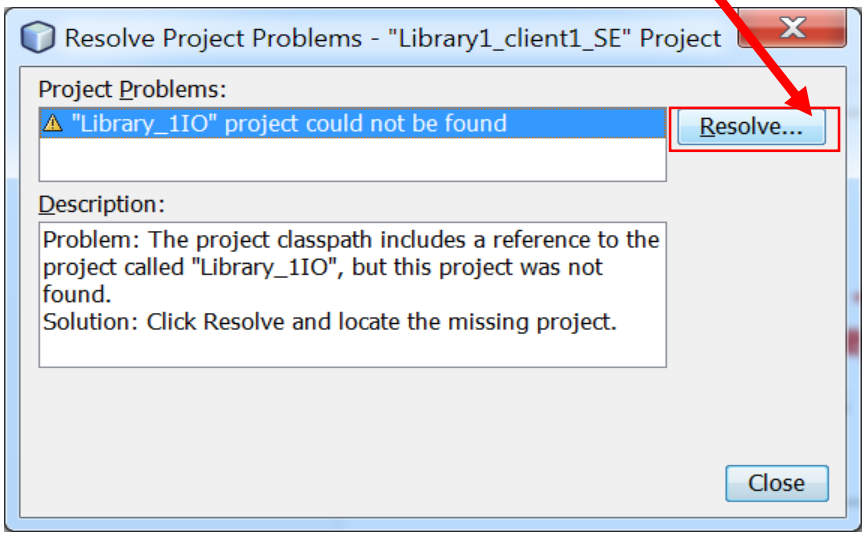


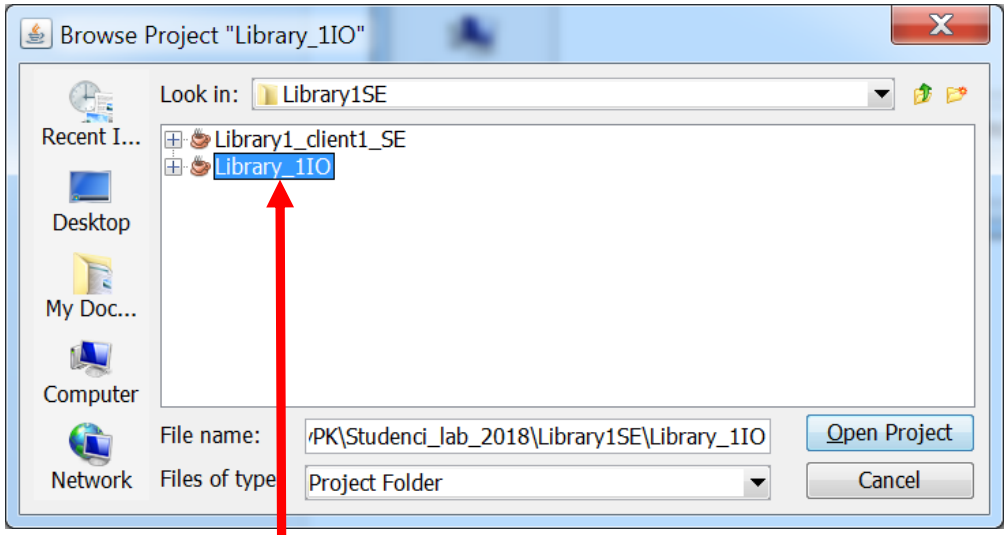
Jeśli po rozpakowaniu dwóch projektów pojawi się informacja o błędzie w projekcie Library1_client_SE, wtedy należy prawym klawiszem myszy wybrać wybrać ten projekt i wybrać pozycję Resolve Project Problems...



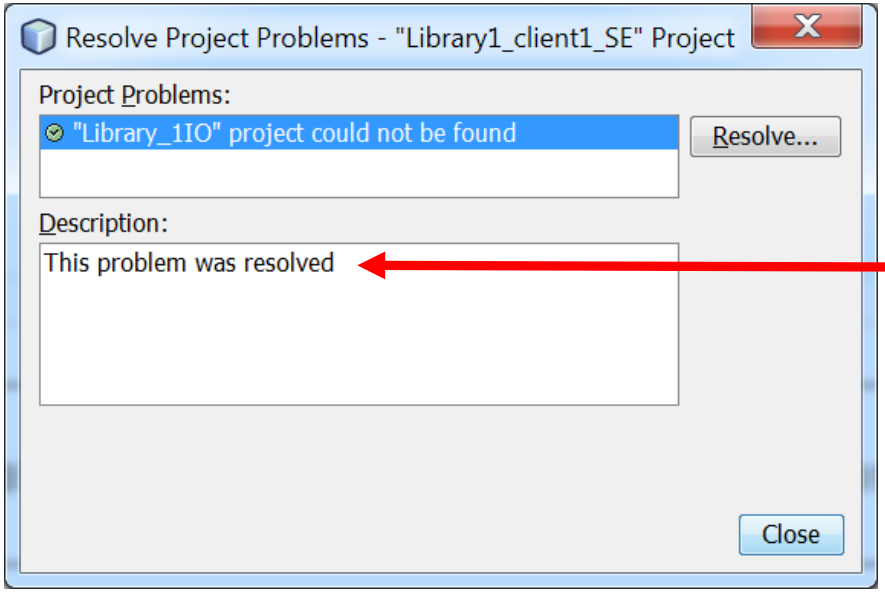


Należy nacisnąć klawisz Resolve...



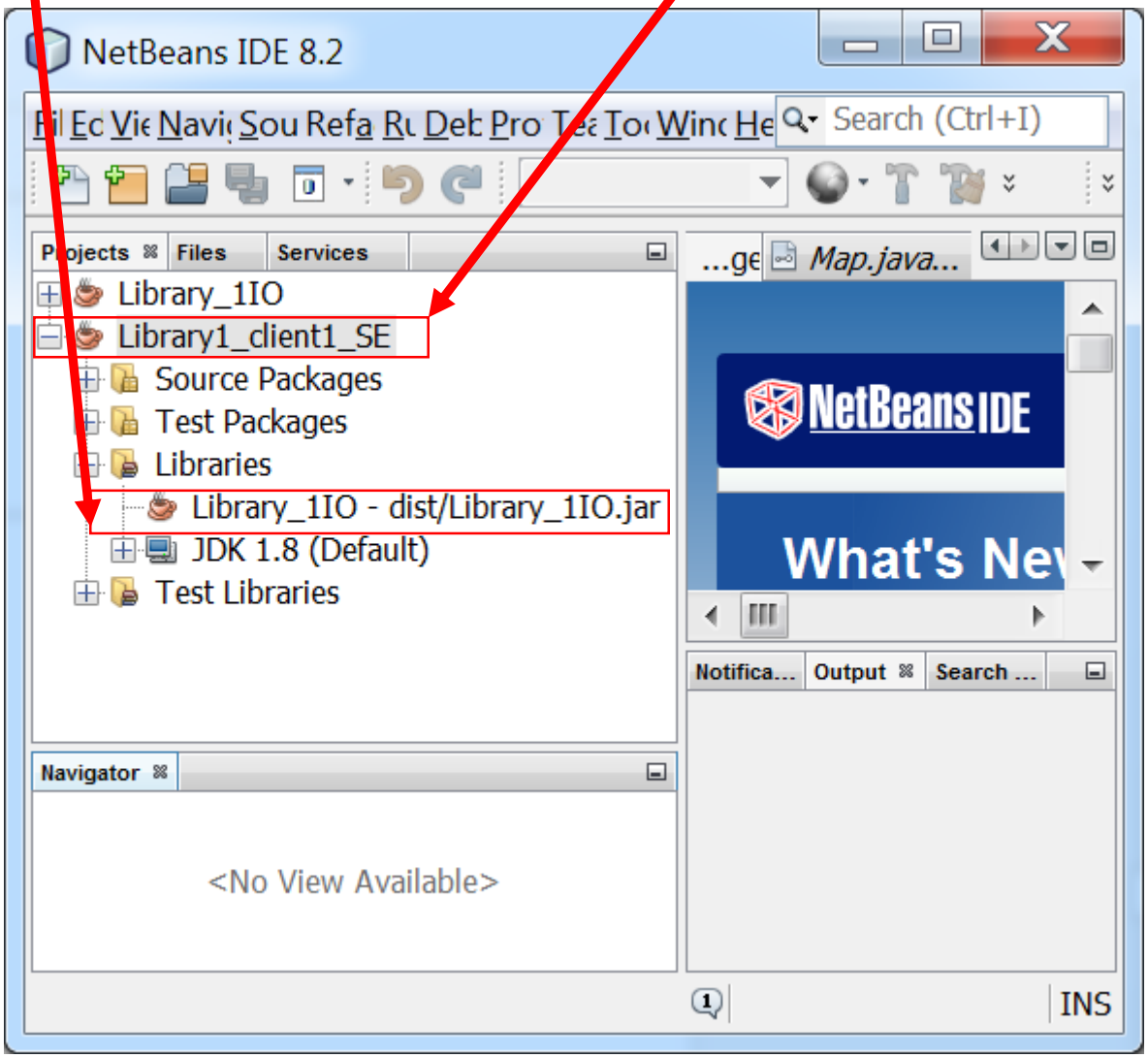


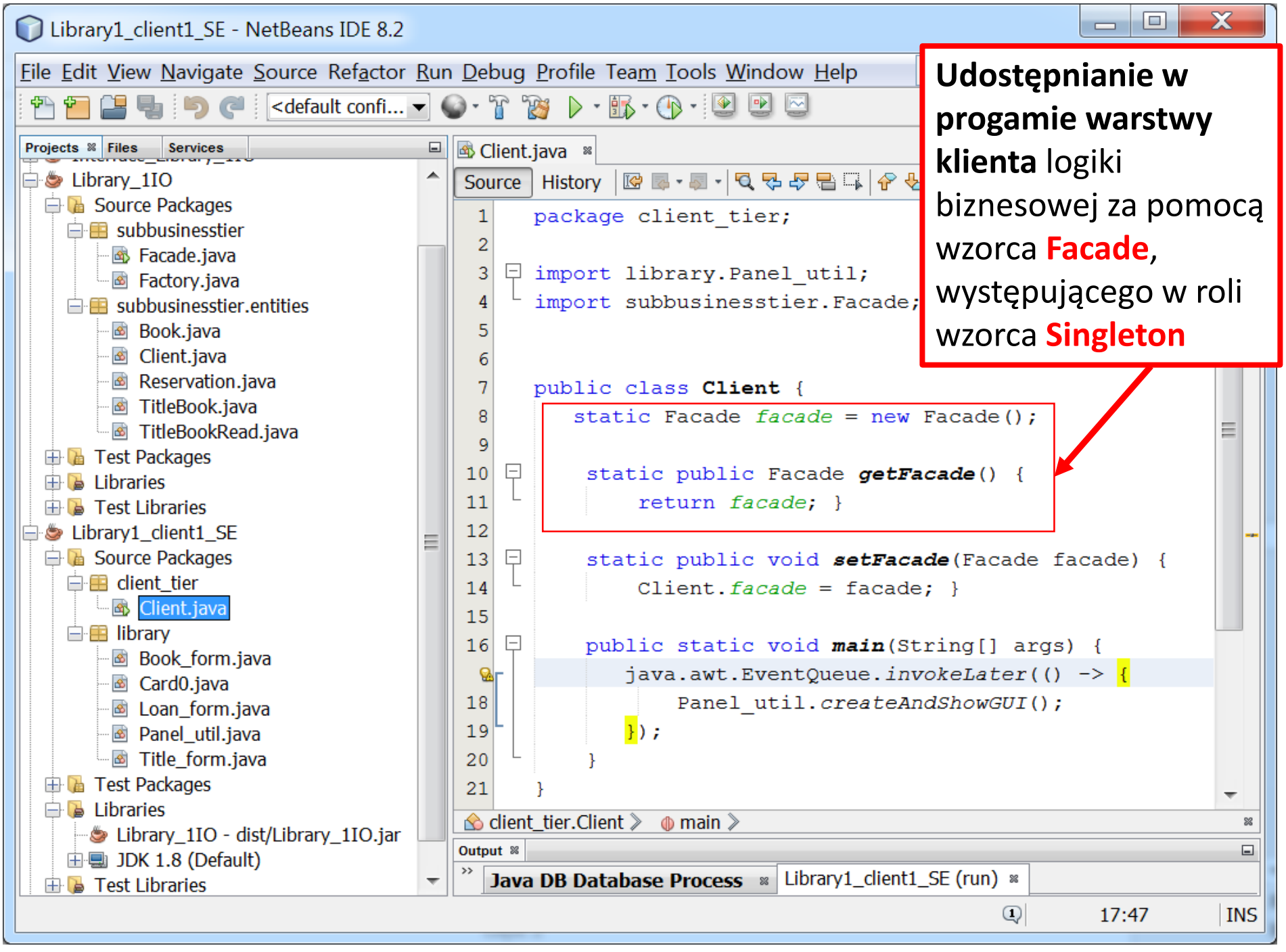
Należy wybrać projekt Library_1IO i nacisnąć klawisz Open Project

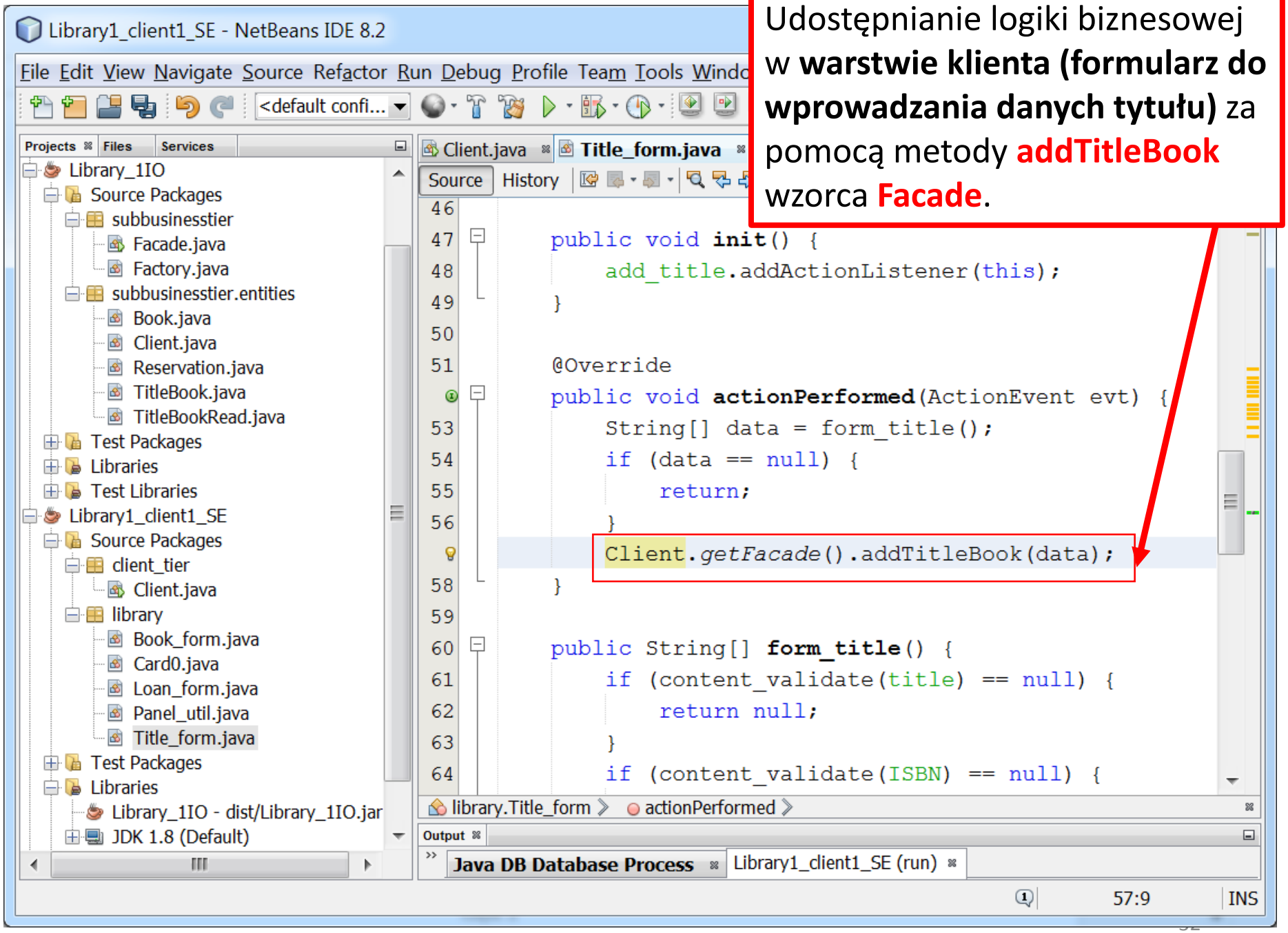


Problem brak kontaktu z projektem zawierającym logikę biznesową, zostal rozwiązany

Rezultat – ponownie zostało dodane powiązanie projektu Library_1IO z programem Library1_client2_SE




Library1_client1_SE - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
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;
3 import library.Panel_util;
4 import subbusinesstier.Facade;
7 public class Client {
8 static Facade facade = new Facade();
10 static public Facade getFacade() {
11 return facade; }
13 static public void setFacade(Facade facade) {
14 Client.facade = facade; }
16 public static void main(String[] args) {
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
```

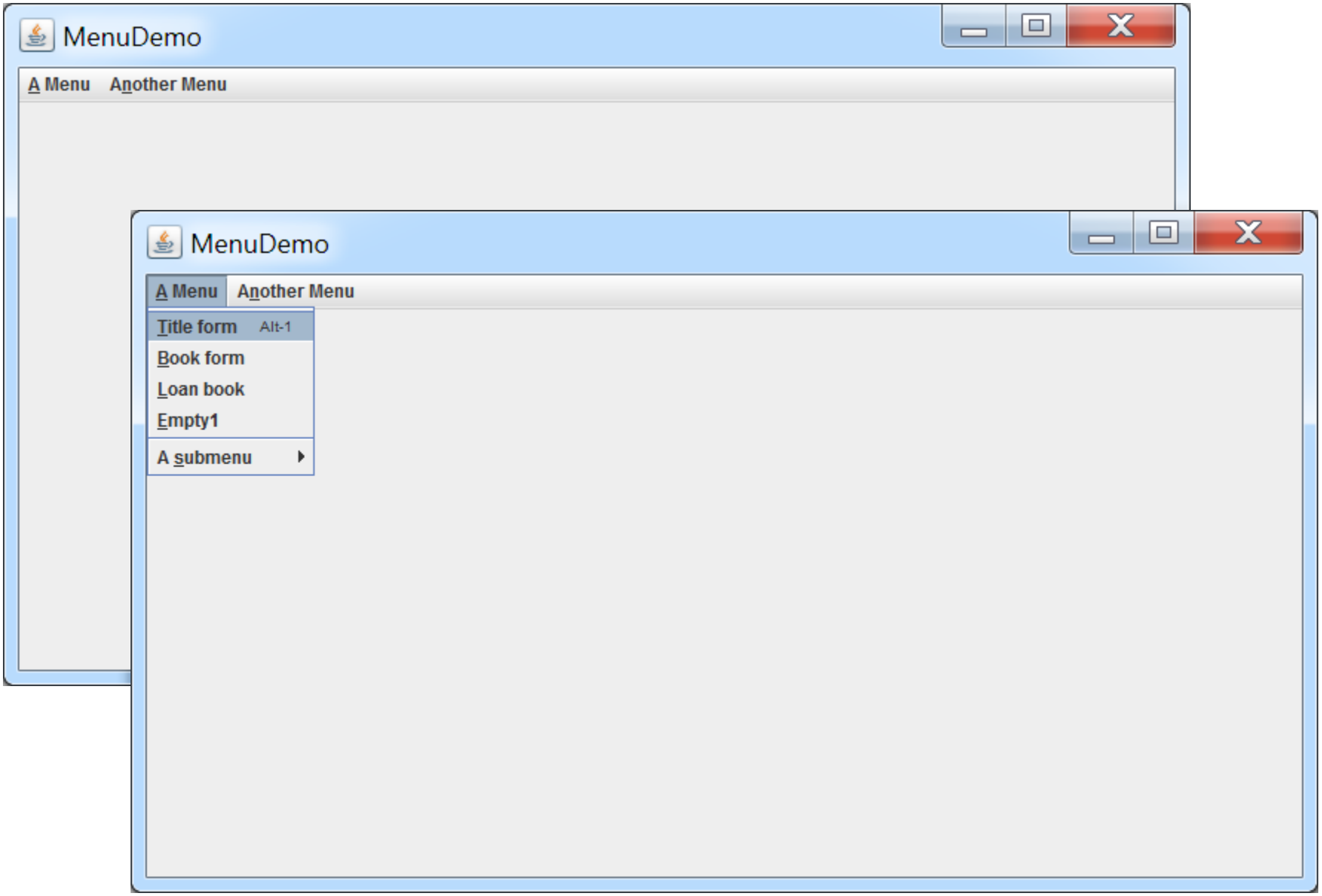
Wynik metody **getTitkeBookModel** jest wykorzystany jako **model widoku JTable**

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

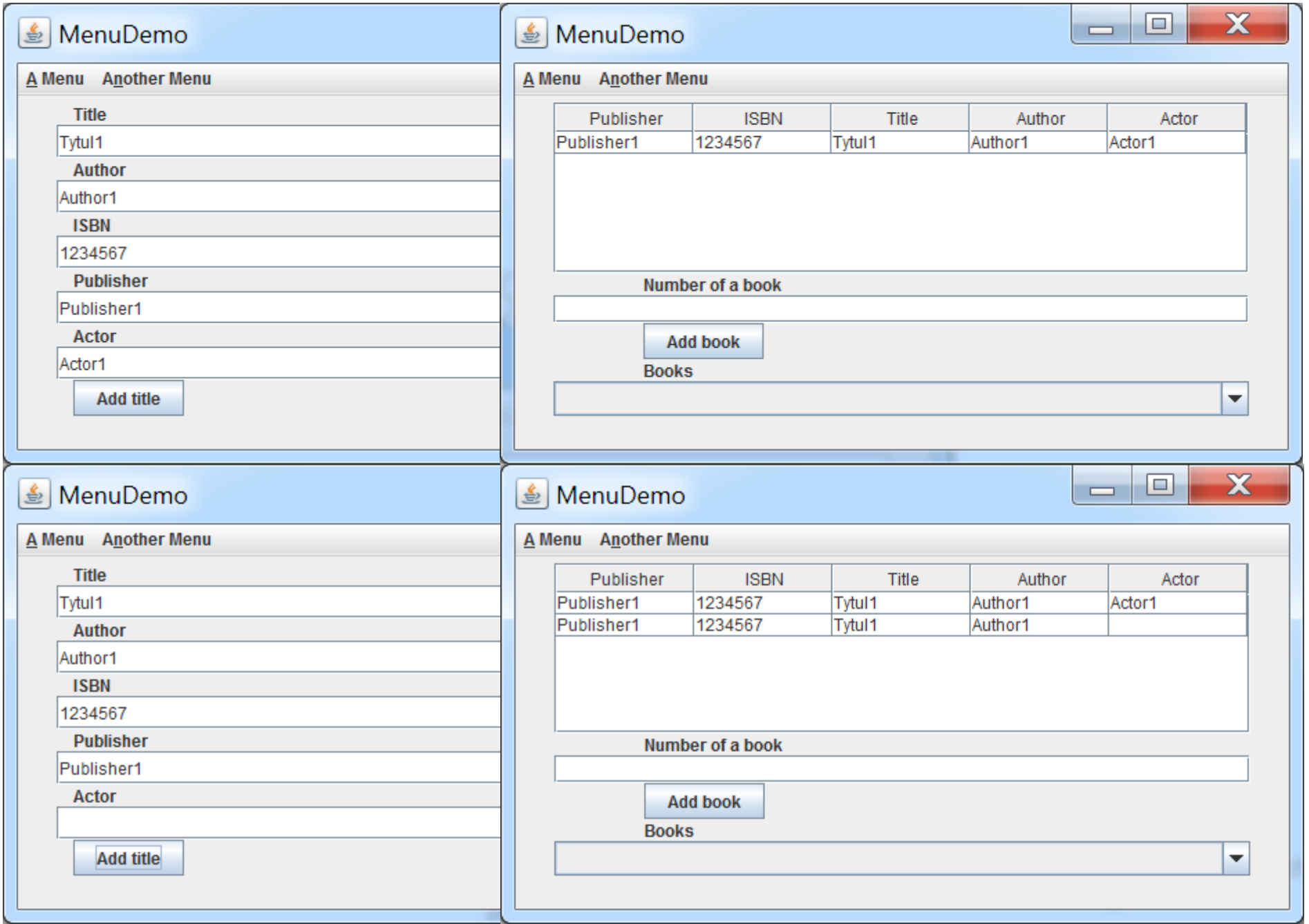
Wynik **help3** metody **addBook** jest wykorzystany jako **model widoku JComboBox**

```
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);  }
   }
```

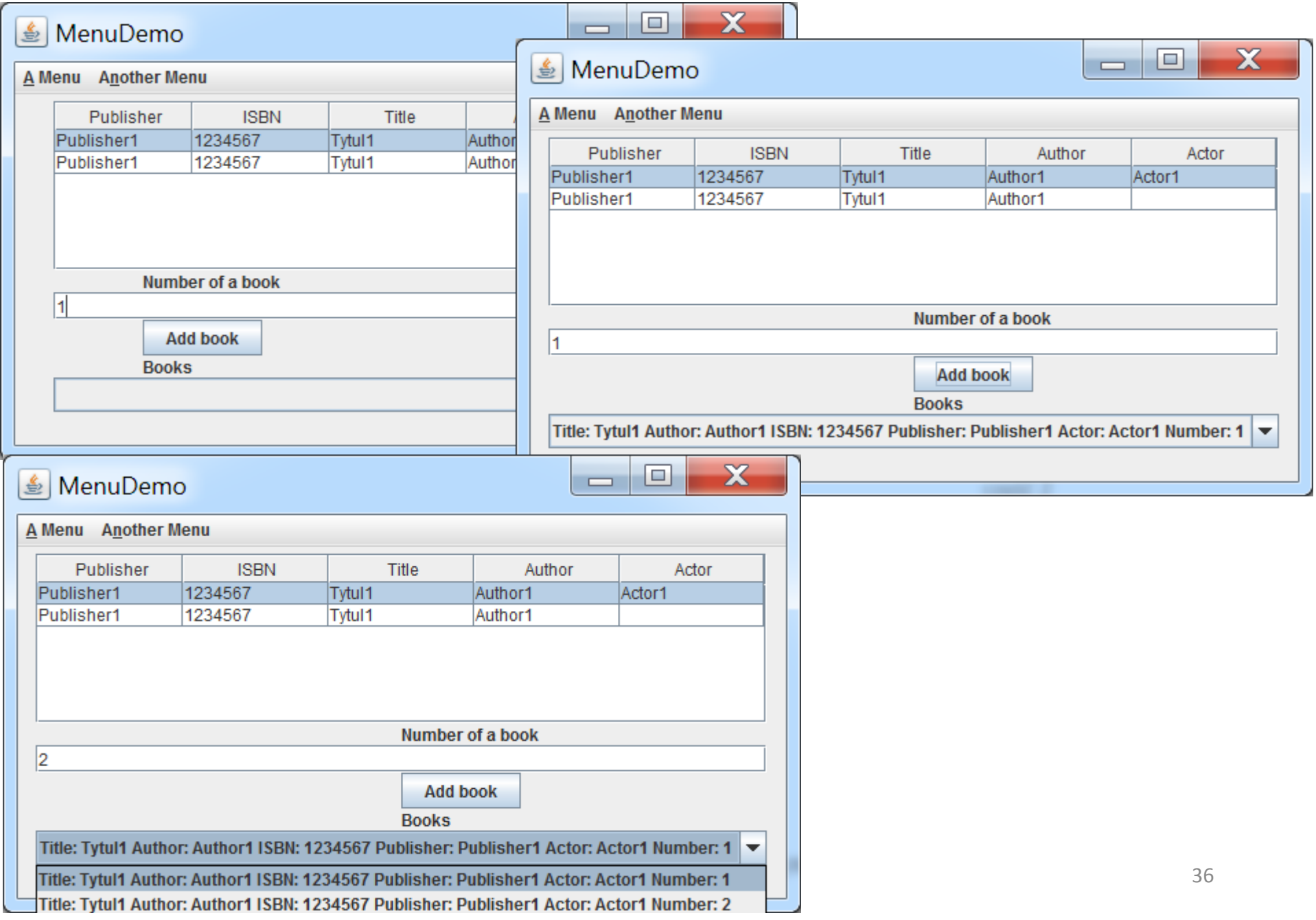
Dostęp do formularzy do wprowadzania danych tytułów i książek



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

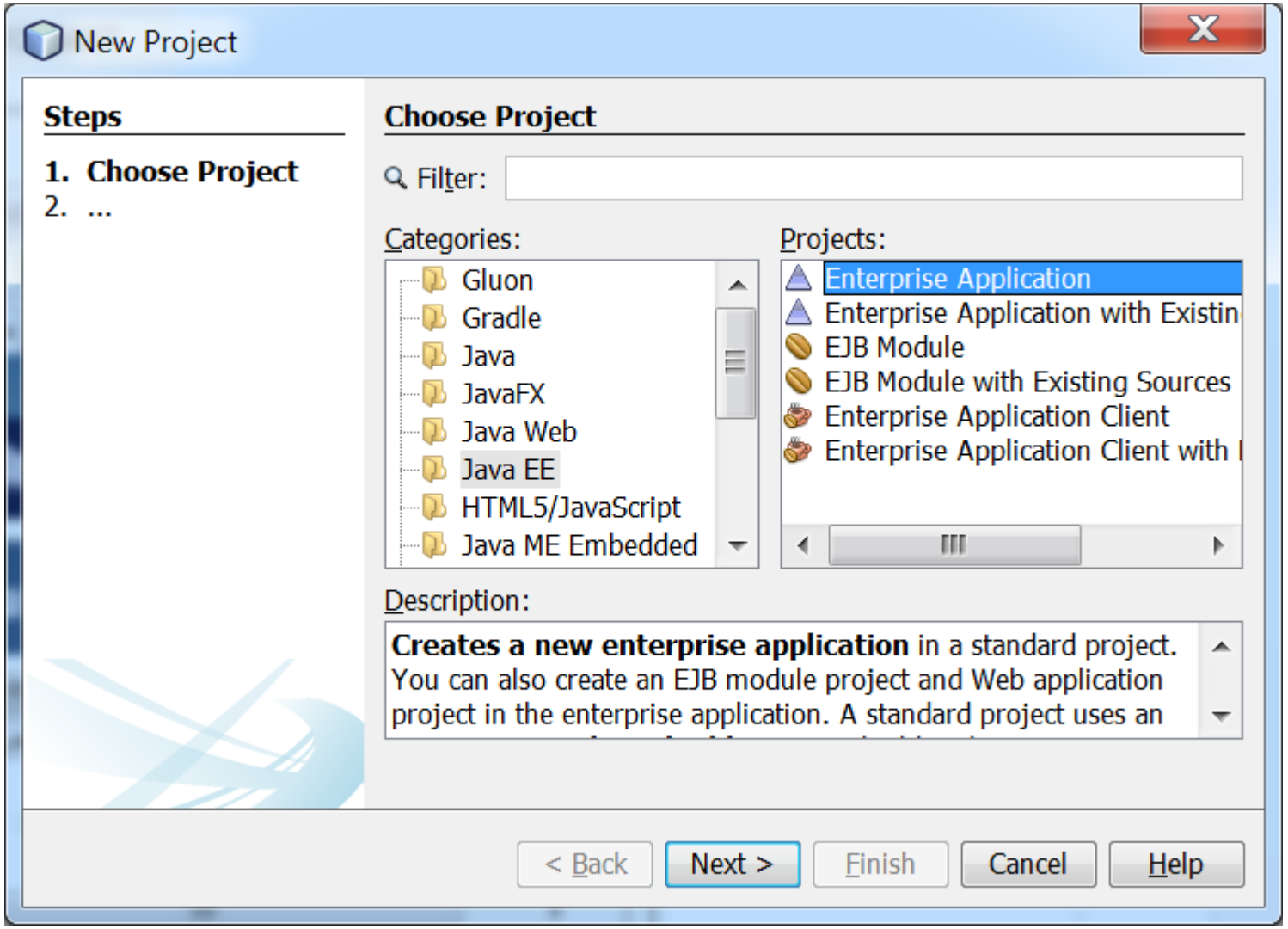


Wprowadzanie danych książek papierowych i nagranych

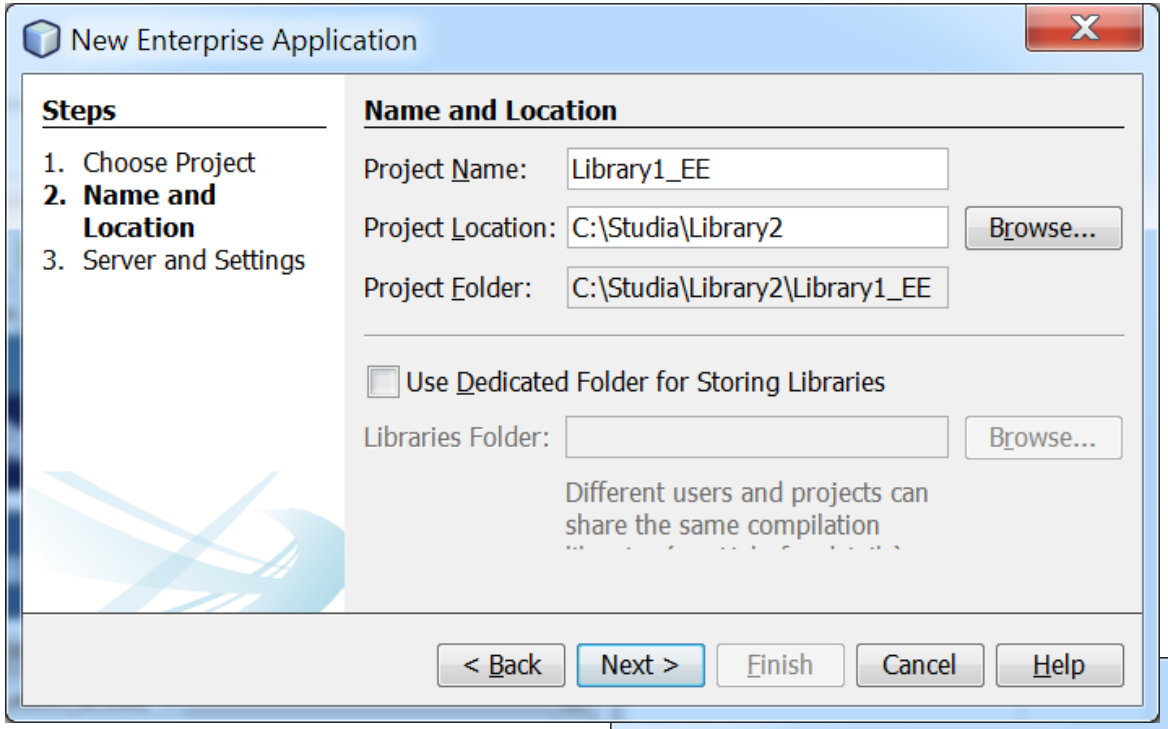


Wykonanie aplikacji typu Enterprise na platformie Java EE7 - wykorzystującej kod wykonany podczas tworzenia aplikacji dwuwarstwowej na platformie SE dla 1 użytkownika

1. Wykonanie projektu głównego: File/New Project.../JavaEE/Enterprise Application i naciśnięcie klawisza Next

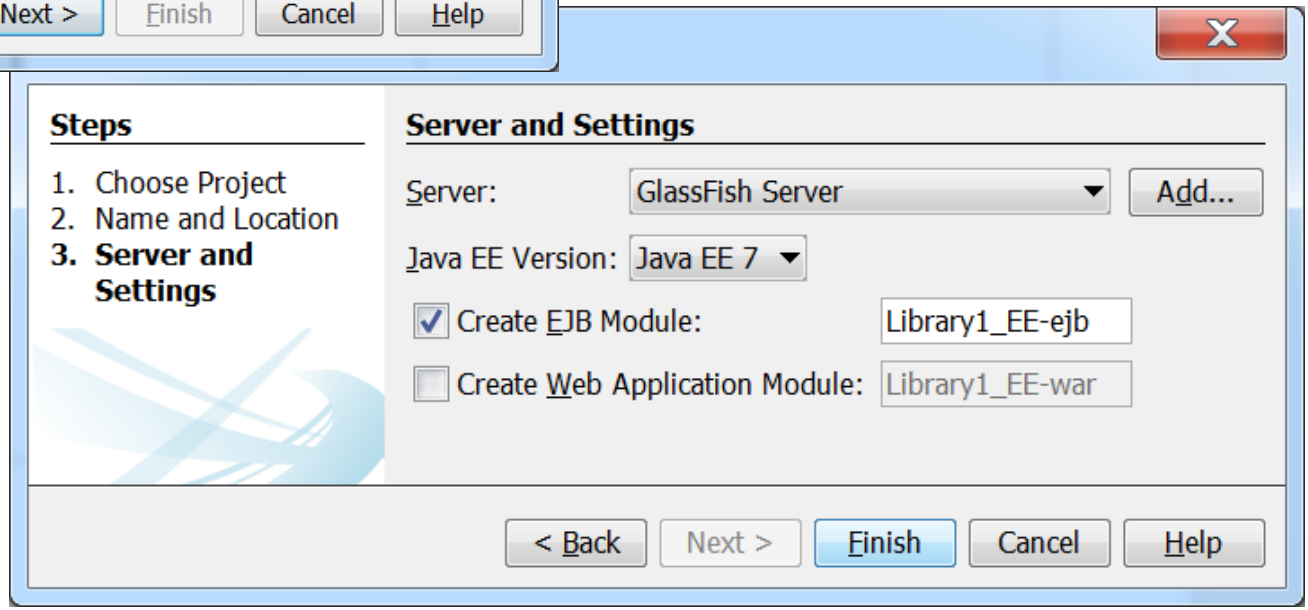


2. Podanie nazwy projektu (Project Name) oraz wybór lokalizacji (Project Location) i naciśnięcie klawisza Next

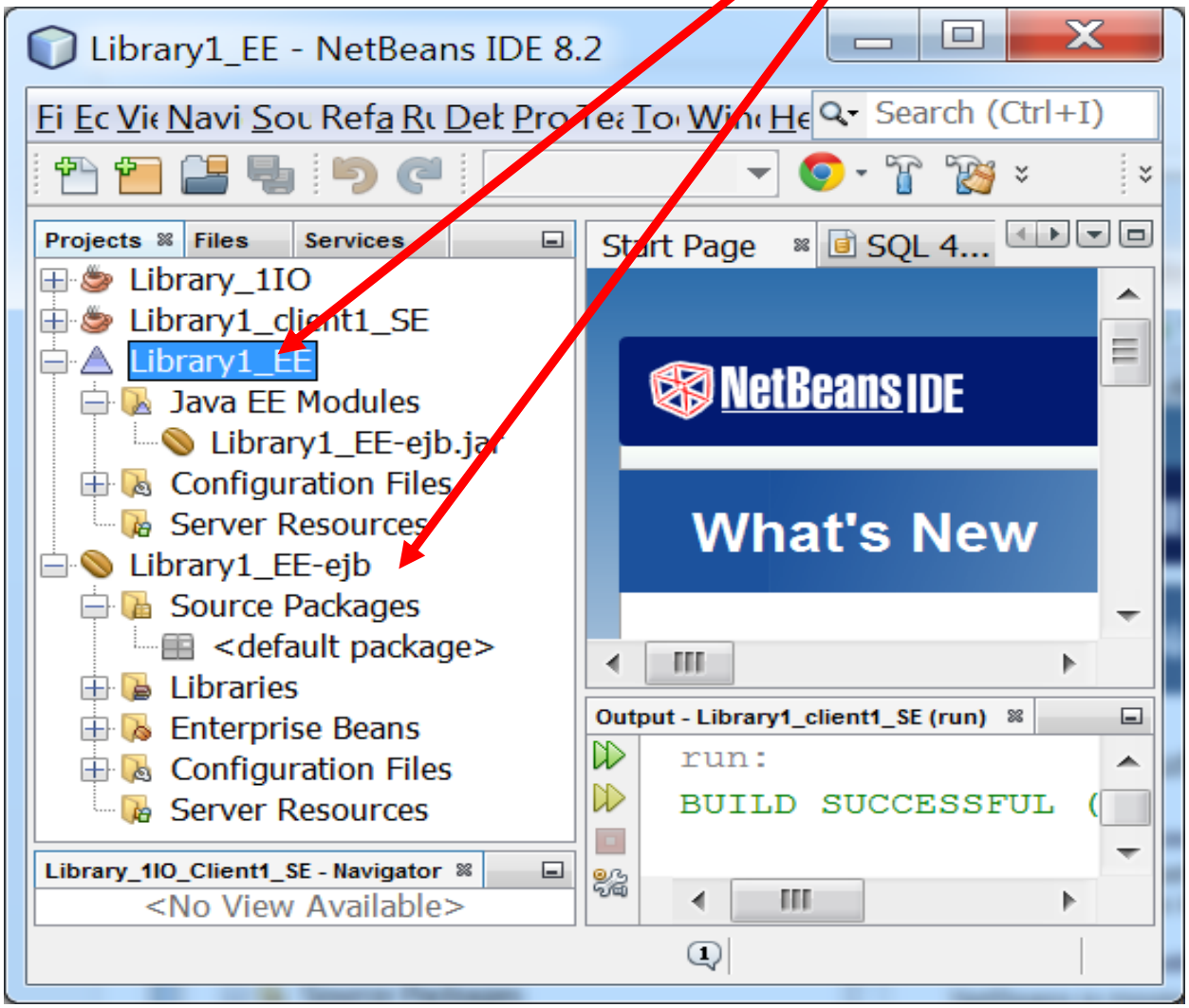


Uwaga: w nazwach katalogów należy stosować jedynie znaki z alfabetu angielskiego (duże i małe litery)

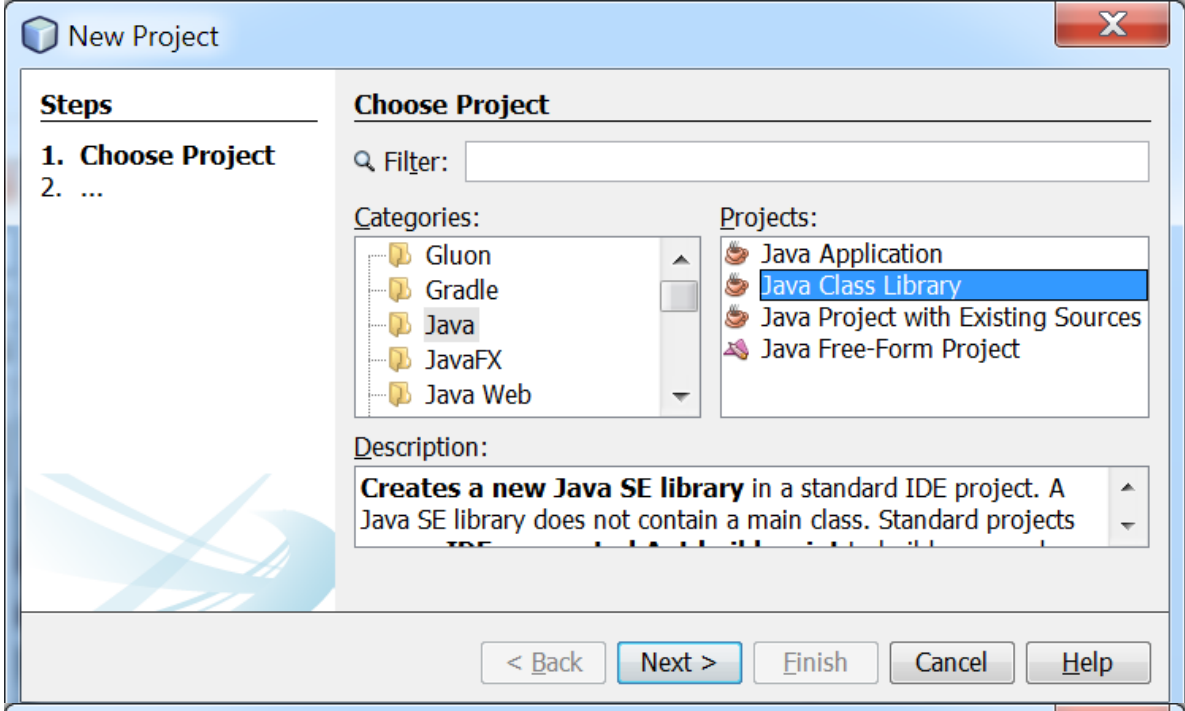
3. Wybór serwera aplikacji (Server) oraz wybór utworzenia tylko modułu EJB (Create EJB Module) i naciśnięcie Finish



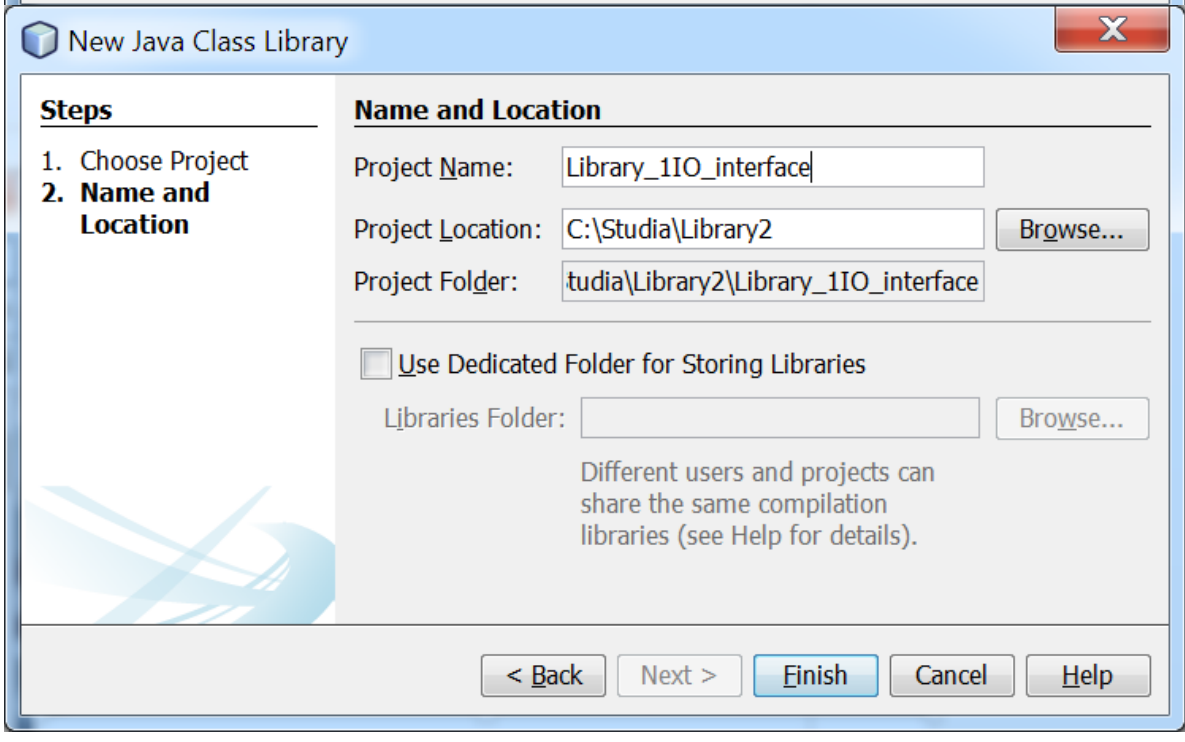
4. Wynik: założony został projekt główny typu Enterprise Application oraz pusty projekt typu moduł EJB



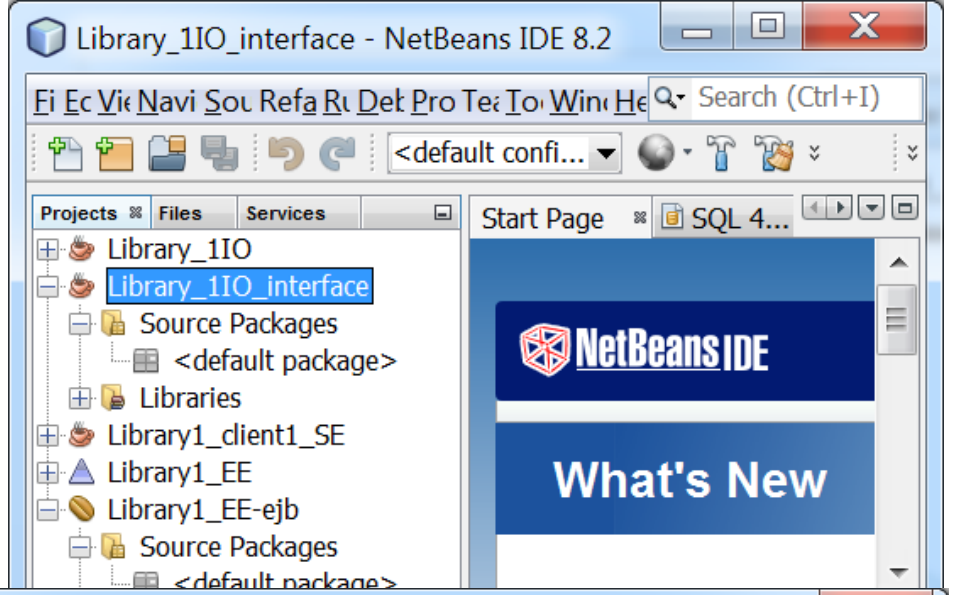
5. Wykonanie projektu typu Java ClassLibrary do przechowania interejsu komponentu EJB typu Session Bean do uzdalnienia dostępu do metod logiki boznesowej klasy Facade



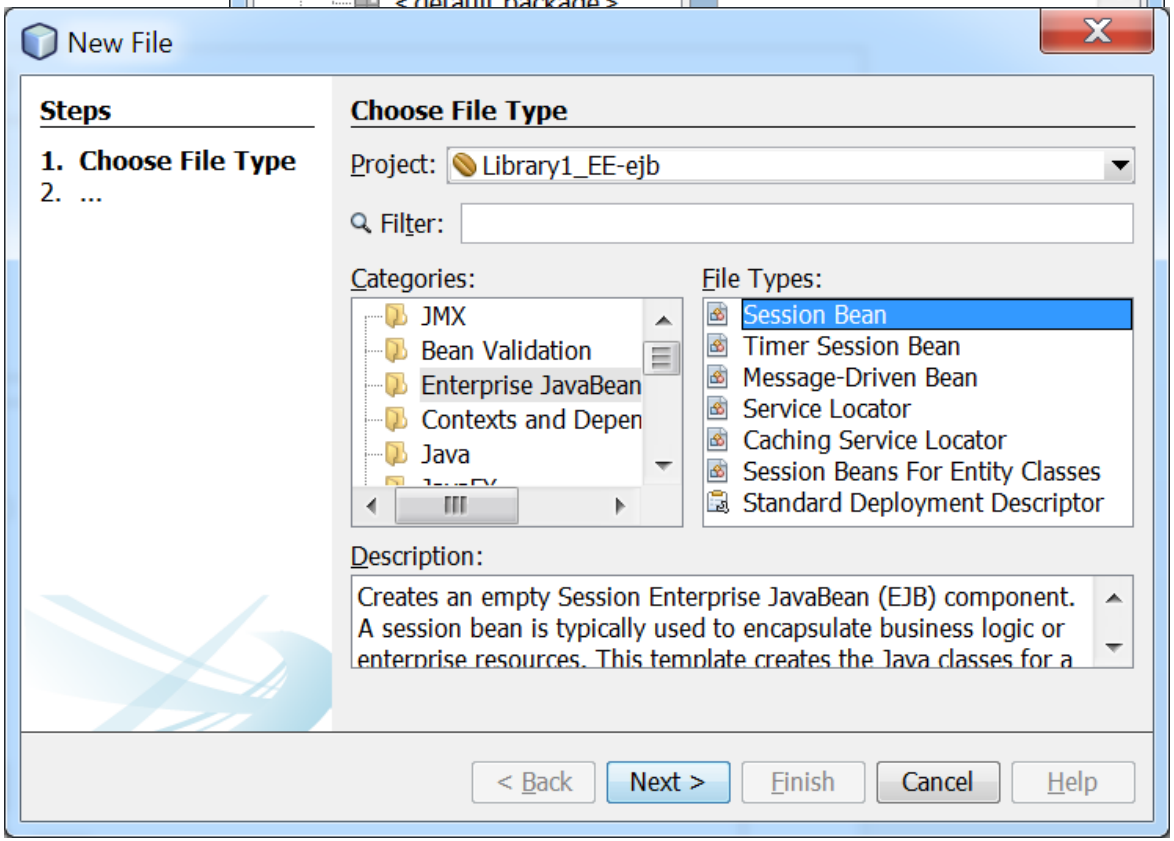
6. Należy wybrać: File/New Project.../Java/Java Class Library i nadać mu nazwę (Project Name) oraz lokalizację (Project Location) – taką samą, jak projekt typu Enterprise Application



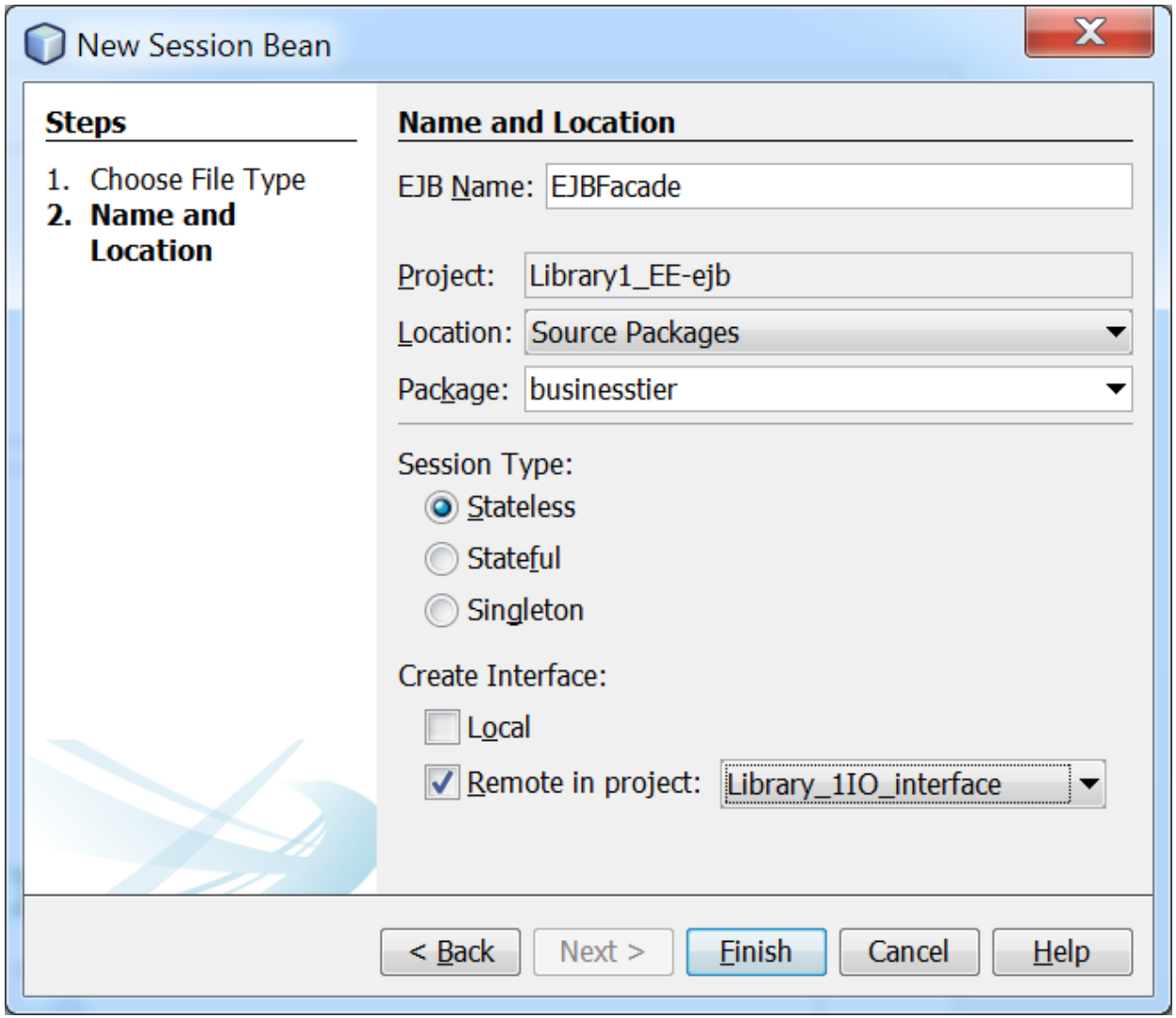
7. Wykonany projekt typu Java ClassLibrary do przechowania interejsu komponentu EJB typu Session Bean do uzdalnienia dostępu do metod logiki biznesowej klasy Facade



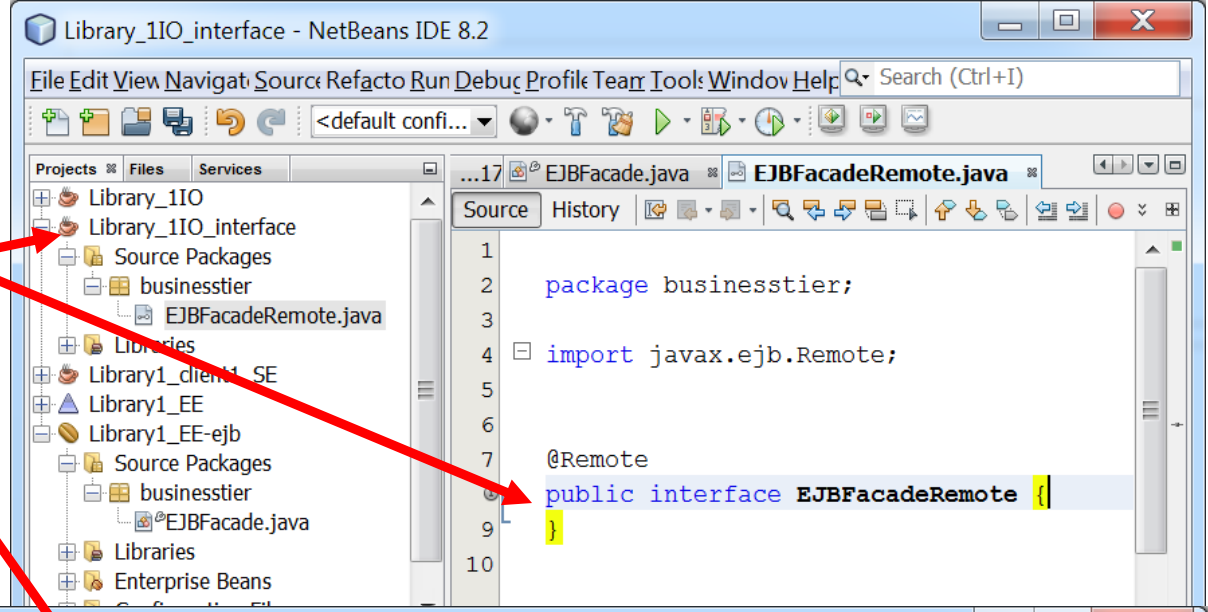
8. Wstawienie komponentu EJB typu Session Bean do modułu EJB do uzdalnienia dostępu do metod logiki biznesowej klasy Facade: prawym klawiszem myszy wybrać projekt typu moduł EJB, i wybrać pozycje: New/Other.../Enterprise JavaBean/Session Bean i nacisnąć na klawisz Next

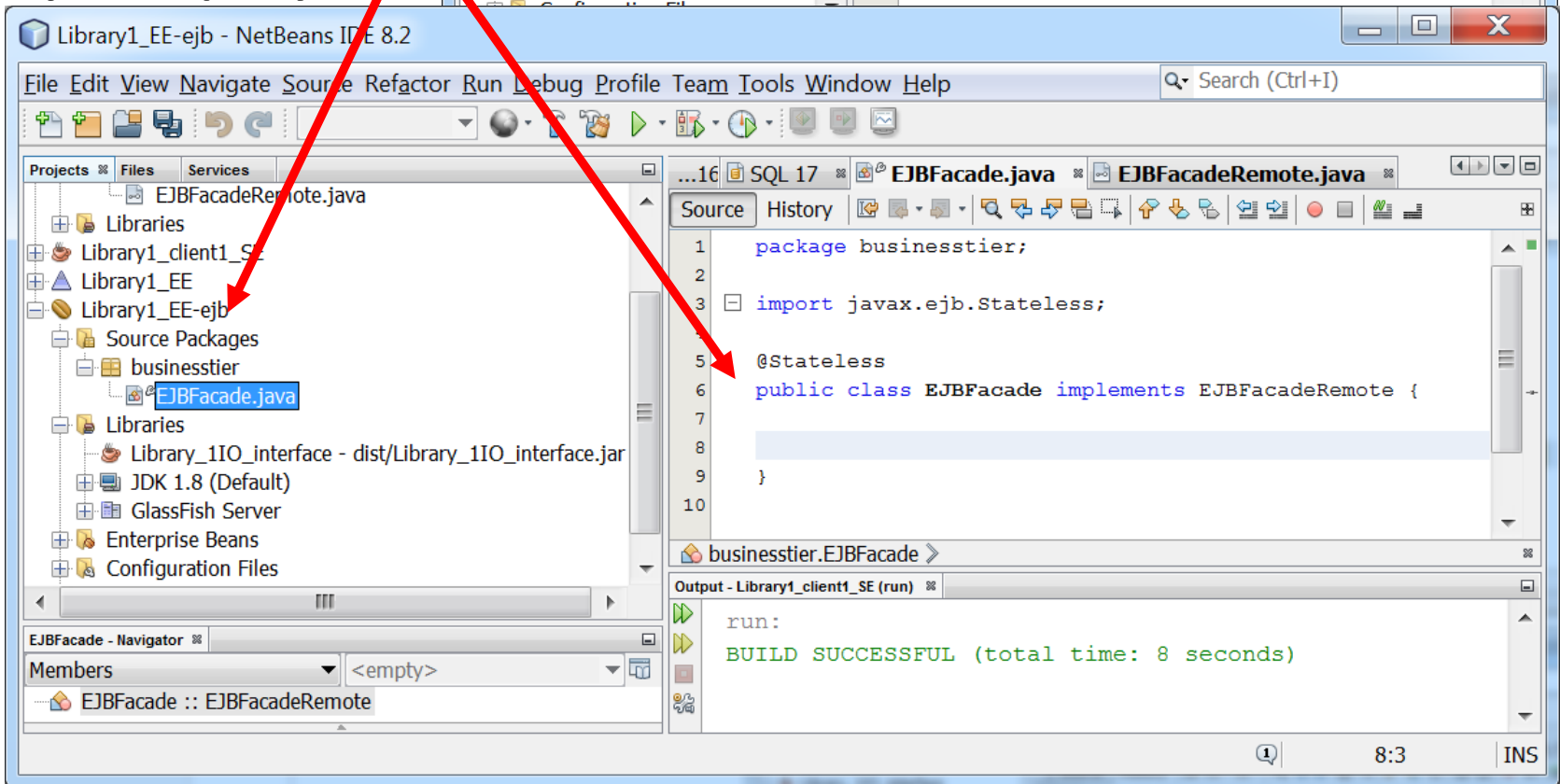


9. W polu EJB Name wpisać nazwę komponentu, w polu Package wpisać nazwę pakietu, w którym będzie przechowany tworzony obecnie komponent EJB. Należy wybrać w Session Type pozycję Stateless i w polu Create Interface zaznaczyć Remote. Wtedy z listy należy wybrać utworzony wcześniej projekt typu Java Class Library. Następnie należy nacisnąć klaswisz Finish.

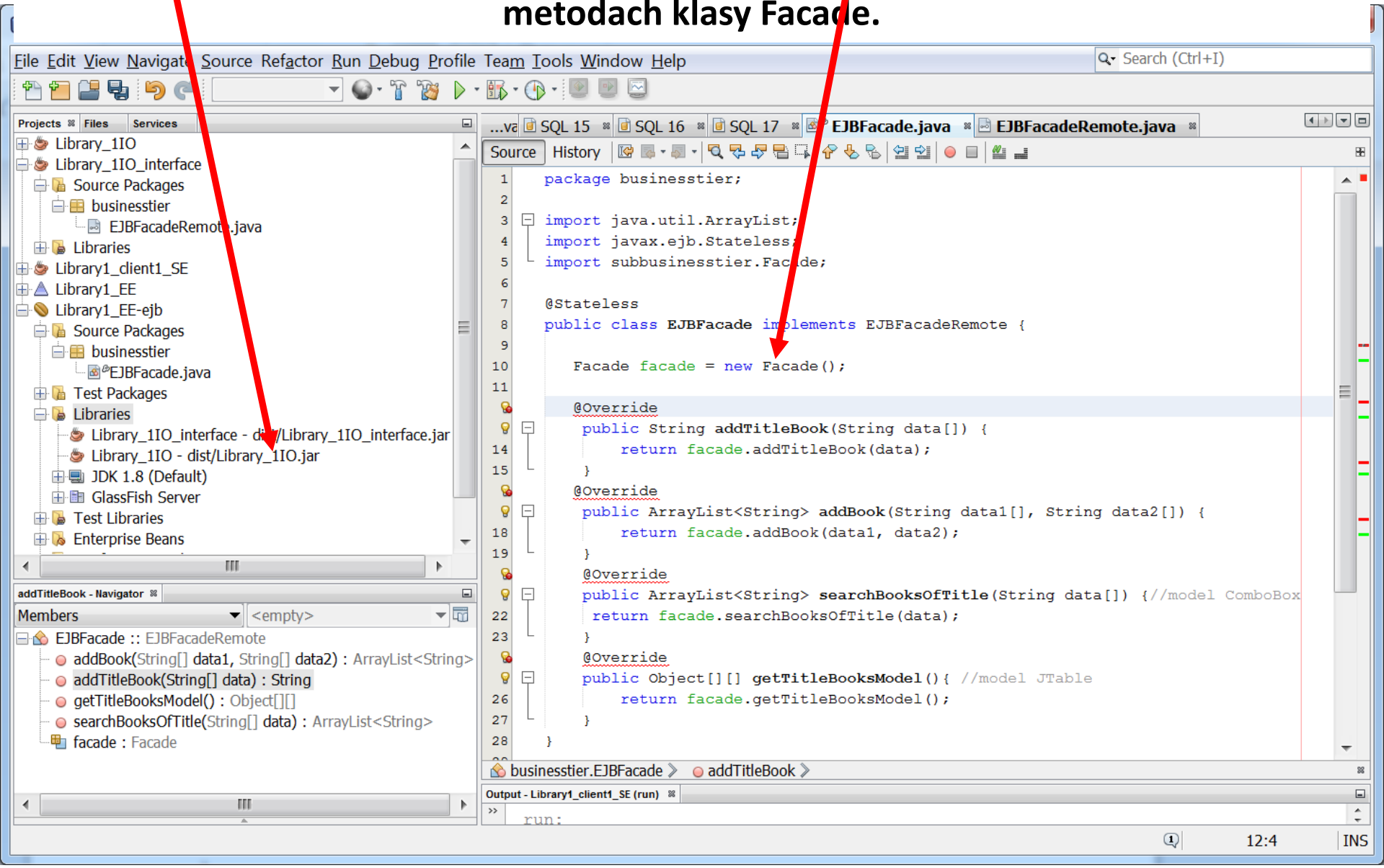


10. W projekcie typu Java Class Library utworzył się pusty interfejs o nazwie komponentu EJB typu Session Bean z przyrostkiem Remote. W projekcie typu moduł EJB utworzył się pusty komponent typu EJB implementujący dany interfejs – też pusty.

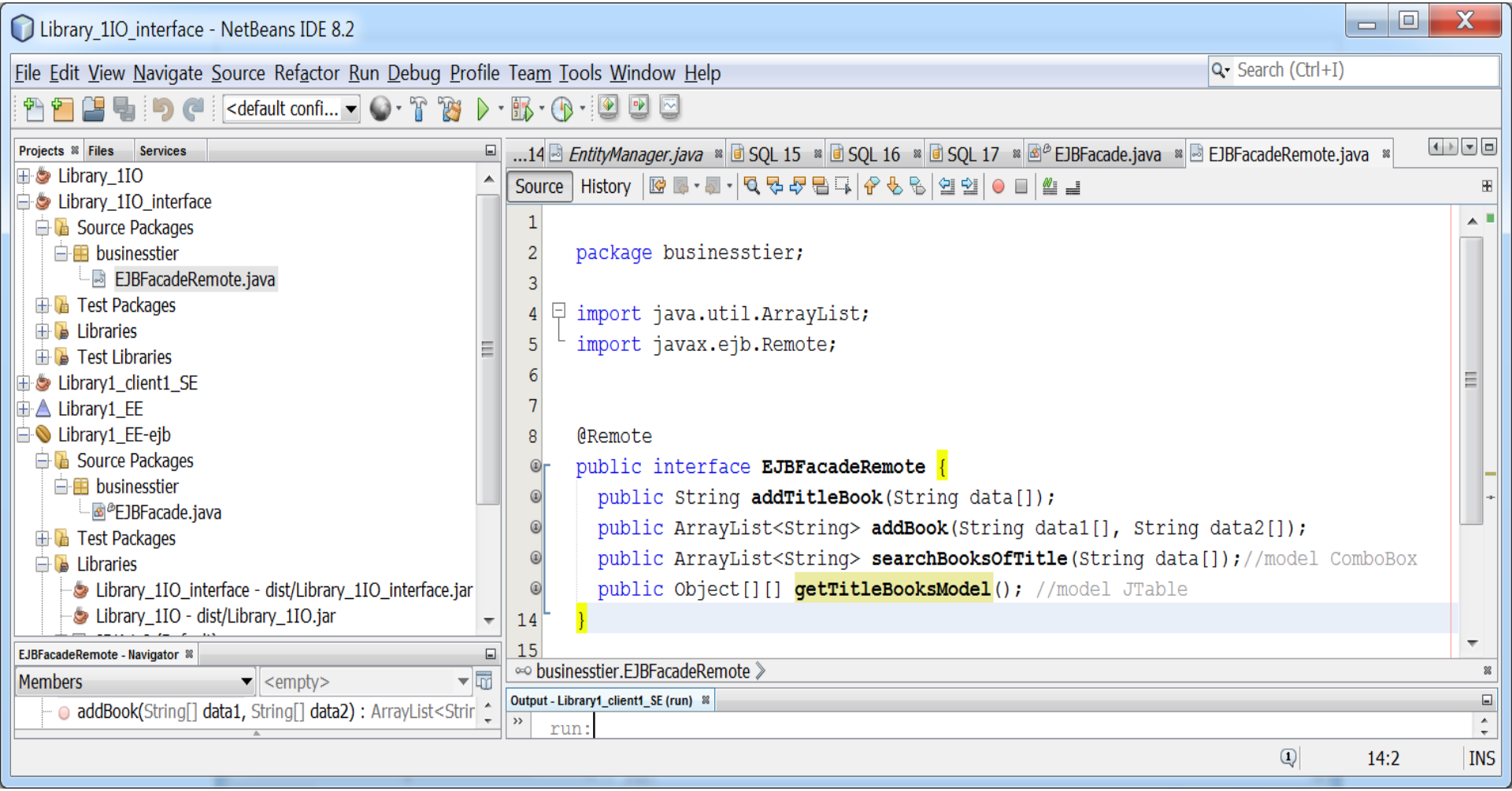




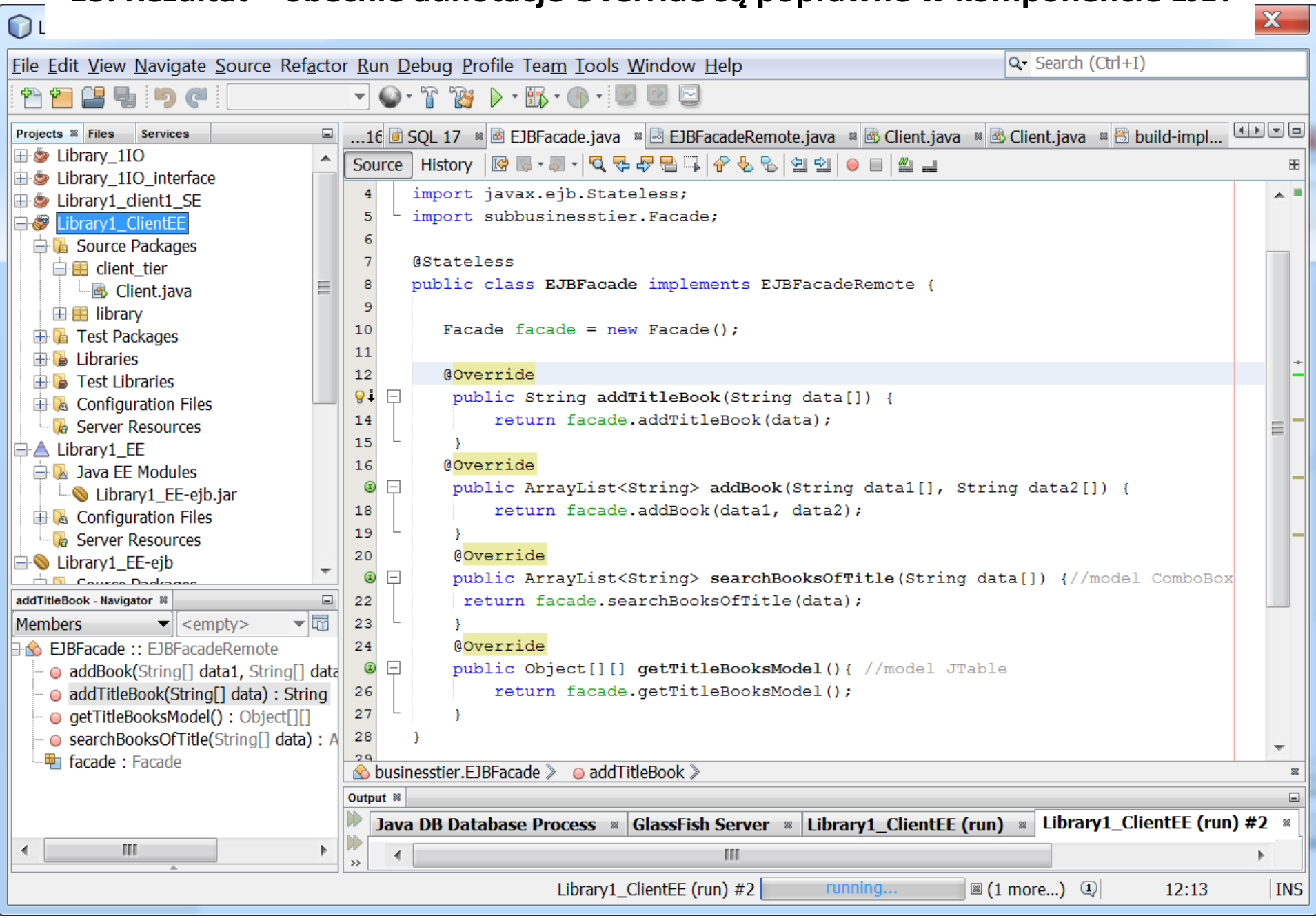
11. Należy w projekcie modułu EJB w folderze Libraries dodać projekt z logiką biznesową. Należy zdefiniować obiekt typu Facade i zdefiniować te metody, które były wywołane w projekcie warstwy klienta. Ciało tych metod jest oparte na metodach klasy Facade.

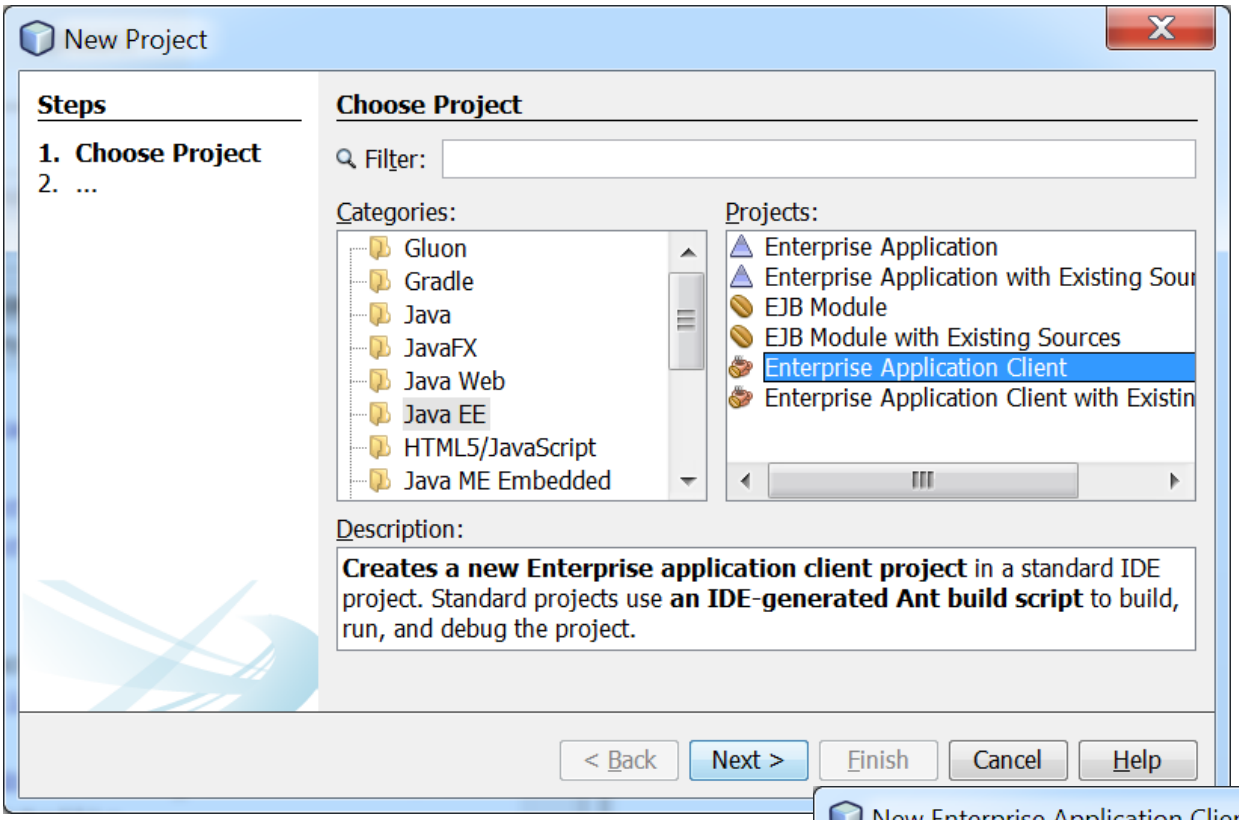


12. Należy w projekcie typu Java Class Library w interfejsie komponentu EJB zadeklarować metody zdefiniowane w komponencie EJB, znajdującym sie w projekcie modułu EJB



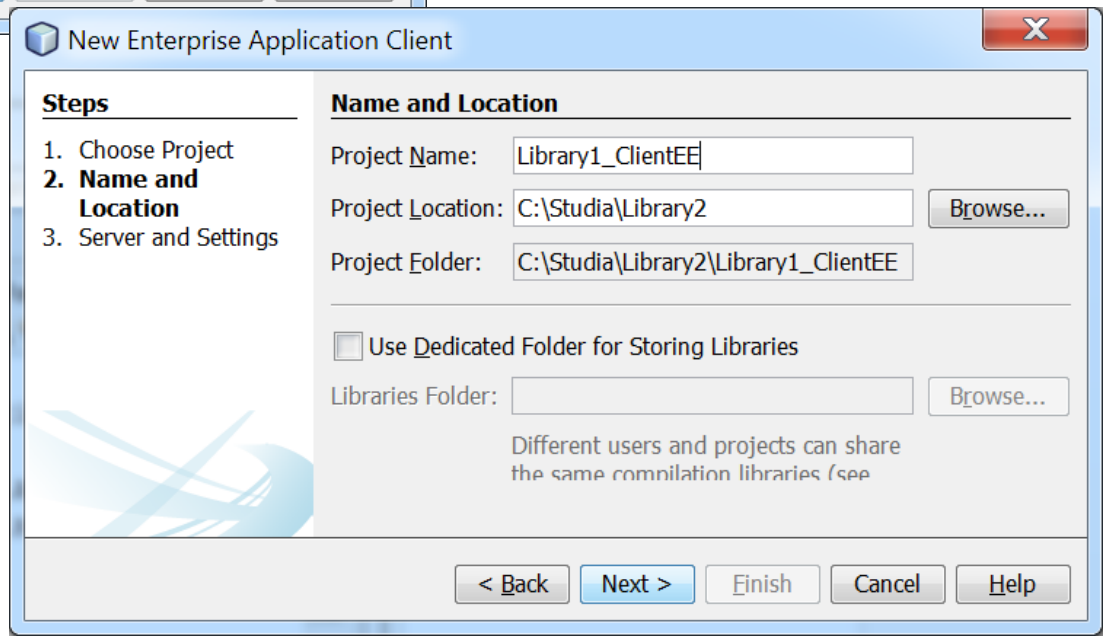
13. Rezultat – obecnie adnotacje Override są poprawne w komponencie EJB.



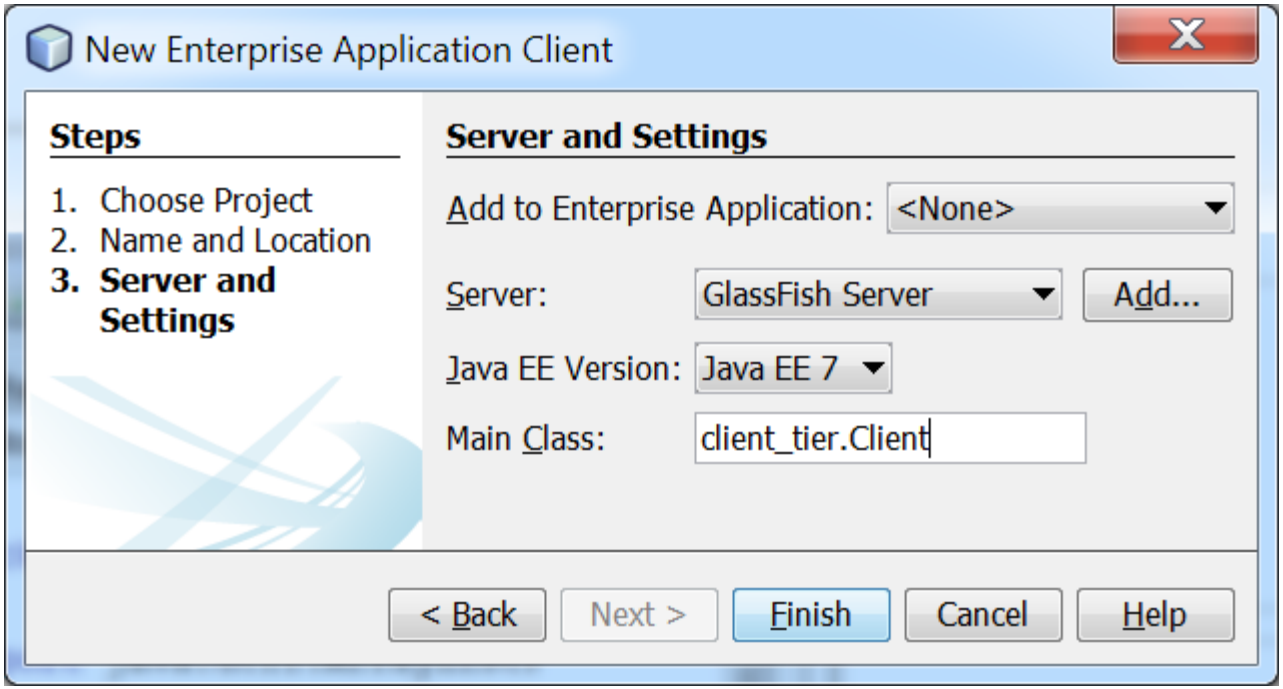


14. Należy wykonać projekt typu Enterprise Application Client: File/New Project.../JavaEE/Enterprise Application Client i nacisnąć klawisz Next

15. Należy wypełnić pole Project Name podając nazwę projektu oraz Project Location podając nazwę katalogu projektu i należy nacisnąć klawisz Next

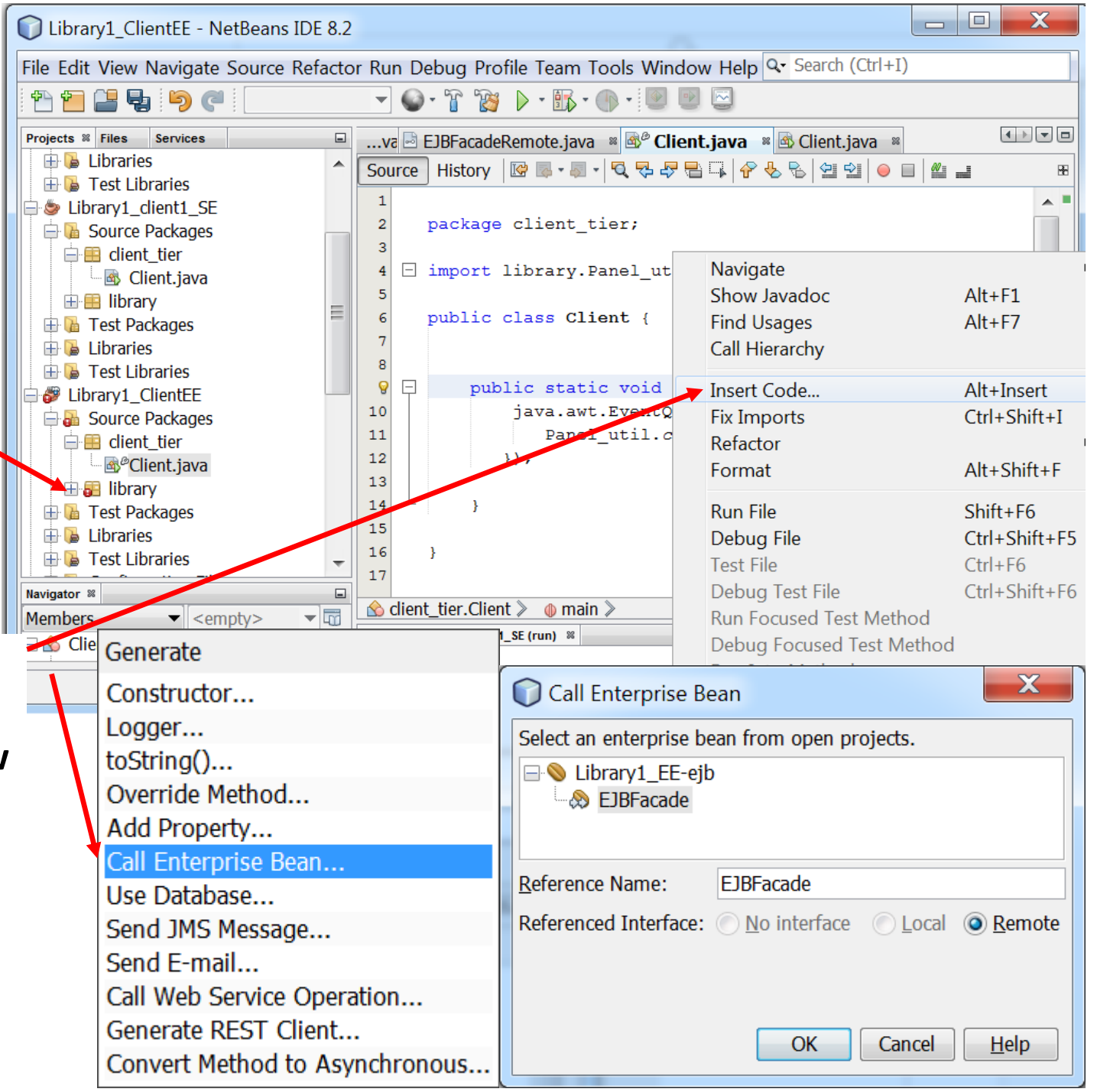


16. Kod tworzonego projektu typu Enterprise Application Client będzie wykonany w opaciu o pakiet library z projektu warstwy klienta (aplikacja dwuwarstwowa). Zdefiniowano domyślną klasę Client w pakiecie client_tier (pole Main Class) w celu uniknięcia modyfikacji kodu klas w pakiecie library.



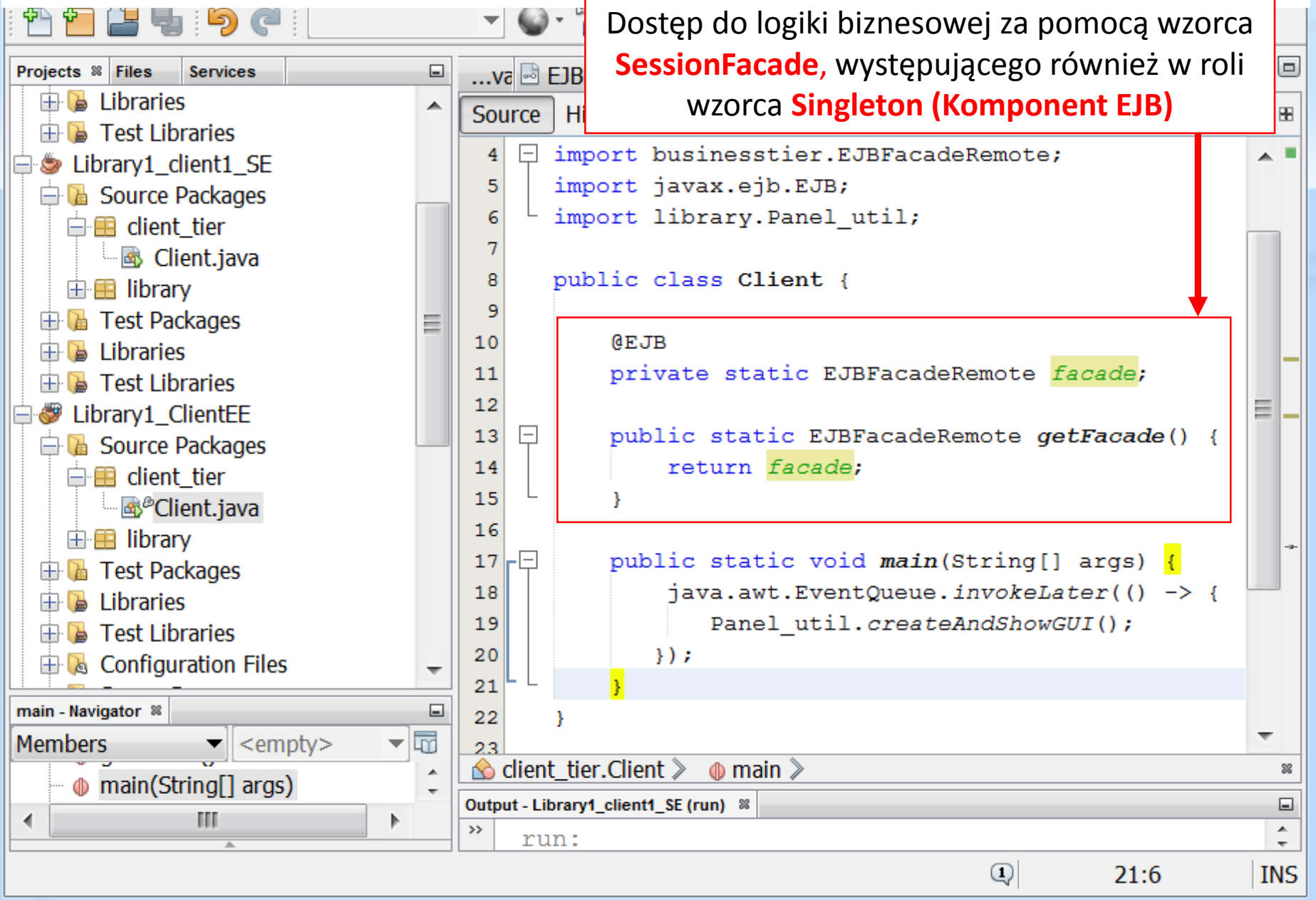
17. Skopiowanie pakietu library z aplikacji reprezentującej aplikację warstwy klienta.

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

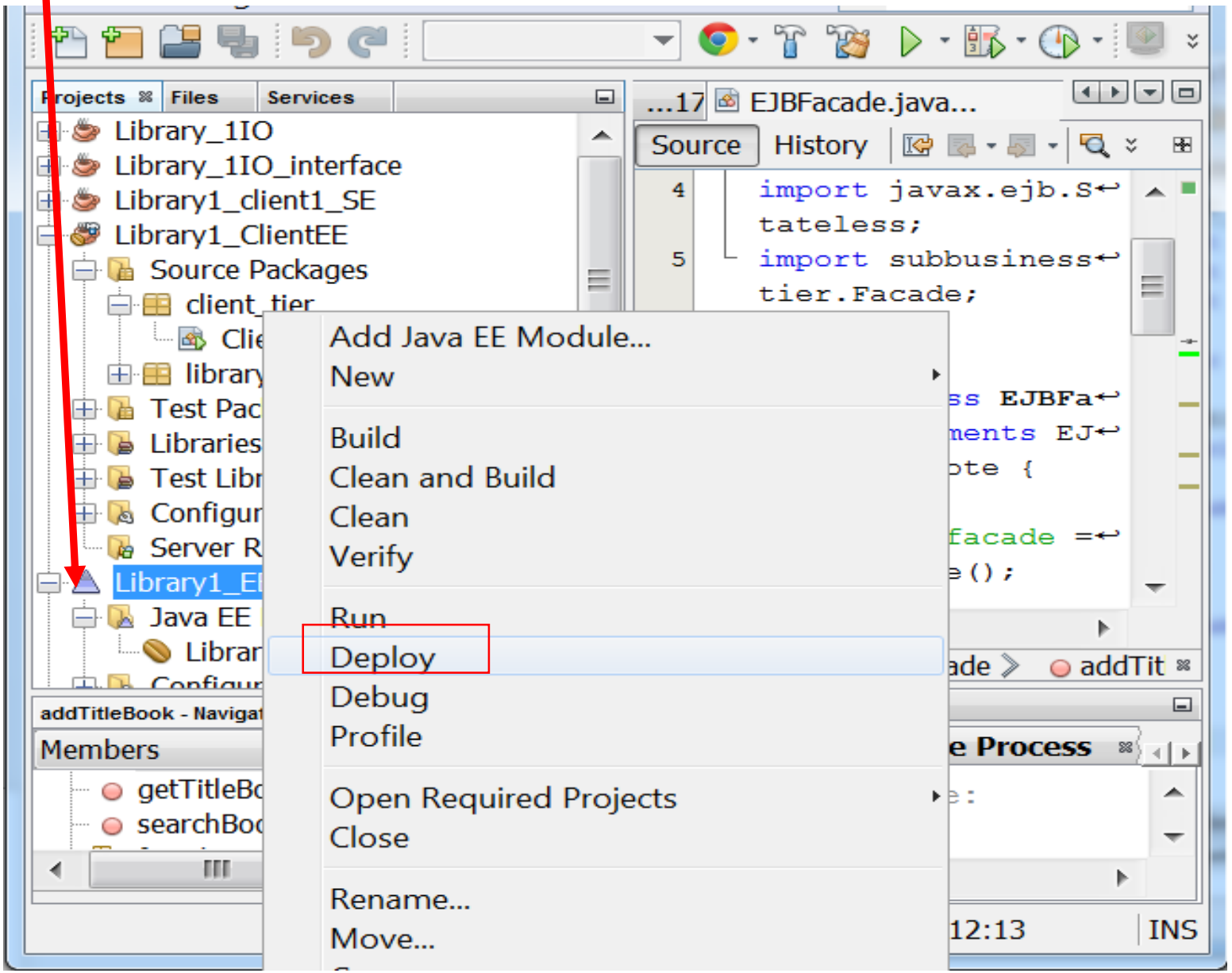


19. Wykonana aplikacja reprezentująca warstwę klienta EE

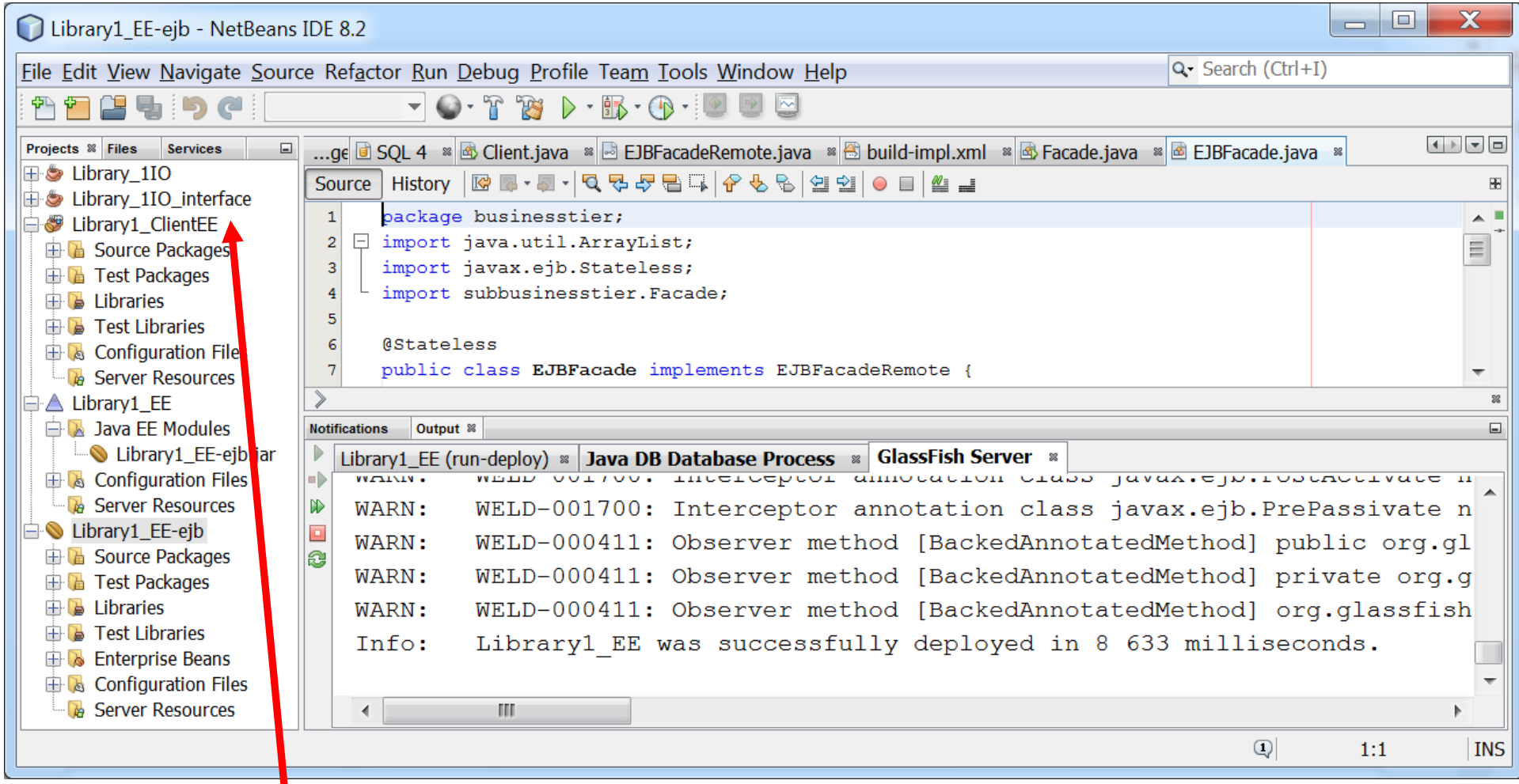
Dostęp do logiki biznesowej za pomocą wzorca **SessionFacade,** występującego również w roli wzorca **Singleton (Komponent EJB)**



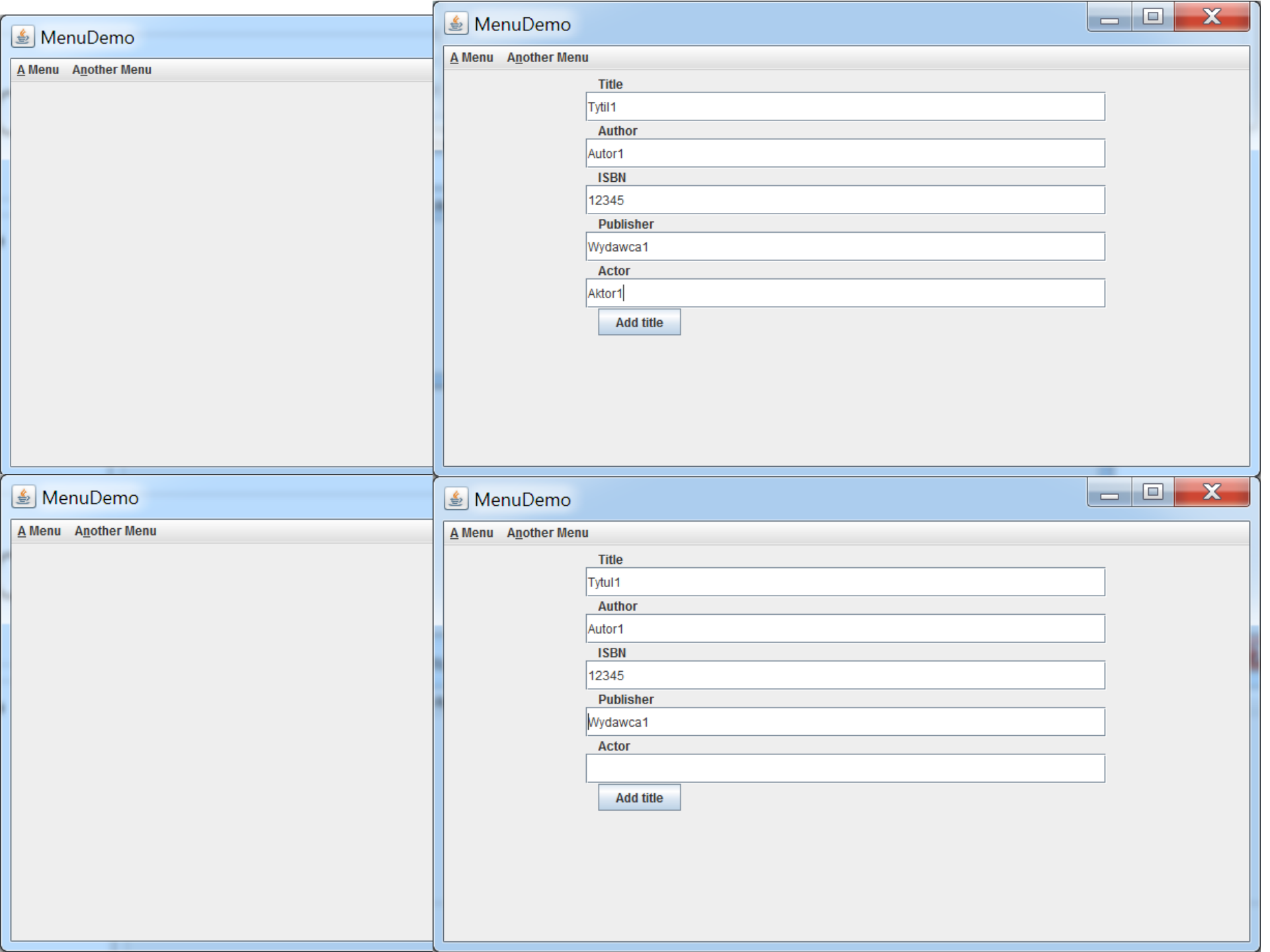
20. Uruchomienie aplikacji typu Enterprise Application za pomocą Deploy

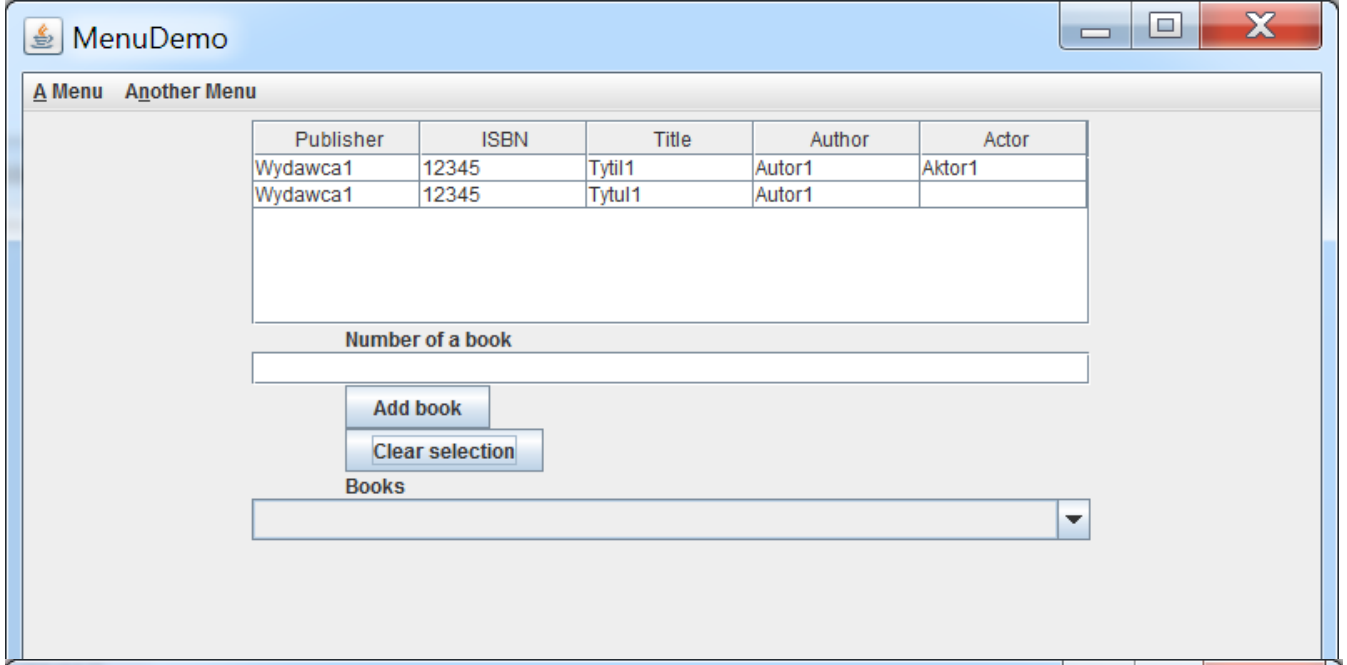


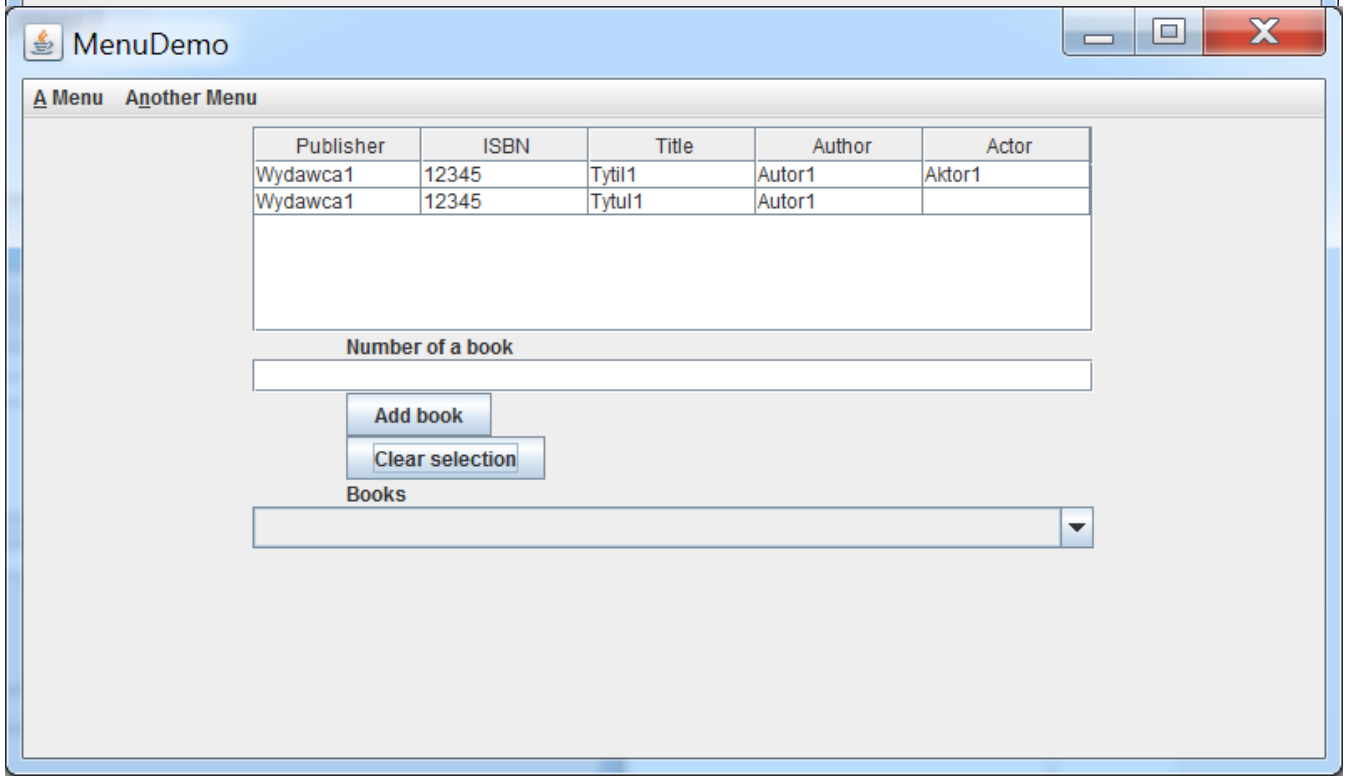
21. Uruchomiona aplikacja wielowarstwowa typu Enterprise Application



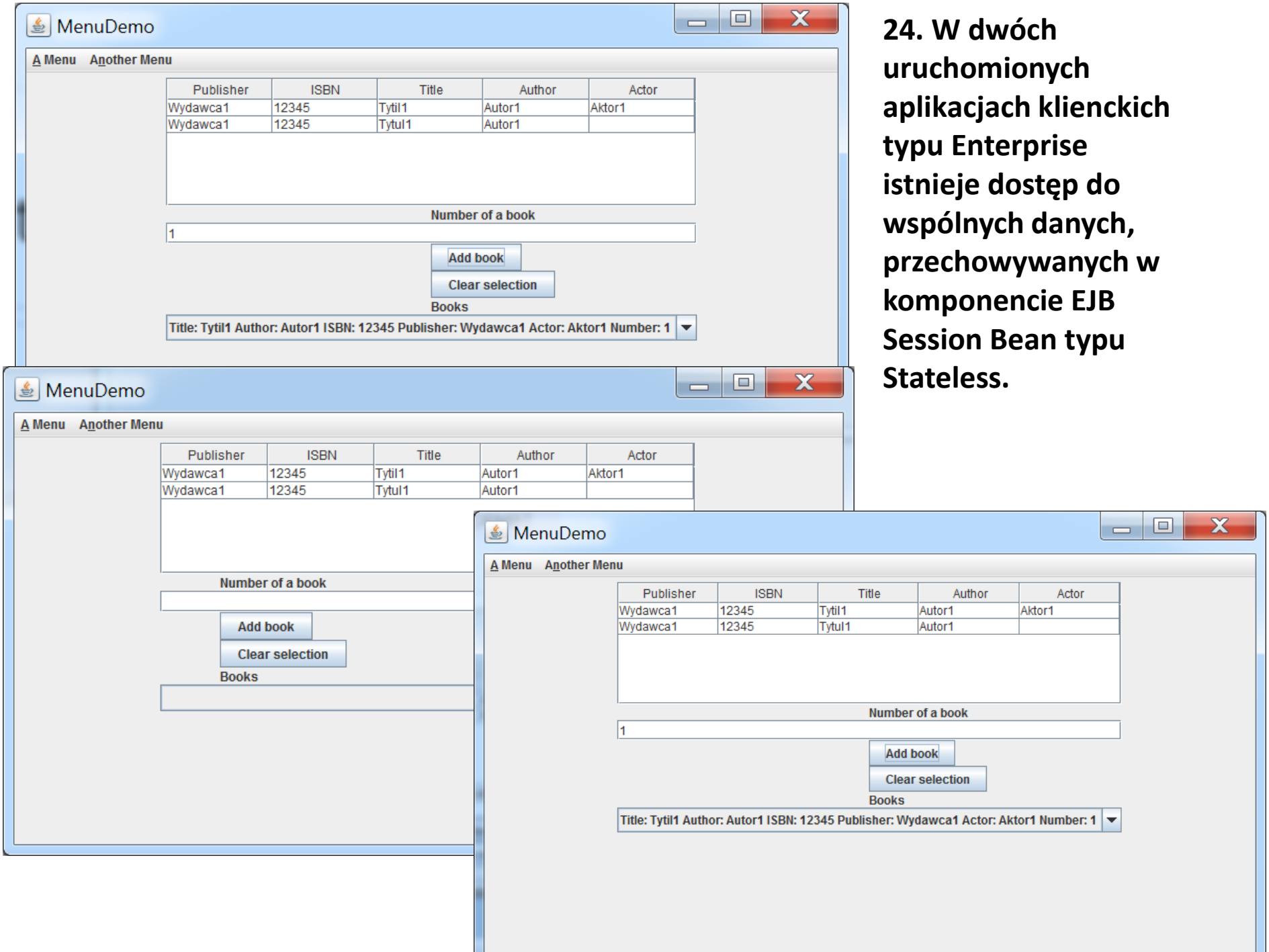
22. Uruchomienie 2 desktopowych aplikacji wielowarstwowych typu EE: 2 razy uruchomić Run na projekcie typu Enterprise Application Client


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





23. W dwóch uruchomionych aplikacjach klienckich typu Enterprise istnieje dostęp do wspólnych danych, przechowywanych w komponencie EJB Session Bean typu Stateless.



24. W dwóch uruchomionych aplikacjach klienckich typu Enterprise istnieje dostęp do wspólnych danych, przechowywanych w komponencie EJB Session Bean typu Stateless.