Laboratorium 2_3_4
Wzorce oprogramowania zastosowane w modelu obiektowym

(wg Alan Shalloway, James R.Trott)

Implementacja warstwy biznesowej stosującej wzorce obiektowe

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)**

System informacyjny wypożyczalni książek

1. Opis biznesowy systemu
2. Sformułowanie wymagań funkcjonalnych i niefunkcjonalnych systemu
3. **Model analizy całego systemu** oparty na diagramie przypadków użycia
4. **Model projektowy warstwy biznesowej** oparty na diagramie klas i diagramie sekwencji tworzony metodą iteracyjno-rozwojową sterowany realizacją przypadków użycia
5. **Implementacja warstwy biznesowej** tworzona w cyklu iteracyjno-rozwojowym sterowana rozwojem modelu projektowego

1. Opis biznesowy systemu

Opis biznesowy aplikacji typu Katalog tytułów i książek

1. Opis zasobów ludzkich

Pracownik wypożyczalni może dodawać do katalogu tytułów nowe tytuły. Każdy tytuł jest reprezentowany przez następujące 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 kasety, dlatego dodatkowo tytuł zawiera dane nagrania np nazwisko aktora. Każdy egzemplarz, niezależnie, czy jest książką czy kasetą, jest opisany odrębną informacją zawierajacą numer egzemplarza i ewentualnie (dotyczy to wyodrębnionych egzemplarzy) informację o liczbie dni, na które można wypożyczyć egzemplarz. Numery egzemplarzy mogą się powtarzać dla różnych tytułów. Pracownik bilioteki (bibliotekarz) może dodawać nowe tytuły i egzemplarze oraz je przeszukiwać, natomiast klient może jedynie przeszukiwać tyutły i sprawdzać egzemplarze wybranych tytułów.

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. Pracownik biblioteki może dodatkowo wstawiać, modyfikować i usuwać dane o tytuiłach oraz egzemplarzach. Zakłada się, że klientów jednocześnie przeglądajądających 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.

2. Sformułowanie wymagań funkcjonalnych i niefunkcjonalnych systemu

System informacyjny dla wypożyczalni

Lista wymagań funkcjonalnych

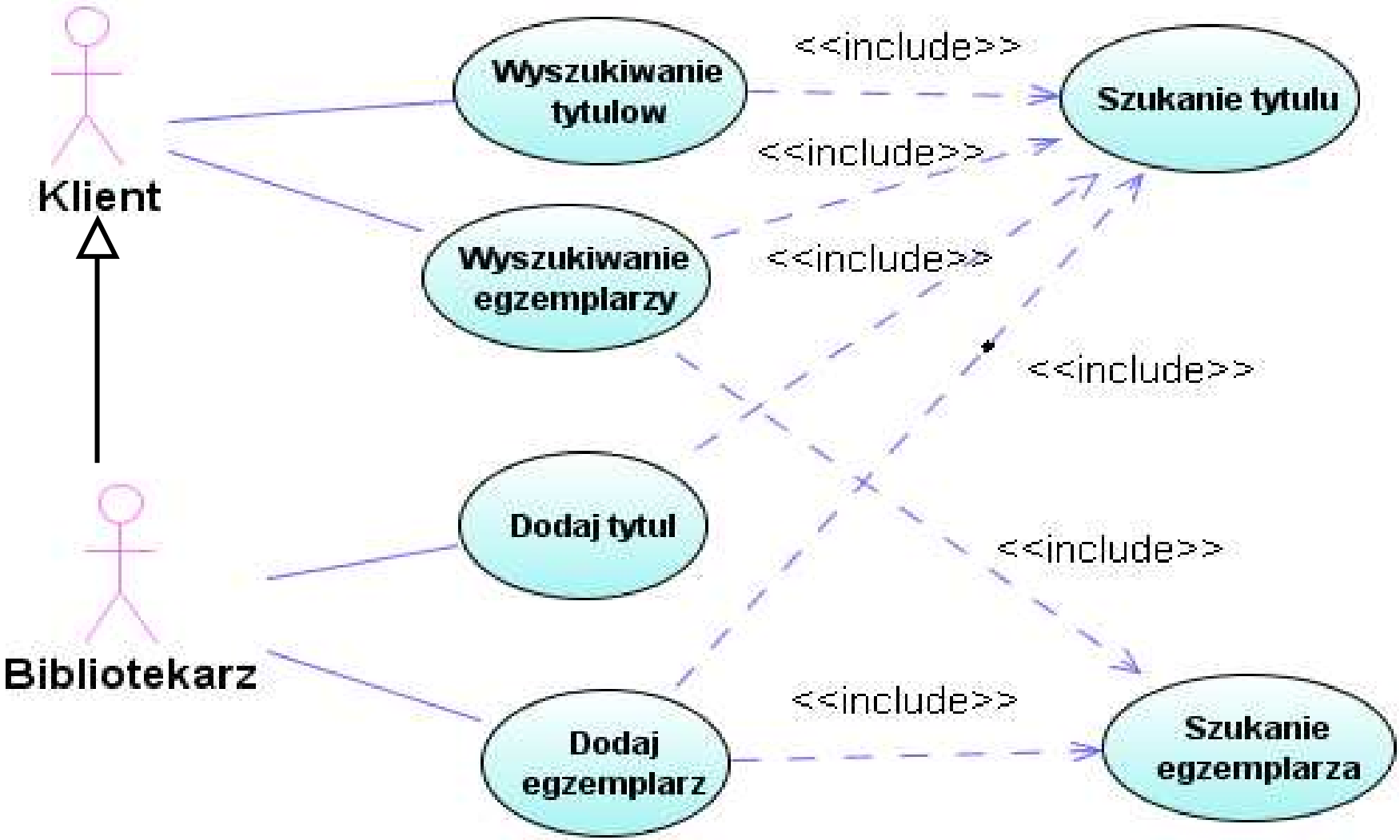
1. System zawiera katalog tytułów
2. System zawiera dwa typy egzemplarzy do wypożyczenia: książki i kasety z nagraniami dźwiękowymi książek.
3. Każdy egzemplarz zawiera tytuł, nazwisko autora, ISBN, wydawnictwo, jeśli jest to książka oraz dodatkowo nazwisko aktora, jeżeli jest to nagranie dźwiękowe. 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 wybrania właściwego egzemplarza należy podać ISBN, jeśli jest to książka oraz dodatkowo nazwisko aktora, jeśli jest to kaseta oraz numer egzemplarza
5. 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

3. Model analizy całego systemu oparty na diagramie przypadków użycia

Wypożyczalnia



AKTOR	OPIS	PRZYPADKI UŻYCIA
Bibliotekarz	*Bibliotekarz jest odpowiedzialny za utrzymywanie zasobów biblioteki (wstawianie i usuwanie: tytułów książek, egzemplarzy książek). Może on również przeszukiwać zasoby katalog tytułów i egzemplarzy książek*	• Dodaj tytul • Dodaj egzemplarz • Wyszukiwanie tytulow • Wyszukiwanie egzemplarzy
Klient	*Klient może jedynie przeszukiwać zasoby katalog tytułów i egzemplarzy książek*	• Wyszukiwanie tytulow • Wyszukiwanie egzemplarzy

PU Szukanie tytulu
OPIS
CEL: Poszukiwanie tytulu
WS (warunki wstępne): inicjalizacja przez uruchomienie programu (np. otwarcie strony WWW, start aplikacji)
WK (warunki końcowe): podanie tytułu zawierającego identyczne dane, jakie posiada tytuł wzorcowy lub podanie informacji o braku tytułu
PRZEBIEG:
1. Szukanie tytułu przebiega według atrybutów: ISBN (obowiązkowo) oraz aktor (jeśli jest to wymagane) zgodnie z danymi tytułu podanego do przypadku użycia
2. Jeśli istnieje tytuł o podanych atrybutach, zwracany jest tytuł z zasobów wypożyczalni, w przeciwnym wypadku zwracana jest informacja o braku tytułu.

PU Wyszukiwanie tytulow
OPIS
CEL: Wyszukiwanie tytulow
WS (warunki wstępne): inicjalizacja przez uruchomienie programu (np. otwarcie strony WWW, start aplikacji)
WK (warunki końcowe): wyszukanie tytułu o podanym atrybutach obowiązkowych ISBN lub ISBN i aktor w przypadku nagrania dźwiękowego lub podanie informacji o braku tytułu
PRZEBIEG:
1. Należy podać atrybuty tytułu: ISBN jako obowiązkowa dana oraz dodatkowo aktor, jeśli poszukiwany jest tytuł książki jako nagranie dźwiękowe. Tworzony jest tytuł wzorcowy do wyszukiwania rzeczywistego tytułu
2. Należy wywołać **PU Szukanie tytułu**. Należy sprawdzić, czy tytuł o podanych atrybutach już istnieje. Jeśli nie, należy zakończyć PU podając informację o braku tytułu, w przeciwnym wypadku należy podać znaleziony tytuł.

PU Szukanie egzemplarza
OPIS
CEL: Poszukiwanie egzemplarza
WS (warunki wstępne): inicjalizacja przez uruchomienie programu (np. otwarcie strony WWW, start aplikacji)
WK (warunki końcowe): podanie egzemplarza zawierającego identyczne dane, jakie posiada egzemplarz wzorcowy lub podanie informacji o braku egzemplarza
PRZEBIEG:
1. Szukanie egzemplarza przebiega według atrybutu: numer egzemplarza (obowiązkowo) zgodnie z danymi tytułu podanego do przypadku użycia. Przeszukiwane są egzemplarze należące do konkretnego tytułu egzemplarza
2. Jeśli istnieje egzemplarz o podanym numerze, zwracany jest egzemplarz z zasobów wypożyczalni, w przeciwnym wypadku zwracana jest informacja o braku egzemplarza.

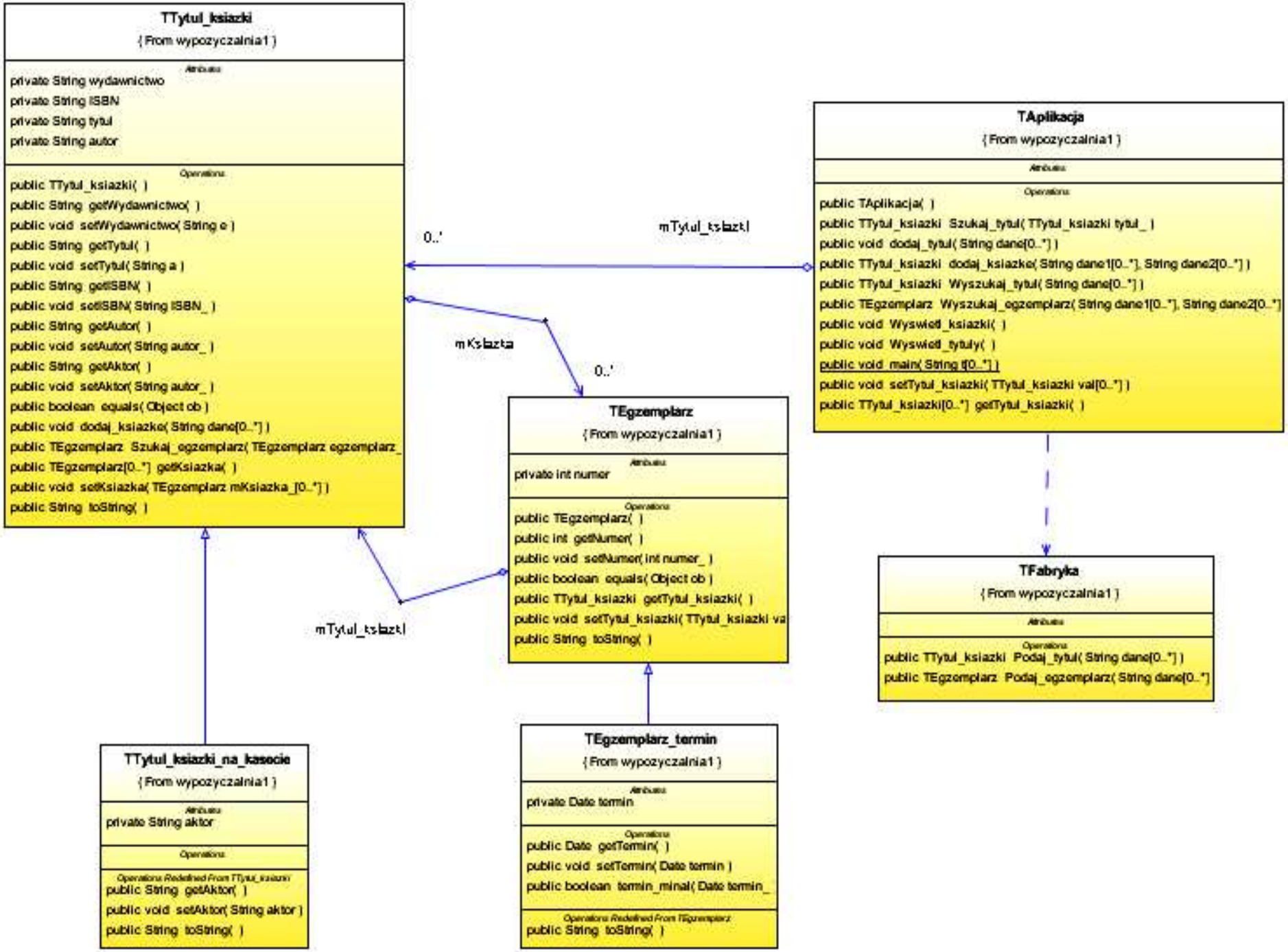
PU Wyszukiwanie egzemplarzy
OPIS
CEL: Wyszukiwanie egzemplarzy książek o podanym tytule
WS (warunki wstępne): inicjalizacja przez uruchomienie programu (np. otwarcie strony WWW, start aplikacji)
WK (warunki końcowe): wyszukanie egzemplarza o tytule zgodnym z podanymi atrybutami obowiązkowymi ISBN lub ISBN i aktor w przypadku nagrania dźwiękowego oraz podanym numerze lub podanie informacji o braku egzemplarza
PRZEBIEG:
1. Należy podać atrybuty tytułu: ISBN jako obowiązkowa dana oraz dodatkowo aktor, jeśli poszukiwany jest tytuł książki jako nagranie dźwiękowe. Tworzony jest tytuł wzorcowy do wyszukiwania rzeczywistego tytułu
2. Należy wywołać **PU Szukanie tytułu**. Należy sprawdzić, czy tytuł o podanych atrybutach już istnieje. Jeśli nie, należy zakończyć PU podając informację o braku tytułu.
3. Należy utworzyć wzorcowy egzemplarz zawierający numer podany do wyszukiwania egzemplarza i przekazać go do **PU Szukanie egzemplarza.** Wynik podany przez wywołany PU należy podać jako wynik końcowy.

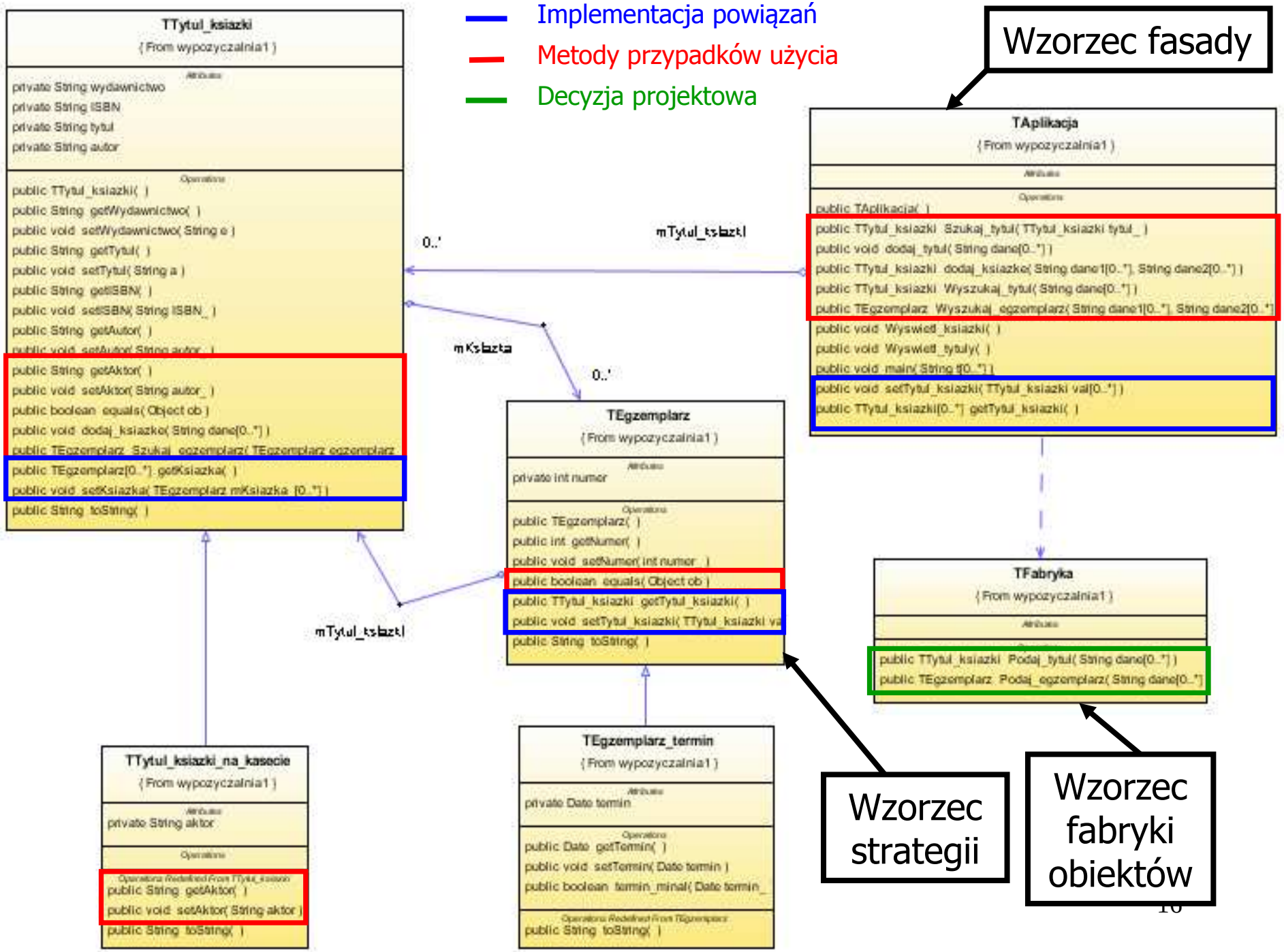
PU Dodaj tytul
OPIS
CEL: Wstawienie nowego tytułu
WS (warunki wstępne): inicjalizacja przez uruchomienie programu (np. otwarcie strony WWW, start aplikacji)
WK (warunki końcowe): dodanie tytułu o podanych atrybutach obowiązkowych: tytul, autor, ISBN, wydawnictwo oraz jeśli jest to nagranie dźwiękowe, to nazwisko aktora lub informacja o istnieniu takiego tytułu
PRZEBIEG:
1. Należy podać atrybuty tytułu: tytul, autor, ISBN, wydawnictwo oraz jeśli jest to nagranie dźwiękowe, to nazwisko aktora. Należy utworzyć tytuł do wyszukiwania i ewentualnego wstawienia.
2. Należy wywołać **PU Szukanie tytulu**. Należy sprawdzić, czy tytuł o podanych atrybutach juz istnieje. Jeśli tak, należy zakończyć PU, w przeciwnym wypadku należy wstawić nowy tytuł.

PU Dodaj egzemplarz
OPIS
CEL: Wstawianie nowego egzemplarza
WS (warunki wstępne): inicjalizacja przez uruchomienie programu (np. otwarcie strony WWW, start aplikacji)
WK (warunki końcowe): wstawienie egzemplarza o tytule zgodnym z podanymi atrybutami obowiązkowymi ISBN lub ISBN i aktor w przypadku nagrania dźwiękowego oraz podanym numerze i ewentualnie atrybucie do określania terminu zwrotu, jeśli należy wstawić egzemplarz z wyznaczonym terminem zwrotu lub podanie informacji o istnieniu takiego egzemplarza
PRZEBIEG:
1. Należy podać atrybuty tytułu: ISBN jako obowiązkowa dana oraz dodatkowo aktor, jeśli poszukiwany jest tytuł książki jako nagranie dźwiękowe. Tworzony jest tytuł wzorcowy do wyszukiwania rzeczywistego tytułu
2. Należy wywołać **PU Szukanie tytułu**. Należy sprawdzić, czy tytuł o podanych atrybutach już istnieje. Jeśli nie, należy zakończyć PU podając informację o braku tytułu.
3. Należy utworzyć egzemplarz zawierający numer podany do wyszukiwania egzemplarza oraz atrybut terminu zwrotu, jeśli jest to wymagane i należy przekazać go do **PU Szukanie egzemplarza.** Jeśli nie istnieje egzemplarz o danym numerze, należy wstawić ten egzemplarz, w przeciwnym wypadku należy podać informację o istnieniu takiego egzemplarza.

Analiza wspólności i zmienności

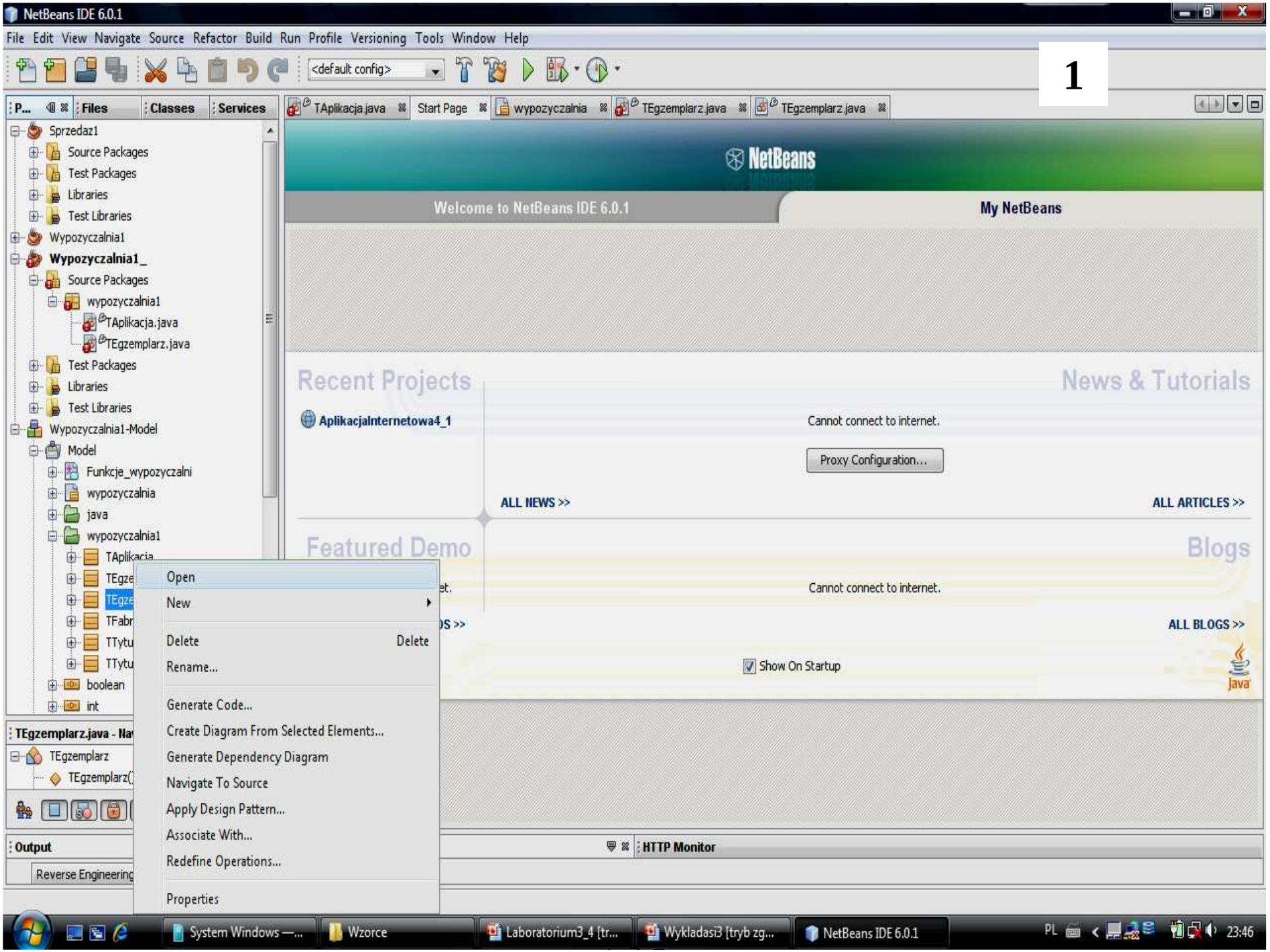
- Wykryto dwie główne klasy typu „Entity” ze względu na odpowiedzialność: TTytul_ksiazki (zawiera atrybuty tytułu, posiada kiążki – wstawia i wyszukuje je), oraz TEgzemplarz (posiada numer). Pojęcia **książki i egzemplarza** są równoważne.
- Wykryto dziedziczenie w właściwościach tytułów, które mogą wystąpić jako zwykłe książki lub jako nagrania dźwiękowe (klasa TTytul_ksiazki_na_kasecie typu „Entity”, która dziedziczy od klasy TTytul_ksiazki). Określono strategię przechowywania danych o tytule na wielu egzemplarzach książek lub kaset. Wyróżniono egzemplarze zwykłe typu TEgzemplarz, rozróżniane w ramach danego tytułu książki zwyklej lub nagranej w postaci dźwiękowej numerem oraz egzemplarze TEgzemplarz_termin z dodatkowo oznaczonym terminem oddania.
- Zależność między obiektami typu TTytul_ksiazki oraz TEgzemplarz są w relacji 1 do 0..*. Związek ten dziedziczą obiekty typu TTytul_ksiazki_na_kasecie. Związek 0..* do 1 między obiektami typu TEgzemplarz oraz TTytul_ksiazki są dziedziczone przez obiekty typu TEgzemplarz_termin. Stąd zwykłe książki mogą być oznaczone jedynie numerami lub numerami i terminem zwrotu. Dotyczy to również książek w postaci nagrań dźwiękowych.
- Wykryto związki silnej agregacji między tytułem i egzemplarzem – egzemplarz nie może istnieć bez tytułu.Wybrano **wzorzec strategii** do implementacji obiektów typu TEgzemplarz
- Zastosowano klasęTAplikacja typu „Control” jako **wzorzec fasady** do oddzielenia obiektów typu „Entity” od pozostałej części systemu oraz klasę typu „Control” jako **wzorzec fabryki obiektów** (TFabryka) do tworzenia różnych typów tytułów oraz egzemplarzy.

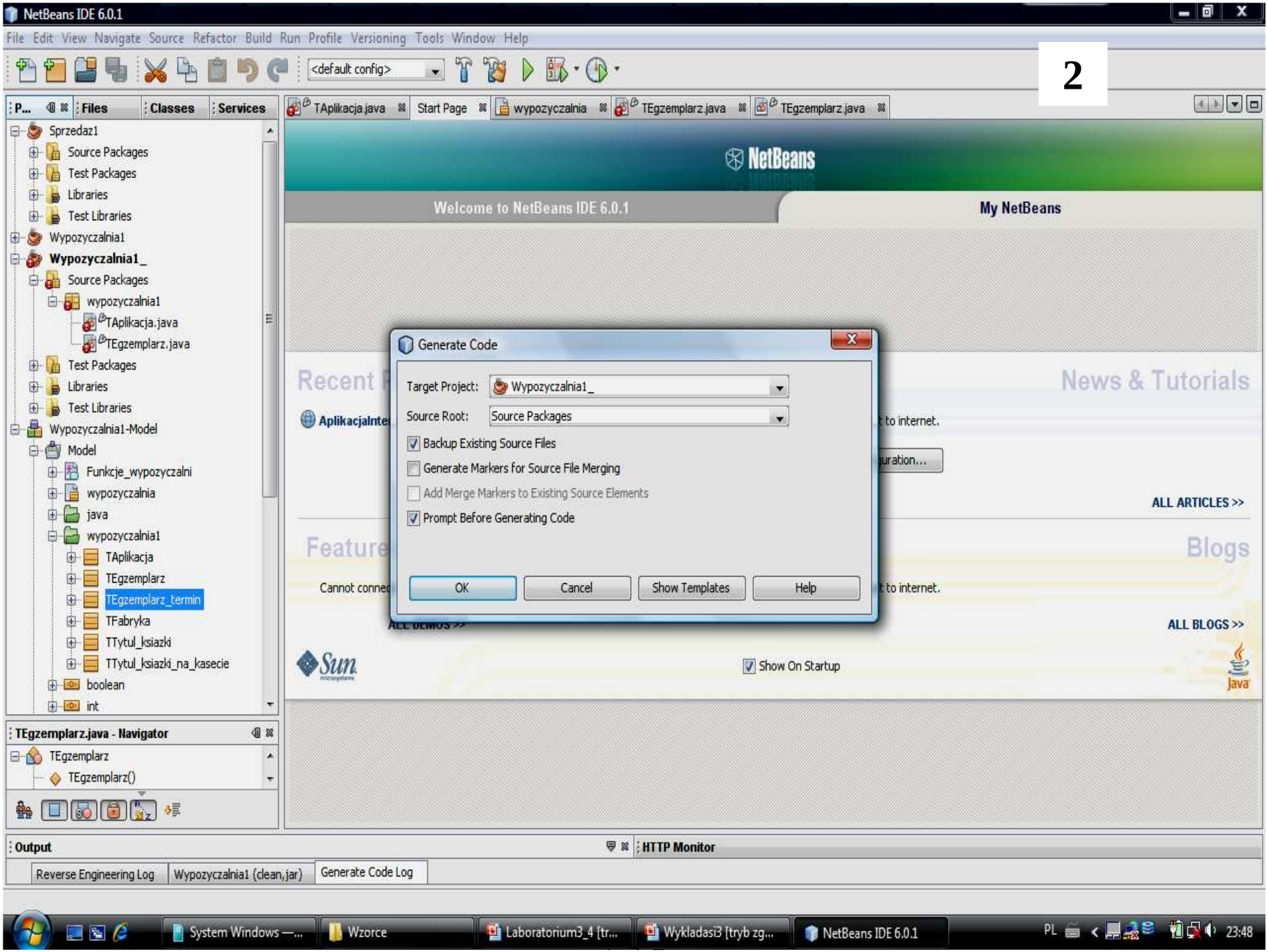

TTytul_ksiazki
{ From wypozyczalnia1 }
Attributes
private String wydawnictwo
private String ISBN
private String tytul
private String autor
Operations
public TTytul_ksiazki( )
public String getWydawnictwo( )
public void setWydawnictwo(String e)
public String getTytul( )
public void setTytul(String a)
public String getISBN( )
public void setISBN(String ISBN_)
public String getAutor( )
public void setAutor(String autor_)
public String getAktor( )
public void setAktor(String autor_)
public boolean equals(Object ob)
public void dodaj_ksiazke(String dane[0..*])
public TEgzemplarz Szukaj_egzemplarz(TEgzemplarz egzemplarz_
public TEgzemplarz[0..*] getKsiazka( )
public void setKsiazka(TEgzemplarz mKsiazka_[0..*])
public String toString( )
0..'
mTytul_ksiazki
mKsiazka
0..'
TAplikacja
{ From wypozyczalnia1 }
Attributes
Operations
public TAplikacja( )
public TTytul_ksiazki Szukaj_tytul(TTytul_ksiazki tytul_)
public void dodaj_tytul(String dane[0..*])
public TTytul_ksiazki dodaj_ksiazke(String dane1[0..*], String dane2[0..*])
public TTytul_ksiazki Wyszukaj_tytul(String dane[0..*])
public TEgzemplarz Wyszukaj_egzemplarz(String dane1[0..*], String dane2[0..*]
public void Wyswietl_ksiazki( )
public void Wyswietl_tytuly( )
public void main(String t[0..*])
public void setTytul_ksiazki(TTytul_ksiazki val[0..*])
public TTytul_ksiazki[0..*] getTytul_ksiazki( )
TEgzemplarz
{ From wypozyczalnia1 }
Attributes
private int numer
Operations
public TEgzemplarz( )
public int getNumer( )
public void setNumer(int numer_)
public boolean equals(Object ob)
public TTytul_ksiazki getTytul_ksiazki( )
public void setTytul_ksiazki(TTytul_ksiazki va
public String toString( )
mTytul_ksiazki
TFabryka
{ From wypozyczalnia1 }
Attributes
Operations
public TTytul_ksiazki Podaj_tytul(String dane[0..*])
public TEgzemplarz Podaj_egzemplarz(String dane[0..*]
TTytul_ksiazki_na_kasecie
{ From wypozyczalnia1 }
Attributes
private String aktor
Operations
Operations Redefined From TTytul_ksiazki
public String getAktor( )
public void setAktor(String aktor)
public String toString( )
TEgzemplarz_termin
{ From wypozyczalnia1 }
Attributes
private Date termin
Operations
public Date getTermin( )
public void setTermin(Date termin)
public boolean termin_minal(Date termin_
Operations Redefined From TEgzemplarz
public String toString( )

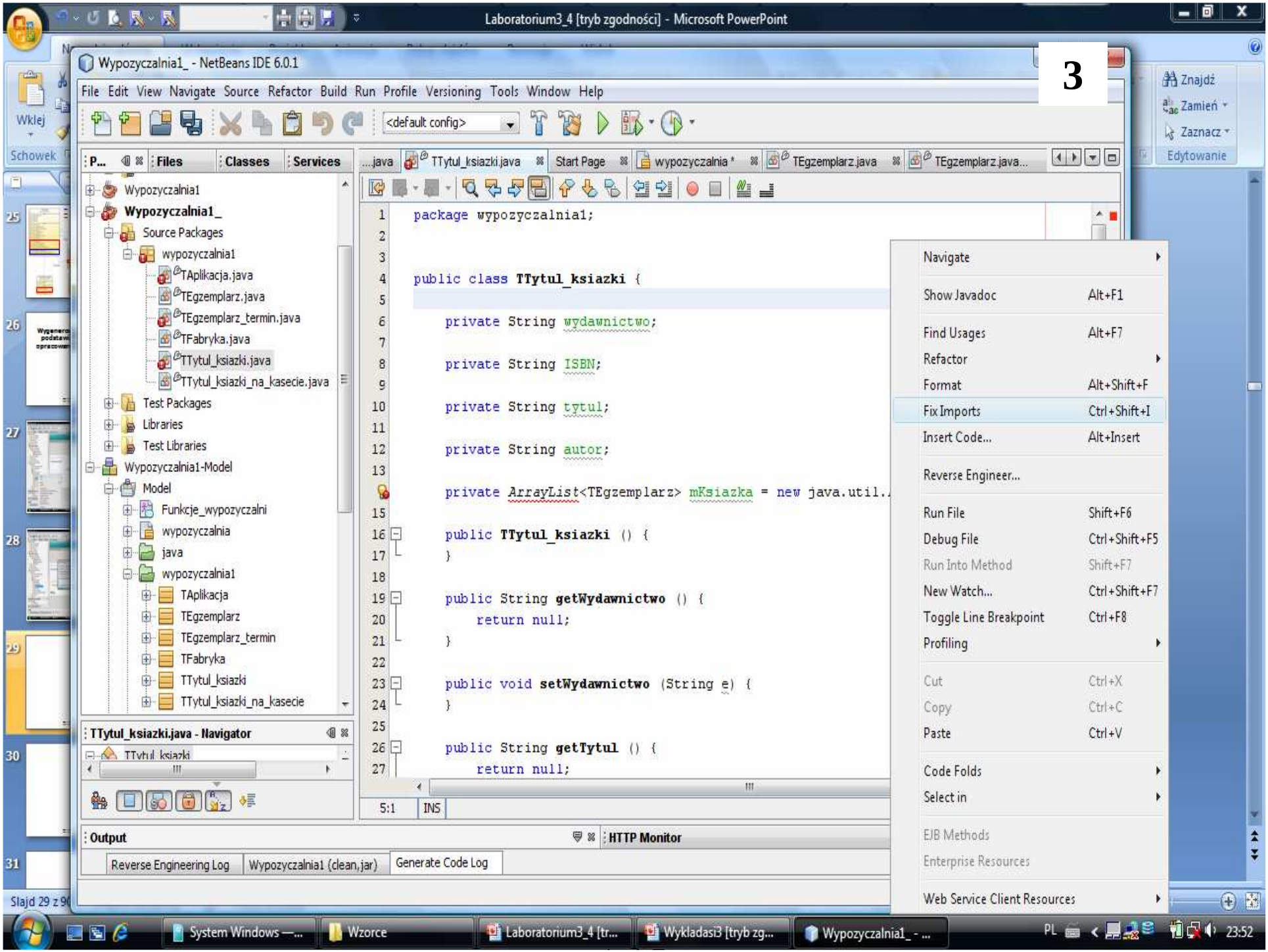

Implementacja powiązań
Metody przypadków użycia
Decyzja projektowa
Wzorzec fasady
TTytul_ksiazki
(From wypozyczalnia1)
private String wydawnictwo
private String ISBN
private String tytul
private String autor
public TTytul_ksiazki( )
public String getWydawnictwo( )
public void setWydawnictwo(String e)
public String getTytul( )
public void setTytul(String a)
public String getISBN( )
public void setISBN(String ISBN_)
public String getAutor( )
public String getAktor( )
public void setAktor(String autor_)
public boolean equals(Object ob)
public void dodaj_ksiazke(String dane[0..*])
public TEgzemplarz[0..*] getKsiazka( )
public void setKsiazka(TEgzemplarz mKsiazka [0..*])
public String toString( )
TAplikacja
(From wypozyczalnia1)
public TAplikacja( )
public TTytul_ksiazki Szukaj_tytul(TTytul_ksiazki tytul_)
public void dodaj_tytul(String dane[0..*])
public TTytul_ksiazki dodaj_ksiazke(String dane1[0..*], String dane2[0..*])
public TTytul_ksiazki Wyszukaj_tytul(String dane[0..*])
public TEgzemplarz Wyszukaj_egzemplarz(String dane1[0..*], String dane2[0..*]
public void Wyswietl_ksiazki( )
public void Wyswietl_tytuly( )
public void main(String t[0..*])
public void setTytul_ksiazki(TTytul_ksiazki val[0..*])
public TTytul_ksiazki[0..*] getTytul_ksiazki( )
mTytul_ksiazki
0..*
mKsiazka
0..*
TEgzemplarz
(From wypozyczalnia1)
private int numer
public TEgzemplarz( )
public int getNumer( )
public void setNumer(int numer_)
public boolean equals(Object ob)
public TTytul_ksiazki getTytul_ksiazki( )
public void setTytul_ksiazki(TTytul_ksiazki va
public String toString( )
mTytul_ksiazki
TFabryka
(From wypozyczalnia1)
public TTytul_ksiazki Podaj_tytul(String dane[0..*])
public TEgzemplarz Podaj_egzemplarz(String dane[0..*]
TTytul_ksiazki_na_kasecie
(From wypozyczalnia1)
private String aktor
public String getAktor( )
public void setAktor(String aktor)
public String toString( )
TEgzemplarz_termin
(From wypozyczalnia1)
private Date termin
public Date getTermin( )
public void setTermin(Date termin)
public boolean termin_minal(Date termin_
public String toString( )
Wzorzec strategii
Wzorzec fabryki obiektów

Wygenerowanie kodu klas na podstawie diagramu klas opracowanego w fazy analizy

- Należy otworzyć projekt kategorii **UML**, typu **Java-Platform Model o nazwie *Wypozyczalnia1-Model***, dostarczony w załączniku do laboratorium
- Należy założyć projekt **Java Application** np. o nazwie *Wypożyczalnia1_* (**File, New Project, Java, Java Application, Next**, *wstawienie nazwy projektu np. Wypozyczalnia1_ w* **Project Name**, *wybór katalogu w* **Project Location**, *w którym znajduje się projekt Wypozyczalnia1-Model*, **Finish**)
- Należy wygenerować kod dla każdej z klas osobno, z projektu *Wypozyczalnia1-Model*, wybierając np. w okienku zakładki **Projects** katalog **Model**, podkatalog **wypozyczalnia1** – i kolejno klasy projektu. Klikając na nie prawym klawiszem myszy z wyskakującego menu należy wybrać opcję **Generate Code** **(slajd 1)**
- W formularzu **Generate Code** w okienku **Target Project** wybrać utworzony projekt *Wypozyczalnia1_ , zaznaczyć Backup Existing Source Files oraz Prompt Before Generating Code* i nacisnąć **OK.** **(slajd 2)**
- Należy uzupełnić import pakietów w tych plikach, gdzie nazwy klas zostały oznaczone jako nieznane – należy wtedy kliknąć prawym klawiszem myszy i wybrać opcję **Fix Imports .** Jeśli pokaże się okienko do wyboru właściwego pakietu, należy dokonać wyboru np. dla klasy **Date** należy wybrać pakiet **java.util.Date (slajd 3).** Wybierając podobnie z wyskakującego menu opcję **Format** można uporządkować kod (wcięcia, wolne linie, bloki kodu itp).


NetBeans IDE 6.0.1
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
<default config>
1
Files
Classes
Services
Sprzedaz1
Source Packages
Test Packages
Libraries
Test Libraries
Wypozyczalnia1
Wypozyczalnia1_
Source Packages
wypozyczalnia1
TAplikacja.java
TEgzemplarz.java
Test Packages
Libraries
Test Libraries
Wypozyczalnia1-Model
Model
Funkcje_wypozyczalni
wypozyczalnia
java
wypozyczalnia1
TAplikacja
TEgze
TEgze
TFabr
TTytu
TTytu
boolean
int
TEgzemplarz.java - Na
TEgzemplarz
TEgzemplarz()
Output
Reverse Engineering
TAplikacja.java
Start Page
wypozyczalnia
TEgzemplarz.java
TEgzemplarz.java
NetBeans
Welcome to NetBeans IDE 6.0.1
My NetBeans
Recent Projects
AplikacjaInternetowa4_1
News & Tutorials
Cannot connect to internet.
Proxy Configuration...
ALL NEWS >>
ALL ARTICLES >>
Featured Demo
Blogs
Cannot connect to internet.
ALL BLOGS >>
Show On Startup
Java
HTTP Monitor
Open
New
Delete
Delete
Rename...
Generate Code...
Create Diagram From Selected Elements...
Generate Dependency Diagram
Navigate To Source
Apply Design Pattern...
Associate With...
Redefine Operations...
Properties
System Windows —...
Wzorce
Laboratorium3_4 [tr...
Wykladasi3 [tryb zg...
NetBeans IDE 6.0.1
PL
23:46


NetBeans IDE 6.0.1
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
<default config>
2
Files
Classes
Services
Sprzedaz1
Source Packages
Test Packages
Libraries
Test Libraries
Wypozyczalnia1
Wypozyczalnia1_
Source Packages
wypozyczalnia1
TAplikacja.java
TEgzemplarz.java
Test Packages
Libraries
Test Libraries
Wypozyczalnia1-Model
Model
Funkcje_wypozyczalni
wypozyczalnia
java
wypozyczalnia1
TAplikacja
TEgzemplarz
TEgzemplarz_termin
TFabryka
TTytul_ksiazki
TTytul_ksiazki_na_kasecie
boolean
int
TEgzemplarz.java - Navigator
TEgzemplarz
TEgzemplarz()
TAplikacja.java
Start Page
wypozyczalnia
TEgzemplarz.java
TEgzemplarz.java
NetBeans
Welcome to NetBeans IDE 6.0.1
My NetBeans
Generate Code
Target Project:
Wypozyczalnia1_
Source Root:
Source Packages
Backup Existing Source Files
Generate Markers for Source File Merging
Add Merge Markers to Existing Source Elements
Prompt Before Generating Code
OK
Cancel
Show Templates
Help
News & Tutorials
ALL ARTICLES >>
Blogs
ALL BLOGS >>
Show On Startup
Output
HTTP Monitor
Reverse Engineering Log
Wypozyczalnia1 (clean,jar)
Generate Code Log
System Windows —...
Wzorce
Laboratorium3_4 [tr...
Wykladasi3 [tryb zg...
NetBeans IDE 6.0.1
PL
23:48


Laboratorium3_4 [tryb zgodności] - Microsoft PowerPoint
3
Wypozyczalnia1_ - NetBeans IDE 6.0.1
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
<default config>
Files
Classes
Services
Wypozyczalnia1
Wypozyczalnia1_
Source Packages
wypozyczalnia1
TAplikacja.java
TEgzemplarz.java
TEgzemplarz_termin.java
TFabryka.java
TTytul_ksiazki.java
TTytul_ksiazki_na_kasecie.java
Test Packages
Libraries
Test Libraries
Wypozyczalnia1-Model
Model
Funkcje_wypozyczalni
wypozyczalnia
java
wypozyczalnia1
TAplikacja
TEgzemplarz
TEgzemplarz_termin
TFabryka
TTytul_ksiazki
TTytul_ksiazki_na_kasecie
TTytul_ksiazki.java - Navigator
TTytul_ksiazki
TTytul_ksiazki.java
Start Page
wypozyczalnia *
TEgzemplarz.java
TEgzemplarz.java...
1 package wypozyczalnia1;
4 public class TTytul_ksiazki {
6 private String wydawnictwo;
8 private String ISBN;
10 private String tytul;
12 private String autor;
15 private ArrayList<TEgzemplarz> mKsiazka = new java.util.
16 public TTytul_ksiazki () {
17 }
19 public String getWydawnictwo () {
20 return null;
21 }
23 public void setWydawnictwo (String e) {
24 }
26 public String getTytul () {
27 return null;
5:1
INS
Navigate
Show Javadoc Alt+F1
Find Usages Alt+F7
Refactor
Format Alt+Shift+F
Fix Imports Ctrl+Shift+I
Insert Code... Alt+Insert
Reverse Engineer...
Run File Shift+F6
Debug File Ctrl+Shift+F5
Run Into Method Shift+F7
New Watch... Ctrl+Shift+F7
Toggle Line Breakpoint Ctrl+F8
Profiling
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Code Folds
Select in
EJB Methods
Enterprise Resources
Web Service Client Resources
Output
HTTP Monitor
Reverse Engineering Log
Wypozyczalnia1 (clean,jar)
Generate Code Log
Znajdź
Zamień
Zaznacz
Edytowanie
Schowek
Wklej
Slajd 29 z 90
System Windows —...
Wzorce
Laboratorium3_4 [tr...
Wykladasi3 [tryb zg...
Wypozyczalnia1_ - ...
PL
23:52

```
package wypozyczalnia1;
import java.util.ArrayList;

public class TAplikacja {
   private ArrayList<TTytul_ksiazki> mTytul_ksiazki = new ArrayList<TTytul_ksiazki>();

   public TAplikacja ()                                                    {   }
   public static void main (String[] t)                                    {   }
   public TTytul_ksiazki Szukaj_tytul (TTytul_ksiazki tytul_)              {     return null;   }
   public void dodaj_tytul (String[] dane)                                 {   }
   public ArrayList<TTytul_ksiazki> getTytul_ksiazki ()                    {     return null;   }
   public void setTytul_ksiazki (ArrayList<TTytul_ksiazki> val)            {   }
   public TTytul_ksiazki dodaj_ksiazke (String[] dane1, String[] dane2)    {     return null;   }
   public TTytul_ksiazki Wyszukaj_tytul (String[] dane)                    {     return null;   }
   public TEgzemplarz Wyszukaj_egzemplarz (String[] dane1, String[] dane2) {     return null;   }
   public void Wyswietl_ksiazki ()                                         {   }
   public void Wyswietl_tytuly ()                                          {   }
}
```

```
package wypozyczalnia1;

public class TEgzemplarz {
   private int numer;
   private TTytul_ksiazki mTytul_ksiazki;

   public TEgzemplarz ()                                {   }
   public int getNumer ()                               {     return 0;    }
   public void setNumer (int numer_)                    {   }
   public boolean equals (Object ob)                    {     return true;   }
   public TTytul_ksiazki getTytul_ksiazki ()            {     return null;   }
   public void setTytul_ksiazki (TTytul_ksiazki val)    {   }
   public String toString ()                            {     return null;   }
}
```

```
package wypozyczalnia1;
import java.util.Date;

public class TEgzemplarz_termin extends TEgzemplarz {
   private Date termin;

   public Date getTermin ()                          {     return null;   }
   public void setTermin (Date termin)               {   }
   public boolean termin_minal (Date termin_)        {     return true;   }
   public String toString ()                         {     return null;   }
}
```

```
package wypozyczalnia1;

public class TFabryka {

   public TTytul_ksiazki Podaj_tytul (String[] dane)          {     return null;   }
   public TEgzemplarz Podaj_egzemplarz (String[] dane)        {     return null;   }
}
```

```
package wypozyczalnia1;
import java.util.ArrayList;
public class TTytul_ksiazki {
   private String wydawnictwo;
   private String ISBN;
   private String tytul;
   private String autor;
   private ArrayList<TEgzemplarz> mKsiazka = new java.util.ArrayList<TEgzemplarz>();
   public TTytul_ksiazki ()                                                    {   }
   public String getWydawnictwo ()                                             {     return null;   }
   public void setWydawnictwo (String e)                                       {   }
   public String getTytul ()                                                   {     return null;   }
   public void setTytul (String a)                                             {   }
   public String getISBN ()                                                    {     return null;   }
   public void setISBN (String ISBN_)                                          {   }
   public String getAutor ()                                                   {     return null;   }
   public void setAutor (String autor_)                                        {   }
   public String getAktor ()                                                   {     return null;   }
   public void setAktor (String autor_)                                        {   }
   public String toString ()                                                   {     return null;   }
   public boolean equals (Object ob)                                           {     return true;   }
   public void dodaj_ksiazke (String[] dane)                                   {   }
   public TEgzemplarz Szukaj_egzemplarz (TEgzemplarz egzemplarz_)              {     return null;   }
   public ArrayList<TEgzemplarz> getKsiazka ()                                 {     return null;   }
   public void setKsiazka (ArrayList<TEgzemplarz> mKsiazka_)                   {   }
}
```

```
package wypozyczalnia1;
public class TTytul_ksiazki_na_kasecie extends TTytul_ksiazki {
   private String aktor;
   public String getAktor ()                    {     return null;   }
   public void setAktor (String aktor)          {   }
   public String toString ()                    {     return null;   }
}
```

Proponowany kod funkcji main w klasie fasadowej TAplikacja

```
public static void main(String t[]) // your code here
   { /*TAplikacja ap = new TAplikacja();
      String t1[] = {"1", "1", "1", "1", "1"}; //t1, t2,  t3 – tablice łańcuchów  do tworzenia tytułu książki zwyklej – pierwszy łańcuch
      String t2[] = {"1", "2", "2", "2", "2"}; // jest informacją dla fabryki, jaki obiekt wygenerować
      String t3[] = {"1", "3", "3", "3", "3"}; //"1" oznacza utworzenie obiektu klasy TTytul_ksiazki, a pozostałe łańcuchy to kolejno
     // autor, tytul, ISBN, wydawnictwo dla uproszczenia w postaci cyfr -  obiekty do wstawiania
      String t4[] = {"3", "1", "1", "1", "1", "1"}; // t4, t5,t6 – tablice łańcuchów  do tworzenia tytułu książki jako nagranie dźwiękowe
      String t5[] = {"3", "2", "2", "2", "2", "2"}; //– pierwszy łańcuch jest informacją dla fabryki, jaki obiekt wygenerować
      String t6[] = {"3", "4", "4", "4", "4", "4"}; //"3" oznacza utworzenie obiektu klasy TTytul_ksiazki_na_kasecie, a pozostałe
     //łańcuchy to kolejno autor, tytul, ISBN, wydawnictwo, aktor dla uproszczenia w postaci cyfr-  obiekty do wstawiania
      ap.dodaj_tytul(t1);
      ap.dodaj_tytul(t2);        ap.dodaj_tytul(t2);
      ap.dodaj_tytul(t3);
      ap.dodaj_tytul(t4);
      ap.dodaj_tytul(t5);        ap.dodaj_tytul(t5);
      ap.dodaj_tytul(t6);
      String lan = ap.getTytul_ksiazki().toString();
      System.out.println(lan);
      String d1[] = {"0", "1"}; // d1, d2, d3 - – tablice łańcuchów  do tworzenia wzorcowego tytułu książki zwyklej  do wyszukiwania
      String d2[] = {"0", "2"}; // – pierwszy łańcuch jest informacją dla fabryki, jaki obiekt wygenerować: „0" oznacza generowanie
      String d3[] = {"0", "5"}; // obiektu klasy TTytul_ksiazki, drugi łańcuch jest ISBN – obiekty do wyszukiwania
      String d4[] = {"2", "1", "1"}; //d4, d5 - tablice łańcuchów  do tworzenia  wzorcowego tytułu książki jako nagranie dźwiękowe
      String d5[] = {"2", "4", "4"}; //pierwszy łańcuch „2" oznacza generowanie obiektu typu TTytul_ksiazki_na_kasecie
                                       // drugi łańcuch to  ISBN, trzeci jest nazwiskiem aktora – obiekty do wyszukiwania
      String tr1[] = {"0", "1"};  //tablice tr1 i tr2 zawierają informację o tworzeniu obiektu typu TEgzemplarz: pierwszy łańcuch
      String tr2[] = {"0", "2"};  //równy „0" oznacza tworzenie obiektu typu typu TEgzemplarz, drugi jest numerem egzemplarza
      String tr3[] = {"1", "3", "April 10, 2008, 00:00:00 GMT"}; //pierwszy łańcuch równy „1" oznacza tworzenie obiektu klasy
      String tr4[] = {"1", "2", "April 10, 2008, 00:00:00 GMT"};   //TEgzemplarz_termin, drugi oznacza numer, trzeci termin
```

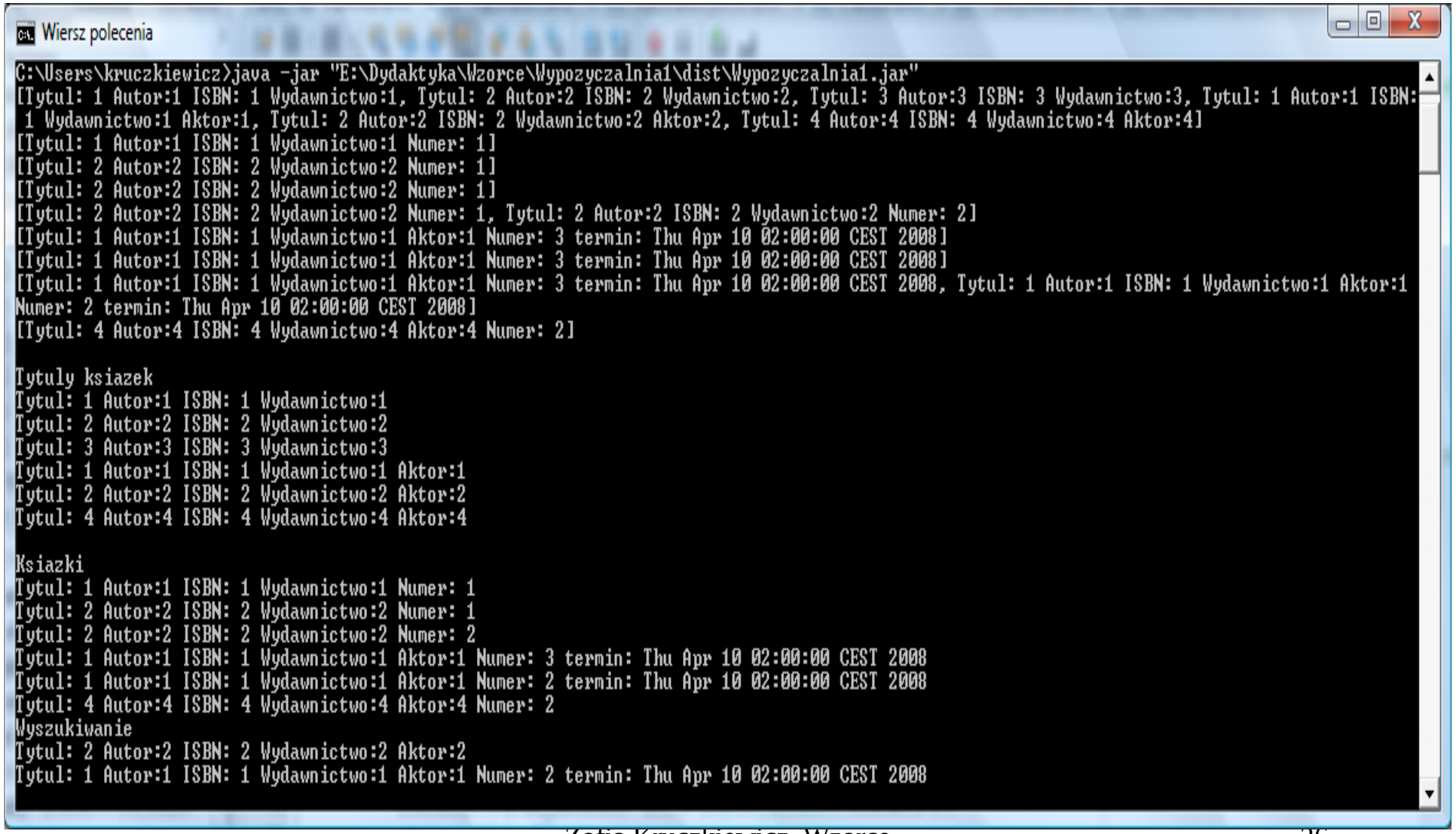
```
// W trakcie tworzenia kodu aplikacji można odsłaniać kod  z
// komentarza w celu przetestowania kolejnych przypadków użycia

   TTytul_ksiazki pom = ap.dodaj_ksiazke(d1, tr1);
    if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
    pom = ap.dodaj_ksiazke(d2, tr1);
    if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
    pom = ap.dodaj_ksiazke(d2, tr1);
    if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
    pom = ap.dodaj_ksiazke(d2, tr2);
    if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
    pom = ap.dodaj_ksiazke(d3, tr2);
    if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
    pom = ap.dodaj_ksiazke(d4, tr3);
    if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
    pom = ap.dodaj_ksiazke(d4, tr3);
    if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
    pom = ap.dodaj_ksiazke(d4, tr4);
    if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
    pom = ap.dodaj_ksiazke(d5, tr2);
    if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }

    ap.Wyswietl_tytuly();
    ap.Wyswietl_ksiazki();

    System.out.println("Wyszukiwanie");
    System.out.println(ap.Wyszukaj_tytul(t5).toString());
    System.out.println(ap.Wyszukaj_egzemplarz(d4, tr4).toString()); */
}
```

Tak może działać aplikacja po wykonaniu poszczególnych przypadków użycia



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

5. Implementacja warstwy biznesowej tworzona w cyklu iteracyjno-rozwojowym sterowana rozwojem modelu projektowego

Projekt przypadku użycia

„ **Szukanie tytulu**”

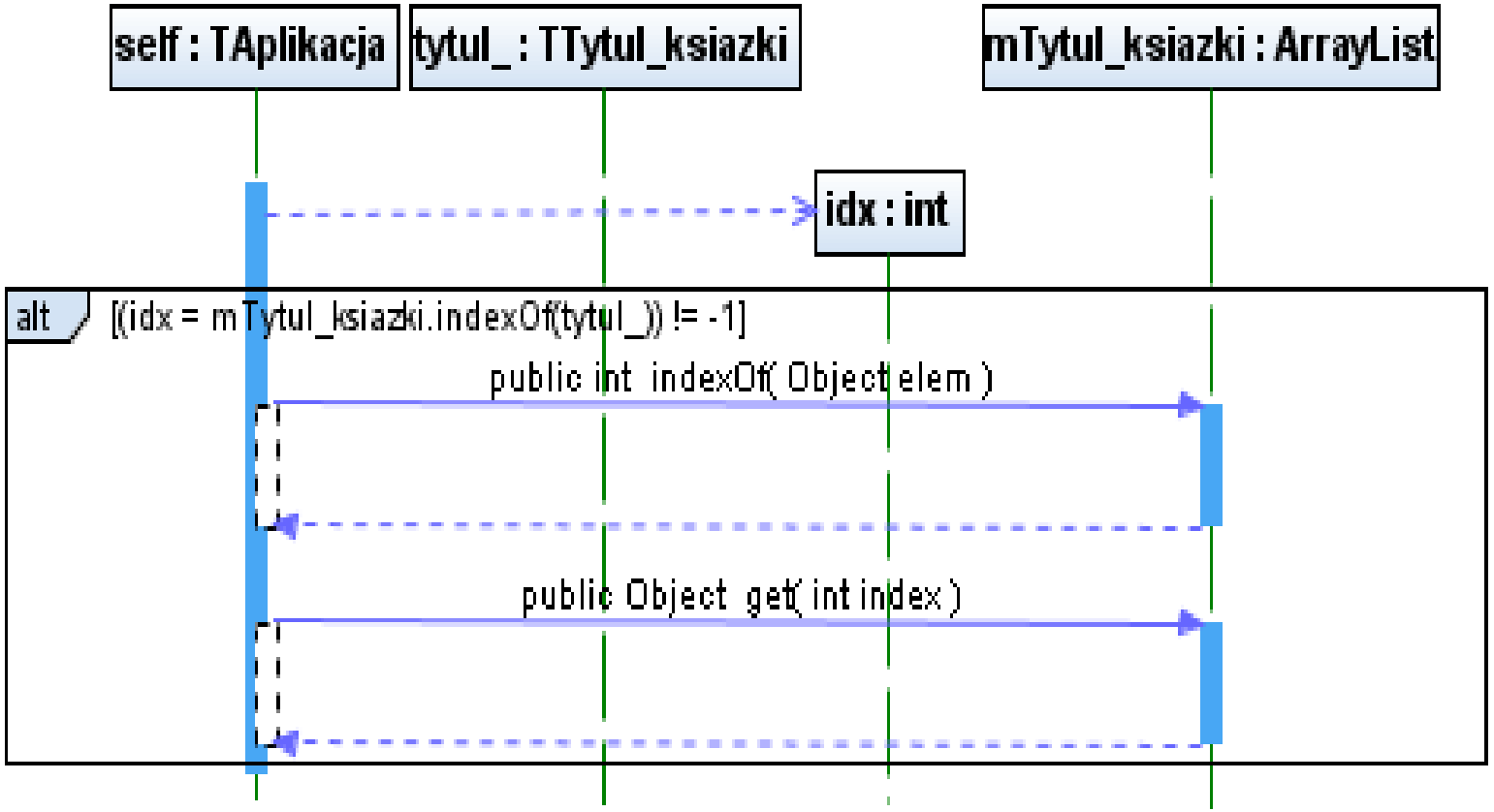
za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

Definiowanie kodu metod realizujących przypadek użycia

na podstawie diagramów sekwencji

(1) Szukanie tytulu

(TTytul_ksiazki TAplikacja::Szukaj_tytul(TTytul_ksiazki tytul_))



boolean TTytul_ksiazki::equals(Object ob)

Projekt przypadku użycia

„ **Dodaj tytul**”

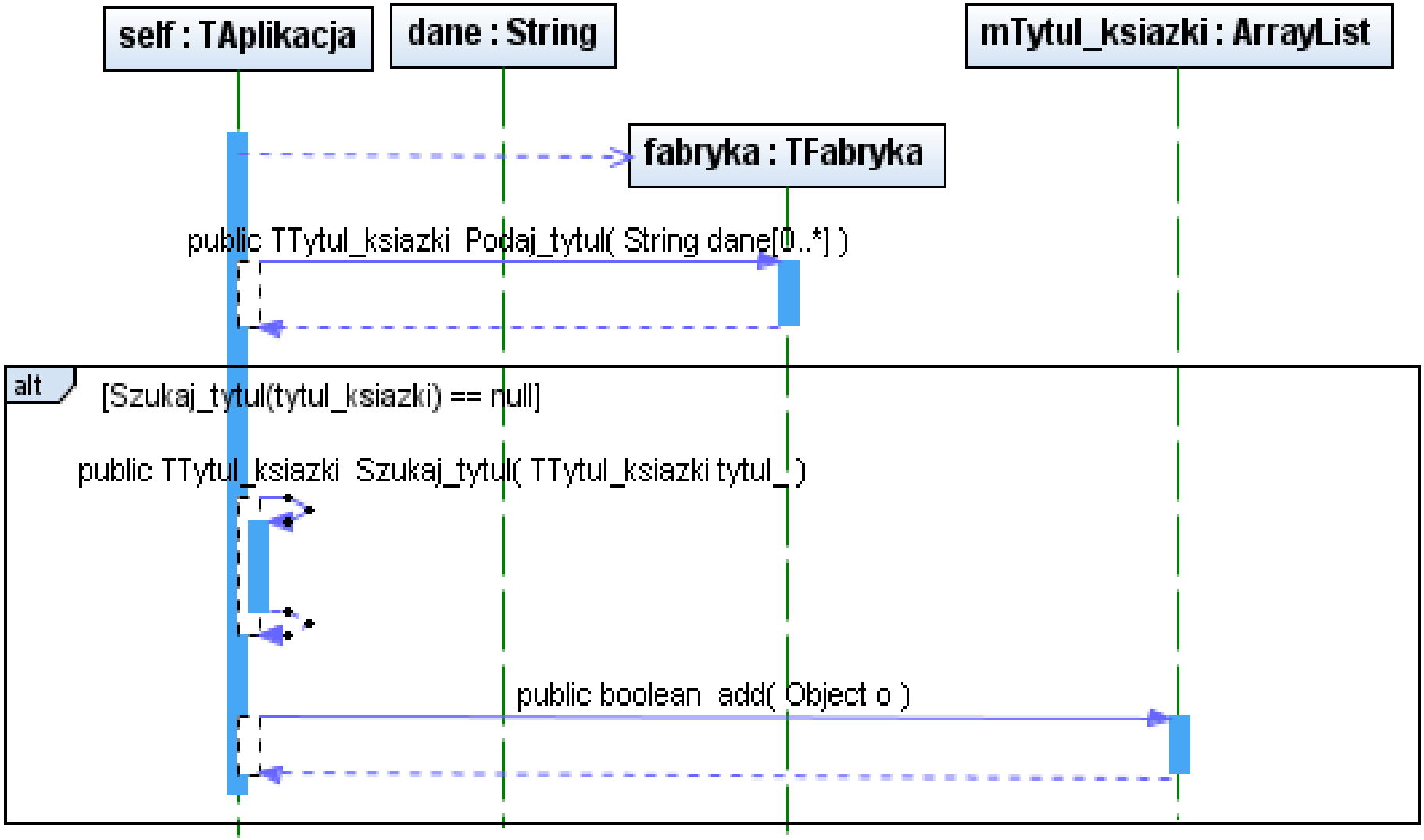
za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

Definiowanie kodu metod realizujących przypadek użycia

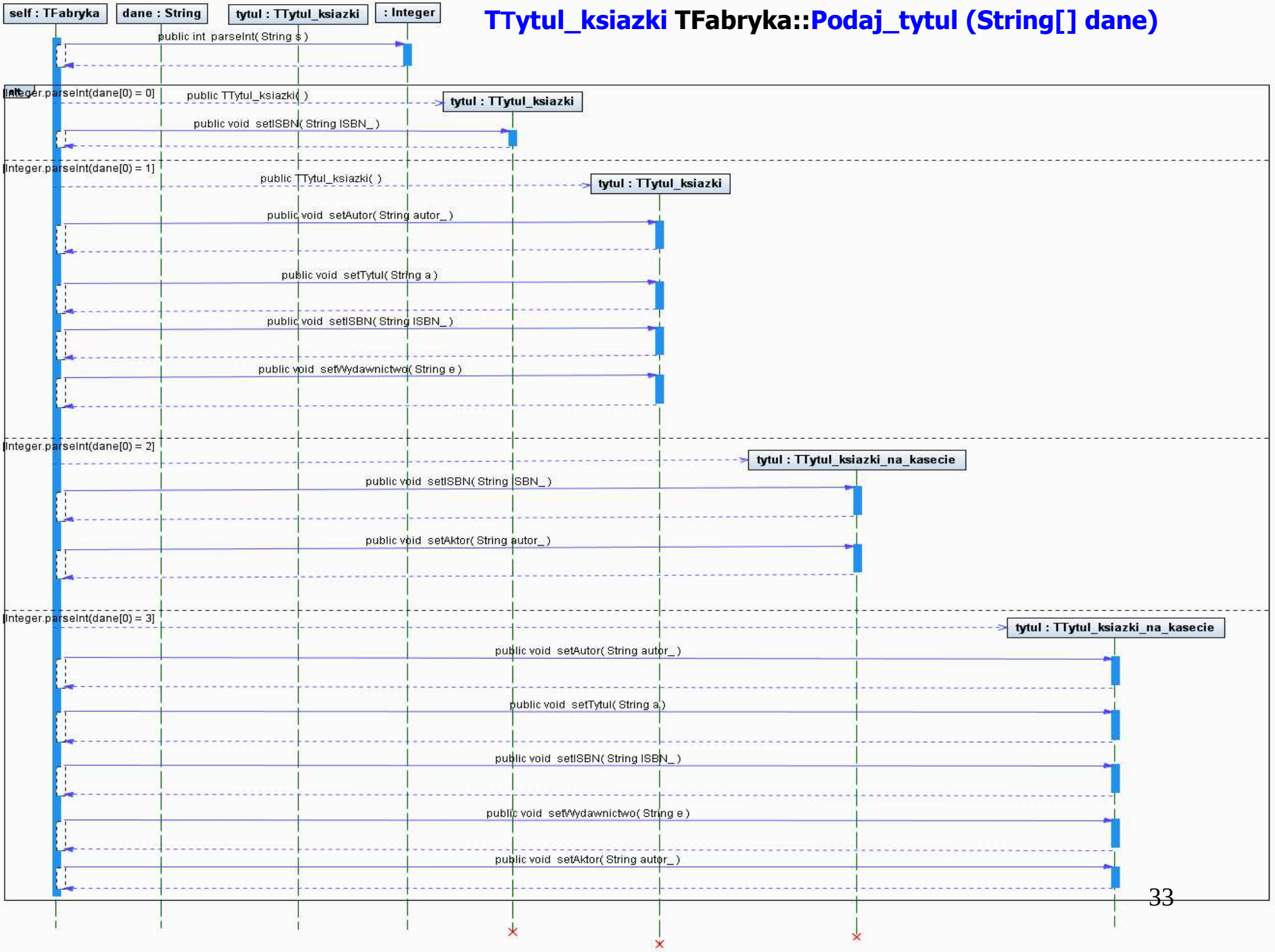
na podstawie diagramów sekwencji

(2) Dodaj tytul

void TAplikacja::dodaj_tytul(String dane[])



TTytul_ksiazki TFabryka::Podaj_tytul (String[] dane)



// ten kod powinien działać po uzupełnieniu kodu dla wskazanych klas
// biorących udział w wykonanych przypadkach użycia oraz
// po wykonaniu metody **toString()** w tych klasach (**TTytul_ksiazki** oraz **TTytul_ksiazki_na_kasecie**) i **getTytul_ksiazki()** w klasie **TAplikacja**

```
public static void main(String t[]) // your code here
   { TAplikacja ap = new TAplikacja();
      String t1[] = {"1", "1", "1", "1", "1"}; //t1, t2,  t3 – tablice łańcuchów  do tworzenia tytułu książki zwyklej – pierwszy łańcuch
      String t2[] = {"1", "2", "2", "2", "2"}; // jest informacją dla fabryki, jaki obiekt wygenerować
      String t3[] = {"1", "3", "3", "3", "3"}; //"1" oznacza utworzenie obiektu klasy TTytul_ksiazki, a pozostałe łańcuchy to kolejno
         // autor, tytul, ISBN, wydawnictwo dla uproszczenia w postaci cyfr -  obiekty do wstawiania
      String t4[] = {"3", "1", "1", "1", "1", "1"}; // t4, t5,t6 – tablice łańcuchów  do tworzenia tytułu książki jako nagranie
                                                    //dźwiękowe
      String t5[] = {"3", "2", "2", "2", "2", "2"}; //– pierwszy łańcuch jest informacją dla fabryki, jaki obiekt wygenerować
      String t6[] = {"3", "4", "4", "4", "4", "4"}; //"3" oznacza utworzenie obiektu klasy TTytul_ksiazki_na_kasecie, a pozostałe
          //łańcuchy to kolejno autor, tytul, ISBN, wydawnictwo, aktor dla uproszczenia w postaci cyfr-  obiekty do wstawiania
      ap.dodaj_tytul(t1);
      ap.dodaj_tytul(t2);        ap.dodaj_tytul(t2);
      ap.dodaj_tytul(t3);
      ap.dodaj_tytul(t4);
      ap.dodaj_tytul(t5);        ap.dodaj_tytul(t5);
      ap.dodaj_tytul(t6);
      String lan = ap.getTytul_ksiazki().toString();
      System.out.println(lan);


     }
```

Projekt przypadku użycia

„Szukaj egzemplarz”

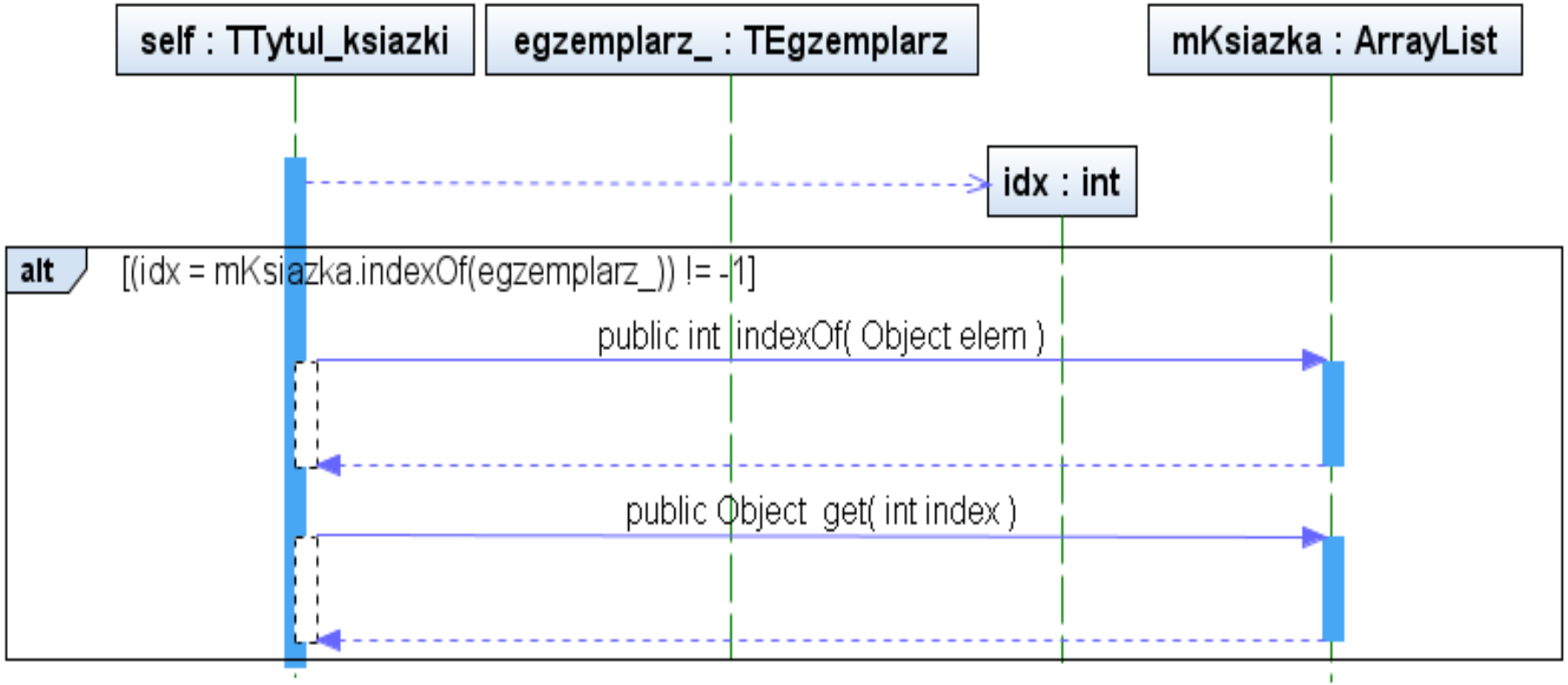
za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

Definiowanie kodu metod realizujących przypadek użycia

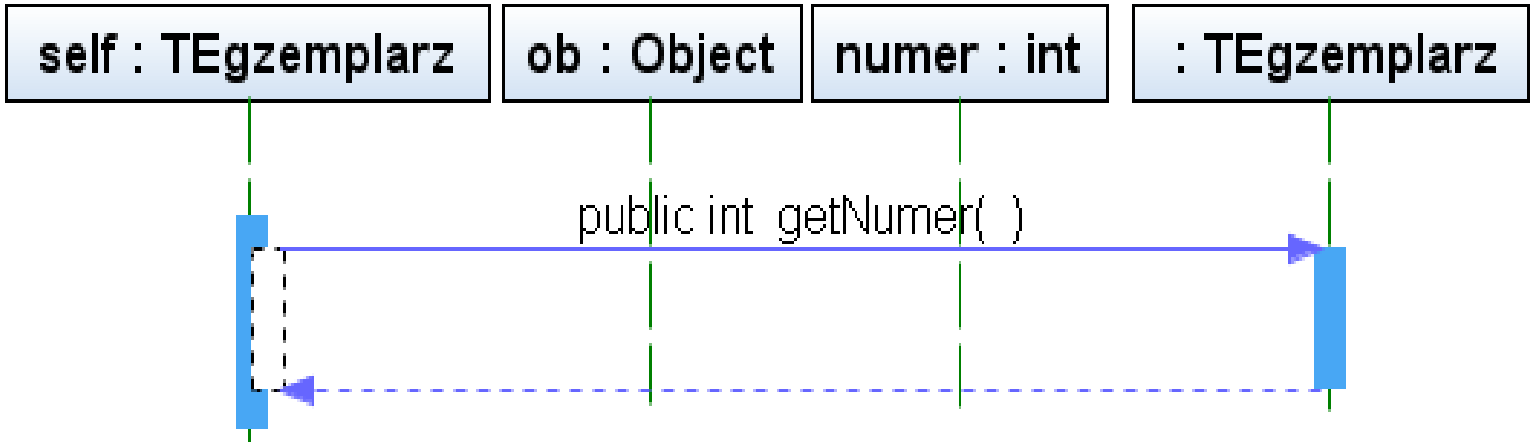
na podstawie diagramów sekwencji

(3) Szukaj egzemplarz

TEgzemplarz TTytul_ksiazki::Szukaj_egzemplarz(TEgzemplarz egzemplarz_)



boolean TEgzemplarz::equals(Object ob)



Projekt przypadku użycia

„ Dodaj egzemplarz ”

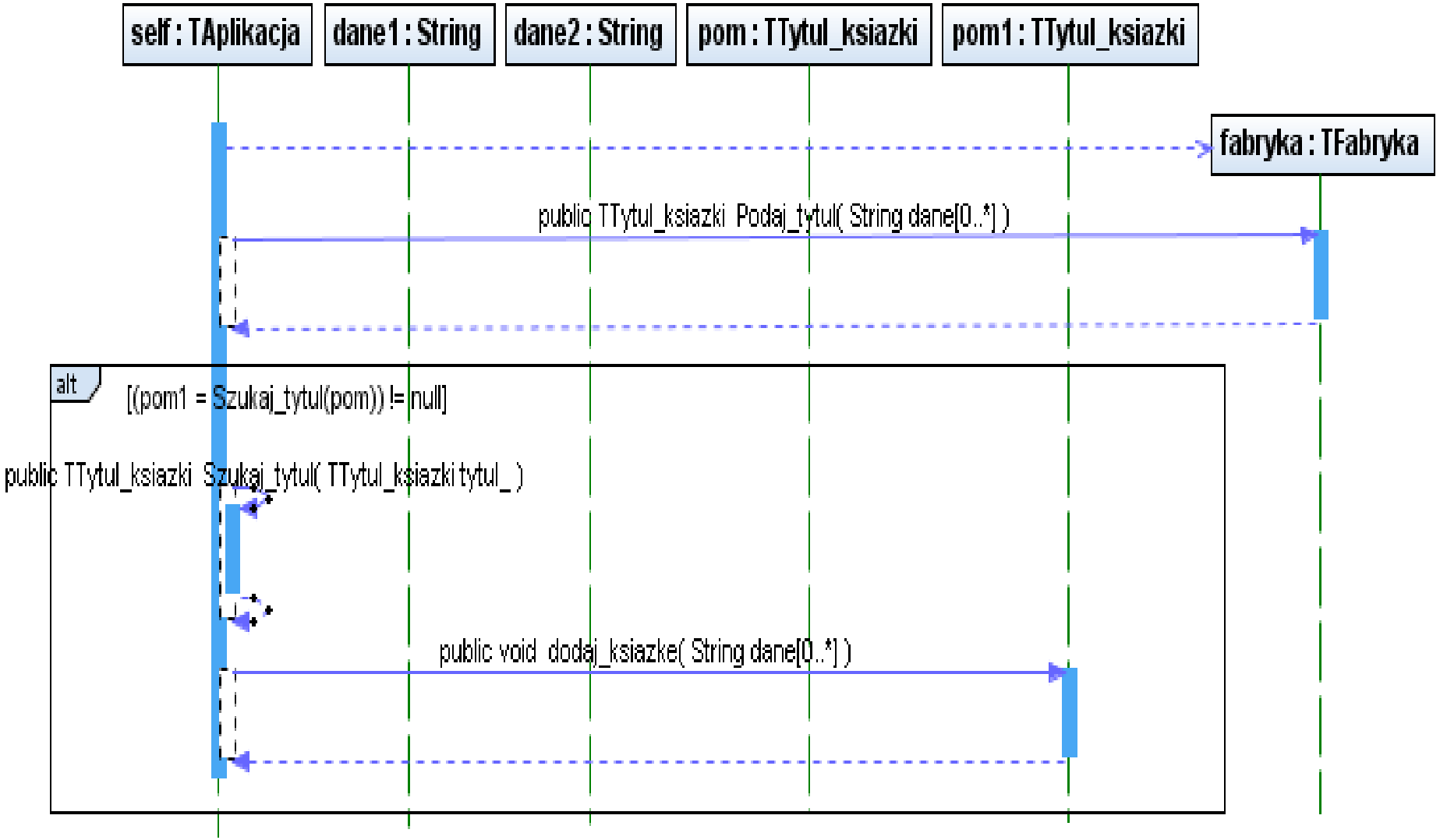
za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

Definiowanie kodu metod realizujących przypadek użycia

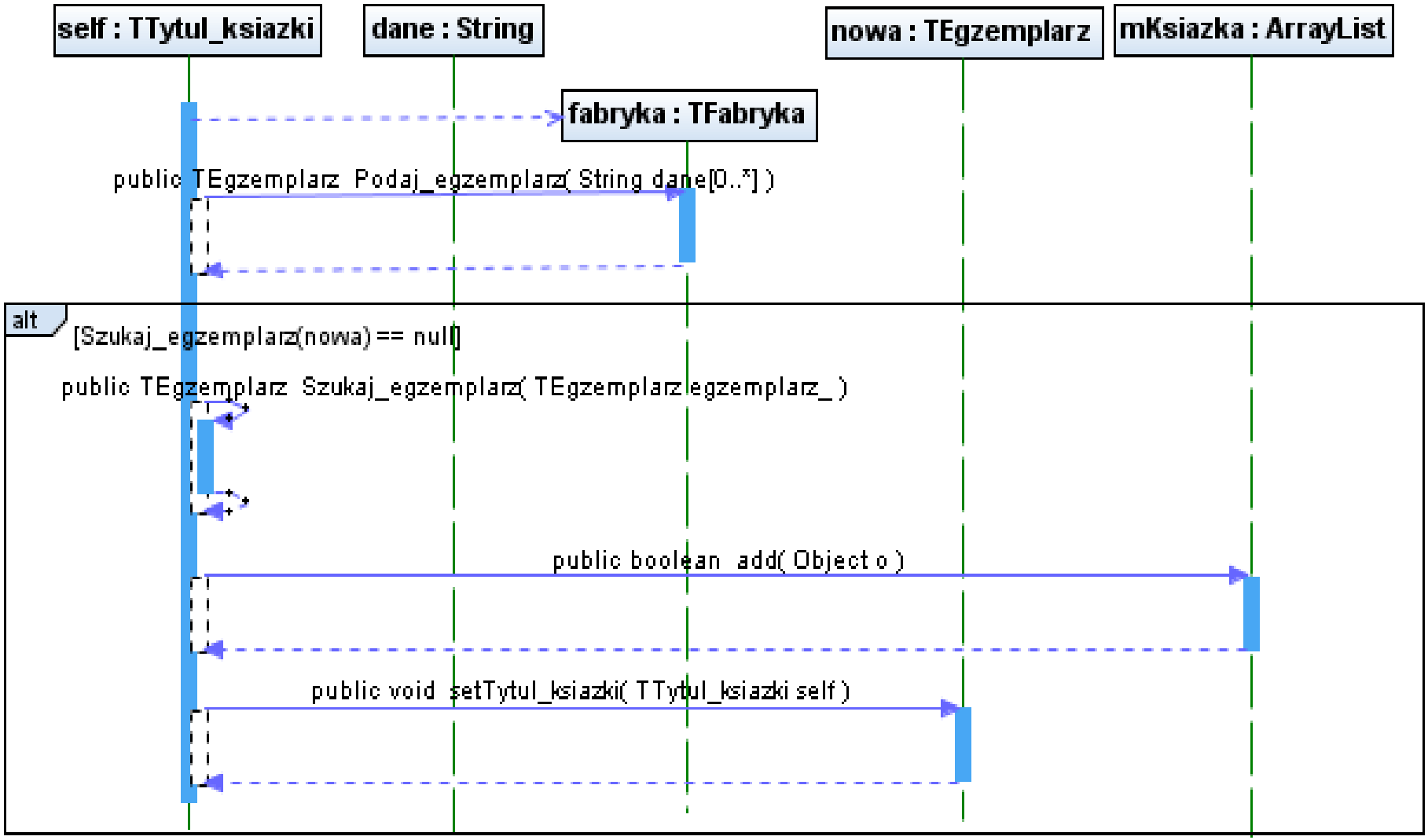
na podstawie diagramów sekwencji

(4) Dodaj egzemplarz

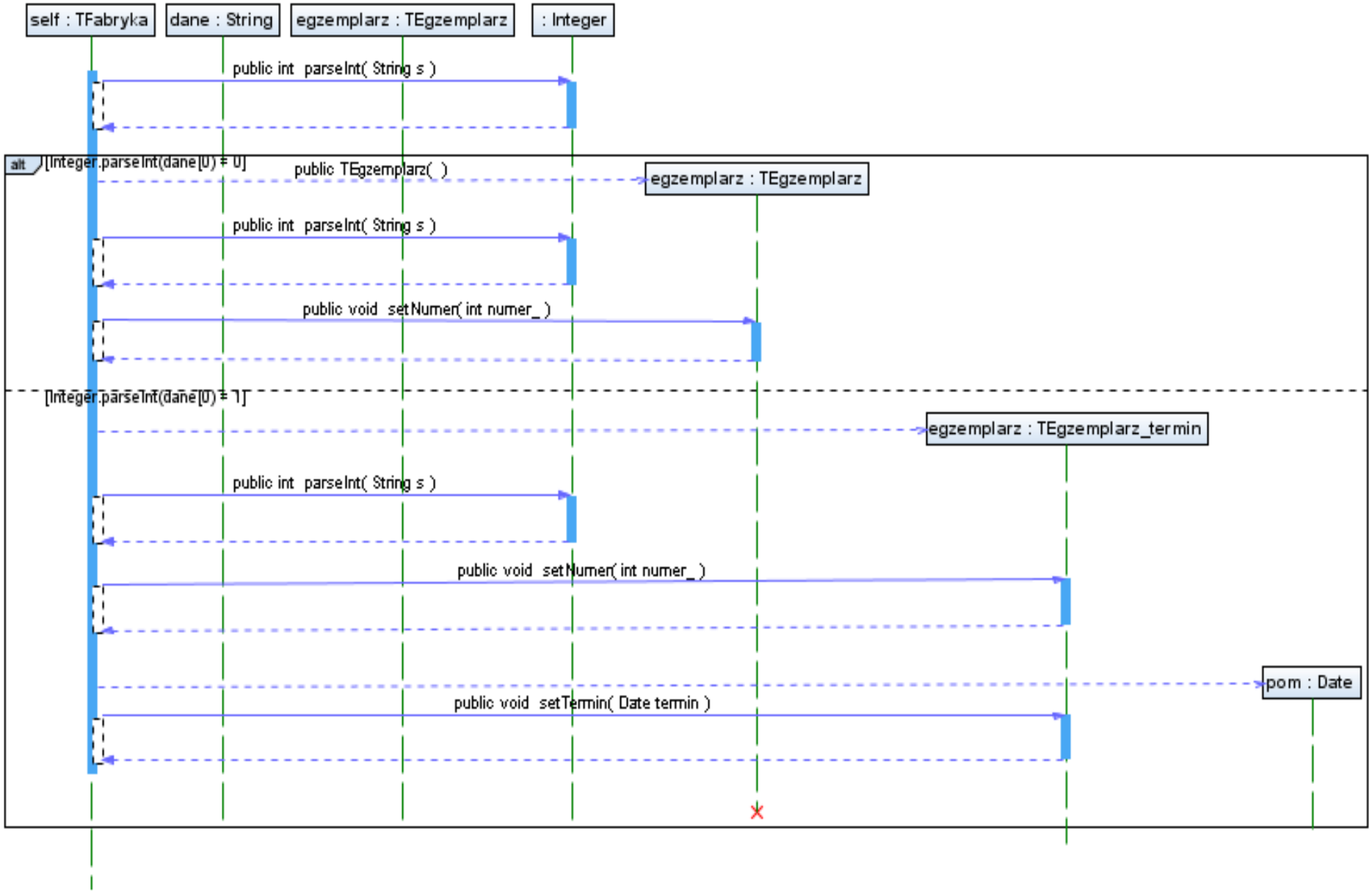
TTytul_ksiazki TAplikacja::dodaj_ksiazke(String dane1[], String dane2[]))



void TTytul_ksiazki::dodaj_ksiazke(String dane[])



TEgzemplarz TFabryka::Podaj_egzemplarz (String[] dane)



Proponowany kod funkcji main w klasie fasadowej TAplikacja

```
public static void main(String t[]) // your code here
   {TAplikacja ap = new TAplikacja();
      String t1[] = {"1", "1", "1", "1", "1"}; //t1, t2,  t3 – tablice łańcuchów  do tworzenia tytułu książki zwyklej – pierwszy łańcuch
      String t2[] = {"1", "2", "2", "2", "2"}; // jest informacją dla fabryki, jaki obiekt wygenerować
      String t3[] = {"1", "3", "3", "3", "3"}; //"1" oznacza utworzenie obiektu klasy TTytul_ksiazki, a pozostałe łańcuchy to kolejno
     // autor, tytul, ISBN, wydawnictwo dla uproszczenia w postaci cyfr -  obiekty do wstawiania
      String t4[] = {"3", "1", "1", "1", "1", "1"}; // t4, t5,t6 – tablice łańcuchów  do tworzenia tytułu książki jako nagranie dźwiękowe
      String t5[] = {"3", "2", "2", "2", "2", "2"}; //– pierwszy łańcuch jest informacją dla fabryki, jaki obiekt wygenerować
      String t6[] = {"3", "4", "4", "4", "4", "4"}; //"3" oznacza utworzenie obiektu klasy TTytul_ksiazki_na_kasecie, a pozostałe
     //łańcuchy to kolejno autor, tytul, ISBN, wydawnictwo, aktor dla uproszczenia w postaci cyfr-  obiekty do wstawiania
      ap.dodaj_tytul(t1);
      ap.dodaj_tytul(t2);        ap.dodaj_tytul(t2);
      ap.dodaj_tytul(t3);
      ap.dodaj_tytul(t4);
      ap.dodaj_tytul(t5);        ap.dodaj_tytul(t5);
      ap.dodaj_tytul(t6);
      String lan = ap.getTytul_ksiazki().toString();
      System.out.println(lan);
      String d1[] = {"0", "1"}; // d1, d2, d3 - – tablice łańcuchów  do tworzenia wzorcowego tytułu książki zwyklej  do wyszukiwania
      String d2[] = {"0", "2"}; // – pierwszy łańcuch jest informacją dla fabryki, jaki obiekt wygenerować: „0" oznacza generowanie
      String d3[] = {"0", "5"}; // obiektu klasy TTytul_ksiazki, drugi łańcuch jest ISBN – obiekty do wyszukiwania
      String d4[] = {"2", "1", "1"}; //d4, d5 - tablice łańcuchów  do tworzenia  wzorcowego tytułu książki jako nagranie dźwiękowe
      String d5[] = {"2", "4", "4"}; //pierwszy łańcuch „2" oznacza generowanie obiektu typu TTytul_ksiazki_na_kasecie
                                       // drugi łańcuch to  ISBN, trzeci jest nazwiskiem aktora - obiekty do wyszukiwania
      String tr1[] = {"0", "1"};  //tablice tr1 i tr2 zawierają informację o tworzeniu obiektu typu TEgzemplarz: pierwszy łańcuch
      String tr2[] = {"0", "2"};  //równy „0" oznacza tworzenie obiektu typu typu TEgzemplarz, drugi jest numerem egzemplarza
      String tr3[] = {"1", "3", "April 10, 2008, 00:00:00 GMT"}; //pierwszy łańcuch równy „1" oznacza tworzenie obiektu klasy
      String tr4[] = {"1", "2", "April 10, 2008, 00:00:00 GMT"};   //TEgzemplarz_termin, drugi oznacza numer, trzeci termin
```

```
    TTytul_ksiazki pom = ap.dodaj_ksiazke(d1, tr1);
     if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
     pom = ap.dodaj_ksiazke(d2, tr1);
     if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
     pom = ap.dodaj_ksiazke(d2, tr1);
     if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
     pom = ap.dodaj_ksiazke(d2, tr2);
     if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
     pom = ap.dodaj_ksiazke(d3, tr2);
     if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
     pom = ap.dodaj_ksiazke(d4, tr3);
     if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
     pom = ap.dodaj_ksiazke(d4, tr3);
     if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
     pom = ap.dodaj_ksiazke(d4, tr4);
     if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
     pom = ap.dodaj_ksiazke(d5, tr2);
     if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }

  }
// ten kod powinien działać  po uzupełnieniu kodu dla wskazanych  klas biorących
//  udział  w wykonanych przypadkach użycia oraz
// po wykonaniu metody toString() w tych klasach
```

Projekt przypadku użycia

„ Wyszukiwanie tytulow”

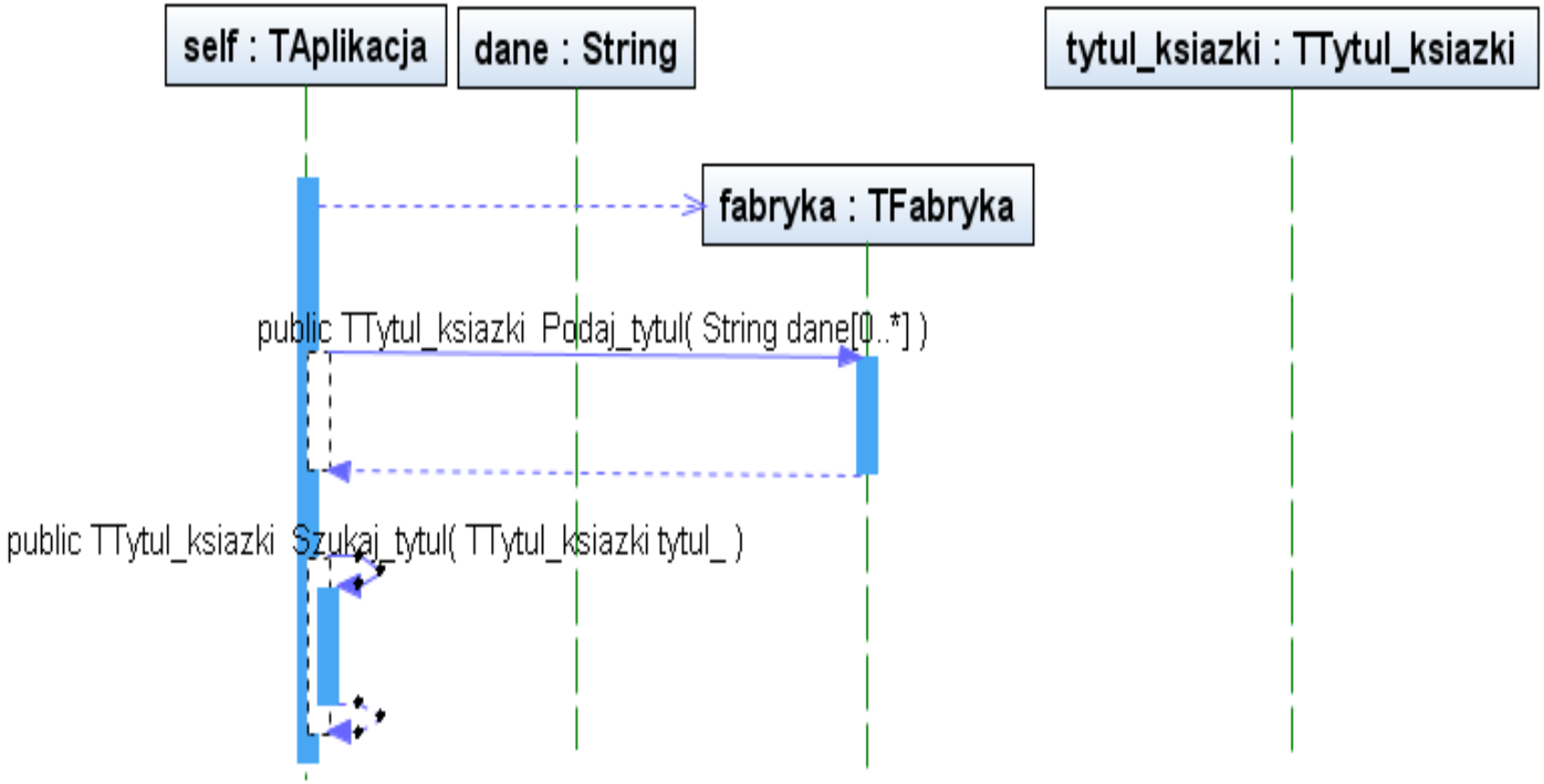
za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

Definiowanie kodu metod realizujących przypadek użycia

na podstawie diagramów sekwencji

(5) Wyszukiwanie tytulow

TTytul_ksiazki TAplikacja::Wyszukaj_tytul (String[] dane)



Proponowany kod funkcji main w klasie fasadowej TAplikacja

```
public static void main(String t[]) // your code here
   {TAplikacja ap = new TAplikacja();
      String t1[] = {"1", "1", "1", "1", "1"}; //t1, t2,  t3 – tablice łańcuchów  do tworzenia tytułu książki zwyklej – pierwszy łańcuch
      String t2[] = {"1", "2", "2", "2", "2"}; // jest informacją dla fabryki, jaki obiekt wygenerować
      String t3[] = {"1", "3", "3", "3", "3"}; //"1" oznacza utworzenie obiektu klasy TTytul_ksiazki, a pozostałe łańcuchy to kolejno
     // autor, tytul, ISBN, wydawnictwo dla uproszczenia w postaci cyfr -  obiekty do wstawiania
      String t4[] = {"3", "1", "1", "1", "1", "1"}; // t4, t5,t6 – tablice łańcuchów  do tworzenia tytułu książki jako nagranie dźwiękowe
      String t5[] = {"3", "2", "2", "2", "2", "2"}; //– pierwszy łańcuch jest informacją dla fabryki, jaki obiekt wygenerować
      String t6[] = {"3", "4", "4", "4", "4", "4"}; //"3" oznacza utworzenie obiektu klasy TTytul_ksiazki_na_kasecie, a pozostałe
     //łańcuchy to kolejno autor, tytul, ISBN, wydawnictwo, aktor dla uproszczenia w postaci cyfr-  obiekty do wstawiania
      ap.dodaj_tytul(t1);
      ap.dodaj_tytul(t2);        ap.dodaj_tytul(t2);
      ap.dodaj_tytul(t3);
      ap.dodaj_tytul(t4);
      ap.dodaj_tytul(t5);        ap.dodaj_tytul(t5);
      ap.dodaj_tytul(t6);
      String lan = ap.getTytul_ksiazki().toString();
      System.out.println(lan);
      String d1[] = {"0", "1"}; // d1, d2, d3 - – tablice łańcuchów  do tworzenia wzorcowego tytułu książki zwyklej  do wyszukiwania
      String d2[] = {"0", "2"}; // – pierwszy łańcuch jest informacją dla fabryki, jaki obiekt wygenerować: „0" oznacza generowanie
      String d3[] = {"0", "5"}; // obiektu klasy TTytul_ksiazki, drugi łańcuch jest ISBN – obiekty do wyszukiwania
      String d4[] = {"2", "1", "1"}; //d4, d5 - tablice łańcuchów  do tworzenia  wzorcowego tytułu książki jako nagranie dźwiękowe
      String d5[] = {"2", "4", "4"}; //pierwszy łańcuch „2" oznacza generowanie obiektu typu TTytul_ksiazki_na_kasecie
                                     // drugi łańcuch to  ISBN, trzeci jest nazwiskiem aktora -obiekty do wyszukiwania
      String tr1[] = {"0", "1"};  //tablice tr1 i tr2 zawierają informację o tworzeniu obiektu typu TEgzemplarz: pierwszy łańcuch
      String tr2[] = {"0", "2"};  //równy „0" oznacza tworzenie obiektu typu typu TEgzemplarz, drugi jest numerem egzemplarza
      String tr3[] = {"1", "3", "April 10, 2008, 00:00:00 GMT"}; //pierwszy łańcuch równy „1" oznacza tworzenie obiektu klasy
      String tr4[] = {"1", "2", "April 10, 2008, 00:00:00 GMT"};   //TEgzemplarz_termin, drugi oznacza numer, trzeci termin
```

```
        TTytul_ksiazki pom = ap.dodaj_ksiazke(d1, tr1);
         if (pom != null) {          System.out.println(pom.getKsiazka().toString());       }
         pom = ap.dodaj_ksiazke(d2, tr1);
         if (pom != null) {          System.out.println(pom.getKsiazka().toString());       }
         pom = ap.dodaj_ksiazke(d2, tr1);
         if (pom != null) {          System.out.println(pom.getKsiazka().toString());       }
         pom = ap.dodaj_ksiazke(d2, tr2);
         if (pom != null) {          System.out.println(pom.getKsiazka().toString());       }
         pom = ap.dodaj_ksiazke(d3, tr2);
         if (pom != null) {          System.out.println(pom.getKsiazka().toString());       }
         pom = ap.dodaj_ksiazke(d4, tr3);
         if (pom != null) {          System.out.println(pom.getKsiazka().toString());       }
         pom = ap.dodaj_ksiazke(d4, tr3);
         if (pom != null) {          System.out.println(pom.getKsiazka().toString());       }
         pom = ap.dodaj_ksiazke(d4, tr4);
         if (pom != null) {          System.out.println(pom.getKsiazka().toString());       }
         pom = ap.dodaj_ksiazke(d5, tr2);
         if (pom != null) {          System.out.println(pom.getKsiazka().toString());       }

         ap.Wyswietl_tytuly();
         ap.Wyswietl_ksiazki();

         System.out.println("Wyszukiwanie");
         System.out.println(ap.Wyszukaj_tytul(t5).toString());
}

// ten kod powinien działać  po uzupełnieniu kodu dla wskazanych  klas biorących
//  udział  w wykonanych przypadkach użycia oraz
// po wykonaniu metody toString() w tych klasach oraz  metod  Wyswietl_tytuly oraz  Wyswietl_ksiazki
```

Projekt przypadku użycia

„Wyszukiwanie egzemplarzy”

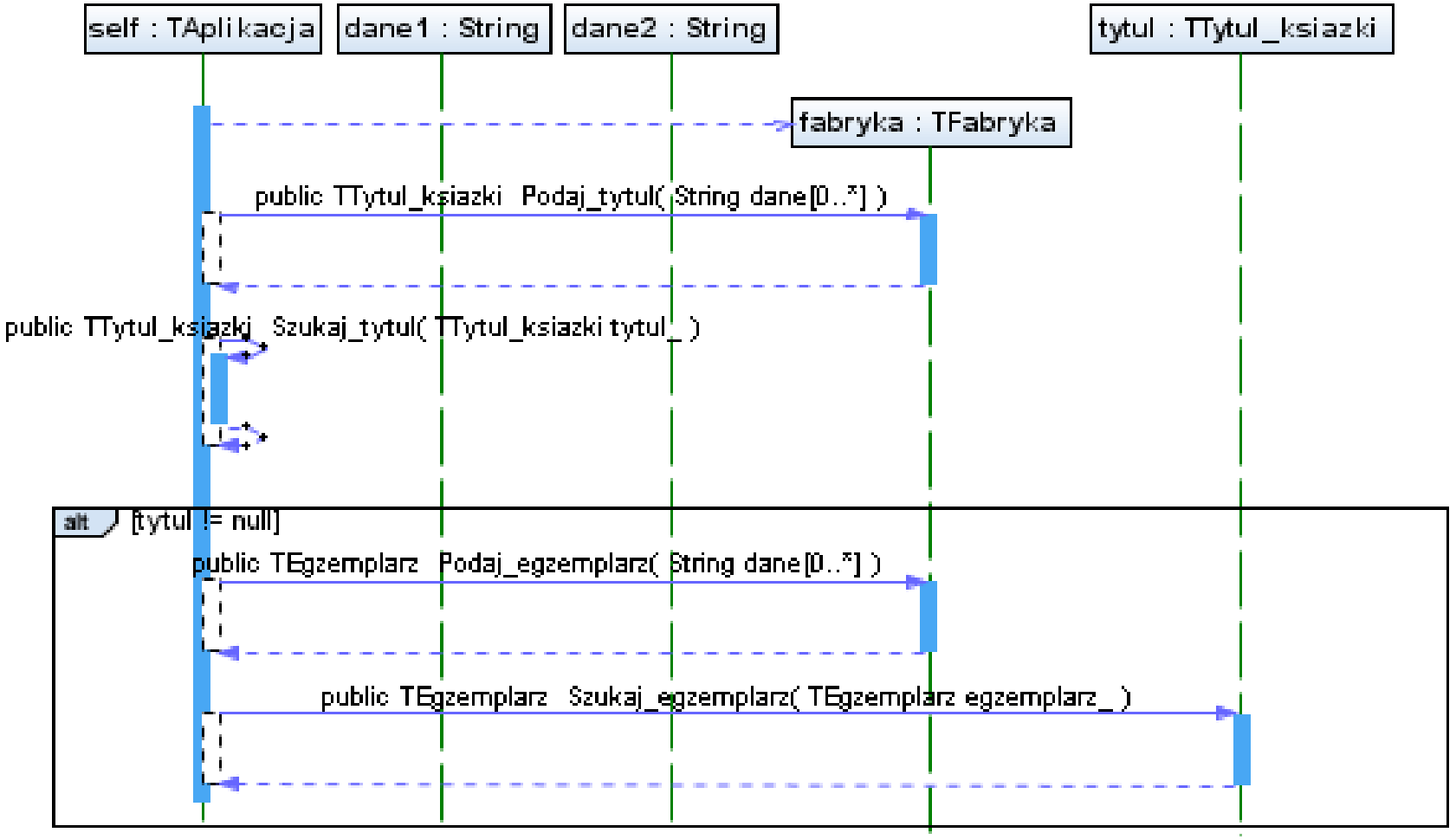
za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

Definiowanie kodu metod realizujących przypadek użycia

na podstawie diagramów sekwencji

(6) Wyszukiwanie egzemplarzy

TEgzemplarz TAplikacja::Wyszukaj_egzemplarz (String[] dane1, String[] dane2)



Proponowany kod funkcji main w klasie fasadowej TAplikacja

```
public static void main(String t[]) // your code here
   {TAplikacja ap = new TAplikacja();
      String t1[] = {"1", "1", "1", "1", "1"}; //t1, t2,  t3 – tablice łańcuchów  do tworzenia tytułu książki zwyklej – pierwszy łańcuch
      String t2[] = {"1", "2", "2", "2", "2"}; // jest informacją dla fabryki, jaki obiekt wygenerować
      String t3[] = {"1", "3", "3", "3", "3"}; //"1" oznacza utworzenie obiektu klasy TTytul_ksiazki, a pozostałe łańcuchy to kolejno
     // autor, tytul, ISBN, wydawnictwo dla uproszczenia w postaci cyfr -  obiekty do wstawiania
      String t4[] = {"3", "1", "1", "1", "1", "1"}; // t4, t5,t6 – tablice łańcuchów  do tworzenia tytułu książki jako nagranie dźwiękowe
      String t5[] = {"3", "2", "2", "2", "2", "2"}; //– pierwszy łańcuch jest informacją dla fabryki, jaki obiekt wygenerować
      String t6[] = {"3", "4", "4", "4", "4", "4"}; //"3" oznacza utworzenie obiektu klasy TTytul_ksiazki_na_kasecie, a pozostałe
     //łańcuchy to kolejno autor, tytul, ISBN, wydawnictwo, aktor dla uproszczenia w postaci cyfr-  obiekty do wstawiania
      ap.dodaj_tytul(t1);
      ap.dodaj_tytul(t2);        ap.dodaj_tytul(t2);
      ap.dodaj_tytul(t3);
      ap.dodaj_tytul(t4);
      ap.dodaj_tytul(t5);        ap.dodaj_tytul(t5);
      ap.dodaj_tytul(t6);
      String lan = ap.getTytul_ksiazki().toString();
      System.out.println(lan);
      String d1[] = {"0", "1"}; // d1, d2, d3 - – tablice łańcuchów  do tworzenia wzorcowego tytułu książki zwyklej  do wyszukiwania
      String d2[] = {"0", "2"}; // – pierwszy łańcuch jest informacją dla fabryki, jaki obiekt wygenerować: „0" oznacza generowanie
      String d3[] = {"0", "5"}; // obiektu klasy TTytul_ksiazki, drugi łańcuch jest ISBN – obiekty do wyszukiwania
      String d4[] = {"2", "1", "1"}; //d4, d5 - tablice łańcuchów  do tworzenia  wzorcowego tytułu książki jako nagranie dźwiękowe
      String d5[] = {"2", "4", "4"}; //pierwszy łańcuch „2" oznacza generowanie obiektu typu TTytul_ksiazki_na_kasecie
                                       // drugi łańcuch to  ISBN, trzeci jest nazwiskiem aktora-obiekty do wyszukiwania
      String tr1[] = {"0", "1"};  //tablice tr1 i tr2 zawierają informację o tworzeniu obiektu typu TEgzemplarz: pierwszy łańcuch
      String tr2[] = {"0", "2"};  //równy „0" oznacza tworzenie obiektu typu typu TEgzemplarz, drugi jest numerem egzemplarza
      String tr3[] = {"1", "3", "April 10, 2008, 00:00:00 GMT"}; //pierwszy łańcuch równy „1" oznacza tworzenie obiektu klasy
      String tr4[] = {"1", "2", "April 10, 2008, 00:00:00 GMT"};   //TEgzemplarz_termin, drugi oznacza numer, trzeci termin
```

```
    TTytul_ksiazki pom = ap.dodaj_ksiazke(d1, tr1);
     if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
     pom = ap.dodaj_ksiazke(d2, tr1);
     if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
     pom = ap.dodaj_ksiazke(d2, tr1);
     if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
     pom = ap.dodaj_ksiazke(d2, tr2);
     if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
     pom = ap.dodaj_ksiazke(d3, tr2);
     if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
     pom = ap.dodaj_ksiazke(d4, tr3);
     if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
     pom = ap.dodaj_ksiazke(d4, tr3);
     if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
     pom = ap.dodaj_ksiazke(d4, tr4);
     if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }
     pom = ap.dodaj_ksiazke(d5, tr2);
     if (pom != null) {          System.out.println(pom.getKsiazka().toString());      }

     ap.Wyswietl_tytuly();
     ap.Wyswietl_ksiazki();

     System.out.println("Wyszukiwanie");
     System.out.println(ap.Wyszukaj_tytul(t5).toString());
     System.out.println(ap.Wyszukaj_egzemplarz(d4, tr4).toString());
   }

// ten kod powinien działać  po uzupełnieniu kodu dla wskazanych  klas biorących
//  udział  w wykonanych przypadkach użycia oraz
// po wykonaniu metody toString() w tych klasach oraz  metod  Wyswietl_tytuly oraz  Wyswietl_ksiazki
```