Przykład 1
Iteracja 1 tworzenia oprogramowania

Opis biznesowy „świata rzeczywistego”

Wymagania funkcjonalne i niefunkcjonalne aplikacji

Diagram przypadków życia

Diagramy klas i sekwencji:

Relacja 1 do 0..*

Cele iteracji 1

Należy:

- wybrać projekt z podanej listy dostępnej za pomocą linku podanego w w laboratorium 1
- sformułować wymagania funkcjonalne i niefunkcjonalne dla wybranego projektu jako zadanie domowe. Zadanie domowe będzie stanowić podstawę do zaprojektowania przypadków użycia na kolejnych laboratorium.
- Wykonać projekt UML i wykonać prosty program stanowiący realizację 1-go etapu wykonania wybranego projektu. Instrukcja zawiera przykłady powiązania 1:* (jeden do wiele) między klasami, użytymi do realizacji 1-go etapu programu „Projekt_lab1” – należy wykorzystać ten typ powiązań w wykonywanym 1-ym etapie projektu i jego implementacji.

Zawartość 1 części projektu

1. Opis „świata rzeczywistego”
2. Wymagania funkcjonalne i niefunkcjonalne programu na podstawie opisu z p.1
3. Diagram przypadków użycia specyfikujący wybrane wymagania z p.2 – opisy poszczególnych przypadków użycia
4. Diagram klas zidentyfikowany na podstawie opisów przypadków użycia
5. Diagramy sekwencji reprezentujące scenariusze ważniejszych przypadków użycia.

Java

język programowania

- obiektowo zorientowany
- wysokiego poziomu

platforma Javy

- z maszyny wirtualnej VM
- API (interfejs programowania aplikacji).

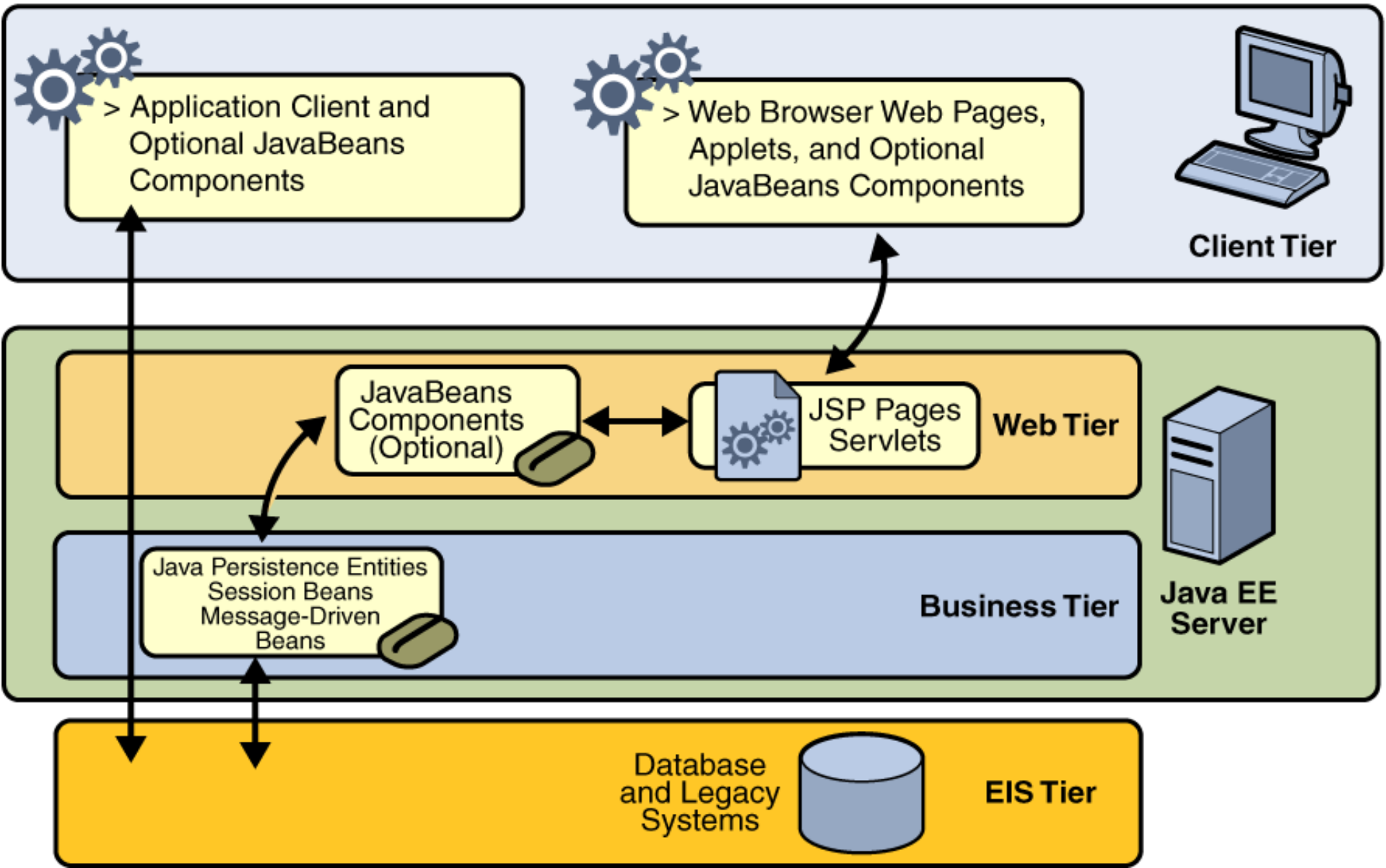
Rezultat

- niezależność od platformy,
- duże możliwości,
- stabilność,
- łatwość rozwoju,
- bezpieczeństwo

Rodzaje platform Javy:

- Java Platform, Standard Edition (Java SE)
- Java Platform, Enterprise Edition (Java EE)
- Java Platform, Micro Edition (Java ME)
- Java Platform CARD

Warstwy aplikacji (Java EE)



1./1.1. Należy wykonać opis biznesowy „świata rzeczywistego” – Katalog tytułów i książek w bibliotece

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. 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.

1.2. Należy zdefiniować wymagania aplikacji

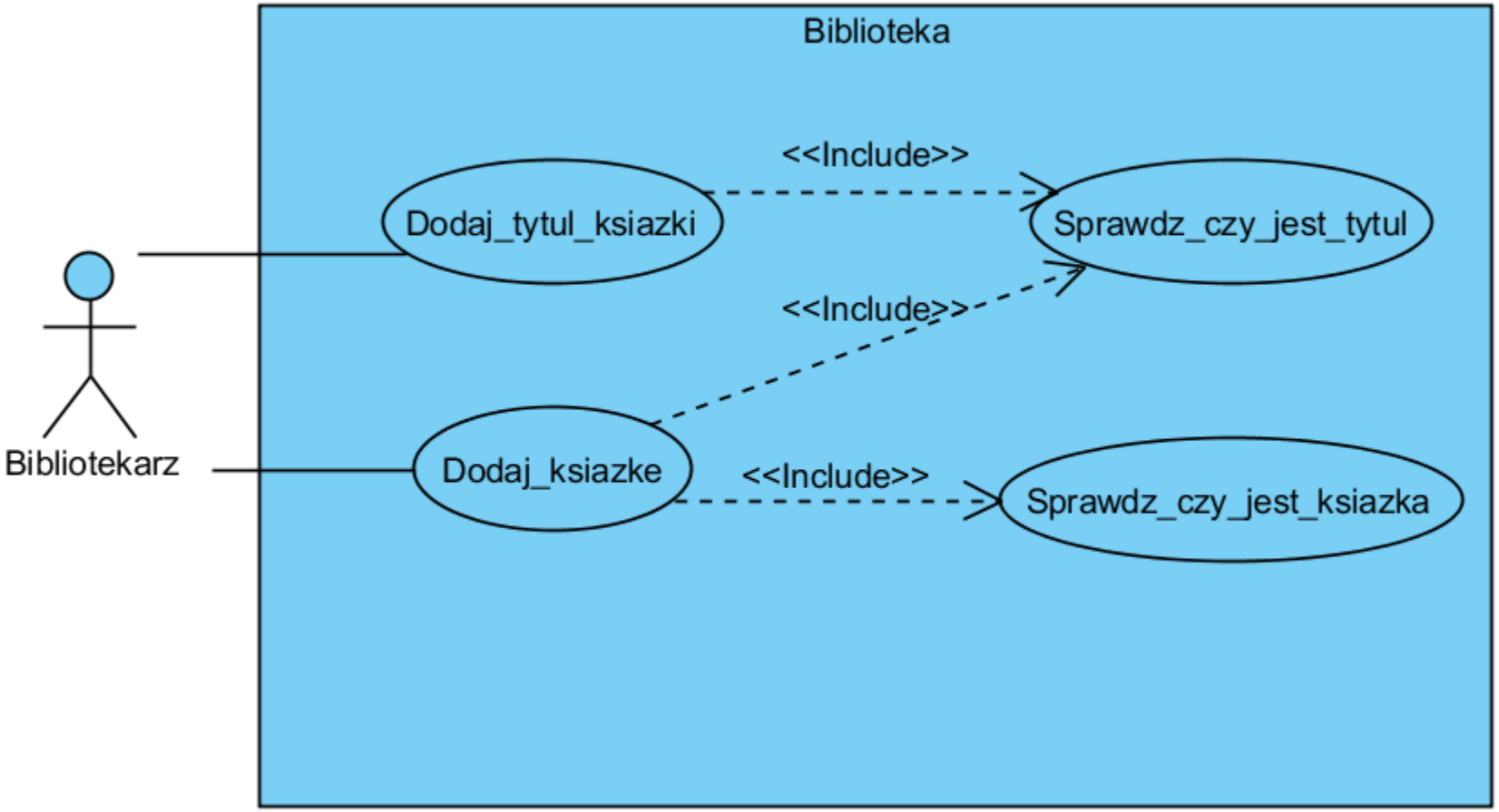
Wymagania funkcjonalne

- Biblioteka wypożycza podane książki i czasopisma osobom zarejestrowanym, o ile je posiada
- Biblioteka dokonuje zakupu nowych książek, przy czym popularne książki kupuje w kilku egzemplarzach. Usuwa zniszczone książki i czasopisma.
- Bibliotekarz jest pracownikiem biblioteki, komunikuje się z wypożyczającym. Jego praca jest wspierana za pomocą systemu
- Wypożyczający może zarezerwować książkę lub czasopismo, które nie jest dostępne w danej chwili, W momencie, kiedy zamówione rzeczy są dostępne- albo po zwrocie lub dzięki zakupowi, można je wypożyczyć i usunąć rezerwację. Rezerwację można usunąć niezależnie.
- Biblioteka może łatwo utworzyć, zmienić i usunąć informację o tytułach, wypożyczających, wypożyczeniach i rezerwacjach

Wymagania niefunkcjonalne

- System powinien pracować w popularnych systemach (UNIX, Windows, OS/2) i powinien mieć nowoczesny graficzny interfejs użytkownika
- System powinien się rozwijać np. wprowadzenie możliwości zawiadamiania rezerwującego książkę o jej dostępności

1.3. Diagram przypadków użycia (częściowa realizacja wymagań funkcjonalnych)



1.4./1.4.1. Scenariusze przypadków użycia

Nazwa: Dodaj_tytul_ksiazki
Cel: Dodanie nowego tytułu książki o unikatowym ISBN
Warunki początkowe:
Uruchomienie programu jako aplikacji www lub aplikacji w architekturze typu "klient-serwer„
Warunki końcowe:
Wprowadzenia danych tytułu o unikalnym numerze ISBN
Scenariusz:
1. Należy podać dane tytułu: imię i nazwisko autora, tytuł książki, wydawnictwo, ISBN
2. Należy sprawdzić, czy dane wprowadzanego tytułu są unikalne za pomocą wywołania PU Sprawdż, czy jest tytul, przekazując ISBN tytułu
3. Jeśli tytuł o podanym ISBN istnieje, należy zakończyć przypadek użycia, w przeciwnym razie należy zapisać dane.

Nazwa: Sprawdz_czy_jest_tytul
Cel: Wyszukiwanie tytułu książki o podanym ISBN
Warunki początkowe:
Jest uruchamiany z PU Dodaj_tytul_ksiazki oraz PU Dodaj_ksiazke
Warunki końcowe:
Zwraca wynik, określający, czy podany ISBN jest unikatowy lub podaje informację, że dany ISBN już istnieje
Scenariusz:
1. Porównuje ISBN podanego tytułu książki i numerami ISBN pozostałych tytułów książek, przechowywanych w bibliotece.
2. W przypadku znalezienia tytułu o takim samym numerze ISBN PU kończy przeglądanie numerów ISBN pozostałych tytułów książek i zwraca znaleziony tytuł książki
3. W przypadku braku tytułu książki o podanym numerze ISBN, po przejrzeniu tytułów książek, zwracany jest wynik negatywny

1.4.2. Scenariusze przypadków użycia

Nazwa: Dodaj_ksiazke
Cel: Dodanie nowej książki
Warunki początkowe:
Uruchomienie programu jako aplikacji www lub aplikacji w architekturze typu "klient-serwer„
Warunki końcowe:
Wprowadzenia danych ksiazki o unikalnym numerze egzemplarza w ramach książek o tym samym tytule
Scenariusz:

1. Należy podać atrybuty tytułu: ISBN jako obowiązkowa. Tworzony jest tytuł wzorcowy do wyszukiwania rzeczywistego tytułu
2. Należy wywołać **PU Sprawdz_czy_jest_tytul**. Należy sprawdzić, czy tytuł o podanym ISBN już istnieje. Jeśli nie, należy zakończyć PU.
3. Należy utworzyć egzemplarz zawierający numer podany do wyszukiwania egzemplarza i należy przekazać go do **PU Sprawdz_czy_jest_ksiazka.** Jeśli nie istnieje egzemplarz o danym numerze, należy wstawić ten egzemplarz, w przeciwnym wypadku należy zakończyć PU.

Nazwa: Sprawdz_czy_jest_ksiazka
Cel: Wyszukiwanie książki o podanym numerze
Warunki początkowe:
Przypadek użycia jest wywoływany z PU Dodaj_ksiazke
Warunki końcowe:
Zwraca wynik, określający, czy podany numer jest unikatowy lub podaje informację, że dany numer już istnieje
Scenariusz:

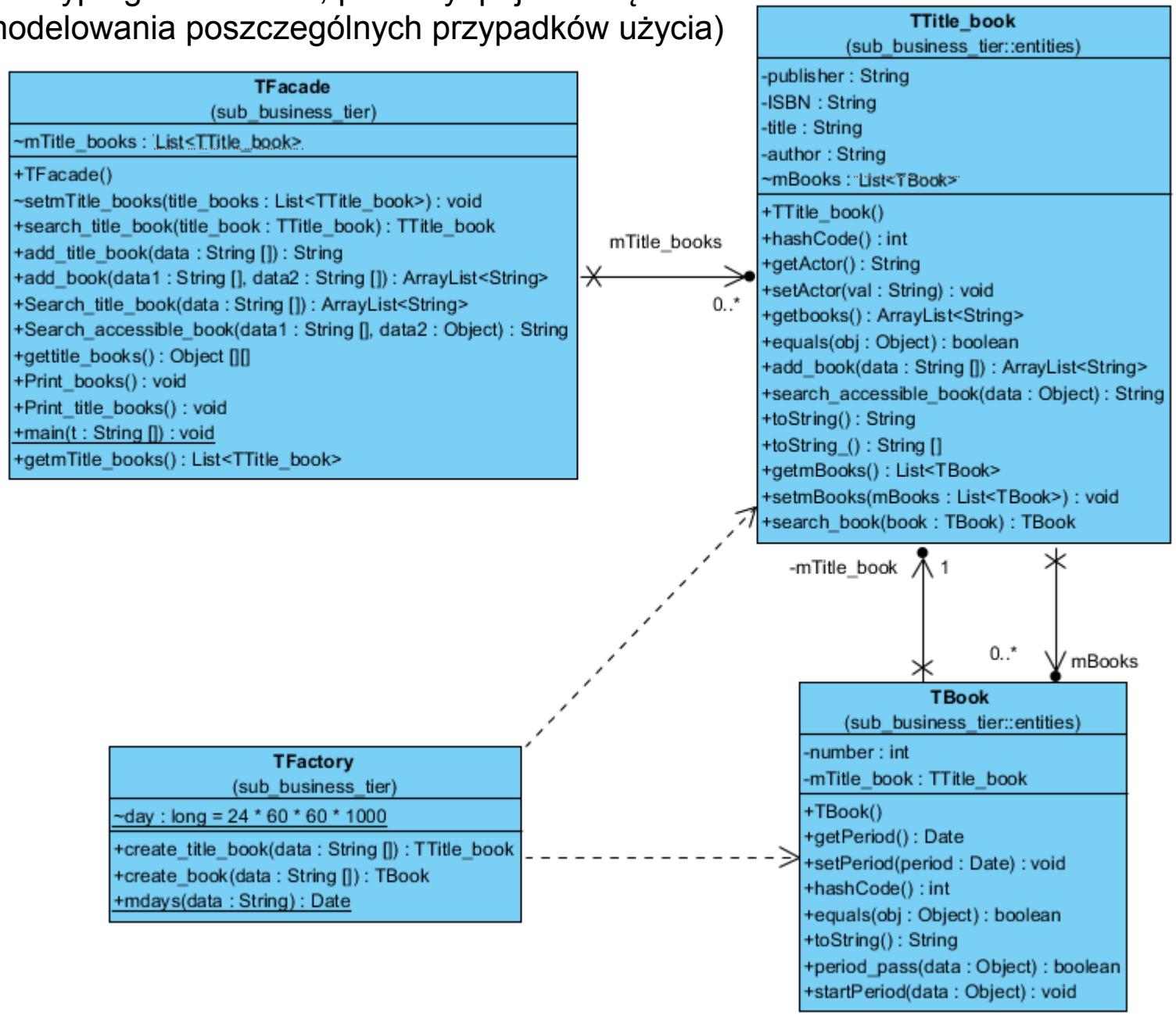
1. Szukanie książki przebiega według atrybutu: numer egzemplarza (obowiązkowo) zgodnie z danymi tytułu podanego do przypadku użycia. Przeszukiwane są egzemplarze należące do konkretnego tytułu
2. Jeśli istnieje egzemplarz o podanym numerze, zwracany jest egzemplarz z zasobów wypożyczalni, w przeciwnym wypadku zwracana jest informacja o braku egzemplarza.

1.5. Identyfikacja klas na podstawie diagramu przypadków użycia

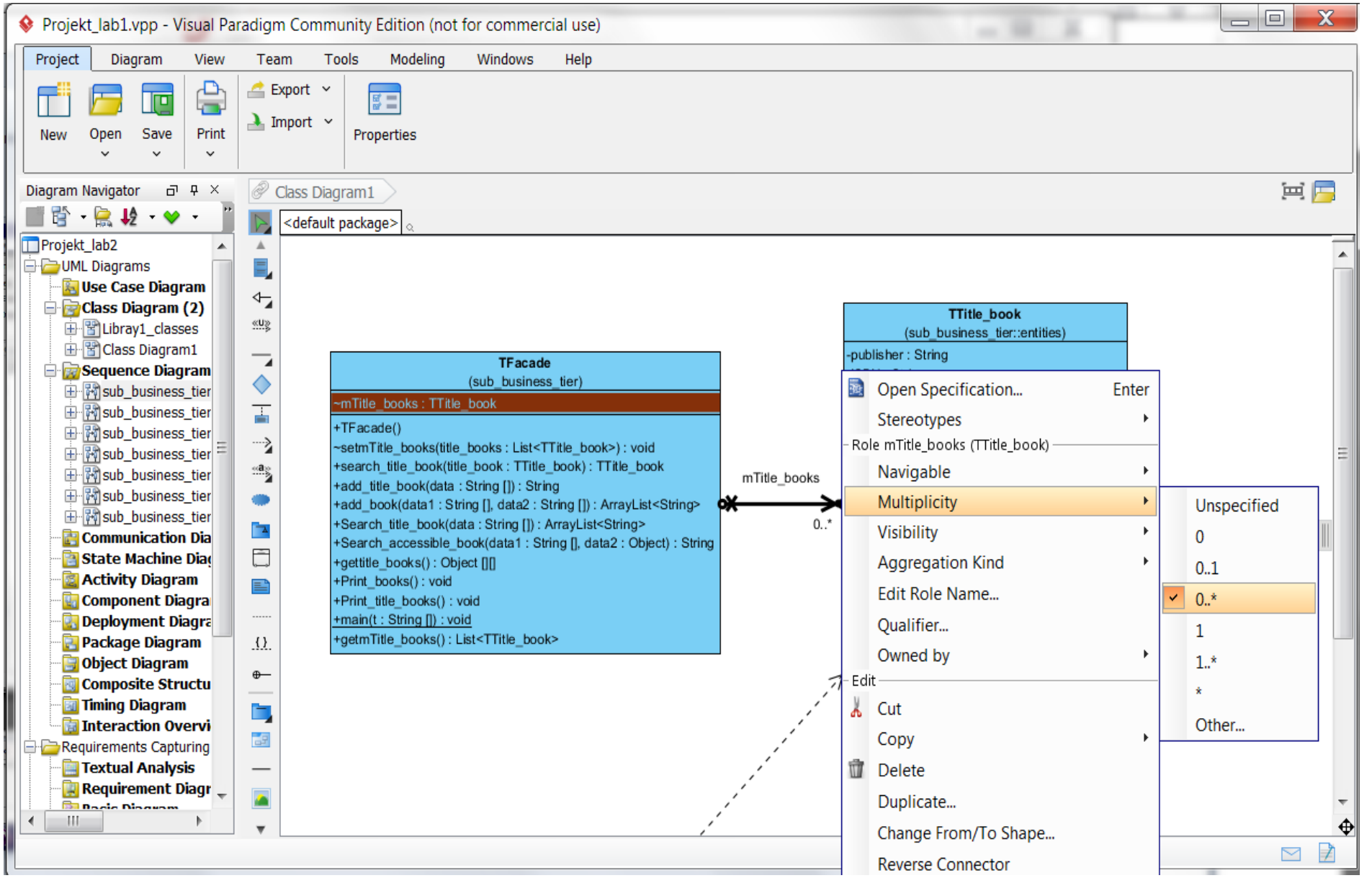
Przypadek użycia	**Implementacja**	**Relacja**	**Klasa**
Dodaj_tytul_ksiazki			TTitle_book
Sprawdz_czy_jest_tytul			TTitle_book
Dodaj_ksiazke	Zbiór obiektów typu TBook	Dwukierunkowa relacja 1 do	TTitle_book
	Obiekt typu TTitle_book	wiele	TBook
Sprawdz_czy_jest_ksiazka			TBook

Klasa typu **TFacade** reprezentuje wzorzec **Fasady**, klasa **TFactory** (wzorce strukturalne) reprezentuje wzorzec **Fabryki** (wzorce wytwórcze), klasa **TTitle_book** reprezentuje wzorzec **Pyłek** (wzorzec strukturalny)

1.5. Diagram zidentyfikowanych klas (metody klas TTitle_book oraz TBook , oprócz metod typu getter i setter, powinny pojawić się dopiero w wyniku modelowania poszczególnych przypadków użycia)



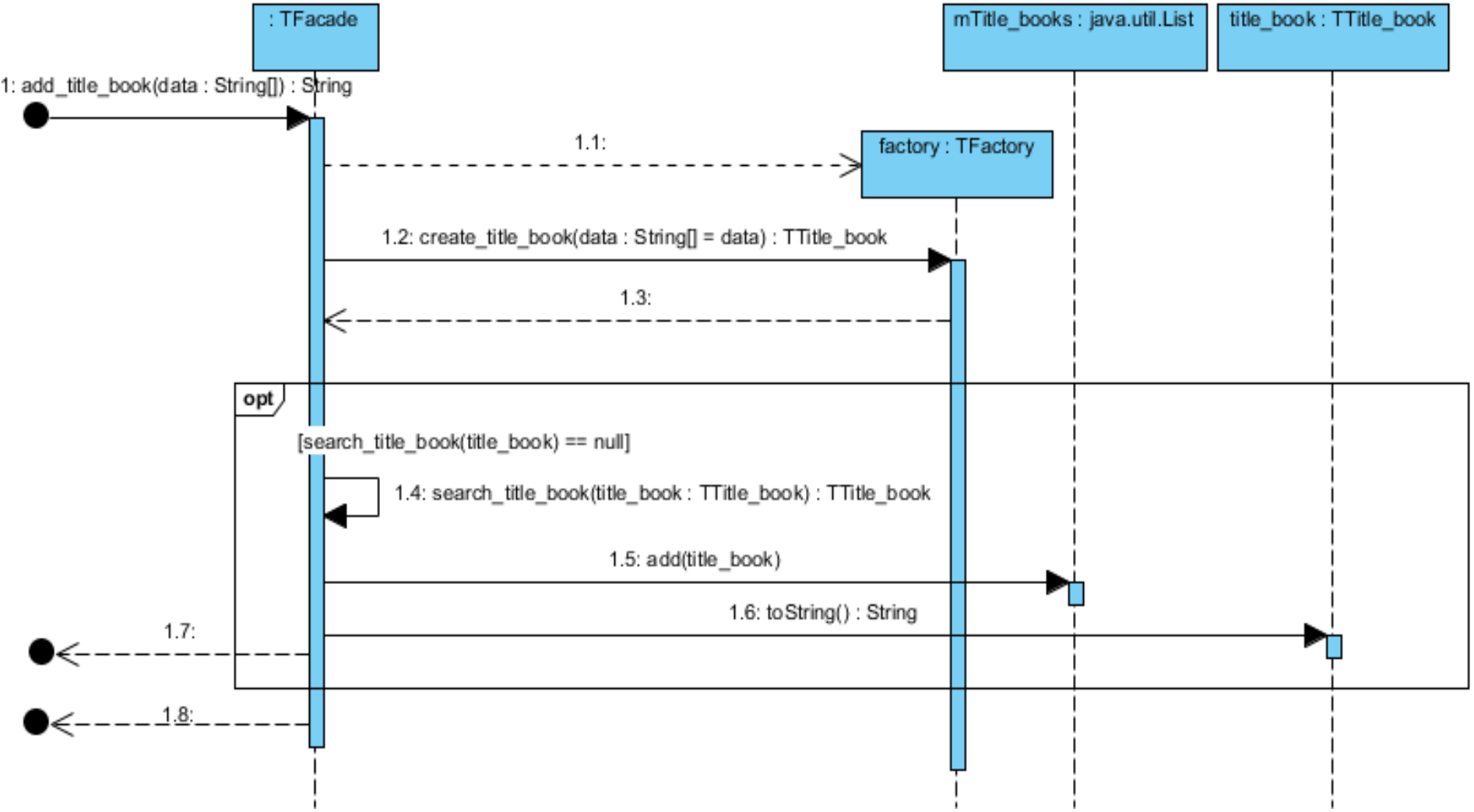
1.6. Możliwość zmiany liczności relacji



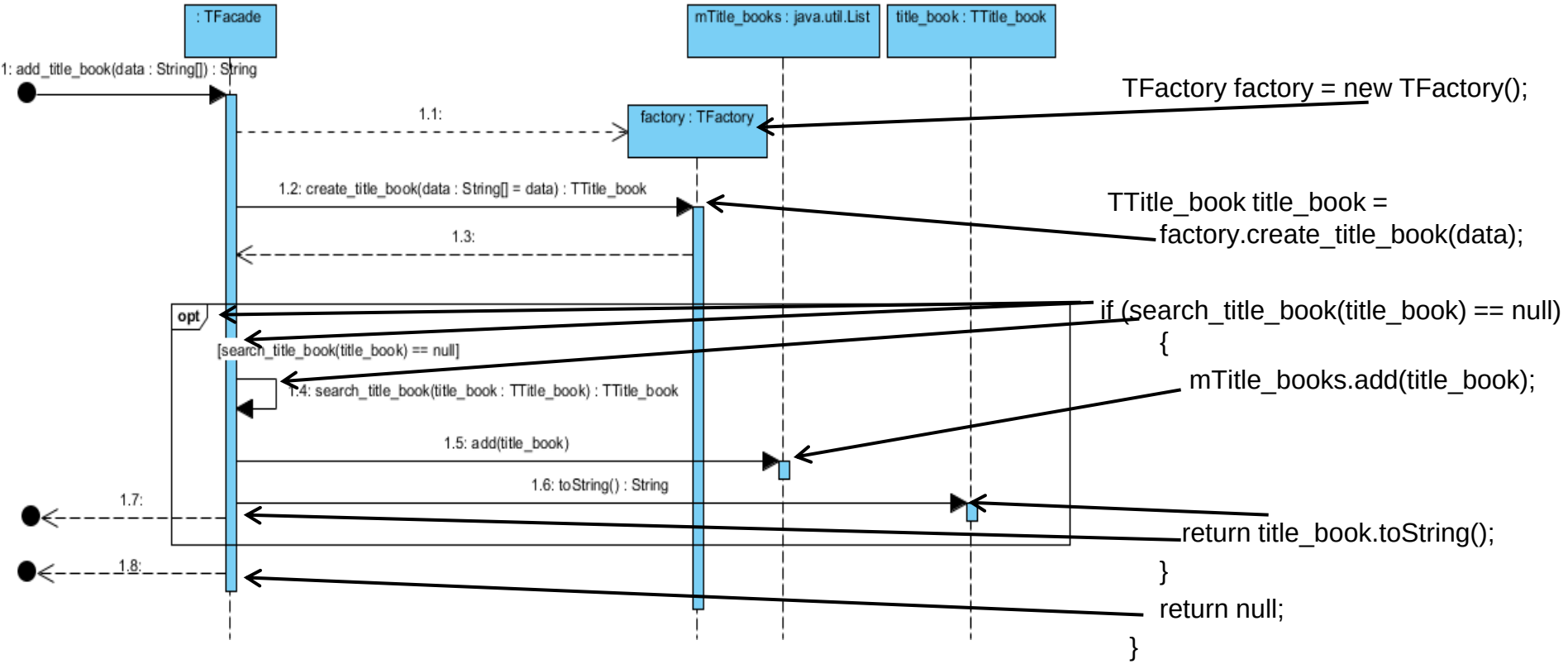
2.Iteracja 1

Dodawanie tytułu książki

2.1. Diagram sekwencji metody klasy TFacade: add_title_book



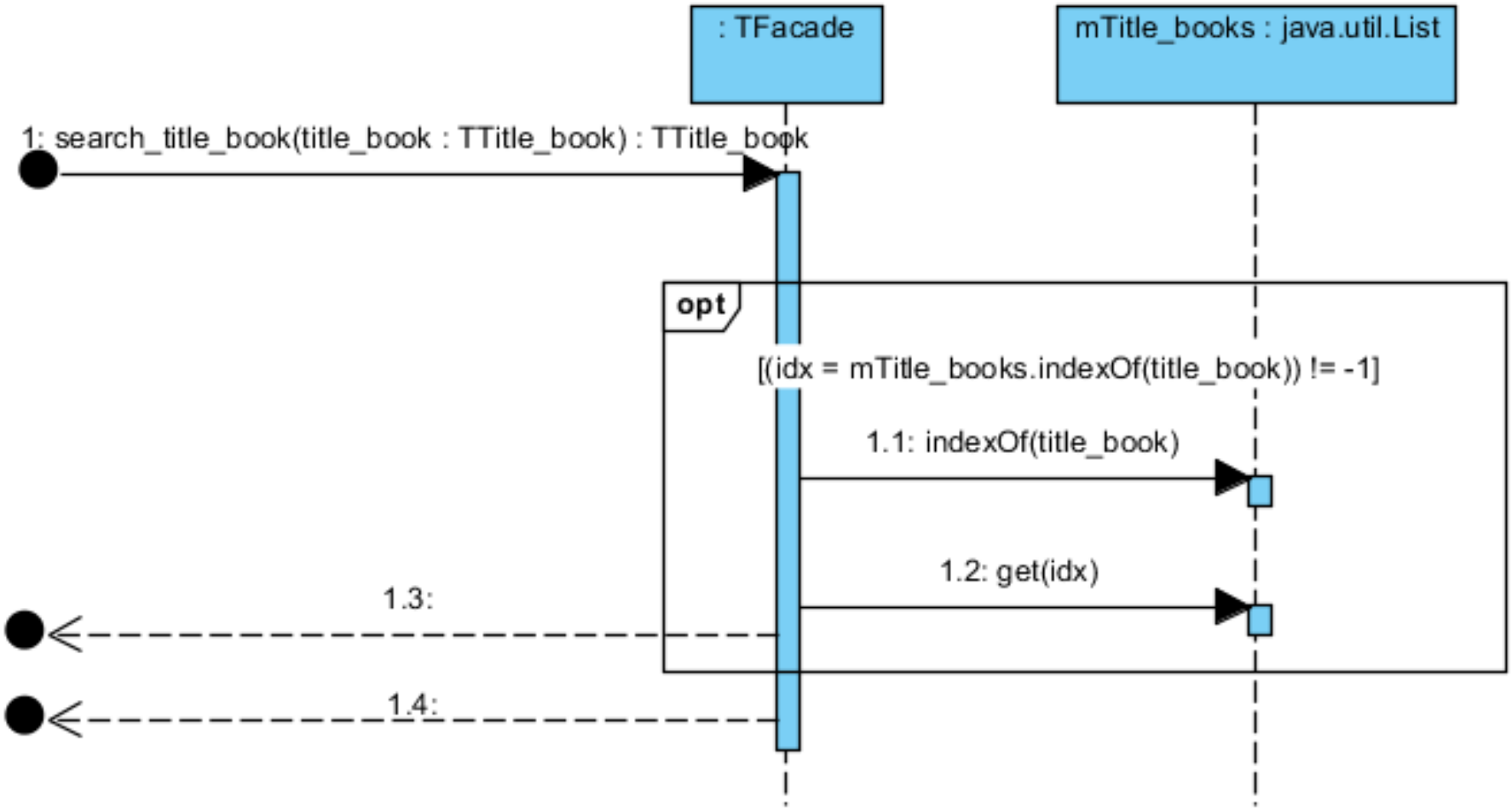
2.2./2.2.1 Kod metody: add_title_book



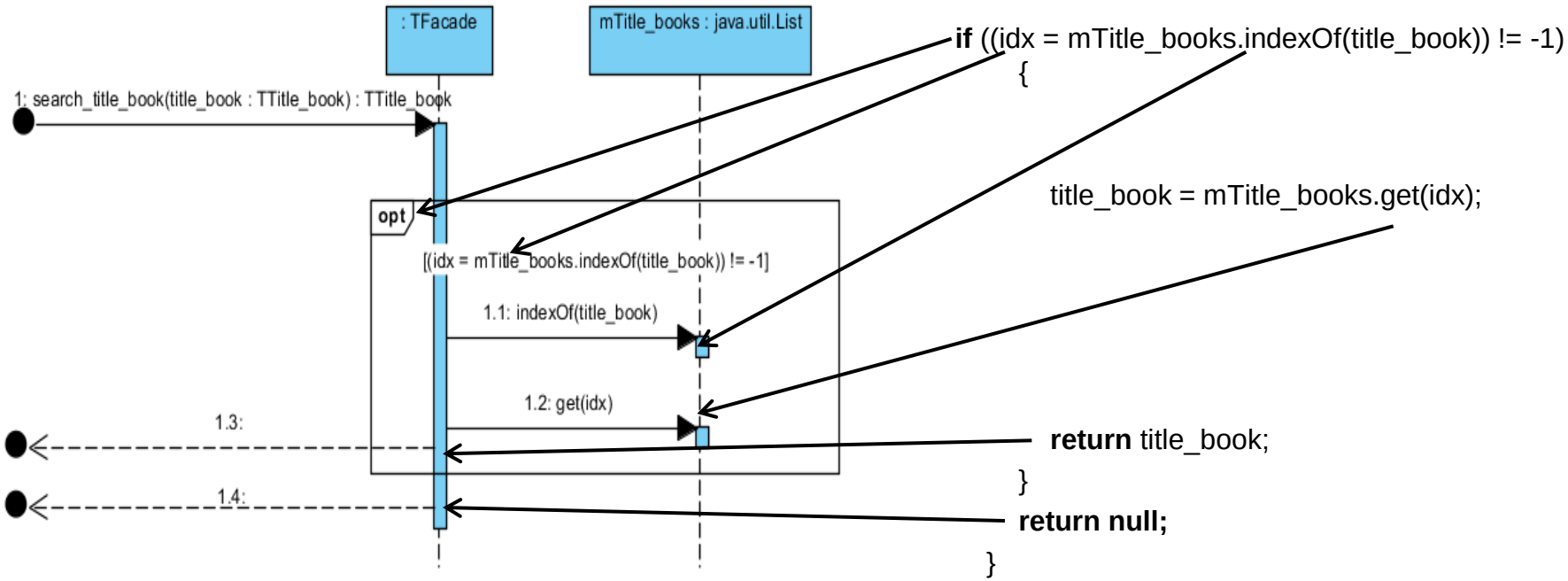
2.2.2. Kod metody: add_title_book

```
public String add_title_book(String data[]) {
        TFactory factory = new TFactory();
        TTitle_book title_book = factory.create_title_book(data);
        if (search_title_book(title_book) == null) {
            mTitle_books.add(title_book);
            return title_book.toString();
        }
        return null;
    }
```

2.3. Diagram sekwencji metody klasy TFacade: search_title_book



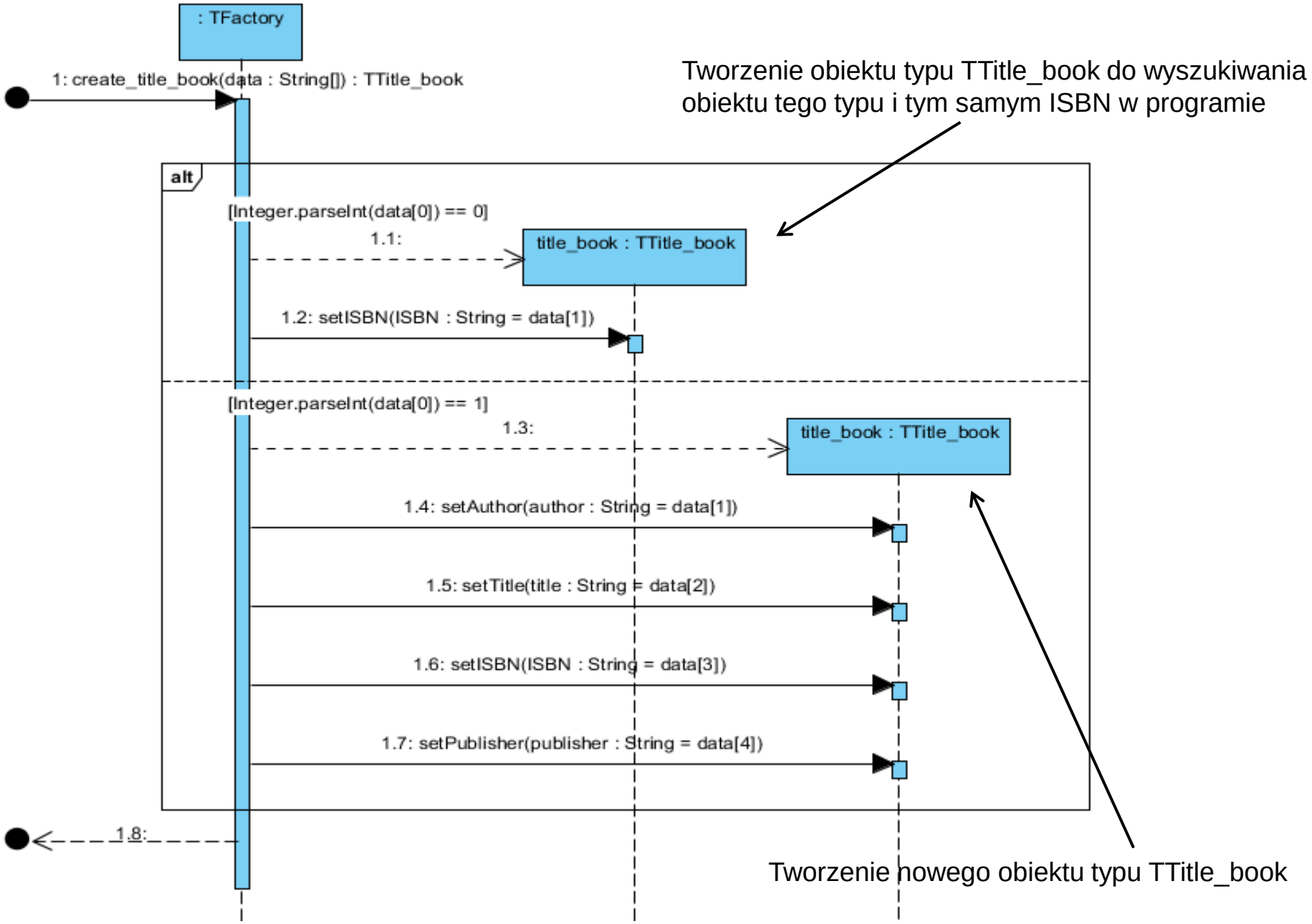
2.4./2.4.1. Kod metody: search_title_book



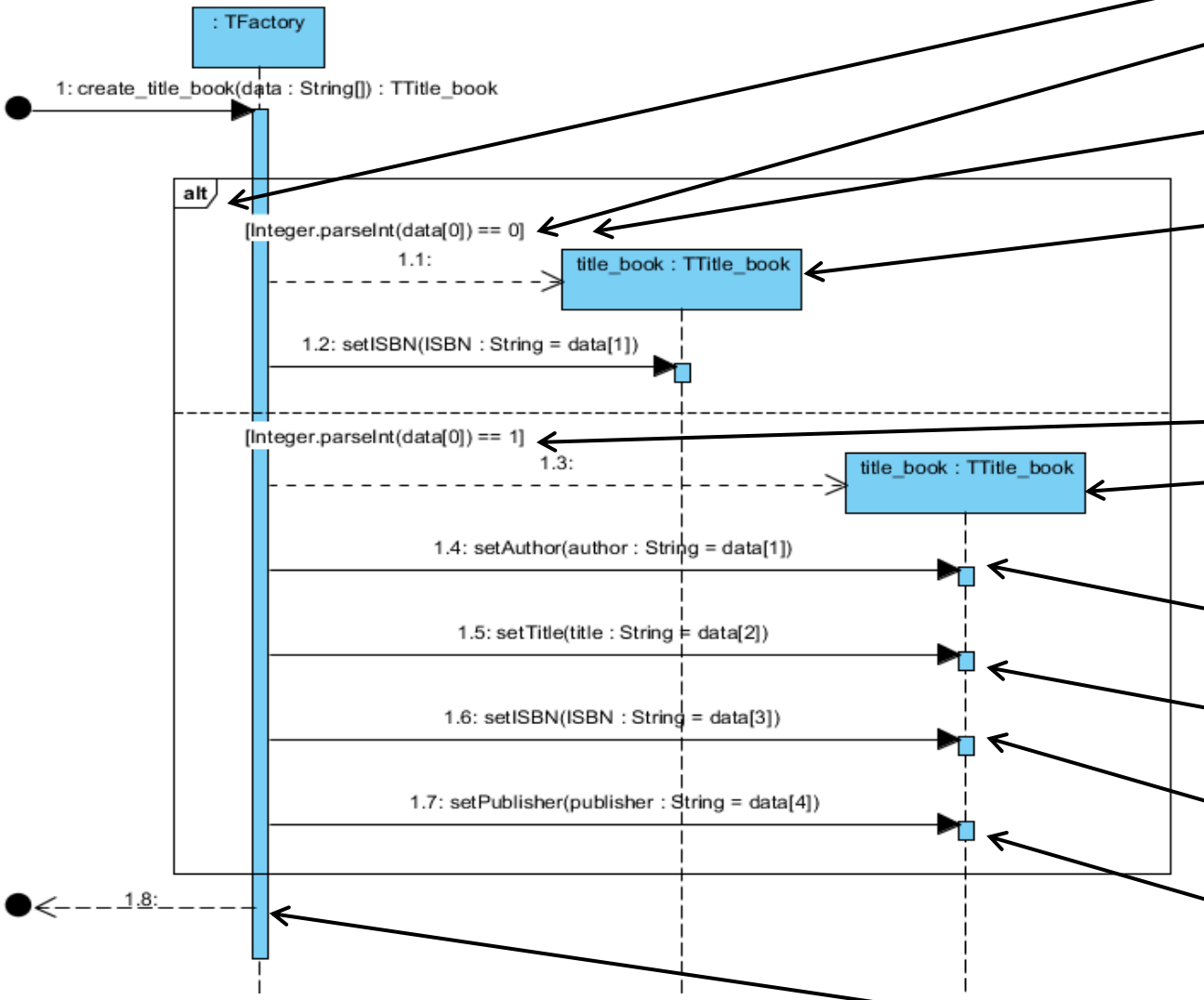
2.4.2. Kod metody: search_title_book

```
public TTitle_book search_title_book(TTitle_book title_book) {
        int idx;
        if ((idx = mTitle_books.indexOf(title_book)) != -1) {
            title_book = mTitle_books.get(idx);
            return title_book;
        }
        return null;
    }
```

2.5. Diagram sekwencji metody klasy TFactory: create_title_book



2.6./2.6.1 Kod metody: create_title_book




```
switch (Integer.parseInt(data[0]))
                //what_title_book_type
{
case 0:
  title_book = new TTitle_book();
//TTitle_book object for searching

   title_book.setISBN(data[1]);
    break;
case 1:
   title_book = new TTitle_book();
//TTitle_book object for persisting

title_book.setAuthor(data[1]);

title_book.setTitle(data[2]);

title_book.setISBN(data[3]);

title_book.setPublisher(data[4]);
break;
  }
return title_book;
```

2.6.2. Kod metody: create_title_book

```
public TTitle_book create_title_book(String data[]) {
        TTitle_book title_book = null;
        switch (Integer.parseInt(data[0])) //what_title_book_type
        {
            case 0:
                title_book = new TTitle_book(); //TTitle_book object for searching
                title_book.setISBN(data[1]);
                break;
            case 1:
                title_book = new TTitle_book(); //TTitle_book object for persisting
                title_book.setAuthor(data[1]);
                title_book.setTitle(data[2]);
                title_book.setISBN(data[3]);
                title_book.setPublisher(data[4]);
                break;
        }
        return title_book;
    }
```

2.7. Przykład kodu sprawdzającego działanie dodawania tytułów książki

```
public static void main(String t[]) {
        TFacade ap = new TFacade();
        String t1[] = {"1", "Author1", "Title1", "ISBN1", "Publisher1"};
        String t2[] = {"1", "Author2", "Title2", "ISBN2", "Publisher2"};
        String t3[] = {"1", "Author3", "Title3", "ISBN3", "Publisher3"};
        ap.add_title_book(t1);
        ap.add_title_book(t2);
        ap.add_title_book(t2);
        ap.add_title_book(t3);
        String lan = ap.getmTitle_books().toString();
        System.out.println(lan);
}
```

Wynik dodawania tytułów książek

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