

Instrukcja 1

Laboratorium 1

**Zapoznanie się z wybranym narzędziem UML –
wprowadzenie do UML**

Cel laboratorium:

Wprowadzenie do UML – wykonanie prostego projektu programu za pomocą wybranych diagramów UML i implementacja projektu programu w języku Java.

1. Opis biznesowy „świata rzeczywistego” zarządzania katalogiem książek wykorzystywanego w bibliotece książek**1.1. 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 zawierającą 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 biblioteki (bibliotekarz) może dodawać nowe tytuły i egzemplarze oraz je przeszukiwać, natomiast klient może jedynie przeszukiwać tytuły i sprawdzać egzemplarze wybranych tytułów.

1.2. Przepisy

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

1.3. Dane techniczne

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

2. Wymagania programu opracowane na podstawie opisu „świata rzeczywistego”**2.1. Wymagania funkcjonalne**

- 2.1.1. Biblioteka wypożycza podane książki i czasopisma osobom zarejestrowanym, o ile je posiada
- 2.1.2. Biblioteka dokonuje zakupu nowych książek, przy czym popularne książki kupuje w kilku egzemplarzach. Biblioteka usuwa wyznaczone książki i czasopisma.
- 2.1.3. Bibliotekarz jest pracownikiem biblioteki, komunikuje się z wypożyczającym. Jego praca jest wspierana za pomocą systemu
- 2.1.4. 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.
- 2.1.5. Biblioteka może łatwo utworzyć, zmienić i usunąć informację o tytułach, wypożyczających, wypożyczeniach i rezerwacjach

2.2. Wymagania нефunkcjonalne programu

- 2.2.1. System powinien pracować w popularnych systemach (LINUX, Windows) i powinien mieć nowoczesny graficzny interfejs użytkownika

2.2.2. System powinien się rozwijać np. wprowadzenie możliwości zawiadamiania rezerwującego książkę o jej dostępności

3. Wykonanie projektu UML np. w środowisku Visual Paradigm

Starting Visual Paradigm

(https://www.visual-paradigm.com/support/documents/vpuserguide/12/3399/6159_startingvisu.html)

Working with projects

1. Creating Project
2. Saving Project
3. Organizing diagrams by Model Structure view
4. Maintaining Backups
5. Manage Project Properties window
6. Switch to Diagram

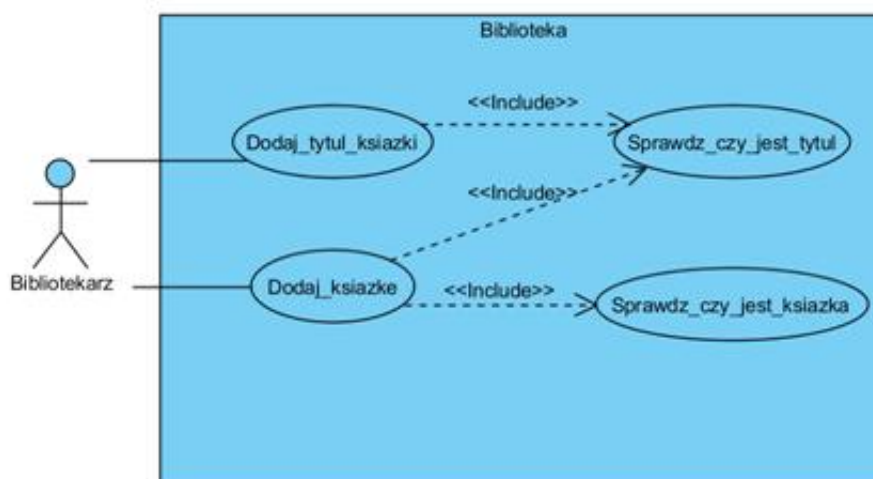
4. Wykonanie diagramu przypadków użycia wg instrukcji na stronie:

Drawing use case diagrams

(https://www.visual-paradigm.com/support/documents/vpuserguide/94/2575/6362_drawinguseca.html)

reprezentującego specyfikację wymagań funkcjonalnych z p. 2

4.1. Diagram przypadków użycia



4.2. Scenariusze przypadków użycia

Nazwa PU: Dodaj_tytul_książki

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 Sprawdź_czy_jest_tytuł**, przekazując ISBN tytułu
3. Jeśli tytuł o podanym ISBN istnieje, należy zakończyć przypadek użycia, w przeciwnym razie należy zapisać dane.

Nazwa PU: Sprawdz_czy_jest_tytul

Cel: Wyszukiwanie tytułu książki o podanym ISBN

Warunki początkowe:

Jest uruchamiany z następujących przypadków użycia: PU Dodaj_tytul_książki_oraz_PU_Dodaj_książke

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 z 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

Nazwa PU: Dodaj_książke

Cel: Dodanie nowej książki

Warunki początkowe:

Uruchomienie programu jako aplikacji www lub aplikacji w architekturze typu "klient-serwer,,

Warunki końcowe:

Wprowadzenia danych książki o unikalnym numerze egzemplarza w ramach książek o tym samym tytule

Scenariusz:

1. Należy podać atrybuty tytułu: ISBN jako obowiązkowa, dlatego tworzony jest tytuł wzorcowy, w którym istotny jest tylko ISBN 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_książka**. Jeśli nie istnieje egzemplarz o danym numerze, należy wstawić ten egzemplarz, w przeciwnym wypadku należy zakończyć PU.

Nazwa PU: Sprawdz_czy_jest_książka

Cel: Wyszukiwanie książki o podanym numerze

Warunki początkowe:

Przypadek użycia jest wywoływany z PU Dodaj_książke

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.

5. Wykonanie diagramu klas, diagramów sekwencji oraz programu w języku Java.

5.1. Identyfikacja klas na podstawie scenariuszy 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

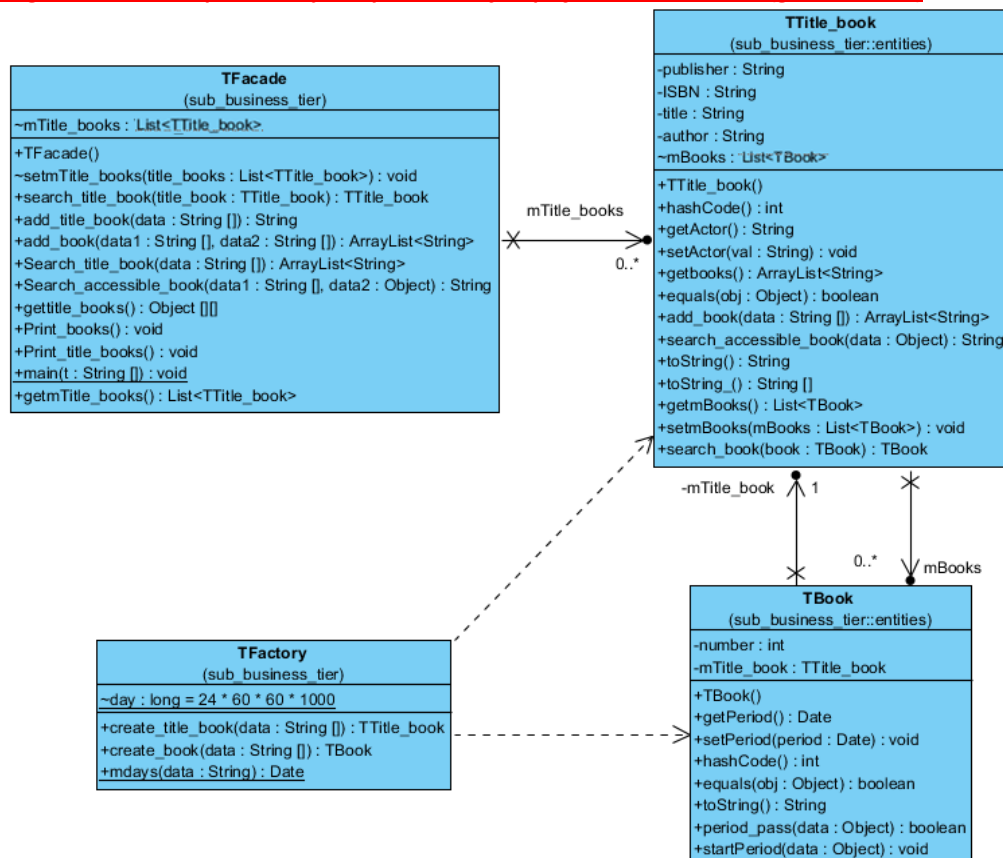
Klasa typu **TFacade** reprezentuje wzorzec **Fasady**, klasa **TFactory** (wzorce strukturalne) reprezentuje wzorzec **Fabryki** (wzorce wytwórcze), klasa **TTitle_book** reprezentuje wzorzec **Pylek** (wzorce strukturalny)

5.2. Wykonanie diagramu klas wg [Drawing class diagrams](https://www.visual-paradigm.com/support/documents/vpuserguide/94/2576/7190_drawingclass.html)

https://www.visual-paradigm.com/support/documents/vpuserguide/94/2576/7190_drawingclass.html reprezentującego strukturę oprogramowania.

Uwaga1: Można pominąć nazwę ścieżki pakietowej podanej w nawiasie, która pojawiła się jako efekt wykonania diagramu klas w wyniku inżynierii odwrotnej (generowanie diagramu z kodu klas), bez utworzenia pakietów lub utworzyć takie zagnieżdżone pakiety o nazwach podanych na diagramie rozmieszczając w nich klasy (ikony pakietów są na palecie diagramu klas) i po zaznaczeniu wszystkich klas i kliknięciu na prawy klawisz myszy wybrać pozycję **Presentation Options**, a następnie pozycję **Show Owner** i kolejną pozycję **Show Fully-Qualified**.

Uwaga2: W Dodatku3 pokazano, jak uzyskać czarną kropkę na końcach strzałki (grocie strzałki)



5.3. Wykonanie projektu typu Java Class Library w środowisku NetBeans, zawierającego kod programu reprezentującego implementację diagramów (należy założyć projekt i zdefiniować 4 klasy wg tab.5.1. Każda z klas powinna posiadać atrybuty podane na diagramie klas. Definicje metod należy wykonać wg tego samego diagramu wpisując nagłówek metody. Jeśli metoda ma wynik różny od **void**, należy wpisać w definicję ciała metody:

throw new UnsupportedOperationException("Not supported yet.");).

Uwaga: narzędzie NetBeans należy zainstalować w wersji All z uwagi na potrzeby wykonania testów funkcjonalnych podczas lab. 14-15.

(<https://netbeans.org/downloads/>).

W tutorialu na stronie:

<https://www.visual-paradigm.com/tutorials/modelinginnetbeans.jsp>

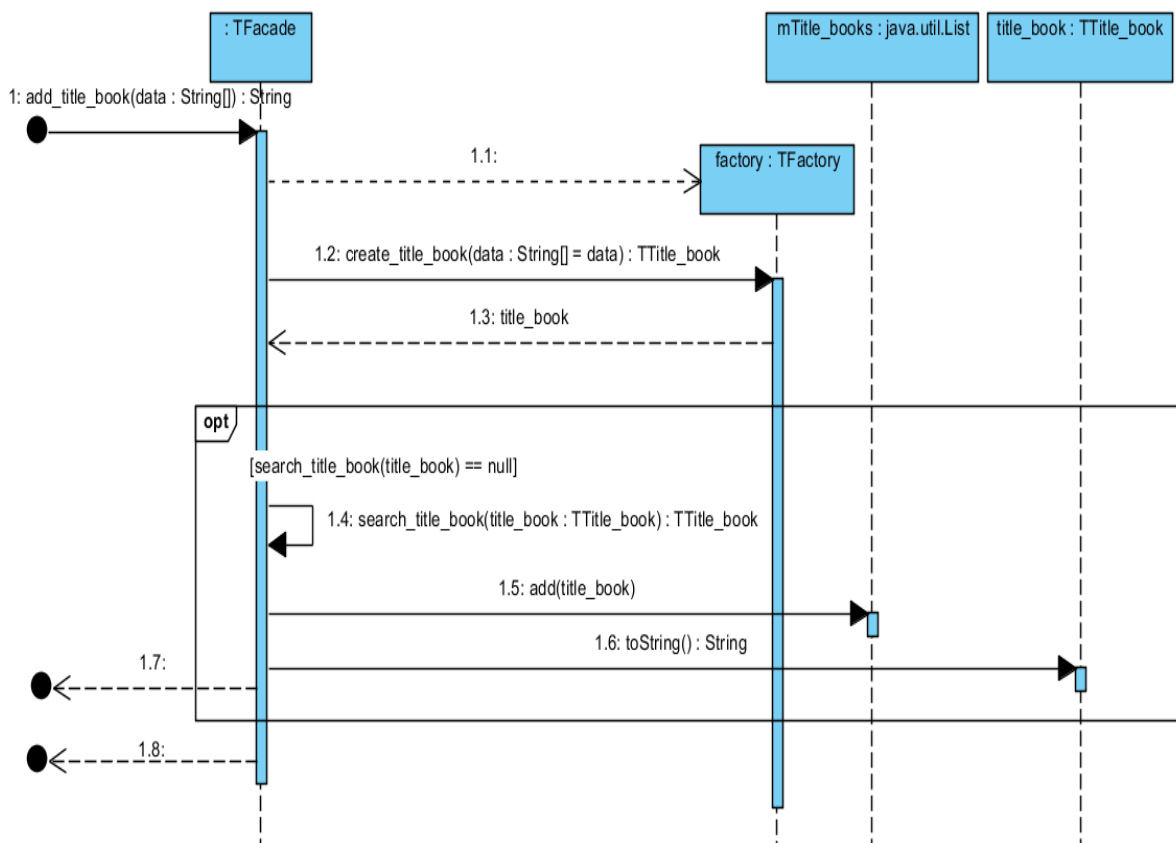
przedstawiono integrację środowiska **NetBeans** i **VisualParadigm**, która ułatwia operacje tworzenia kodu Java i diagramów UML. Jednak wersja narzędzia **Visual Paradigm CE** uniemożliwia wykonanie „inżynierii do przodu” oraz „inżynierii odwrotnej”.

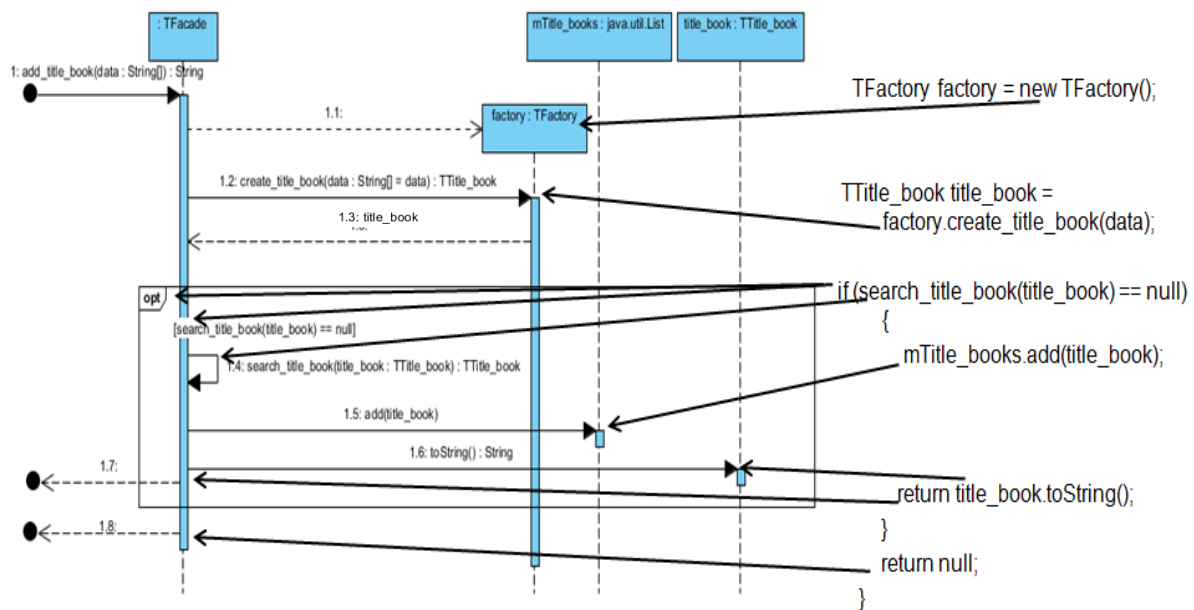
5.4. Wykonanie diagramów sekwencji związanych z projektem przypadku użycia **Dodaj_tytuł_książki – 1-a iteracja rozwoju programu**

Drawing sequence diagrams

(https://www.visual-paradigm.com/support/documents/vpuserguide/94/2577/7025_drawingseque.html)

5.4.1. Diagram sekwencji metody klasy TFacade: add_title_book

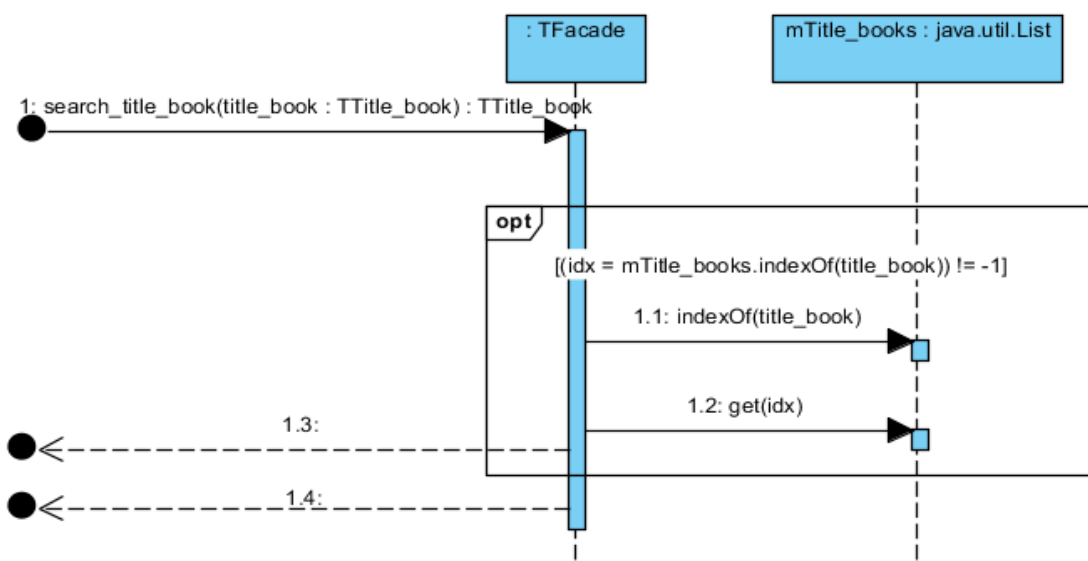


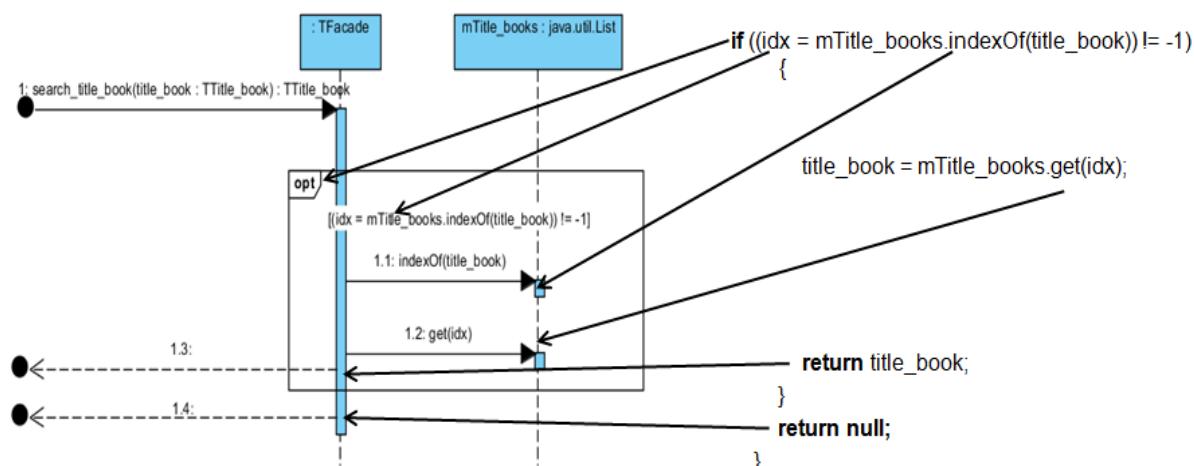
Kod metody: add_title_book

Kod metody: add_title_book w klasie TFacade, reprezentowany przez diagram sekwencji add_title_book, który należy wprowadzić do programu

```

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;
}
  
```

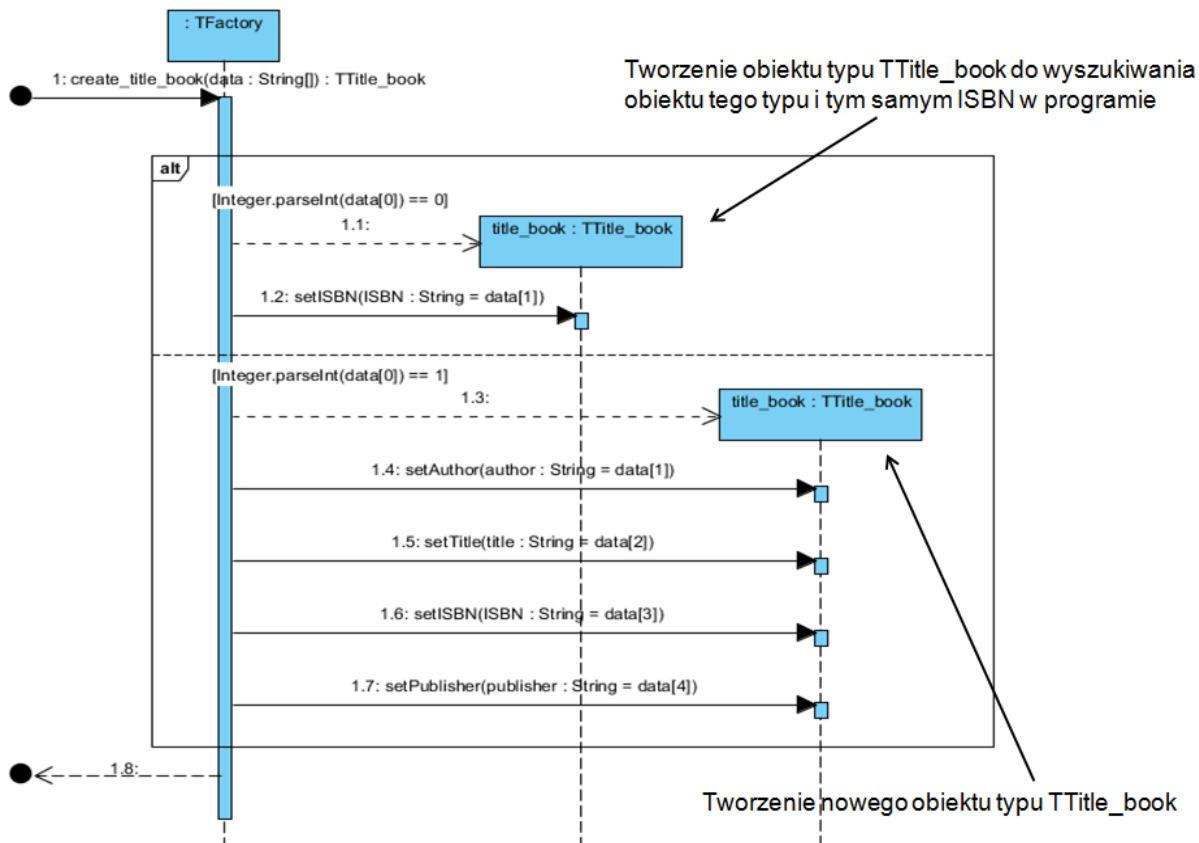
5.4.2. Diagram sekwencji metody search_title_book klasy TFacade

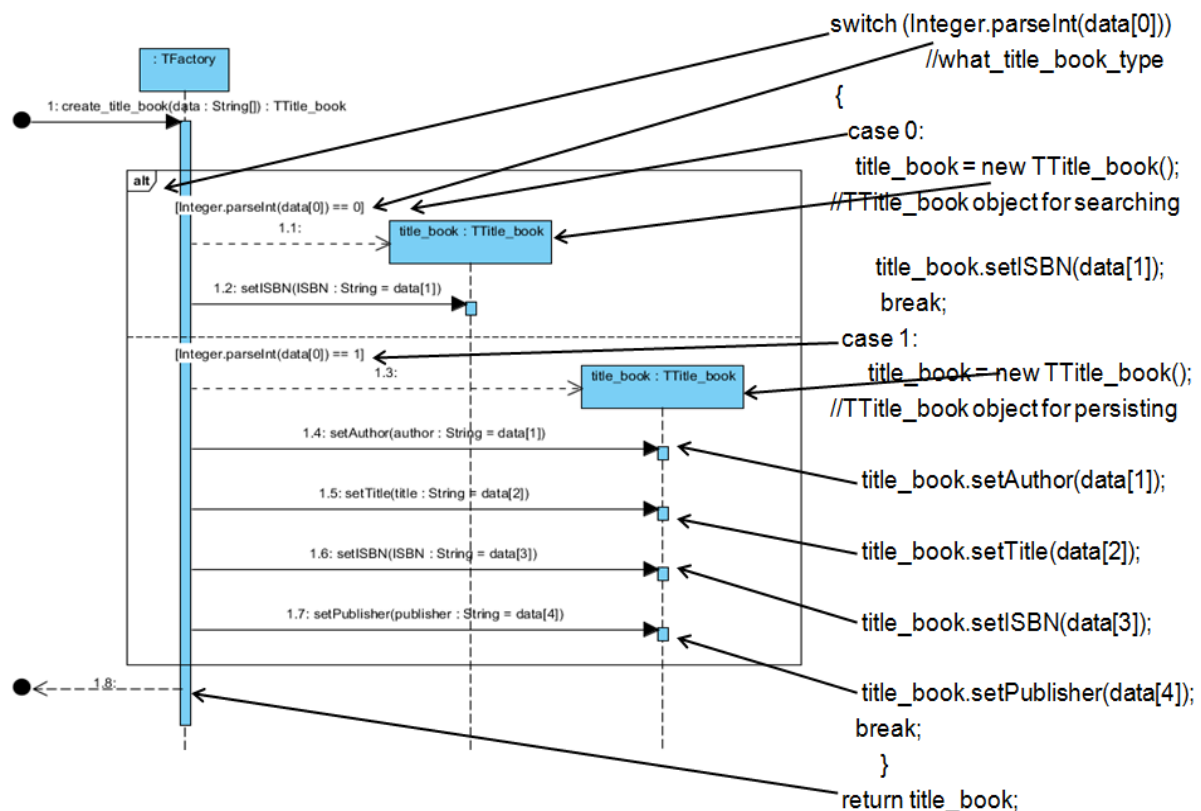
Kod metody search_title_book klasy TFacade:**Kod metody search_title_book klasy TFacade, reprezentowany przez diagram sekwencji search_title_book, który należy wprowadzić do programu**

```

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;
}

```

5.4.3. Diagram sekwencji metody klasy TFactory: create_title_book

Kod metody klasy TFactory: create_title_book**Kod metody klasy TFactory create_title_book, reprezentowany przez diagram sekwencji create_title_book, który należy wprowadzić do programu**

```

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;
}

```

Kod pozostałych metod należy uzupełnić wg informacji podanej w Dodatku.

5.4.4. Przykład kodu sprawdzającego działanie dodawania tytułów książki, który należy wprowadzić do metody main klasy TFacade.

```
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

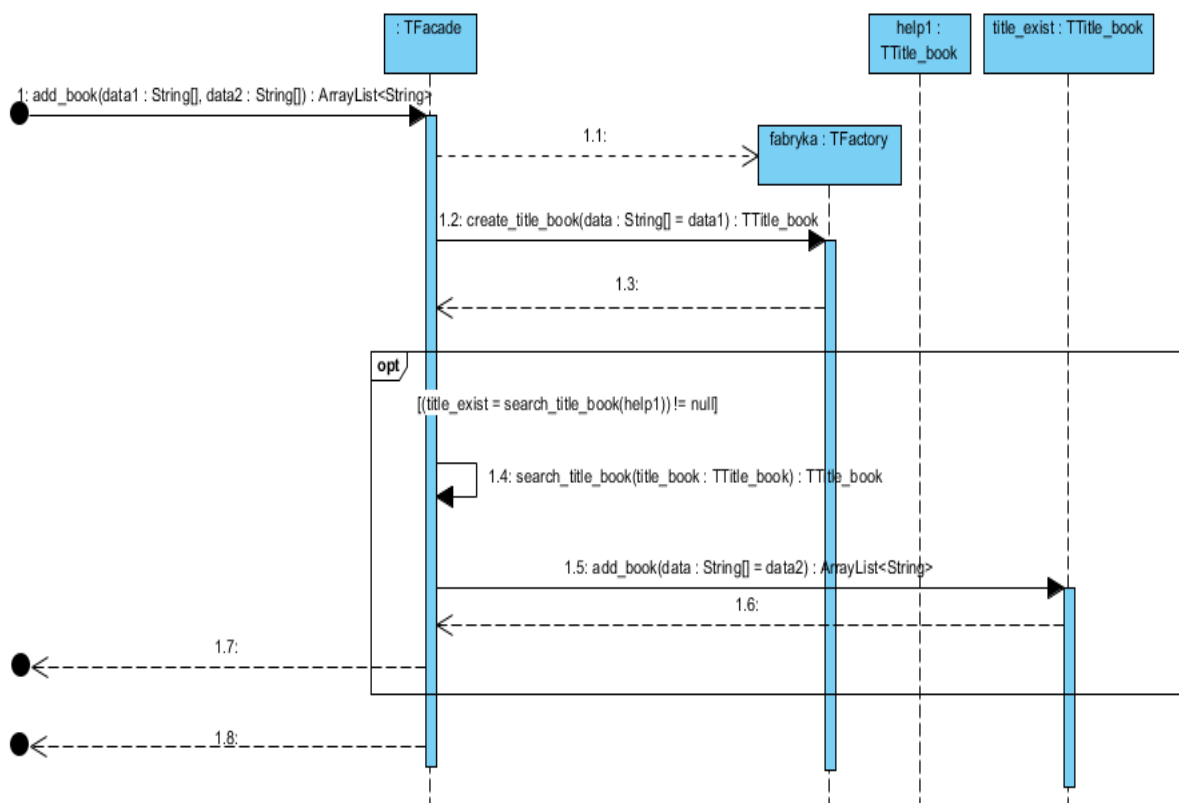
```
[
Title: Title1 Author: Author1 ISBN: ISBN1
Publisher: Publisher1,
Title: Title2 Author: Author2 ISBN: ISBN2
Publisher: Publisher2,
Title: Title3 Author: Author3 ISBN: ISBN3
Publisher: Publisher3]
```

5.5. Wykonanie diagramów sekwencji związanych z projektem przypadku użycia Dodaj_książkę – 2-ga iteracja rozwoju programu

Drawing sequence diagrams

(https://www.visual-paradigm.com/support/documents/vpuserguide/94/2577/7025_drawingseque.html)

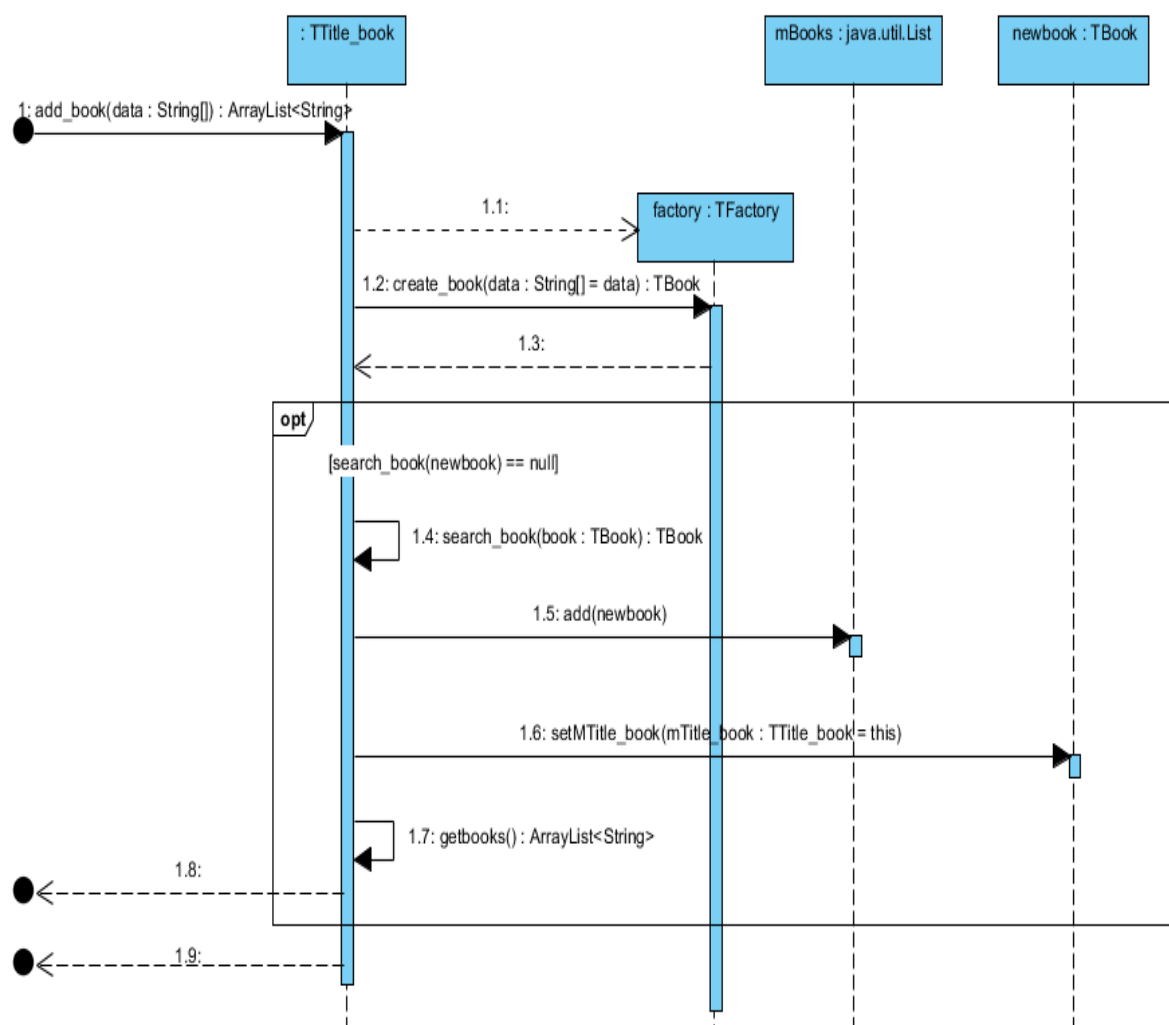
5.5.1. Diagram sekwencji metody add_book klasy TFacade



Kod metody: add_book klasy TFacade, reprezentowany przez diagram sekwencji add_book, który należy wprowadzić do programu

```
public ArrayList<String> add_book(String data1[], String data2[]) {
    TTitle_book help1, title_exist;
    TFactory fabryka = new TFactory();
    help1 = fabryka.create_title_book(data1);
    if ((title_exist = search_title_book(help1)) != null) {
        return title_exist.add_book(data2);
    }
    return null;
}
```

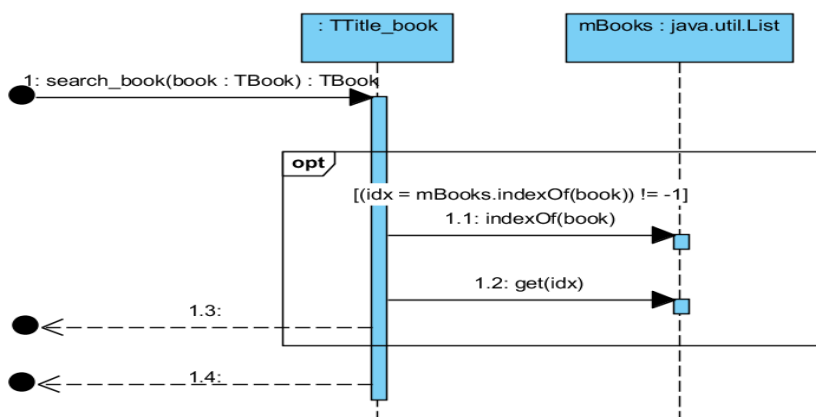
5.5.2. Diagram sekwencji metody add_book klasy TTitle_book



Kod metody: add_book klasy TTitle_book, reprezentowany przez diagram sekwencji add_book, który należy wprowadzić do programu

```
public ArrayList<String> add_book(String data[]) {
    TFactory factory = new TFactory();
    TBook newbook;
    newbook = factory.create_book(data);
    if (search_book(newbook) == null) {
        mBooks.add(newbook);
        newbook.setmTitle_book(this);
        return getbooks();
    }
    return null;
}
```

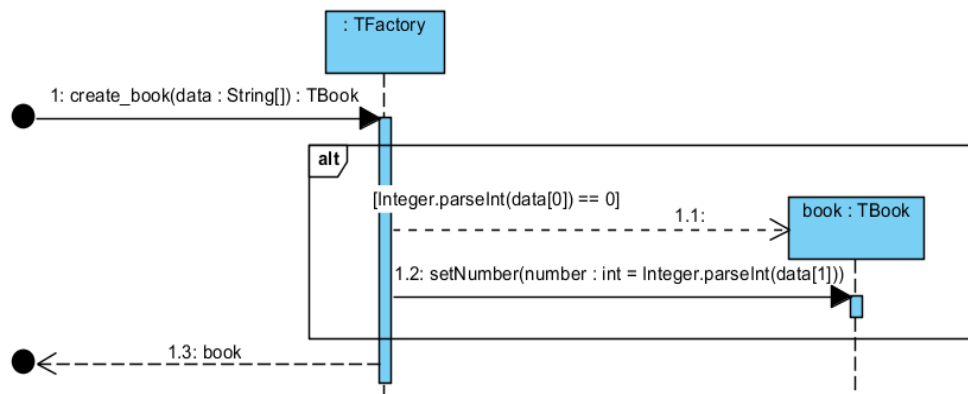
5.5.3. Diagram sekwencji metody search_book klasy TTitle_book



Kod metody: search_book klasy TTitle_book, reprezentowany przez diagram sekwencji search_book, który należy wprowadzić do programu

```
public TBook search_book(TBook book) {
    int idx;
    if ((idx = mBooks.indexOf(book)) != -1) {
        book = (TBook) mBooks.get(idx);
        return book;
    }
    return null;
}
```

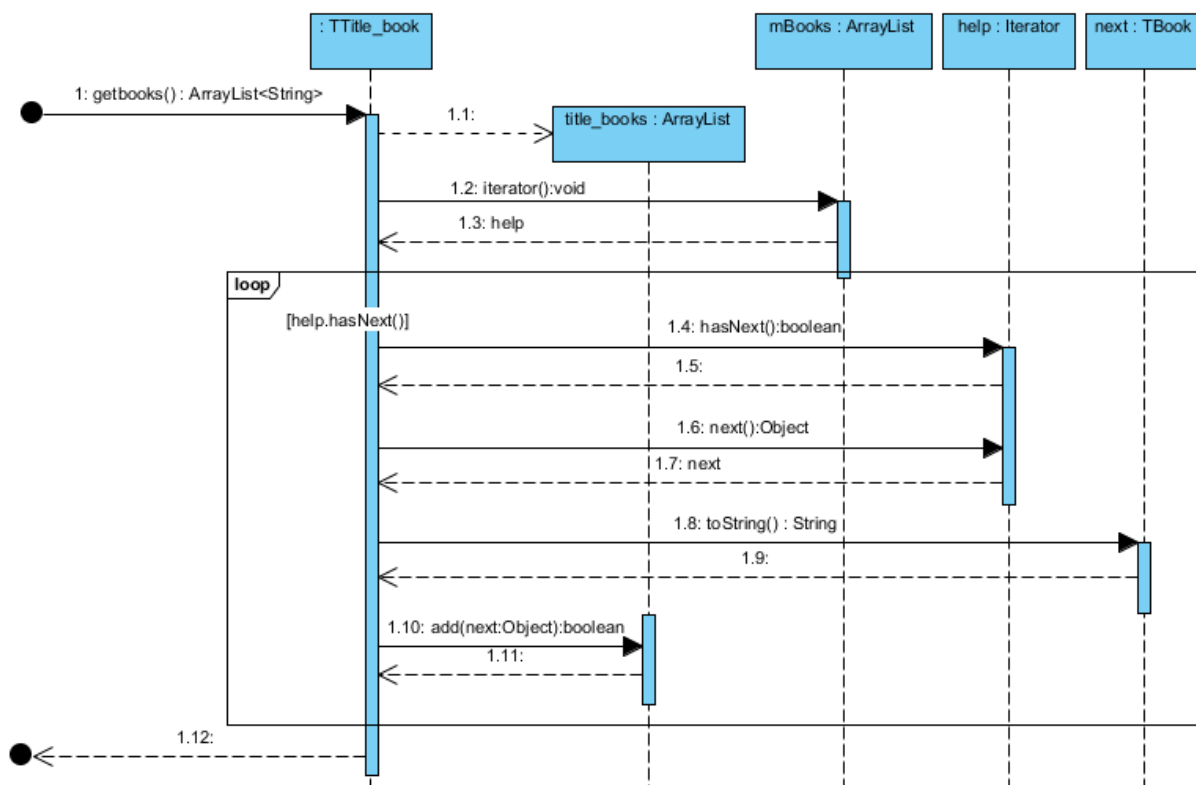
5.5.4. Diagram sekwencji metody create_book klasy TFactory



Kod metody: create_book klasy TFactory, reprezentowany przez diagram sekwencji create_book, który należy wprowadzić do programu

```
public TBook create_book(String data[]) {
    TBook book = null;
    switch (Integer.parseInt(data[0])) //what_book_type
    {
        case 0:
            book = new TBook(); //TBook object for persisting
            book.setNumber(Integer.parseInt(data[1]));
            break;
    }
    return book;
}
```

5.5.5. Diagram sekwencji metody metody ArrayList<String> getbooks() klasy TTitle_book



Kod metody: getbooks() klasy TTitle_book, reprezentowany przez diagram sekwencji getbooks(), który należy wprowadzić do programu

```
public ArrayList<String> getbooks() {
    ArrayList<String> title_books = new ArrayList<>();
    Iterator<TBook> help = mBooks.iterator();    while (help.hasNext())
    {
        TBook next = help.next();
        title_books.add(next.toString());
    }
    return title_books;
}
```

5.5.6. Przykład kodu sprawdzającego działanie dodawania tytułów książki oraz książek, który należy wprowadzić do metody main klasy TFacade

```
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);
    String d1[] = {"0", "ISBN1"};
    String d2[] = {"0", "ISBN2"};
    String d3[] = {"0", "ISBN5"};
    String tr1[] = {"0", "1"};
    String tr2[] = {"0", "2"};

    String d1[] = {"0", "ISBN1"};
    String d2[] = {"0", "ISBN2"};
    String d3[] = {"0", "ISBN5"};
    String tr1[] = {"0", "1"};
    String tr2[] = {"0", "2"};
    ArrayList<String> pom = ap.add_book(d1, tr1);
    if (pom != null) { System.out.print(pom);    }
    pom = ap.add_book(d2, tr1);
    if (pom != null) { System.out.print(pom);    }
    pom = ap.add_book(d2, tr1);
    if (pom != null) { System.out.print(pom);    }
    pom = ap.add_book(d2, tr2);
    if (pom != null) { System.out.print(pom);    }
    pom = ap.add_book(d3, tr2);
    if (pom != null) { System.out.print(pom);    }
}
```

Wynik działania programu: dodawanie tytułów książek i 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]
[
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]
```

Dodatek 1

Kod źródłowy wykonanego programu – w komentarzach umieszczono metody, które powinny być zrealizowane w kolejnej iteracji rozwoju programu. Kod w komentarzach `/**/` służy do projektowania i implementacji kodu w kolejnych iteracjach, które pominięto w instrukcji. Kod oznaczony komentarzem „//2-a iteracja” podczas wykonania pierwszej iteracji powinien być zdefiniowany jako `throw new UnsupportedOperationException("Not supported yet.");`

```
package sub_business_tier.entities;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Objects;

public class TTitle_book {
    private String publisher;           //1-a iteracja
    private String ISBN;                //1-a iteracja
    private String title;               //1-a iteracja
    private String author;              //1-a iteracja
    List<TBook> mBooks;                 //1-a iteracja

    public TTitle_book() {
        mBooks = new ArrayList();      //1-a iteracja
    }
    public String getPublisher() {      //1-a iteracja
        return publisher;
    }
    public void setPublisher(String publisher) { //1-a iteracja
        this.publisher = publisher;
    }
    public String getISBN() {           //1-a iteracja
        return ISBN;
    }
    public void setISBN(String ISBN) {  //1-a iteracja
        this.ISBN = ISBN;
    }
    public String getTitle() {          //1-a iteracja
        return title;
    }
    public void setTitle(String title) { //1-a iteracja
        this.title = title;
    }
    public String getAuthor() {         //1-a iteracja
        return author;
    }
    public void setAuthor(String author) { //1-a iteracja
        this.author = author;
    }
    public List<TBook> getmBooks() {    //1-a iteracja
        return mBooks;
    }
}
```

```

    public void setmBooks(List<TBook> mBooks) { //1-a iteracja
        this.mBooks = mBooks;
    }
    @Override
    public int hashCode() { //1-a iteracja
        int hash = 5;
        hash = 53 * hash + Objects.hashCode(this.ISBN);
        return hash;
    }
    @Override
    public boolean equals(Object obj) { //1-a iteracja
        boolean result = false;
        if (getISBN().equals(((TTitle_book) obj).getISBN()))
            result = true;
        return result;
    }
    @Override
    public String toString() { //1-a iteracja
        String help = "\nTitle: " + getTitle();
        help += " Author: " + getAuthor();
        help += " ISBN: " + getISBN();
        help += " Publisher: " + getPublisher();
        return help;
    }
    public ArrayList<String> add_book(String data[]) { //2-a iteracja
        TFactory factory = new TFactory();
        TBook newbook;
        newbook = factory.create_book(data);
        if (search_book(newbook) == null) {
            mBooks.add(newbook);
            newbook.setmTitle_book(this);
            return getbooks();
        }
        return null;
    }
    public TBook search_book(TBook book) { //2-a iteracja
        int idx;
        if ((idx = mBooks.indexOf(book)) != -1) {
            book = (TBook) mBooks.get(idx);
            return book;
        }
        return null;
    }
    public ArrayList<String> getbooks() { //2-a iteracja
        ArrayList<String> title_books = new ArrayList<>();
        Iterator<TBook> help = mBooks.iterator();
        while (help.hasNext()) {
            TBook next = help.next();
            title_books.add(next.toString());
        }
        return title_books;
    }

```



```

/* public String[] toString_() {
    throw new UnsupportedOperationException("Not supported yet."); } //kolejne iteracje
public String search_accessible_book(Object data) {
    throw new UnsupportedOperationException("Not supported yet."); }
public String getActor() {
    throw new UnsupportedOperationException("Not supported yet."); }
public void setActor(String val) { }
*/
}

package sub_business_tier.entities;

public class TBook {
    private int number; //1-a iteracja
    private TTitle_book mTitle_book; //1-a iteracja
    public TBook() { //2-a iteracja
    }
    public int getNumber() { //2-a iteracja
        return number;
    }
    public void setNumber(int number) { //2-a iteracja
        this.number = number;
    }
    public TTitle_book getmTitle_book() { //2-a iteracja
        return mTitle_book;
    }
    public void setmTitle_book(TTitle_book mTitle_book) { //2-a iteracja
        this.mTitle_book = mTitle_book;
    }
    @Override
    public int hashCode() { //2-a iteracja
        int hash = 0;
        hash += (number != 0 ? number : 0);
        return hash;
    }
    @Override
    public boolean equals(Object obj) { //2-a iteracja
        return number == ((TBook) obj).getNumber();
    }
    @Override
    public String toString() //2-a iteracja
    {
        String help = mTitle_book.toString();
        help += " Number: " + getNumber();
        return help;
    }

    /* public Date getPeriod() { //kolejne iteracje
        throw new UnsupportedOperationException("Not supported yet."); }
    public void setPeriod(Date period) { }
    */
}

```

```
package sub_business_tier;

public class TFactory {

    // static final long day = 24 * 60 * 60 * 1000;           //kolejne iteracje

    public TTitle_book create_title_book(String data[]) {      //1-a iteracja
        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;
    }

    public TBook create_book(String data[]) {                  //2-a iteracja
        TBook book = null;
        switch (Integer.parseInt(data[0])) //what_book_type
        {
            case 0:
                book = new TBook(); //TBook object for persisting
                book.setNumber(Integer.parseInt(data[1]));
                break;
        }
        return book;
    }

    /* static public Date mdays(String data) {                //kolejne iteracje
        throw new UnsupportedOperationException("Not supported yet.");
    }*/
}
```

```

package sub_business_tier;

import java.util.ArrayList;
import java.util.List;

public class TFacade {
    List<TTitle_book> mTitle_books;

    public TFacade() { //1-a iteracja
        mTitle_books = new ArrayList<>();
    }
    public List<TTitle_book> getmTitle_books() { //1-a iteracja
        return mTitle_books;
    }
    void setmTitle_books(List<TTitle_book> title_books) { //1-a iteracja
        mTitle_books = title_books;
    }
    public TTitle_book search_title_book(TTitle_book title_book) { //1-a iteracja
        int idx;
        if ((idx = mTitle_books.indexOf(title_book)) != -1) {
            title_book = mTitle_books.get(idx);
            return title_book;
        }
        return null;
    }
    public String add_title_book(String data[]) { //1-a iteracja
        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;
    }
    public ArrayList<String> add_book(String data1[], String data2[]) { //2-a iteracja
        TTitle_book help1, title_exist;
        TFactory fabryka = new TFactory();
        help1 = fabryka.create_title_book(data1);
        if ((title_exist = search_title_book(help1)) != null) {
            return title_exist.add_book(data2);
        }
        return null;
    }
    /* public ArrayList<String> gettitle_books() { //kolejne iteracje
        throw new UnsupportedOperationException("Not supported yet."); }
    public TTitle_book search_title_book(TTitle_book title_book) {
        throw new UnsupportedOperationException("Not supported yet."); }
    public TTitle_book add_title_book(String[] data) {
        throw new UnsupportedOperationException("Not supported yet."); }
    public TTitle_book add_book(String[] data1, String[] data2) {
        throw new UnsupportedOperationException("Not supported yet."); }

```

```

public TTitle_book Search_title_book(String[] data) {
    throw new UnsupportedOperationException("Not supported yet."); }
public TBook Search_accessible_book(String[] data1, Object data2) {
    throw new UnsupportedOperationException("Not supported yet."); }
public void Print_books() {
}
public void Print_title_books() {
}*/

public static void main(String t[]) {
    TFacade ap = new TFacade();
    String t1[] = {"1", "Author1", "Title1", "ISBN1", "Publisher1"}; //1-a iteracja
    String t2[] = {"1", "Author2", "Title2", "ISBN2", "Publisher2"}; //1-a iteracja
    String t3[] = {"1", "Author3", "Title3", "ISBN3", "Publisher3"}; //1-a iteracja
    ap.add_title_book(t1); //1-a iteracja
    ap.add_title_book(t2); //1-a iteracja
    ap.add_title_book(t2); //1-a iteracja
    ap.add_title_book(t3); //1-a iteracja
    String lan = ap.getmTitle_books().toString(); //1-a iteracja
    System.out.println(lan); //1-a iteracja
    String d1[] = {"0", "ISBN1"}; //2-a iteracja
    String d2[] = {"0", "ISBN2"}; //2-a iteracja
    String d3[] = {"0", "ISBN5"}; //2-a iteracja
    String tr1[] = {"0", "1"}; //2-a iteracja
    String tr2[] = {"0", "2"}; //2-a iteracja
    ArrayList<String> pom = ap.add_book(d1, tr1); //2-a iteracja
    if (pom != null) { //2-a iteracja
        System.out.print(pom); } //2-a iteracja
    pom = ap.add_book(d2, tr1); //2-a iteracja
    if (pom != null) { //2-a iteracja
        System.out.print(pom); } //2-a iteracja
    pom = ap.add_book(d2, tr1); //2-a iteracja
    if (pom != null) { //2-a iteracja
        System.out.print(pom); } //2-a iteracja
    pom = ap.add_book(d2, tr2); //2-a iteracja
    if (pom != null) { //2-a iteracja
        System.out.print(pom); } //2-a iteracja
    pom = ap.add_book(d3, tr2); //2-a iteracja
    if (pom != null) { //2-a iteracja
        System.out.print(pom); } //2-a iteracja
    System.out.println(); //2-a iteracja
    }
}

```

Dodatek 2

1. Wybór tematu aplikacji do prac podczas laboratoriów 2-13, dotyczących modelowania, implementacji i testowania jednostkowego oraz akceptacyjnego:

- 4.3. Program obsługujący konto bankowe
- 4.4. Program realizujący edytor tekstu
- 4.5. Program obsługujący system alarmowy
- 4.6. Program obsługujący Dział Ewidencji Ludności
- 4.7. Program obsługujący system informacyjny linii autobusowych
- 4.8. Program obsługujący wypożyczalnię kaset video
- 4.9. Program obsługujący wypożyczalnię sprzętu turystycznego
- 4.10. Program obsługujący zakład transportowy
- 4.11. Program obsługujący zapisy na zajęcia na wyższych uczelniach
- 4.12. Program wystawiający rachunki
- 4.13. Program obsługujący zakład naprawy sprzętu gospodarstwa domowego
- 4.14. Program obsługujący biuro informacji turystycznej
- 4.15. Program obsługujący system rezerwacji miejsc w hotelach
- 4.16. Program obsługujący pasażerskie linie lotnicze

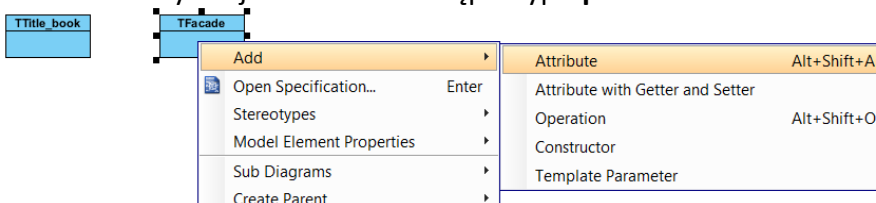
Uwagi:

- Użyto celowo słowo **program** określające temat projektu, ponieważ diagramy UML, stosowane podczas projektowania mają służyć do projektowania kodu programu napisanego w języku obiektowym.
- Można zaproponować **własny temat aplikacji** do realizacji.

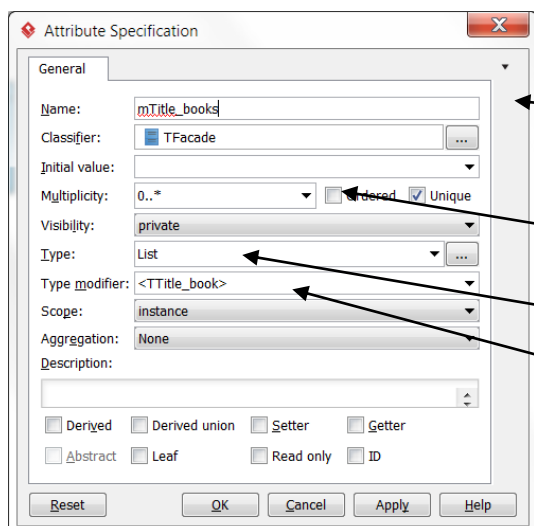
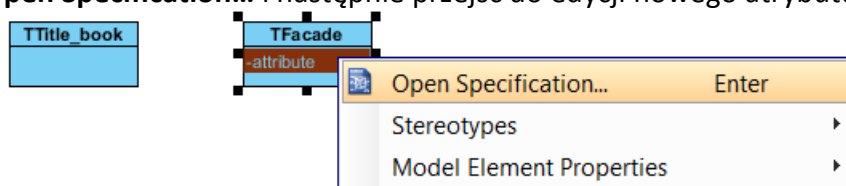
Dodatek 3

Przykład wstawiania powiązania typu **Association** pomiędzy dwiema klasami:

1. Należy do diagramu klas wstawić dwie klasy: **TFacade** i **TTitle_book**
2. Należy dodać atrybut do klasy np. **TFacade** – po wyborze pozycji **Attribute** pojawi się atrybut o nazwie domyślnej **attribute** i dostępie typu **private**.



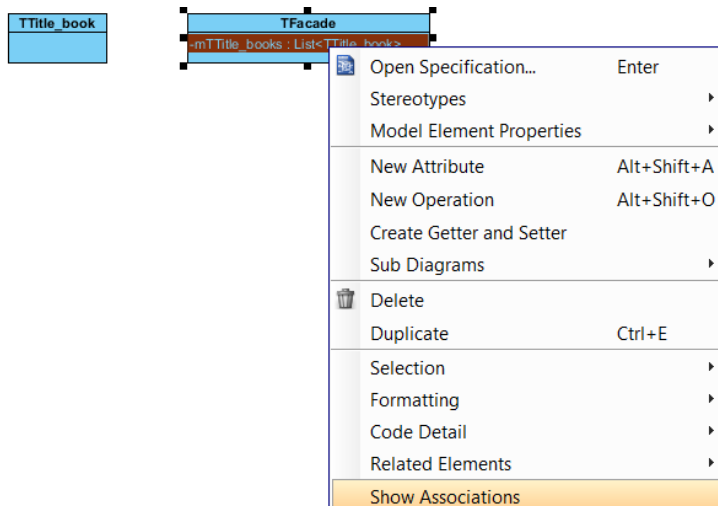
3. Po zaznaczeniu dodanego atrybutu należy kliknąć na prawy klawisz myszy i wybrać pozycję **Open Specification...** i następnie przejść do edycji nowego atrybutu.



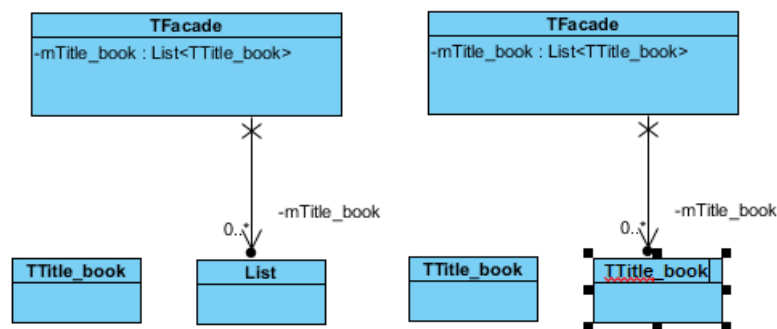
Należy wypełnić pola:

- Name:** tutaj należy wpisać nazwę atrybutu reprezentującego połączenie z wieloma obiektami typu TTitle_book
- Multiplicity:** tutaj należy wybrać pozycję z listy 0..*
- Type** – tutaj należy wpisać np. nazwę pojemnika List
- Type modifier** tutaj należy wpisać do wybranej pozycji **<Unspecified>** nazwę klasy powiązanej w relacji 0..*, czyli TTitle_book

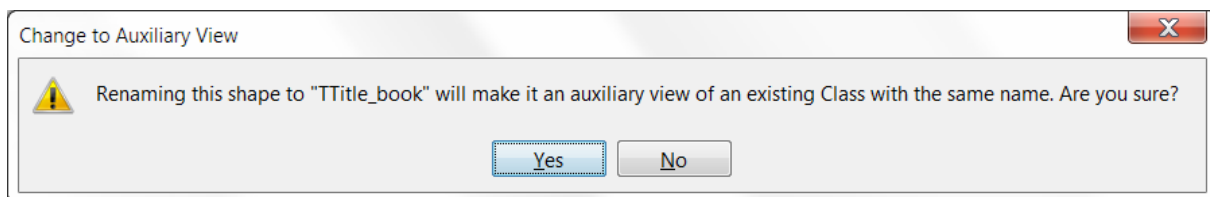
4. Po zaznaczeniu dodanego atrybutu należy kliknąć na prawy klawisz myszy i wybrać pozycję **Show Associations**.



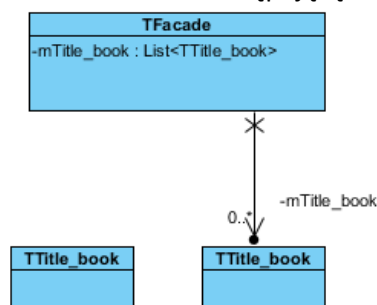
5. Na diagramie klas pojawi się klasa o nazwie **List**, której nazwę należy zmienić na **TTitle_book**.



W momencie zatwierdzania nazwy pojawi się komunikat (poniżej) – w przypadku istniejącej już klasy:



Po zatwierdzeniu komunikatu diagram klas zawiera następującą konstrukcję:



Można wtedy usunąć klasę TTitle_book z widoku (ponieważ ta sama klasa ma dwa „widoki”).

