

Instrukcja 2

Laboratorium z Podstaw Inżynierii Oprogramowania

Opis biznesowy „świata rzeczywistego”

Wymagania funkcjonalne i нефunkcjonalne aplikacji

Diagram przypadków życia

Diagramy klas i sekwencji:

Relacja 1 do 1..0– instrukcja z lab1

Relacja 1 do 0..* - nowy materiał

Cele laboratorium 2

Należy:

- wybrać projekt z podanej listy dostępnej za pomocą linku podanego w w laboratorium 1
- sformułować wymagania funkcjonalne i нефункционалне 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 powiązań 1:1 (jeden do jeden) oraz 1:* (jeden do wiele) między klasami, użytymi do realizacji 1-go etapu programu „Projekt_lab1” – należy wykorzystać te dwa typy powiązań w wykonywanym 1-ym etapie projektu i jego implementacji.

Zawartość 1 części projektu

1. Opis „świata rzeczywistego”
2. Wymagania funkcjonalne i нефункционаłne 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.
6. Program typu Java Application wg slajdów 41- 49.

Java

język programowania

- obiektowo zorientowany
- wysokiego poziomu

platforma Javy

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

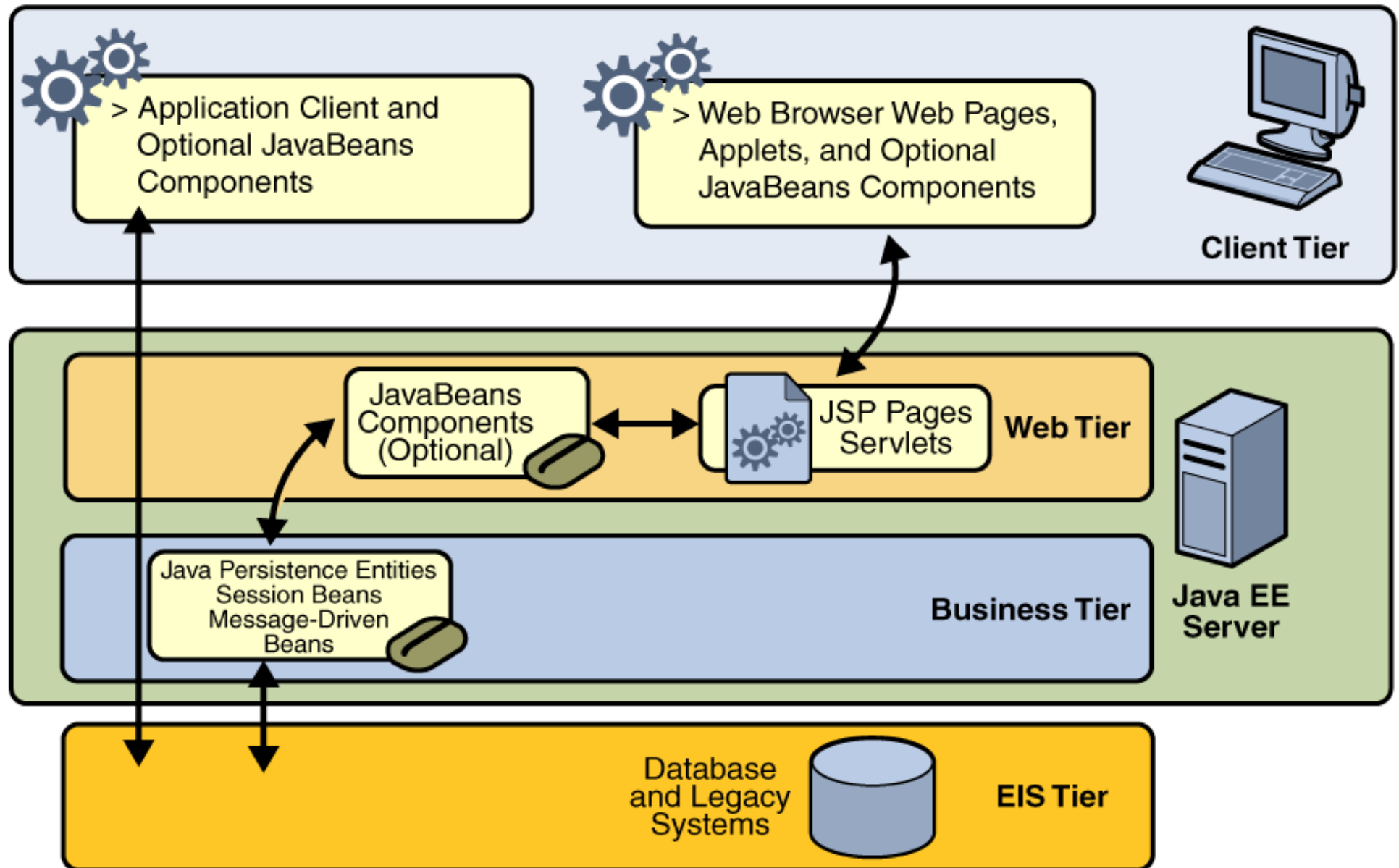
Rezultat

- 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)

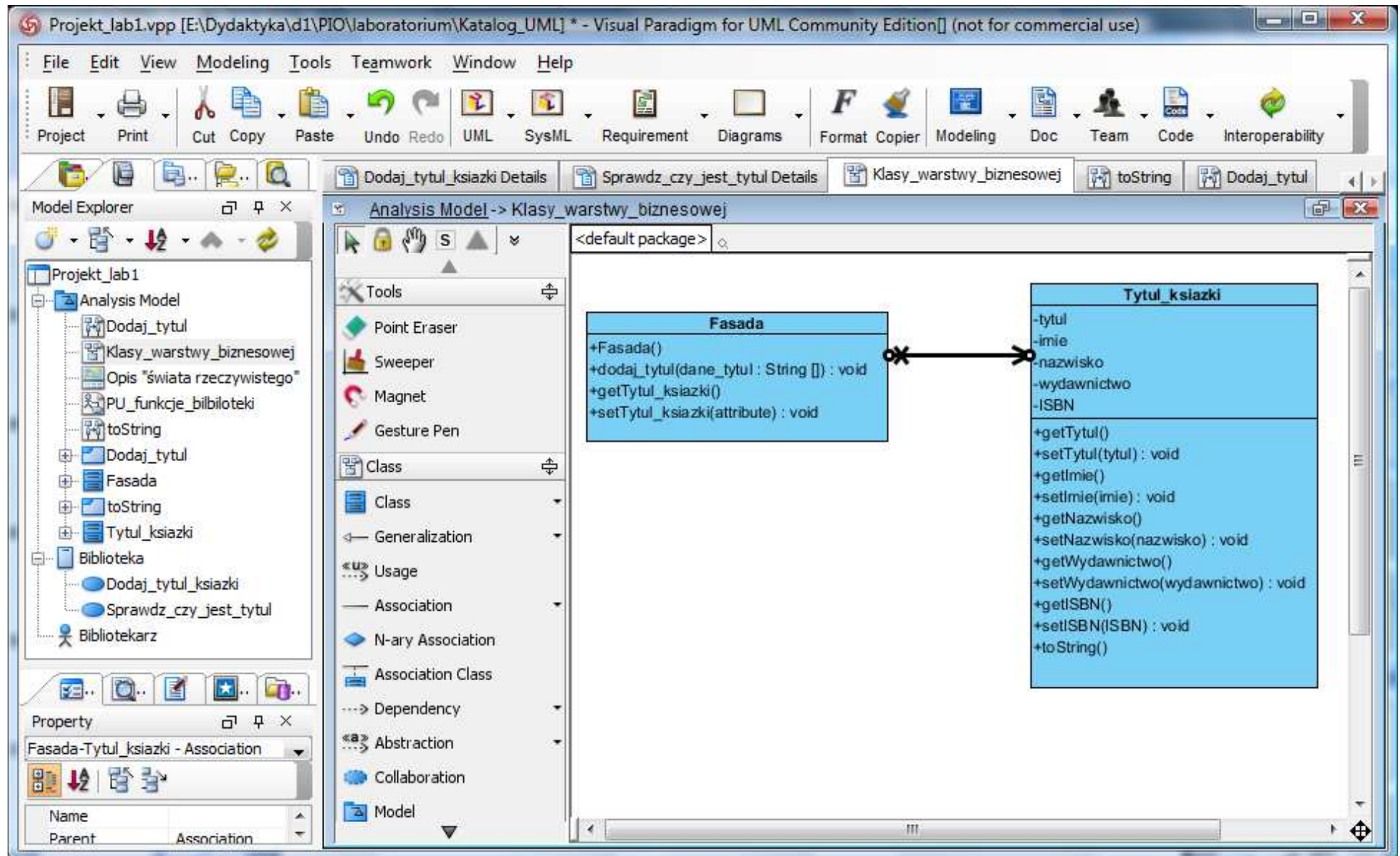


Materiał pomocniczy z lab2

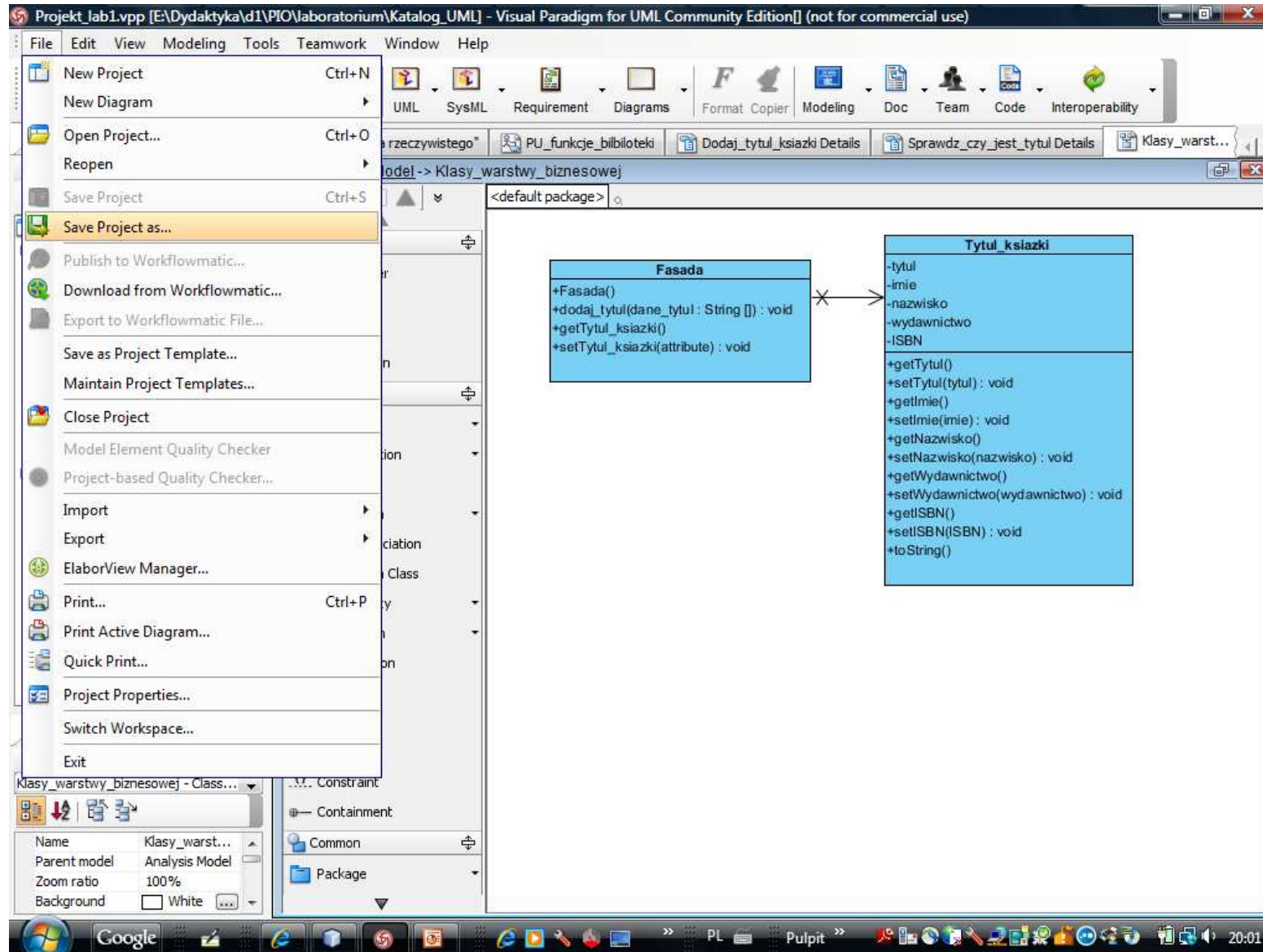
Relacja jeden do wiele

**Realizacja głównego celu
przypadku użycia „Dodaj_tytul”
– wstawianie nowych tytułów bez
powtórzeń**

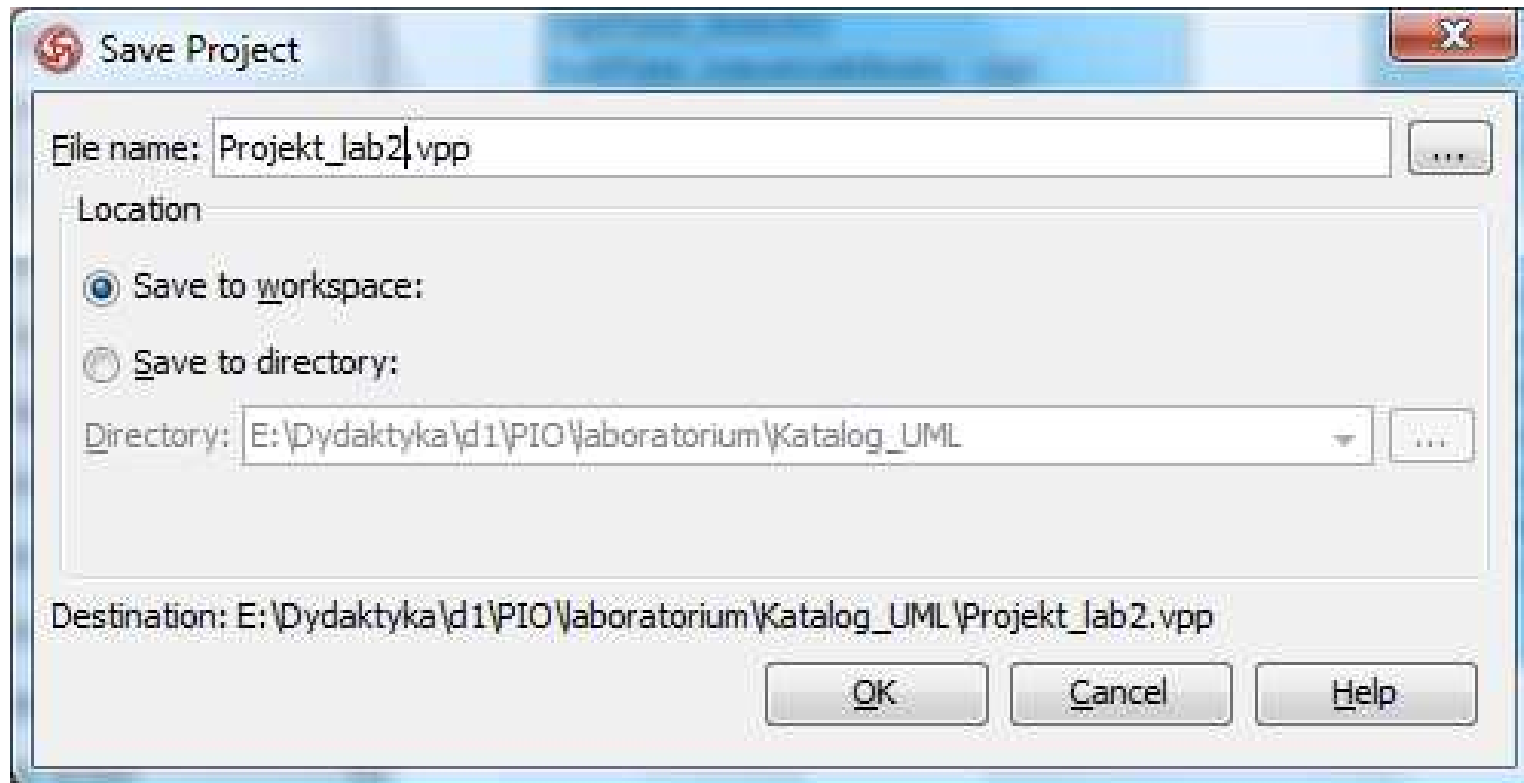
1.1. Należy kontynuować projekt z lab1



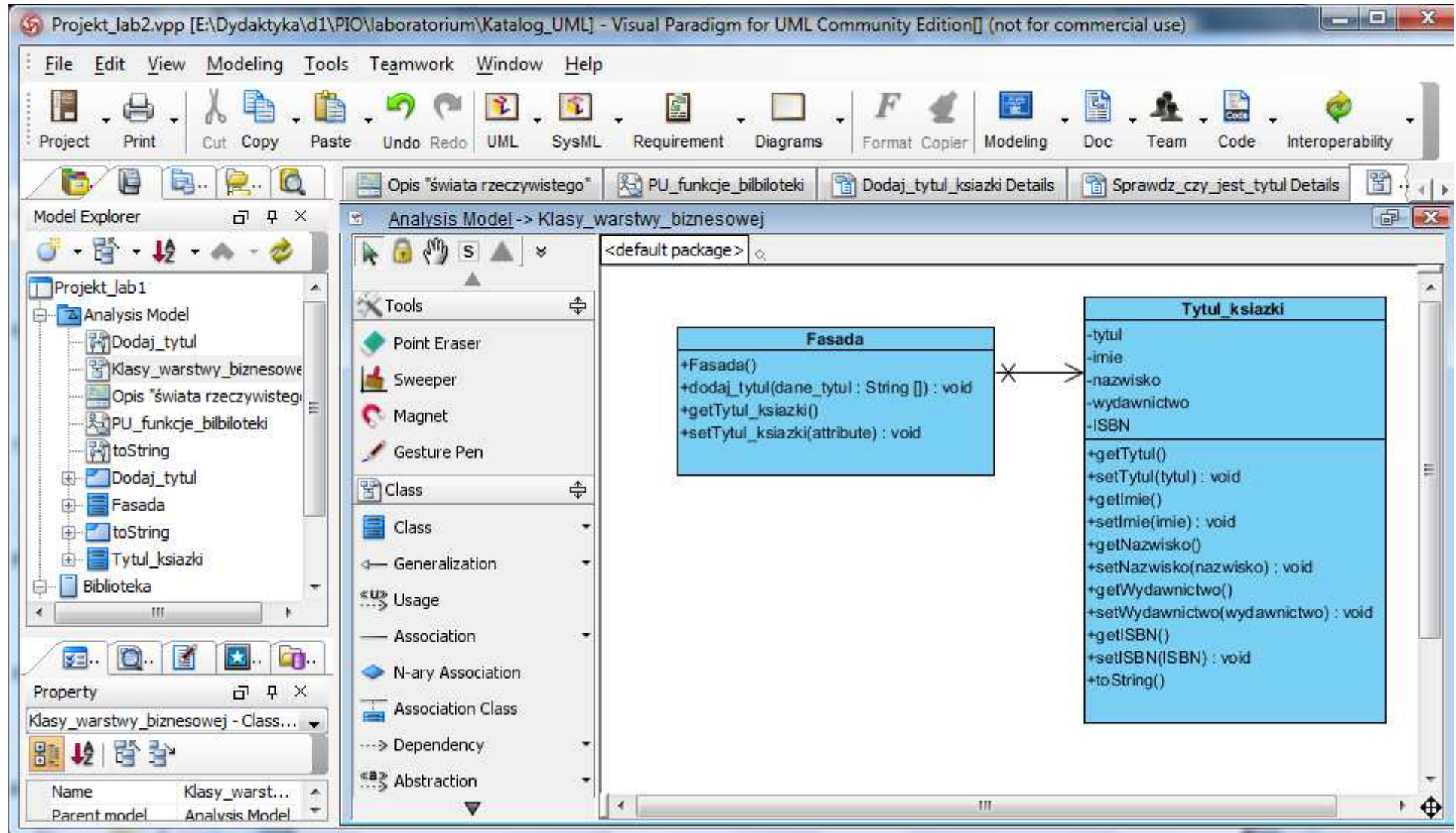
1.2. Należy utworzyć kopię projektu Projekt_lab1 za pomocą *File/Save Project as*



1.3. Utworzenie kopii o nazwie Projekt_lab2



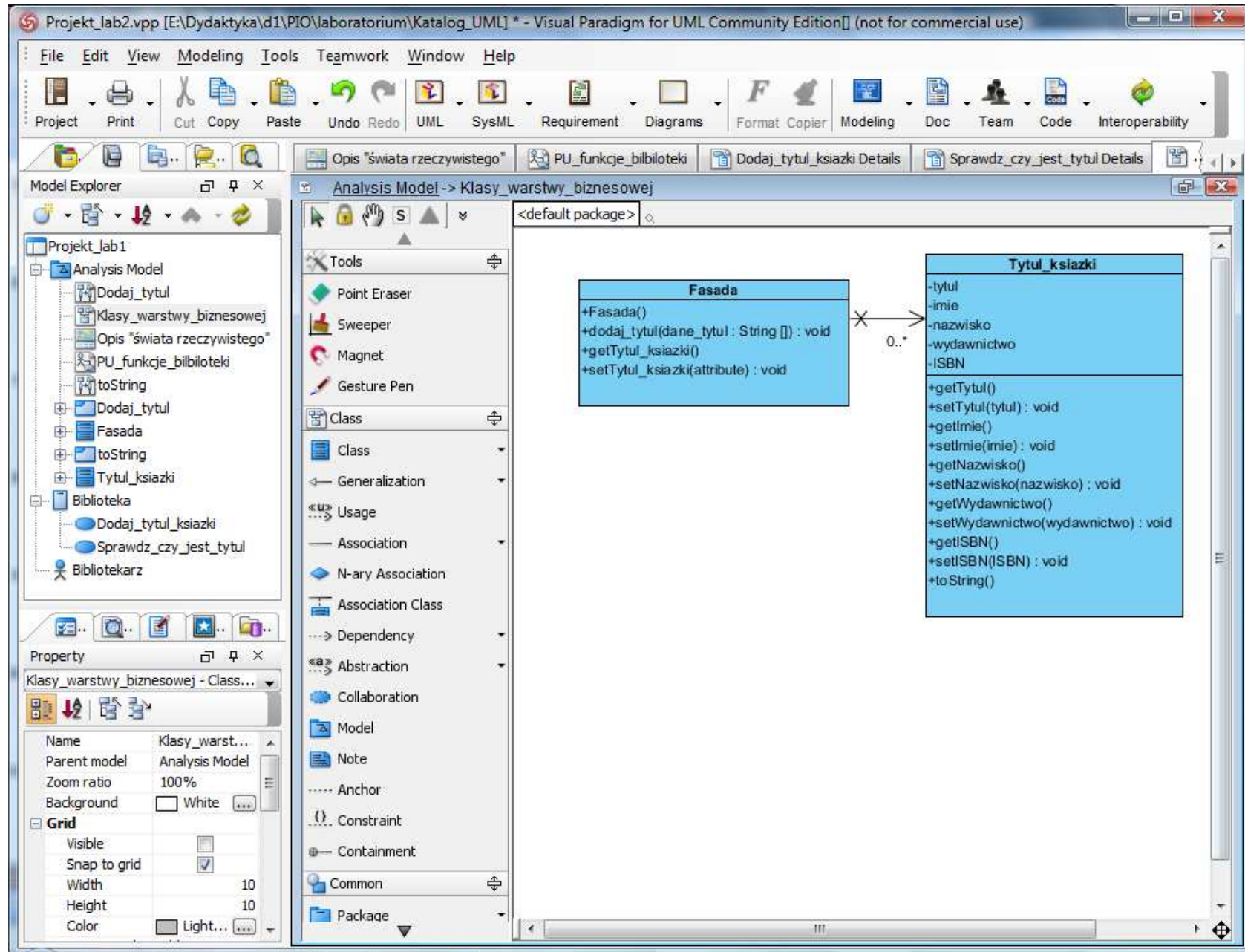
2.1. Zmiana relacji 1 : 1 między klasami **Fasada** i **Tytul_książki** na 1 : 0..*



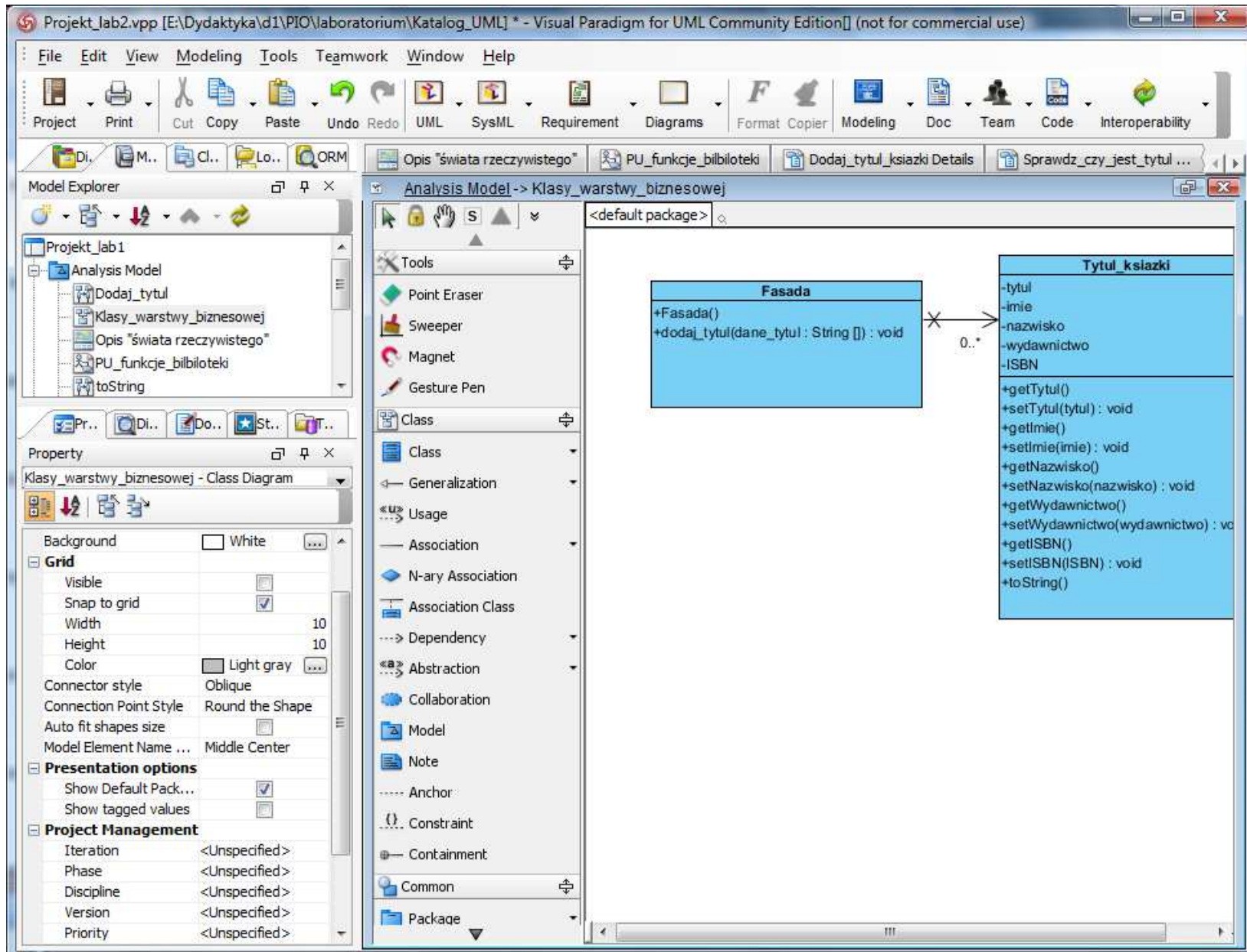
2.2. Relacja jeden do wiele – zmiana Multiplicity w relacji typu Association dla klasy Tytul_książki na 1:0..* (jeden do wiele)

The screenshot shows the Visual Paradigm for UML interface. The main workspace displays a class diagram with a class **Fasada** and an association to the class **Tytul_książki**. The association has a multiplicity of **0..*** at the **Tytul_książki** end. A context menu is open over the association line, showing the **Multiplicity** option selected. The menu options include: Open Specification..., Stereotypes, Role B (Tytul_książki), Navigable, Multiplicity (selected), Visibility, Aggregation Kind, Edit Role Name..., Qualifier..., Edit, Cut, Copy, Delete, Duplicate..., Change From/To Shape..., Reverse Connector, Scroll to Fasada, Pin, Reset Caption Position, Remove All Turning Point(s), Selection, Styles and Formatting, Presentation Options, Layer, Layout, and Related Elements. The **Property** window at the bottom left shows the association's name as **Fasada-Tytul_książki - Association** and its background color as **(122...)**. The **Model Explorer** on the left shows the project structure, including the **Analysis Model** and **Klasy_warstwy_biznesowej** package.

2.3. Rezultat zmian



2.4. Zmiana metod typu set i get w klasie **Fasada** ze względu na zmianę typu atrybutu implementującego relację 1:0..* na kolekcje typu np.. ArrayList



2.5. Dodawanie nowych metod typu set i get do klasy Fasada

The screenshot displays the Visual Paradigm for UML Community Edition interface. The main workspace shows a UML class diagram with two classes: **Fasada** and **Tytul_książki**. The **Fasada** class has a constructor `+Fasada()` and a method `+dodaj_tytul(dane_tytul : String []): void`. The **Tytul_książki** class has attributes `-tytul`, `-imie`, `-nazwisko`, and `-wydawnictwo ISBN`, and methods `+tTytul()`, `+tTytul(tytul) : void`, `+tImie()`, `+tImie(imie) : void`, `+tNazwisko()`, `+tNazwisko(nazwisko) : void`, `+tWydawnictwo()`, `+tWydawnictwo(wydawnictwo) : void`, `+tISBN()`, and `+tISBN(ISBN) : void`. A context menu is open over the **Fasada** class, with the **Operation** option selected. A sub-menu is also open, showing the **Add** option selected. The **Model Explorer** on the left shows the project structure, and the **Property** window at the bottom left shows the properties of the **Fasada** class.

Visual Paradigm for UML Community Edition (not for commercial use)

File Edit View Modeling Tools Teamwork Window Help

Project Print Cut Copy Paste Undo Redo UML SysML Requirement Diagrams Format Copier Modeling Doc Team Code Interoperability

Model Explorer

Projekt_lab1

Analysis Model

Dodaj_tytul

Klasy_warstwy_biznesowej

Opis "świata rzeczywistego"

PU_funkcje_biblioteki

toString

Property

Fasada - Class

Fill (122, 20...)

Line Black

Font Dialog

Connection point Follow diagram

Transparency 0

Opaque

Model Element Na... Follow Diagram

Show attribute opt... Show all

Show operation op... Show all

Attribute sort type No Sorting

Operation sort type No Sorting

Show initial attribu...

Show operation sig...

Visibility style UML

Wrap Members

Visibility public

Abstract

Leaf

Root

Active

Attributes

Association

N-ary Association

Association Class

Dependency

Abstraction

Collaboration

Model

Note

Anchor

Constraint

Containment

Common

Package

Tools

Point Eraser

Sweeper

Magnet

Analysis Model -> Klasy_warstwy_biznesowej

<default package>

Fasada

+Fasada()

+dodaj_tytul(dane_tytul : String []): void

Tytul_książki

-tytul

-imie

-nazwisko

-wydawnictwo ISBN

+tTytul()

+tTytul(tytul) : void

+tImie()

+tImie(imie) : void

+tNazwisko()

+tNazwisko(nazwisko) : void

+tWydawnictwo()

+tWydawnictwo(wydawnictwo) : void

+tISBN()

+tISBN(ISBN) : void

String()

0..*

Operation Alt+Shift+O

Add

Open Specification... Enter

Stereotypes

Model Element Properties

Move/Copy Members...

Sub Diagrams

Create Parent

Edit

Cut

Copy

Delete

Duplicate Ctrl+E

Selection

Order

Grouping

Styles and Formatting

Presentation Options

Layer

Related Elements

Java Round-trip

C++ Round-trip

22:31

2.6. Dodawanie nowych metod typu set i get do klasy Fasada

The screenshot displays the Visual Paradigm for UML interface. The main workspace shows a class diagram with two classes: **Fasada** and **Tytul_książki**. The **Fasada** class has a constructor `+Fasada()` and a method `+dodaj_tytul(dane_tytul : String []) : void`. The **Tytul_książki** class has attributes `-tytul`, `-imie`, `-nazwisko`, `-wydawnictwo`, and `-ISBN`, and methods `+getTytul()`, `+setTytul(tytul) : void`, `+getImie()`, `+setImie(imie) : void`, `+getNazwisko()`, `+setNazwisko(nazwisko) : void`, `+getWydawnictwo()`, `+setWydawnictwo(wydawnictwo) : void`, `+getISBN()`, `+setISBN(ISBN) : void`, and `+toString()`. An association line connects the two classes, with a multiplicity of `0..*` at the **Tytul_książki** end.

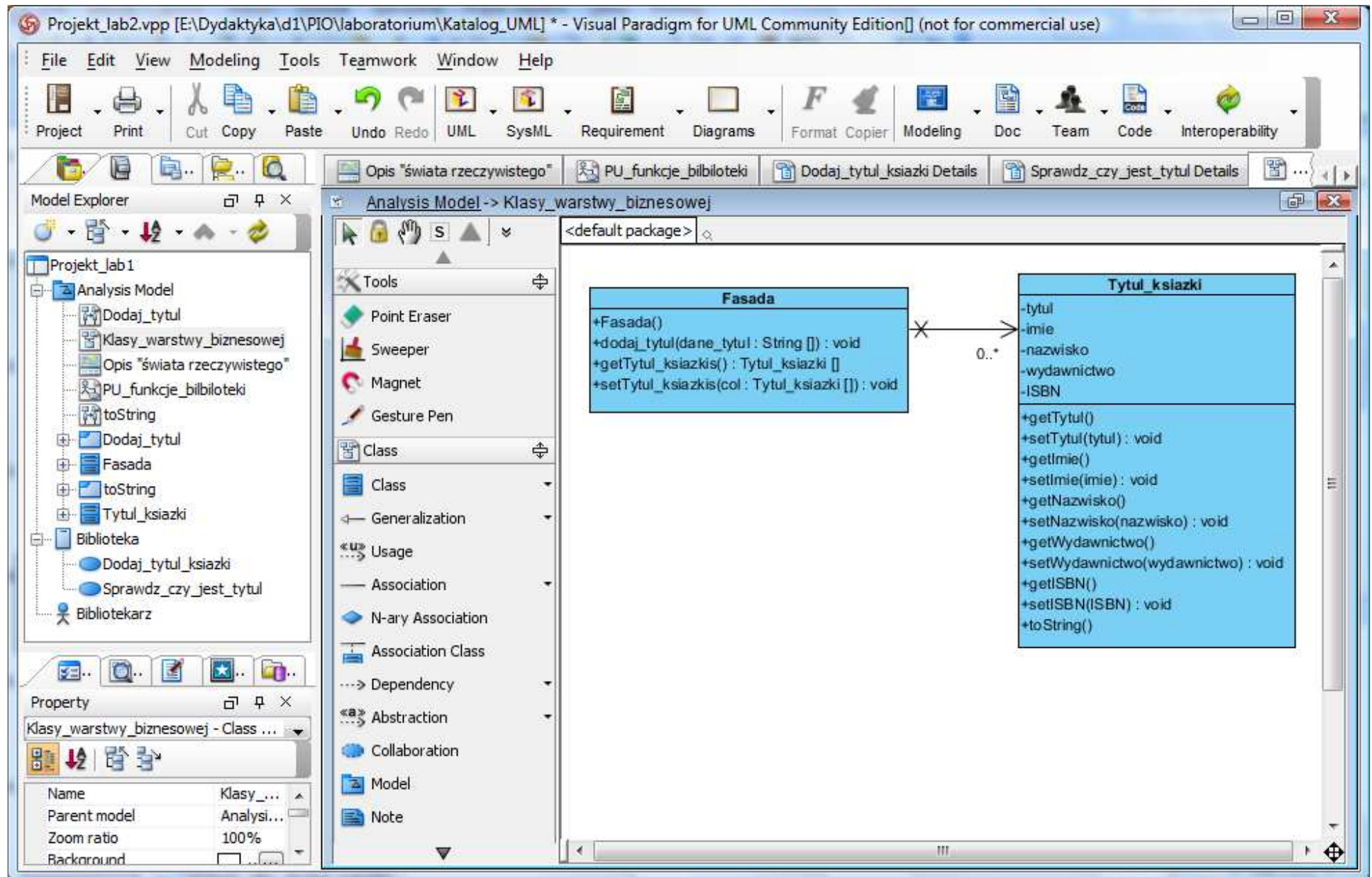
The left sidebar shows the Model Explorer with the following structure:

- Projekt_lab1
 - Analysis Model
 - Dodaj_tytul
 - Klasy_warstwy_biznesowej
 - Opis "świata rzeczywistego"
 - PU_funkcje_bibiloteki
 - toString

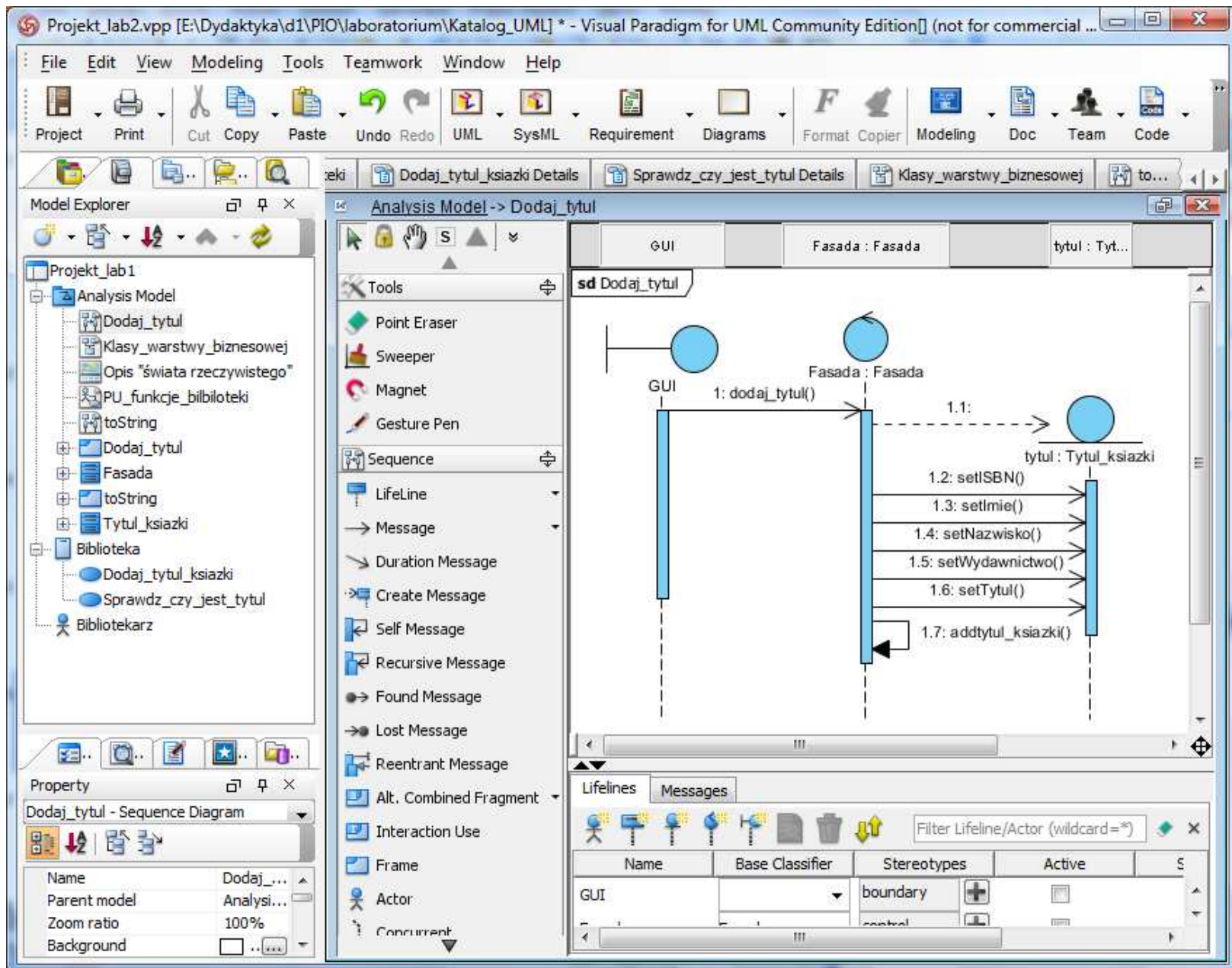
The Property window shows the following details for the `operation` class:

Name	operation
View	
Foreground	Black
Background	<Unspec...>
Return Type	
Visibility	public
Abstract	<input type="checkbox"/>
Query	<input type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Stereotypes	<Unspecified>
Tagged Values	
Comments	
Project Management	
Iteration	<Unspecified>
Phase	<Unspecified>
Discipline	<Unspecified>
Version	<Unspecified>
Priority	<Unspecified>
Status	<Unspecified>
Difficulty	<Unspecified>
Author	kruczkiewicz

2.7. Dodano nowe metody typu set i get do klasy Fasada



3.1. Dodanie do metody `dodaj_tytul` nowej metody `addTytul_ksiazki`, która realizuje sprawdzenie unikalności numeru ISBN wśród innych tytułów książek



3.2. Tworzenie nowej metody `addtytul_książki` w klasie `Fasada`

The screenshot displays the Visual Paradigm for UML interface. The main workspace shows a sequence diagram titled "sd Dodaj_tytul" with two lifelines: "GUI" and "Fasada : Fasada". A message arrow labeled "1: dodaj_tytul()" points from the GUI lifeline to the Fasada lifeline. A context menu is open over the Fasada lifeline, with the "Create Operation" option selected. A secondary menu is visible below it, showing the option "Create Operation 'addtytul_książki()'", which is highlighted. The left sidebar contains the Model Explorer, showing a project structure with "Projekt_lab1" and "Analysis Model" containing various elements like "Dodaj_tytul", "Klasy_warstwy_biznesowej", "Opis 'świata rzeczywistego'", "PU_funkcje_bibiloteki", "toString", "Dodaj_tytul", "Fasada", "Tytul_książki", "Biblioteka", "Dodaj_tytul_książki", "Sprawdzy_czy_jest_tytul", and "Bibliotekarz". The bottom left shows the Property window for the selected element, displaying "addtytul_książki() - Message" with fields for Name, Parent, and View.

3.3. Tworzenie nowej metody `addtytul_książki` w klasie `Fasada`

Operation Specification

Constraints | Traceability | References | Project Management | Quality | Comments

Operation Code Details | Java Annotations | Template Parameters | Stereotypes | Tagged Values

General | Parameters | Raised Exceptions | Preconditions | Postconditions

Name:

Classifier:

Return type:

Type modifier:

Visibility:

Scope:

Lower:

Upper:

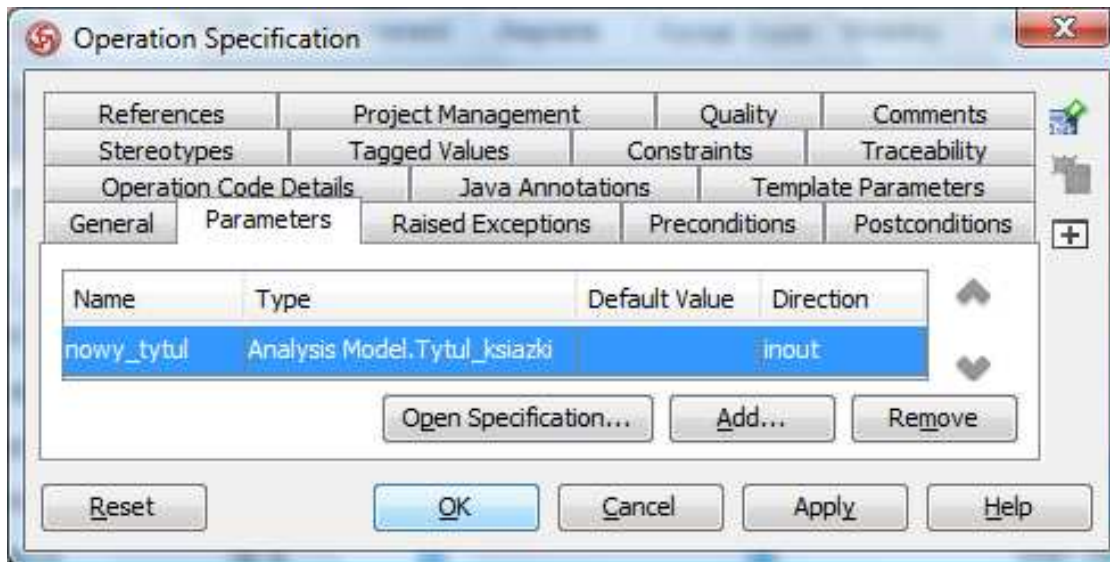
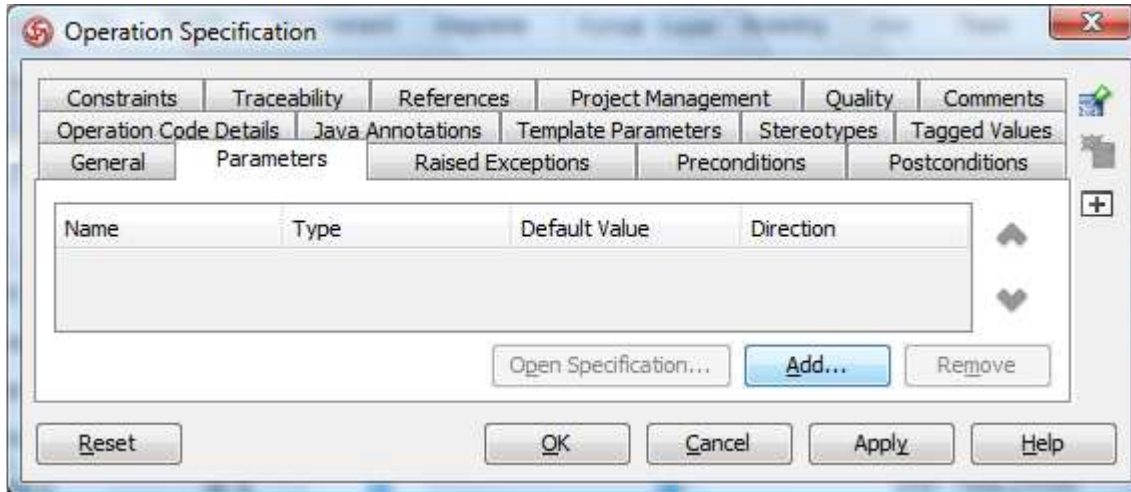
Body condition:

Documentation:

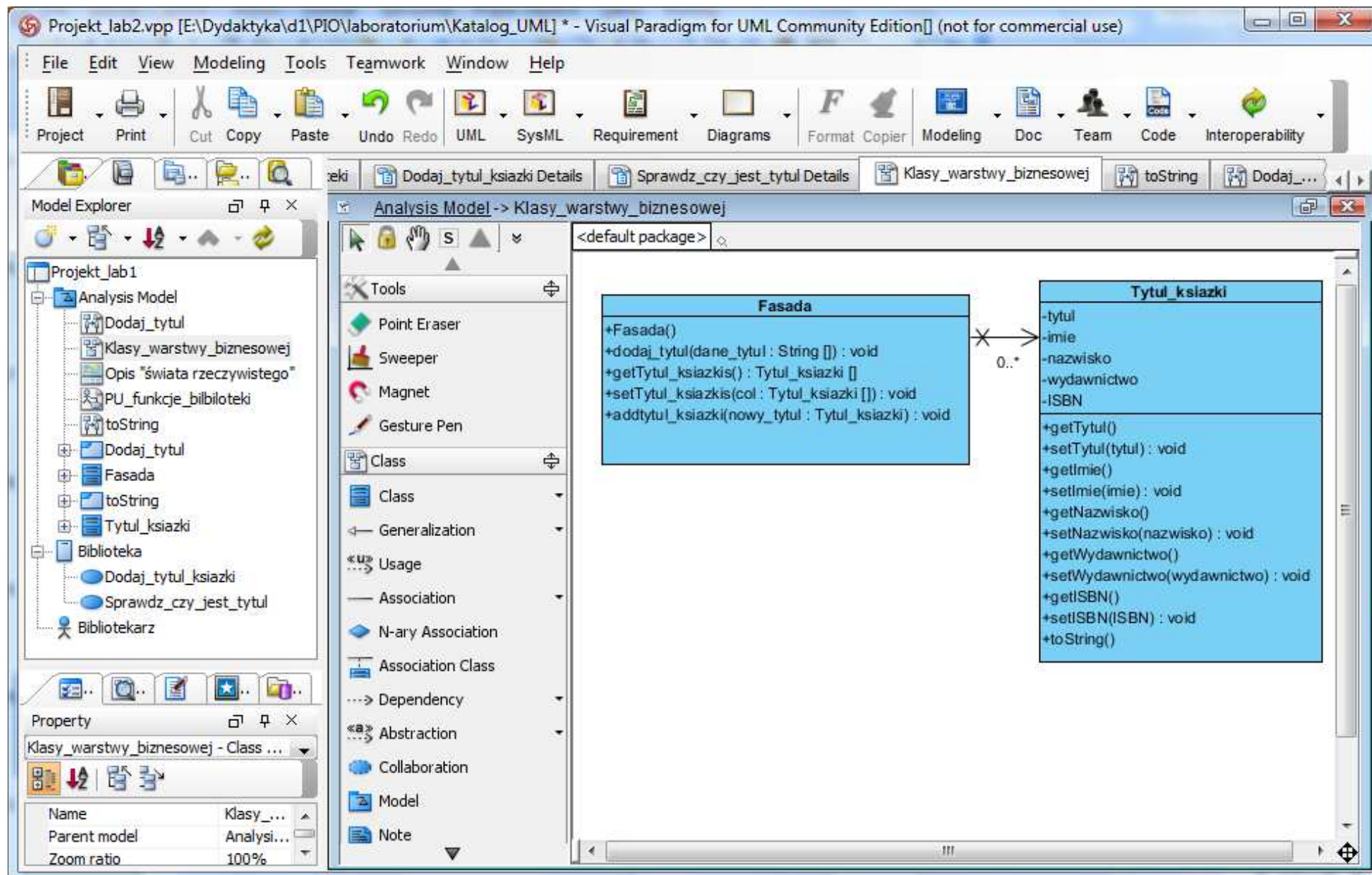
Abstract Leaf Query Ordered Unique

Reset OK Cancel Apply Help

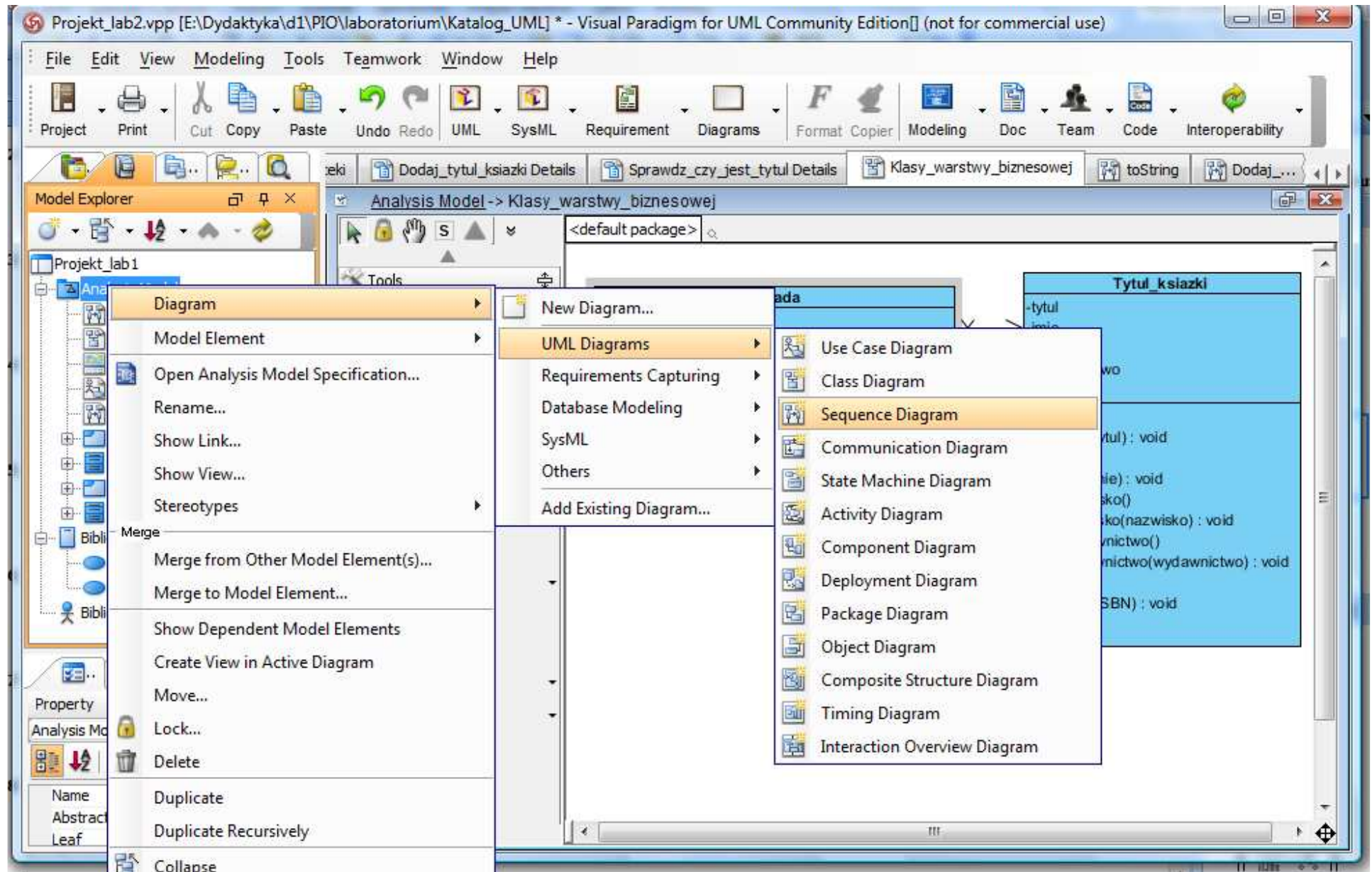
3.4. Definiowanie parametru przekazywanego do metody **addtytul_książki** typu **Tytul_książki**



3.5. Widok diagramu klas po dodaniu metody **addtytul_książki** do klasy **Fasada**,



4.1. Dodanie nowego diagramu sekwencji reprezentującego metodę `addtytul_książki` w klasie `Fasada`



4.2. Nadanie nazwy diagramowi – addtytul_książki

The screenshot displays the Visual Paradigm for UML Community Edition interface. The main window shows a sequence diagram titled "sd addtytul_książki" within the "Analysis Model" project. The diagram is currently empty, with only the title bar visible. The left sidebar shows the "Model Explorer" with a tree view of the project structure, including "Projekt_lab1", "Analysis Model", and "Biblioteka". The bottom-left pane shows the "Property" window for the selected diagram, with the name "addtytul_książki" and the parent model "Analysi...". The right sidebar contains a "Tools" palette with various UML diagram elements like "LifeLine", "Message", and "Create Message". The bottom-right pane shows the "Lifelines" and "Messages" tabs, with a filter for lifelines/actors.

Project: Projekt_lab2.vpp [E:\Dydaktyka\d1\PIO\laboratorium\Katalog_UML] * - Visual Paradigm for UML Community Edition[] (not for comm...)

File Edit View Modeling Tools Teamwork Window Help

Project Print Cut Copy Paste Undo Redo UML SysML Requirement Diagrams Format Copier Modeling Doc Team

Model Explorer

Projekt_lab1

- Analysis Model
 - Dodaj_tytul
 - Klasy_warstwy_biznesowej
 - Opis "świata rzeczywistego"
 - PU_funkcje_bilbiloteki
 - addtytul_książki
 - toString
 - addtytul_książki
- Dodaj_tytul
- Fasada
- toString
- Tytul_książki

Biblioteka

- Dodaj_tytul_książki
- Sprawdz_czy_jest_tytul

Bibliotekarz

Property

addtytul_książki - Sequence Diagram

Name addtyt...
Parent model Analysi...

sd addtytul_książki

Tools

- Point Eraser
- Sweeper
- Magnet
- Gesture Pen

Sequence

- LifeLine
- Message
- Duration Message
- Create Message
- Self Message
- Recursive Message
- Found Message
- Lost Message
- Reentrant Message
- Alt. Combined Fragment
- Interaction Use
- Frame

Lifelines Messages

Filter Lifeline/Actor (wildcard=*)

Name	Base Classifier	Stereotypes	Active
------	-----------------	-------------	--------

4.3. Dodanie dwóch linii życia typu *Entity Lifeline*

The screenshot displays the Visual Paradigm for UML Community Edition interface. The main window shows a sequence diagram titled "sd addtytul_książki" with two lifelines: "self" and "kolekcja". The "self" lifeline is a blue circle with a vertical dashed line, and the "kolekcja" lifeline is a blue circle with a horizontal line and a vertical dashed line. The diagram is contained within a rectangular frame labeled "sd addtytul_książki".

The left sidebar shows the Model Explorer with a tree view of the project structure. The "addtytul_książki" package is expanded, showing its contents. The bottom-left pane shows the Property window for the selected "addtytul_książki - Sequence Diagram".

The bottom-right pane shows the Lifelines and Messages table. The table has columns for Name, Base Classifier, Stereotypes, and Active. The "self" lifeline has a "control" stereotype, and the "kolekcja" lifeline has an "entity" stereotype.

Name	Base Classifier	Stereotypes	Active
self		control	<input type="checkbox"/>
kolekcja		entity	<input type="checkbox"/>

4.4. Powiązanie linii życia self z klasą Fasada

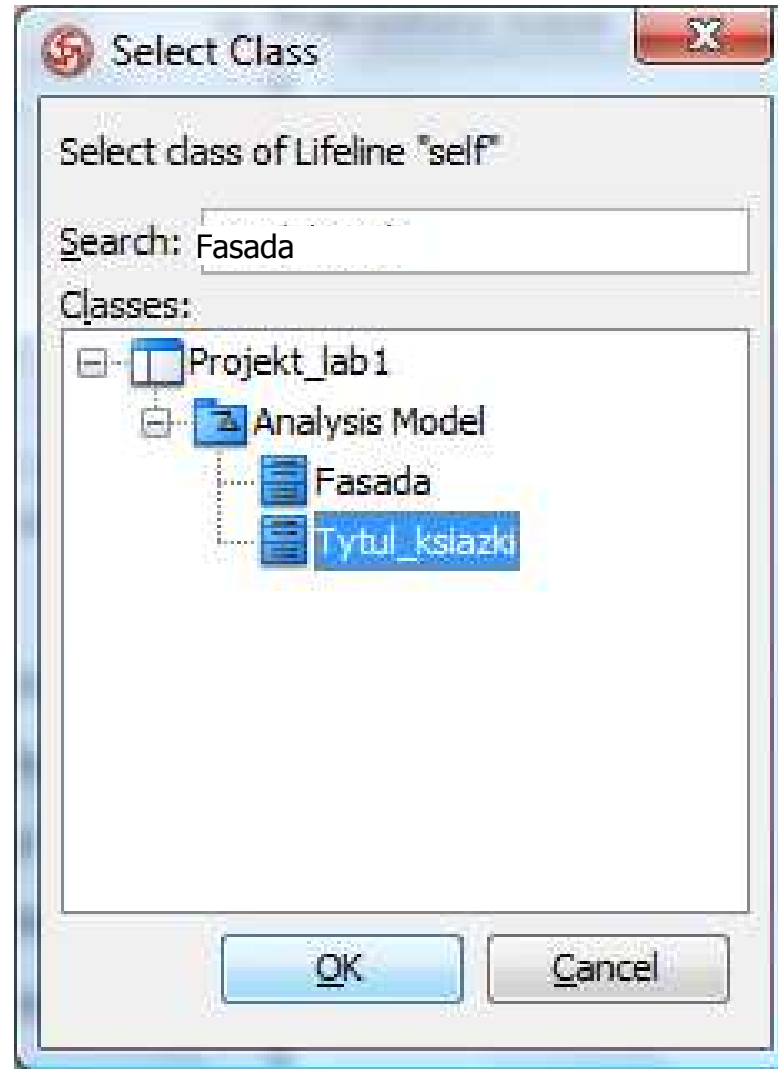
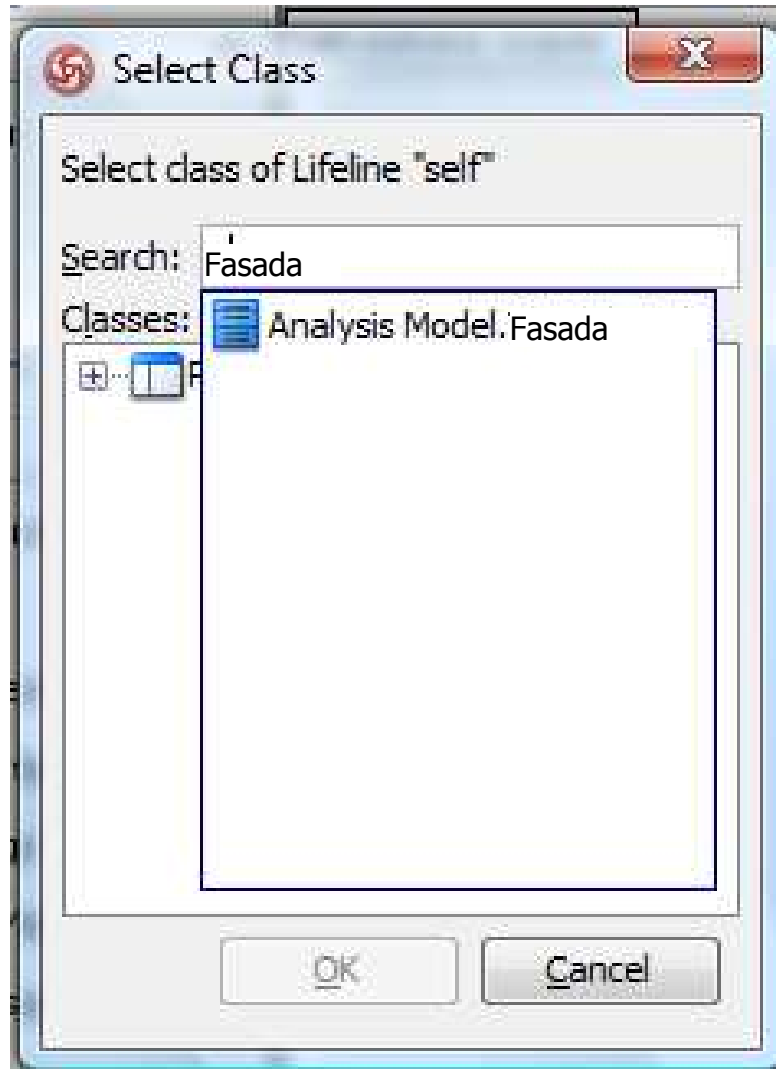
The screenshot shows the Visual Paradigm for UML interface. The main window displays a sequence diagram titled "sd addtytul_książki" with two lifelines: "self" and "kolekcja". A context menu is open over the "self" lifeline, with the "Select Class" option highlighted. A sub-menu is also open, showing options like "Select Class...", "Create Class...", and "Create Class 'Self'".

The Model Explorer on the left shows the project structure, including the "Analysis Model" and "Biblioteka" packages. The Property window at the bottom left shows the properties of the selected "self" lifeline.

Name	Parent
self	addyt...

The bottom of the screen shows the Windows taskbar with the system clock at 00:16.

4.5. Dokonanie selekcji klasy Fasada

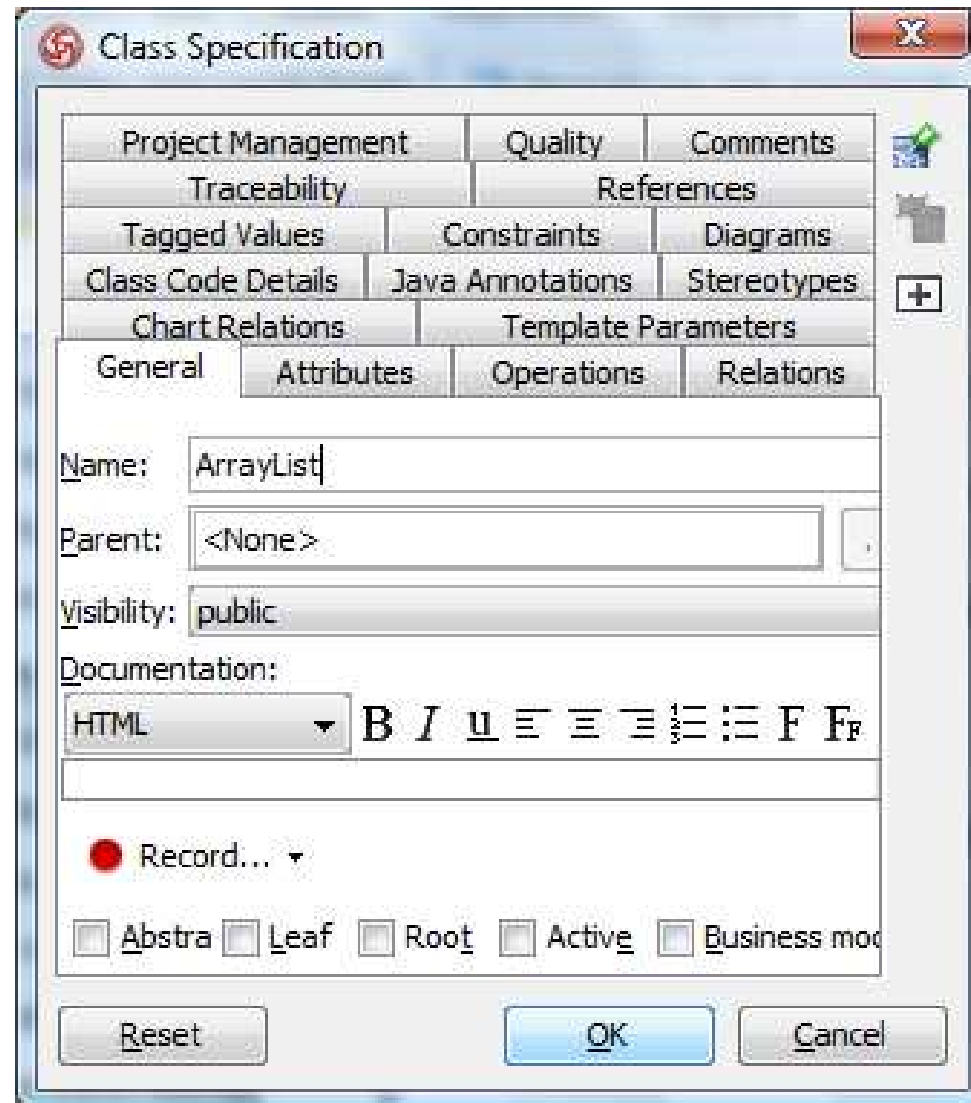


4.6. Powiązanie linii życia kolekcja z nową klasą pomocniczą **ArrayList** (odpowiednik klasy Javy)

The screenshot displays the Visual Paradigm for UML interface. The main workspace shows a sequence diagram titled 'sd addtytul_ksiazki' with two lifelines: 'self : Tytu...' and 'kolekcja'. A context menu is open over the 'kolekcja' lifeline, with the 'Create Class...' option selected. A sub-menu is visible, showing 'Create Class "Kolekcja"' as the chosen option. The left sidebar contains the Model Explorer, showing a project structure with 'Projekt_Jab1' and 'Analysis Model'. The bottom status bar shows the system tray with the time 00:23.

Name	Base Classifier	Ste
self	Fasada	control
kolekcja		entity

4.7. Utworzenie klasy pomocniczej typu **ArrayList**



4.8. Dodanie bloku warunkowego Alt Combined Fragment z palety oraz edycja wyrażenia warunkowego bloku typu Alt Combined Fragment

The screenshot displays the Visual Paradigm for UML Community Edition interface. The main workspace shows a sequence diagram titled "sd addtytul_książki" with two lifelines: "self : Fasada" and "kolekcja : ArrayList". The diagram includes a message "1: contains():boolean" from "self" to "kolekcja", and a return message "1.1: boolean" from "kolekcja" to "self". An "alt" block is attached to the "self" lifeline, containing a guard condition "[nie istnieje_tytul_książki]".

The left sidebar shows the Model Explorer with a tree structure for "Projekt_lab1" containing "Analysis Model" and "Biblioteka". The "Analysis Model" contains "Dodaj_tytul", "Klasy_warstwy_biznesowej", "Opis 'świata rzeczywistego'", "PU_funkcje_biblioteki", "addtytul_książki", "toString", "Fasada", "Tytul_książki", and "ArrayList". The "Biblioteka" contains "Dodaj_tytul_książki", "Sprawdz_czy_jest_tytul", and "Bibliotekarz".

The bottom-left pane shows the Property window for "addtytul_książki - Sequence Diagram" with fields for Name, Parent model, and Zoom ratio.

The bottom-right pane shows the Lifelines and Messages table:

Name	Base Classifier	Stereotypes	Active
self	Fasada	control	<input type="checkbox"/>
kolekcja	ArrayList	entity	<input type="checkbox"/>

4.9. Usunięcie niepotrzebnej linii reprezentującej dodatkowy operand elementu typu **Alt Combined Fragment**

The screenshot displays the Visual Paradigm for UML interface. The main workspace shows a sequence diagram for the 'addtytul_książki' use case. It features two lifelines: 'self : Fasada' and 'kolekcja : ArrayList'. A message '1: contains():boolean' is sent from 'self' to 'kolekcja', and a return message '1.1: boolean' is sent back. An 'alt' fragment is attached to the return message, containing a constraint '[[nie istnieje_tytul_książki]]'. A context menu is open over the 'alt' fragment, with the 'Remove Operand' option selected. The 'Remove Operand' submenu is also visible, showing the operand to be removed: '[[nie istnieje_tytul_książki]]'.

The interface includes a menu bar (File, Edit, View, Modeling, Tools, Teamwork, Window, Help), a toolbar, a Model Explorer on the left showing the project structure, and a Property window at the bottom left. The bottom of the screen shows the Windows taskbar with the Google search bar and system tray.

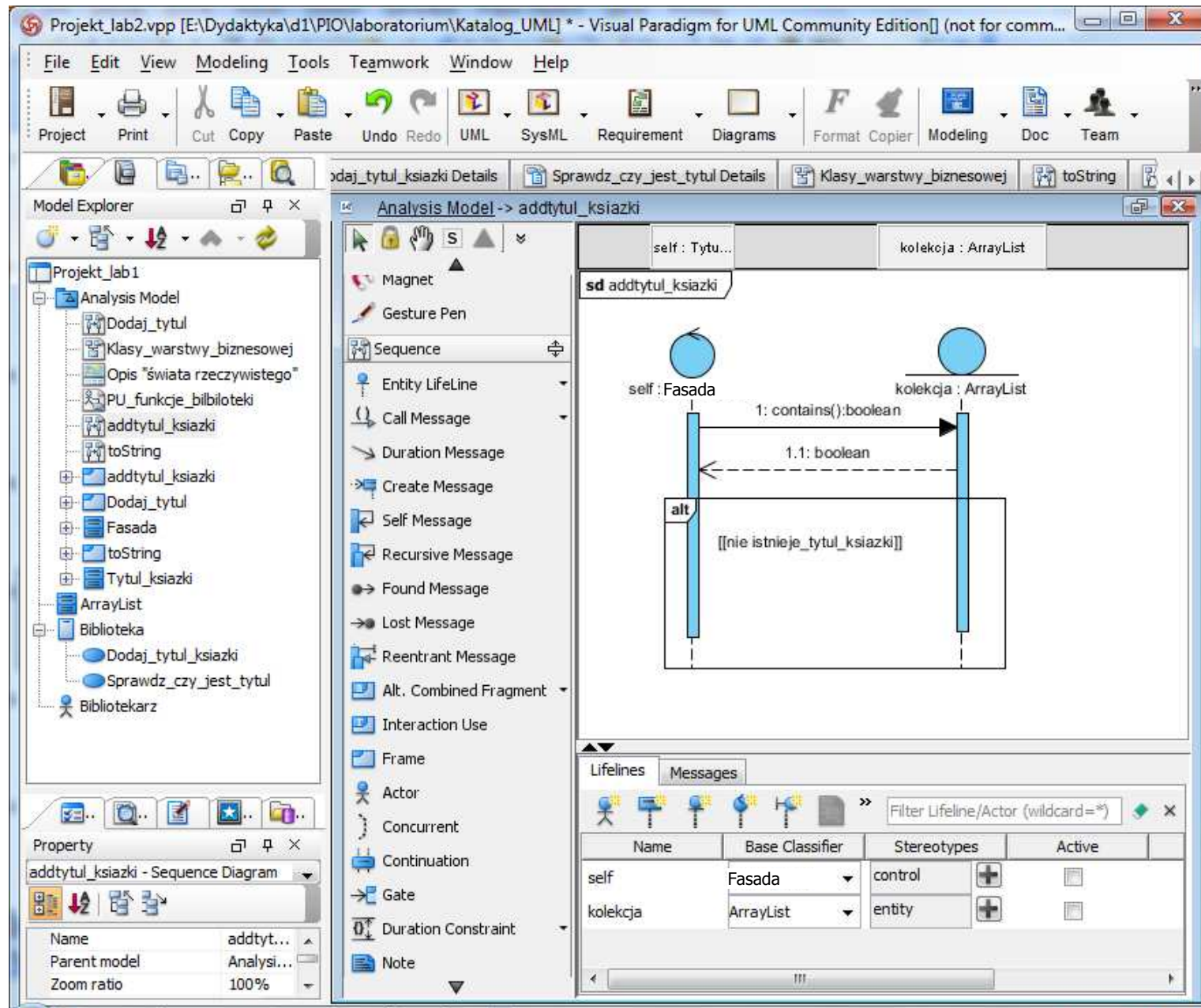
4.10. Usunięcie niepotrzebnej linii reprezentującej dodatkowy operand elementu typu **Alt Combined Fragment** – potwierdzenie operacji usuwania

The screenshot displays the Visual Paradigm for UML Community Edition interface. The main workspace shows an interaction diagram for the 'sd addtytul_książki' sequence diagram. It features two lifelines: 'self : Fasada' and 'kolekcja : ArrayList'. A message '1: contains():boolean' is sent from 'self' to 'kolekcja'. Below this, an 'alt' fragment is shown with a guard condition '[[nie istnieje_tytul_książki]]'. A 'Confirm Removal' dialog box is overlaid on the diagram, asking: 'Remove operand "Operand2 (no constraint)"?' with 'Yes' and 'No' buttons.

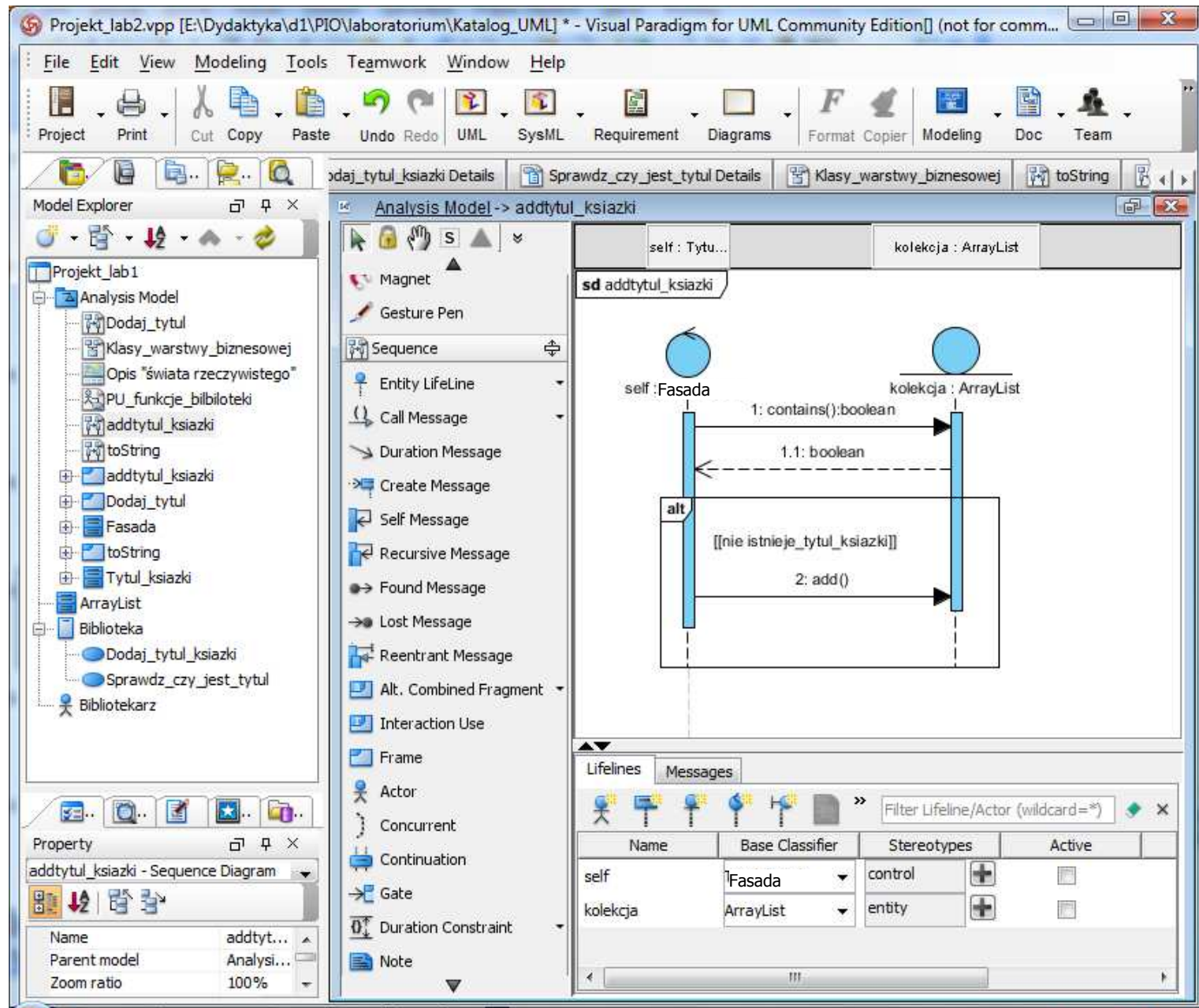
The left sidebar shows the 'Model Explorer' with a tree view of the project structure, including 'Projekt_lab1', 'Analysis Model', and various diagrams like 'Dodaj_tytul', 'Klasy_warstwy_biznesowej', and 'addtytul_książki'. The bottom of the interface shows a table with the following data:

Name	Base Classifier	Stereotypes	Active	Stopped	Multi-object
self	Fasada	control	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
kolekcja	ArrayList	entity	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

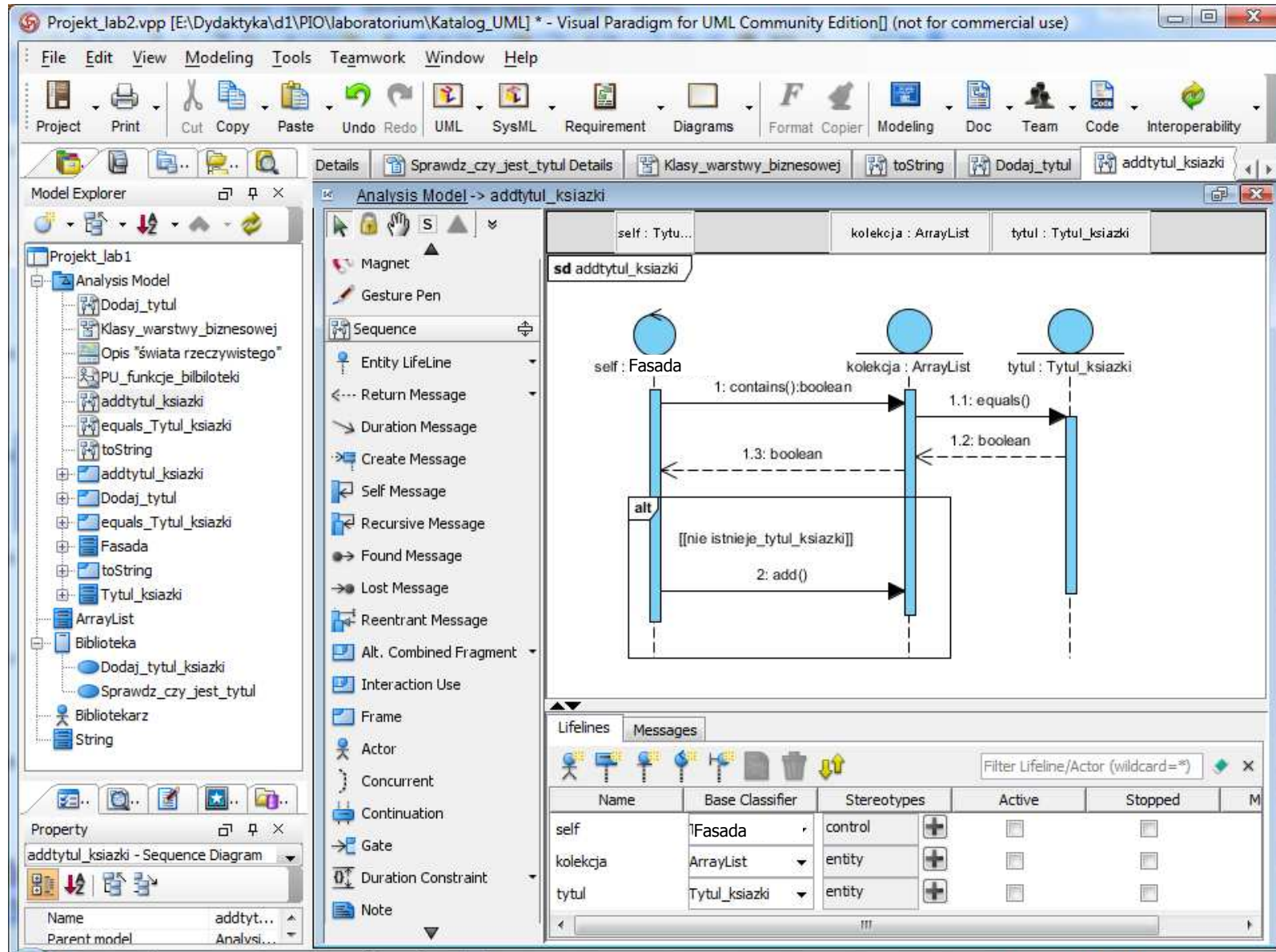
4.11. Zakończenie zmiany w bloku typu **Alt Combined Fragment**.



4.12. Dodanie do metody **dodaj_tytul** nowej metody **addTytul_książki** metody **add** wywoływanej przez linię życia **typu Fasada** z linii życia **typu ArrayList**



4.13. Uzupełniono scenariusz metody `addtytul_książki` o wywołanie metody `equals` z klasy `Tytul_książki` przesłaniającej metodę `equals` dziedziczoną od klasy `Object` – teraz w metodzie `contains` wywoływana jest metoda `equals` z klasy `Tytul_książki`



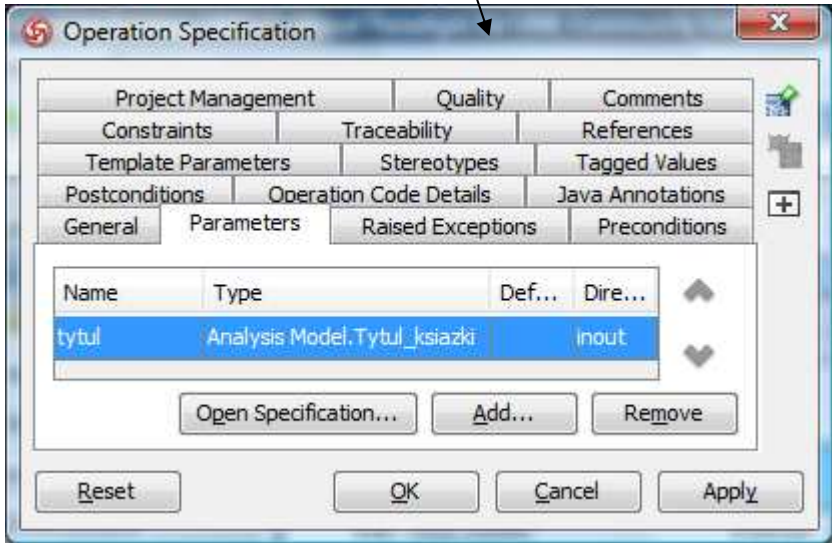
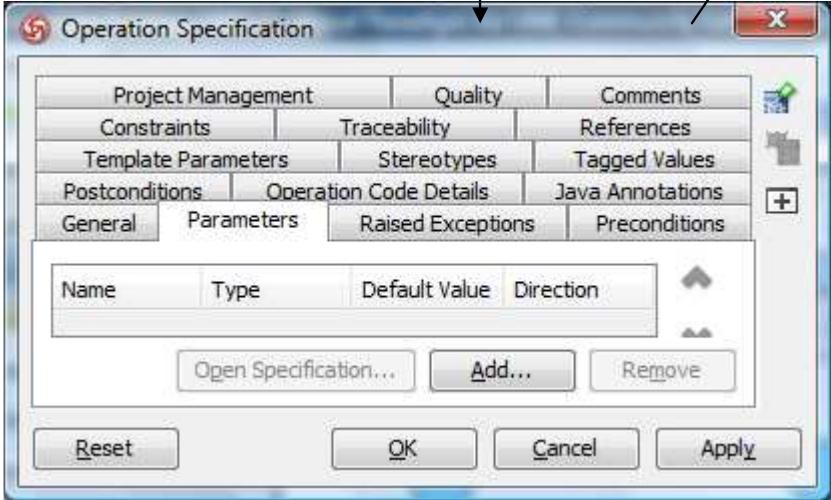
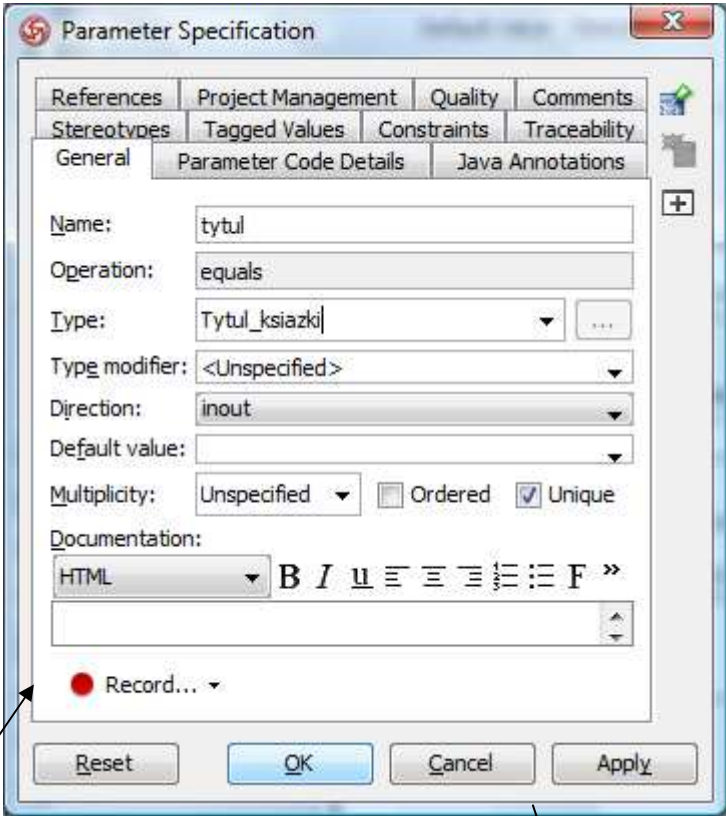
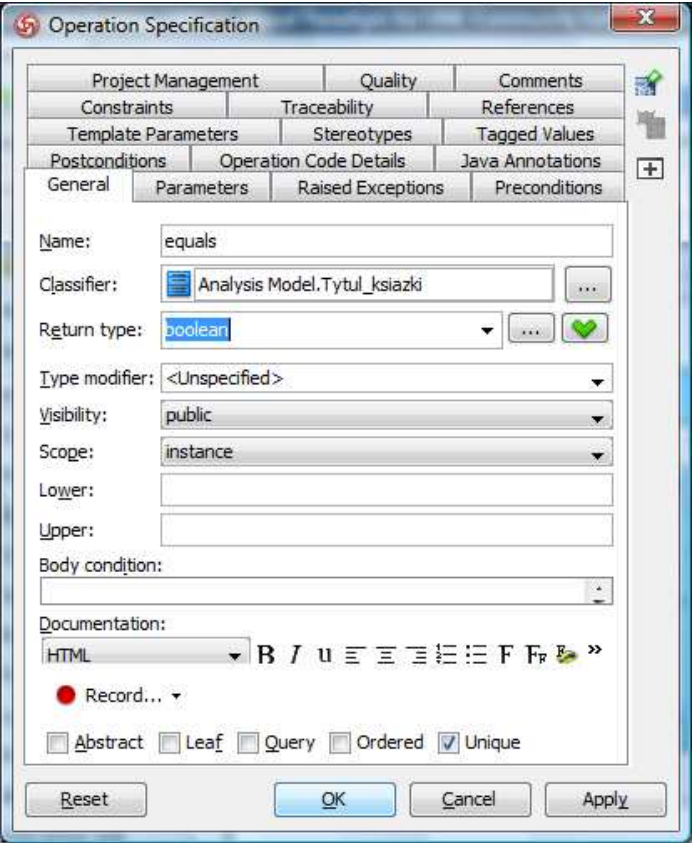
4.14. Zapisanie metody equals w klasie Tytul_książki.

The screenshot displays the Visual Paradigm for UML interface. The main workspace shows a sequence diagram for the `addtytul_książki` operation. The diagram includes lifelines for `self : Fasada` and `kolekcja : ArrayList`. A message `1: contains():boolean` is sent from `self` to `kolekcja`. A second message `2: add()` is sent from `self` to `kolekcja`. A context menu is open over the diagram, with the `Create Operation "equals()"` option selected. The `Messages` tab is active, showing a table of messages:

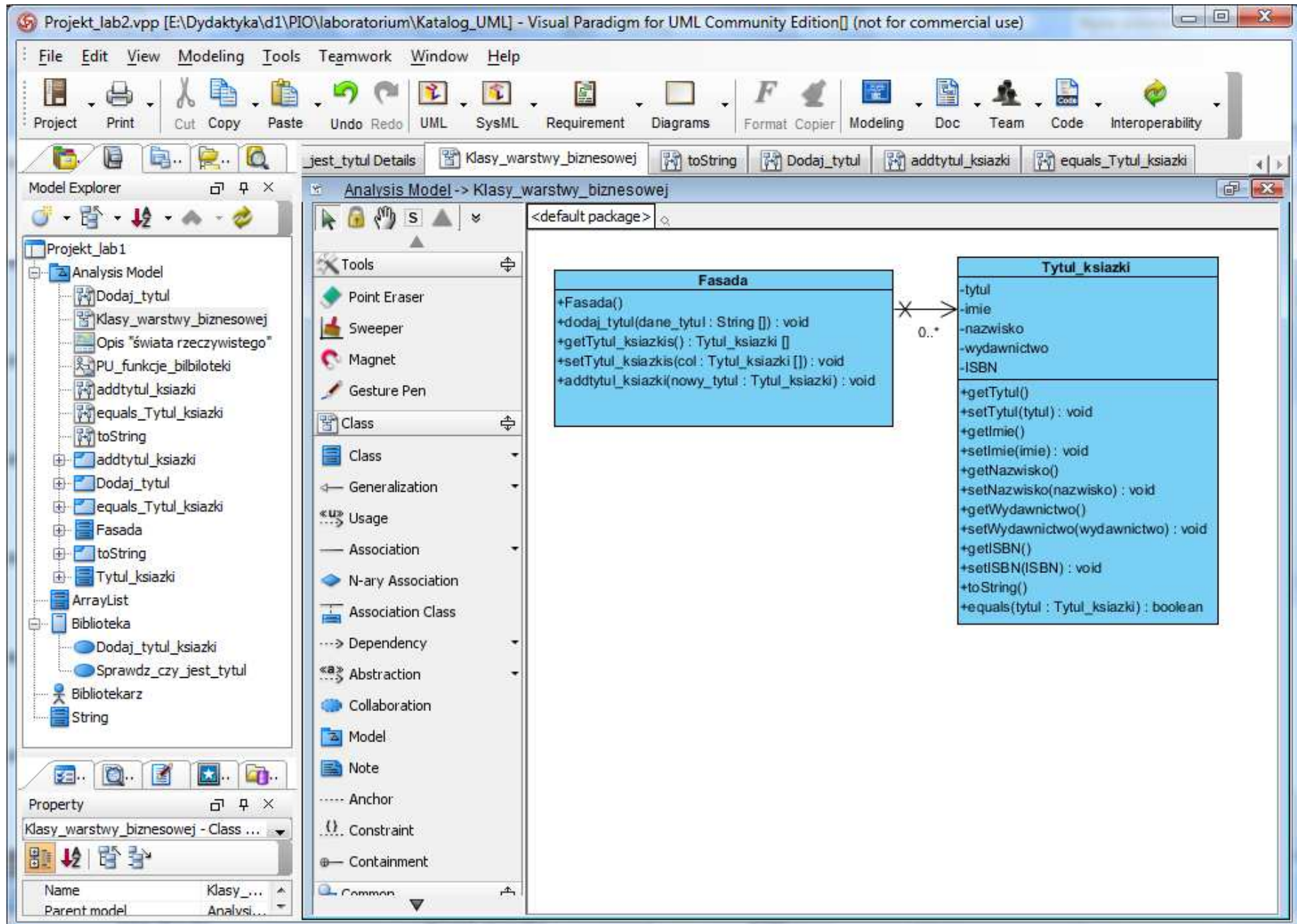
Name	Base Classifier	Stereotypes
self	Fasada	control
kolekcja	ArrayList	entity
tytul	Tytul_książki	entity

The `Property` window at the bottom left shows the details for the `equals() - Message`, with the name `equals()` and parent `Message`.

4.15
Definiowanie metody equals w klasie Tytul_ksiazki



4.16. Nowe metody w klasach **Fasada** (typu get i set oraz addtytul_książki) oraz **Tytul_książki** (equals)



5.1. Utworzenie nowego diagramu sekwencji do zdefiniowania metody equals w klasie Tytul_książki

The screenshot shows the Visual Paradigm for UML interface. The main window displays an Analysis Model with a diagram titled 'sd addtytul_książki'. A context menu is open over the diagram, and the 'UML Diagrams' sub-menu is selected, showing a list of diagram types. The 'Sequence Diagram' option is highlighted. The 'Lifelines' and 'Messages' panels are visible at the bottom of the diagram area.

UML Diagrams List:

- Use Case Diagram
- Class Diagram
- Sequence Diagram
- Communication Diagram
- State Machine Diagram
- Activity Diagram
- Component Diagram
- Deployment Diagram
- Package Diagram
- Object Diagram
- Composite Structure Diagram
- Timing Diagram
- Interaction Overview Diagram

Lifelines Table:

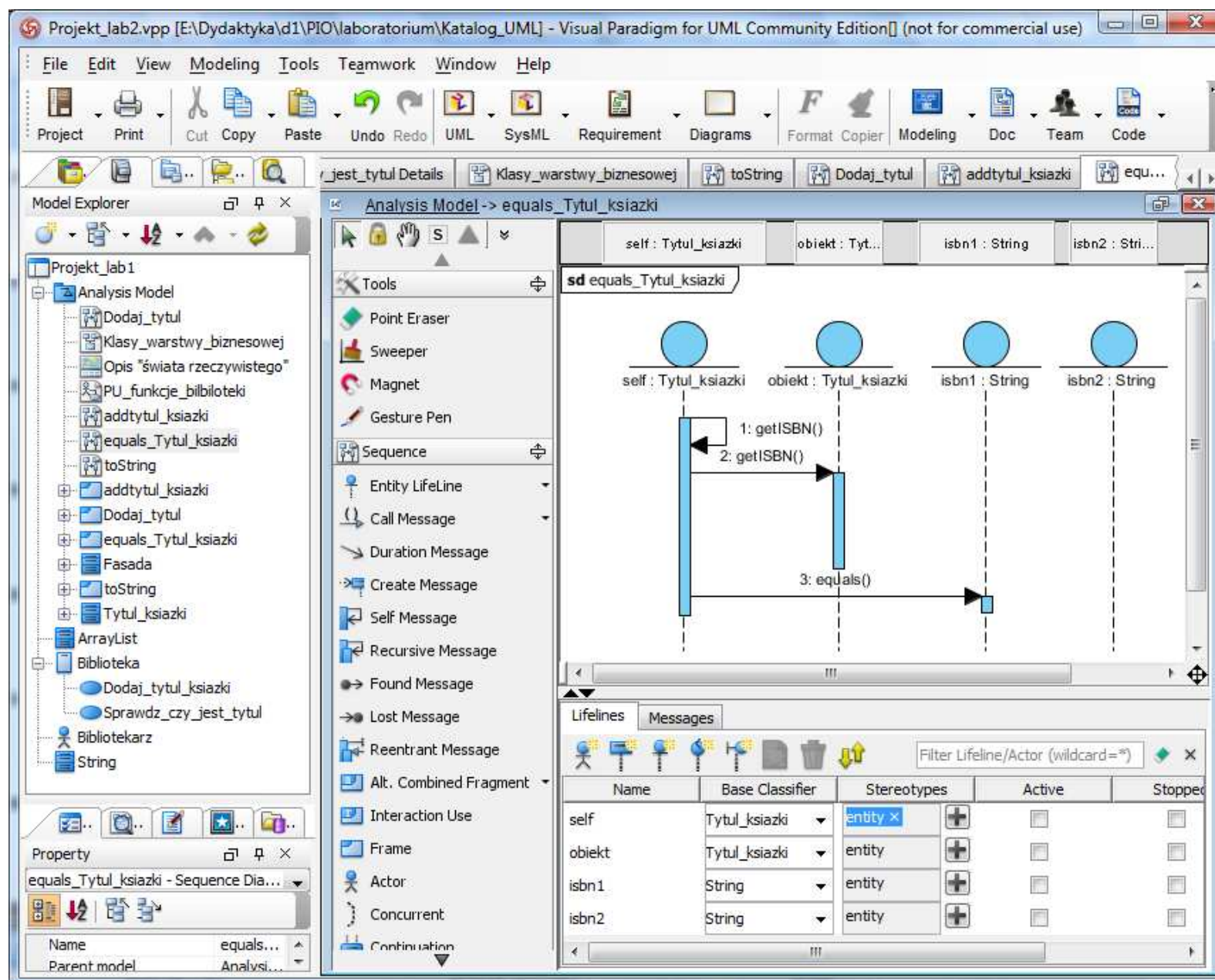
Name	Base Classifier	Stereotypes	Active	Stopped	Multi-object
self	Tytuł_książki	control	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
kolekcja	ArrayList	entity	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

5.2. Nadanie nazwy diagramowi sekwencji

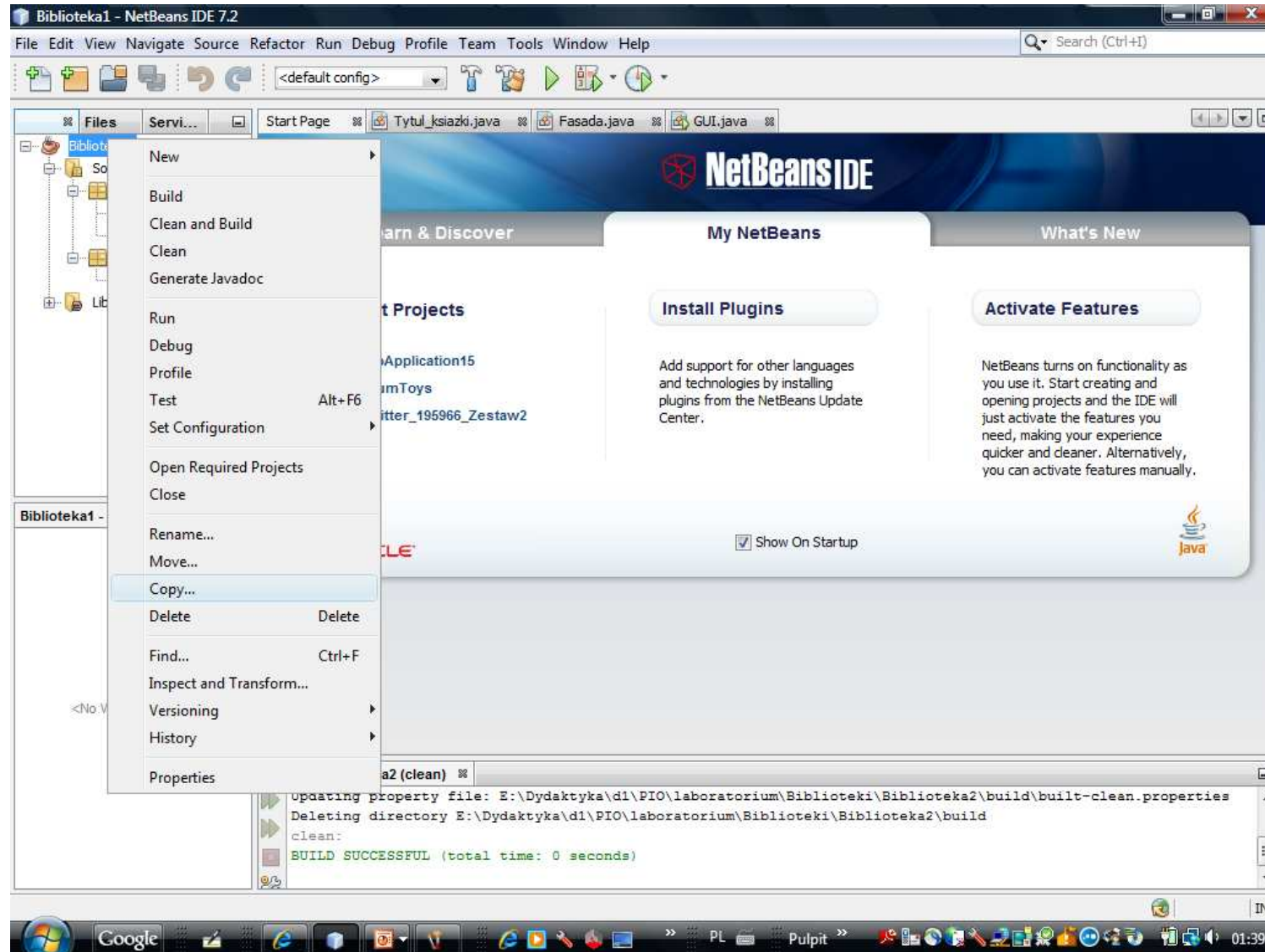
The screenshot displays the Visual Paradigm for UML Community Edition interface. The main workspace shows a sequence diagram titled "sd equals_Tytul_ksiazki". The diagram is currently empty, with only the title bar visible. The left sidebar contains the Model Explorer, showing a project structure with "Projekt_lab1" and "Analysis Model". The "Analysis Model" folder is expanded, revealing several elements: "Dodaj_tytul", "Klasy_warstwy_biznesowej", "Opis 'świata rzeczywistego'", "PU_funkcje_bilbiloteki", "addtytul_ksiazki", "equals_Tytul_ksiazki", "toString", "addtytul_ksiazki", "Dodaj_tytul", "equals_Tytul_ksiazki", "Fasada", "toString", "Tytul_ksiazki", "ArrayList", "Biblioteka", "Dodaj_tytul_ksiazki", "Sprawdz_czy_jest_tytul", and "Bibliotekarz". The bottom-left pane shows the Property window for the selected diagram, displaying "equals_Tytul_ksiazki - Sequence Dia...". The bottom-right pane shows the Lifelines and Messages table, which is currently empty.

Name	Base Classifier	Stereotypes	Active
------	-----------------	-------------	--------

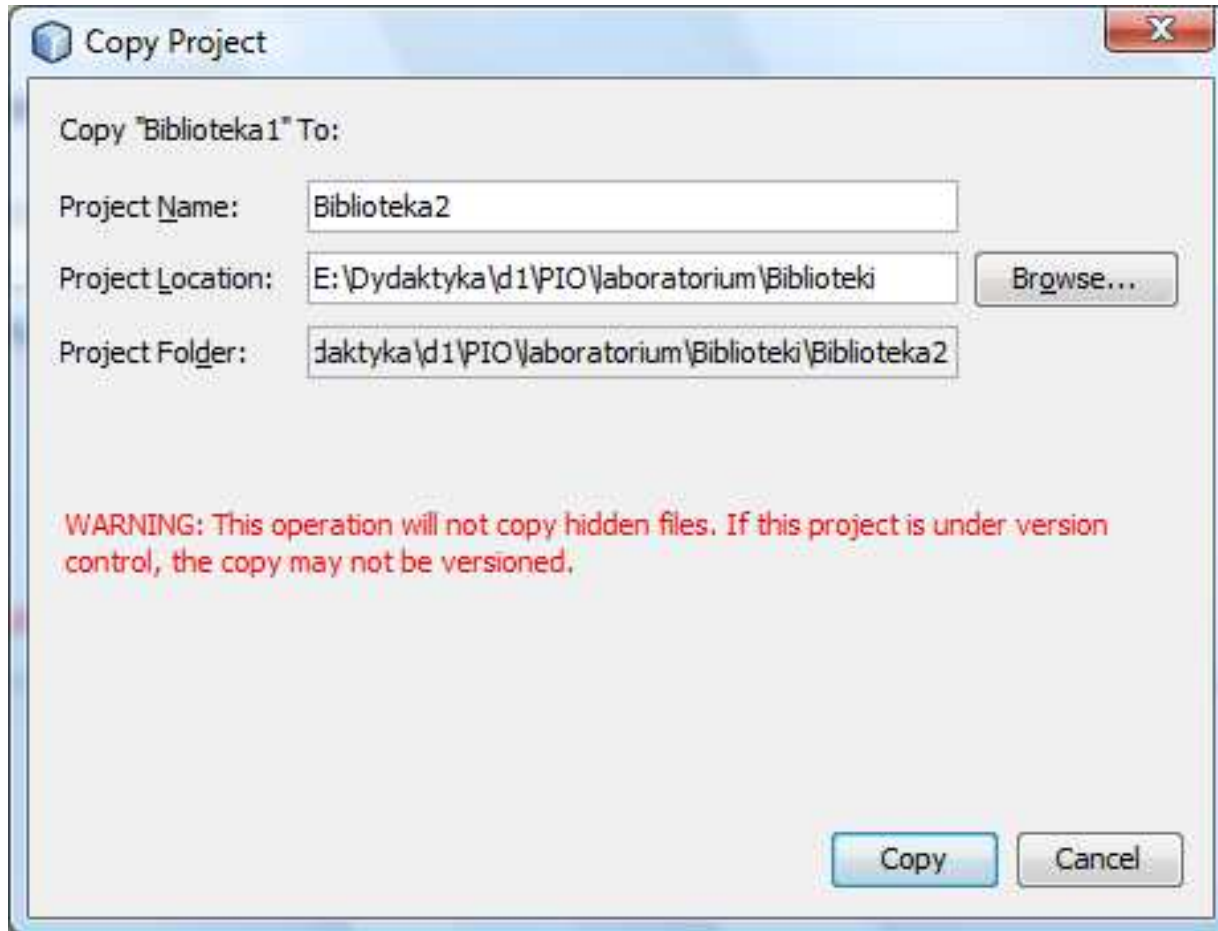
5.3. Wykonanie diagramu sekwencji reprezentującego metodę **equals** w klasie **Tytul_książki**



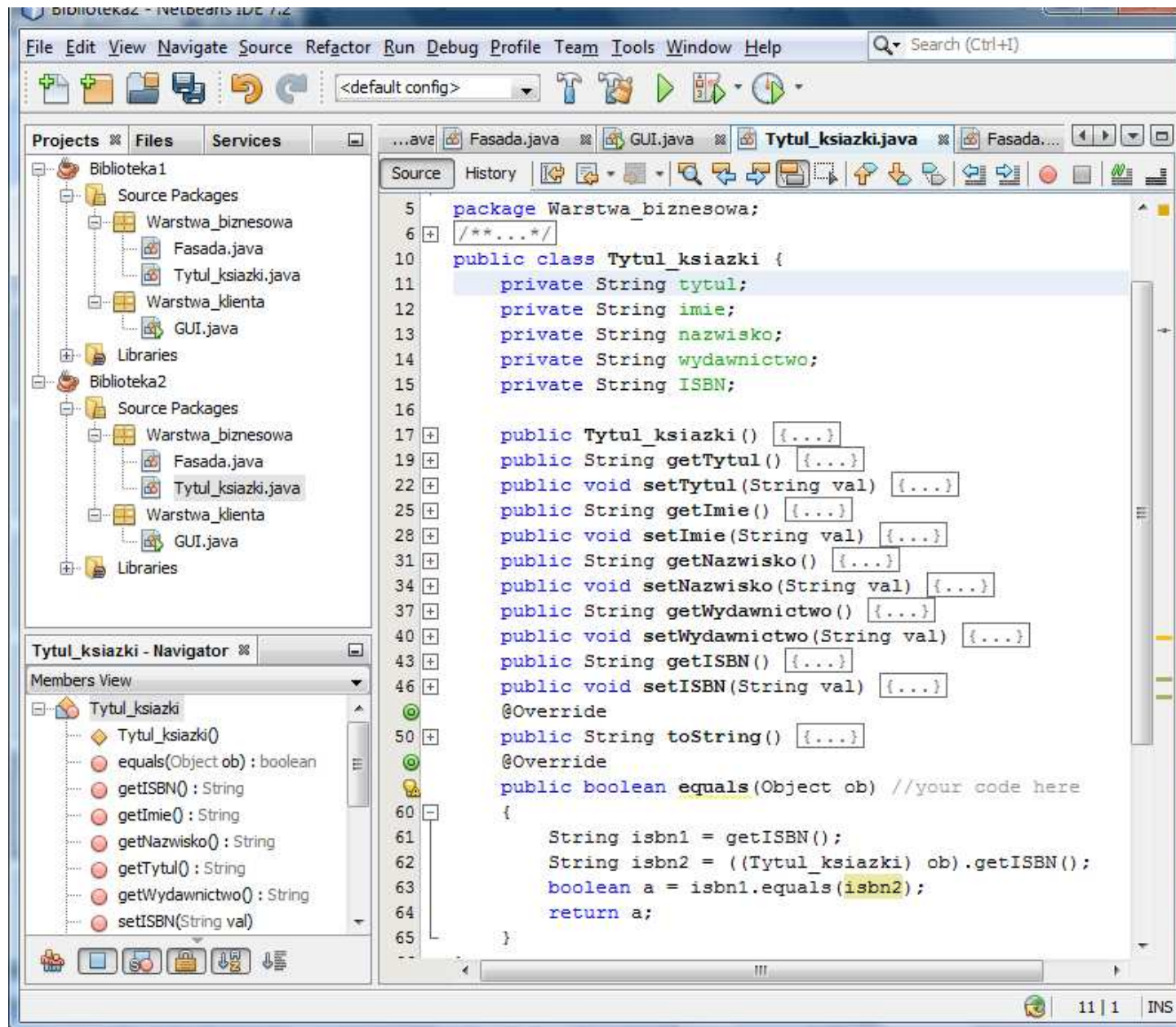
6.1. Modyfikacja kodu programu na podstawie diagramu klas i zmodyfikowanych diagramów sekwencji



6.2. Utworzenie kopii programu z projektu reprezentującego relację 1:1 z lab1.



6.3. Kod klasy **Tytul Książki**: metody **toString** i **equals** napisane na podstawie ich diagramów sekwencji



Uzupełniona definicja klasy `Tytul_ksiazki` o metodę `equals`, która przesłania metodę `equals` dziedziczoną od klasy `Object`

```
package Warstwa_biznesowa; public class Tytul_ksiazki {
    private String tytul;
    private String imie;
    private String nazwisko;
    private String wydawnictwo;
    private String ISBN;

    public Tytul_ksiazki() { }
    public String getTytul() { return tytul; }
    public void setTytul(String val) { this.tytul = val; }
    public String getImie() { return imie; }
    public void setImie(String val) { this.imie = val; }
    public String getNazwisko() { return nazwisko; }
    public void setNazwisko(String val) { this.nazwisko = val; }
    public String getWydawnictwo() { return wydawnictwo; }
    public void setWydawnictwo(String val) { this.wydawnictwo = val; }
    public String getISBN() { return ISBN; }
    public void setISBN(String val) { this.ISBN = val; }
```

`@Override`

```
public String toString() {
    String pom = "Tytul: " + getTytul();
    pom += " Autor:" + getNazwisko() + " " + getImie();
    pom += " ISBN: " + getISBN();
    pom += " Wydawnictwo:" + getWydawnictwo();
    return pom; }
```

`@Override`

```
public boolean equals(Object ob) {
    String isbn1 = getISBN();
    String isbn2 = ((Tytul_ksiazki) ob).getISBN();
    boolean a = isbn1.equals(isbn2);
    return a; }
```

```
}
```

6.4. Kod w klasie **Fasada** implementujący relację **jeden:wiele**. Metody **dodaj_tytul** oraz **addtytul_książki** zostały napisane na podstawie diagramów sekwencji

```
5 package Warstwa_biznesowa;
6
7 import java.util.ArrayList;
8 /**...*/
12 public class Fasada {
13
14     private ArrayList <Tytul_książki> tytul_książek=new ArrayList<Tytul_książki>();
15
16     public Fasada() { }
17
18     public void dodaj_tytul(String dane_tytul[]) {
19         Tytul_książki tytul_książki = new Tytul_książki();
20         tytul_książki.setTytul(dane_tytul[0]);
21         tytul_książki.setNazwisko(dane_tytul[1]);
22         tytul_książki.setImie(dane_tytul[2]);
23         tytul_książki.setISBN(dane_tytul[3]);
24         tytul_książki.setWydawnictwo(dane_tytul[4]);
25         addtytul_książki(tytul_książki);
26     }
27
28     public void addtytul_książki(Tytul_książki val) {
29         boolean czy_jest = tytul_książek.contains(val);
30         if (!czy_jest)
31             tytul_książek.add(val);
32     }
33
34     public ArrayList <Tytul_książki> getTytuly_książek() {
35         return tytul_książek;
36     }
37
38     public void setTytuly_książek(ArrayList<Tytul_książki> val) {
39         this.tytuly_książek = val;
40     }
41 }
```

ArrayList jako kolekcja referencji typu do obiektu klasy **Tytul_książki** reprezentuje relację 1 do 0..* po stronie klasy **Fasada**, która jest „właścicielem” relacji

Uzupełniono kod metody `dodaj_tytul` oraz dodano metodę `addTytul_książki`

```
package Warstwa_biznesowa;
```

```
import java.util.ArrayList;  
public class Fasada {
```

Kolekcja referencji do obiektu klasy `Tytul_książki` reprezentuje relację 1 do 0..* po stronie klasy Fasada, która jest „właścicielem” relacji

```
    private ArrayList <Tytul_książki> tytuly_książek=new ArrayList<Tytul_książki>();
```

```
    public Fasada() { }
```

```
    public void dodaj_tytul(String dane_tytul[]) {  
        Tytul_książki tytul_książki = new Tytul_książki();  
        tytul_książki.setTytul(dane_tytul[0]);  
        tytul_książki.setNazwisko(dane_tytul[1]);  
        tytul_książki.setImie(dane_tytul[2]);  
        tytul_książki.setISBN(dane_tytul[3]);  
        tytul_książki.setWydawnictwo(dane_tytul[4]);  
        addtytul_książki(tytul_książki);  
    }
```

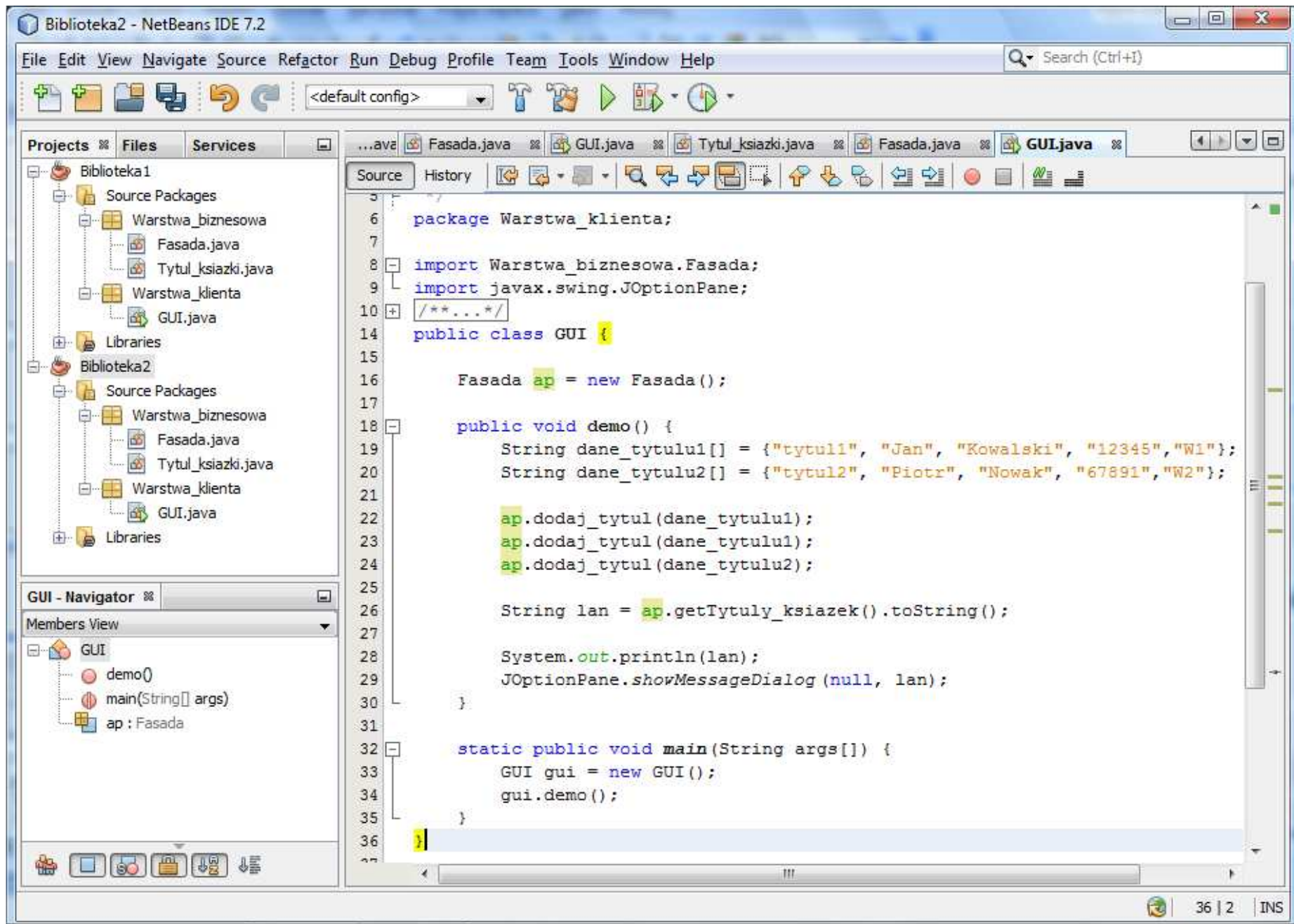
```
    public void addtytul_książki(Tytul_książki val) {  
        boolean czy_jest = tytuly_książek.contains(val);  
        if (!czy_jest)  
            tytuly_książek.add(val);  
    }
```

```
    public ArrayList <Tytul_książki> getTytuly_książek() { return tytuly_książek; }
```

```
    public void setTytuly_książek(ArrayList<Tytul_książki> val) { this.tytuly_książek = val; }
```

```
}
```

6.5. Zmodyfikowana metoda **demo** w klasie **GUI** z warstwy klienta.



The screenshot shows the NetBeans IDE 7.2 interface. The main editor displays the source code for the `GUI` class in the `Warstwa_klienta` package. The code includes imports for `Warstwa_biznesowa.Fasada` and `javax.swing.JOptionPane`. The `demo` method is defined, which creates a `Fasada` object and calls `dodaj_tytul` with two sets of book data. It then retrieves the list of titles and displays them in a message dialog. The `main` method creates a `GUI` instance and calls `demo`.

```
6 package Warstwa_klienta;
7
8 import Warstwa_biznesowa.Fasada;
9 import javax.swing.JOptionPane;
10 /**...*/
11
14 public class GUI {
15
16     Fasada ap = new Fasada();
17
18     public void demo() {
19         String dane_tytulu1[] = {"tytul1", "Jan", "Kowalski", "12345", "W1"};
20         String dane_tytulu2[] = {"tytul2", "Piotr", "Nowak", "67891", "W2"};
21
22         ap.dodaj_tytul(dane_tytulu1);
23         ap.dodaj_tytul(dane_tytulu1);
24         ap.dodaj_tytul(dane_tytulu2);
25
26         String lan = ap.getTytuly_ksiazek().toString();
27
28         System.out.println(lan);
29         JOptionPane.showMessageDialog(null, lan);
30     }
31
32     static public void main(String args[]) {
33         GUI gui = new GUI();
34         gui.demo();
35     }
36 }
```

The GUI - Navigator window shows the `GUI` class with members `demo()`, `main(String[] args)`, and `ap : Fasada`.

36 | 2 | INS

```
package Warstwa_klienta;

import Warstwa_biznesowa.Fasada;
import javax.swing.JOptionPane;

public class GUI {

    Fasada ap = new Fasada();

    public void demo() {
        String dane_tytulu1[] = {"tytul1", "Jan", "Kowalski", "12345","W1"};
        String dane_tytulu2[] = {"tytul2", "Piotr", "Nowak", "67891","W2"};

        ap.dodaj_tytul(dane_tytulu1);
        ap.dodaj_tytul(dane_tytulu1);
        ap.dodaj_tytul(dane_tytulu2);

        String lan = ap.getTytuly_ksiazek().toString();

        System.out.println(lan);
        JOptionPane.showMessageDialog(null, lan);
    }

    static public void main(String args[]) {
        GUI gui = new GUI();
        gui.demo();
    }
}
```


7. Poprawne wstawianie obiektów typu `Tytul_książki` w klasie `Fasada`– każdy tytuł może mieć inny ISBN dzięki metodzie `equals` zdefiniowanej w klasie `Tytul_książki` i wywoływanej w metodzie `contains` klasy typu `ArrayList`

