Instrukcja 2
Laboratorium z Podstaw Inżynierii Oprogramowania

Opis biznesowy „świata rzeczywistego”

Wymagania funkcjonalne i niefunkcjonalne aplikacji

Diagram przypadków życia

Diagramy klas i sekwencji:

Relacja 1 do 1..0– instrukcja z lab1

Relacja 1 do 0..* - nowy materiał

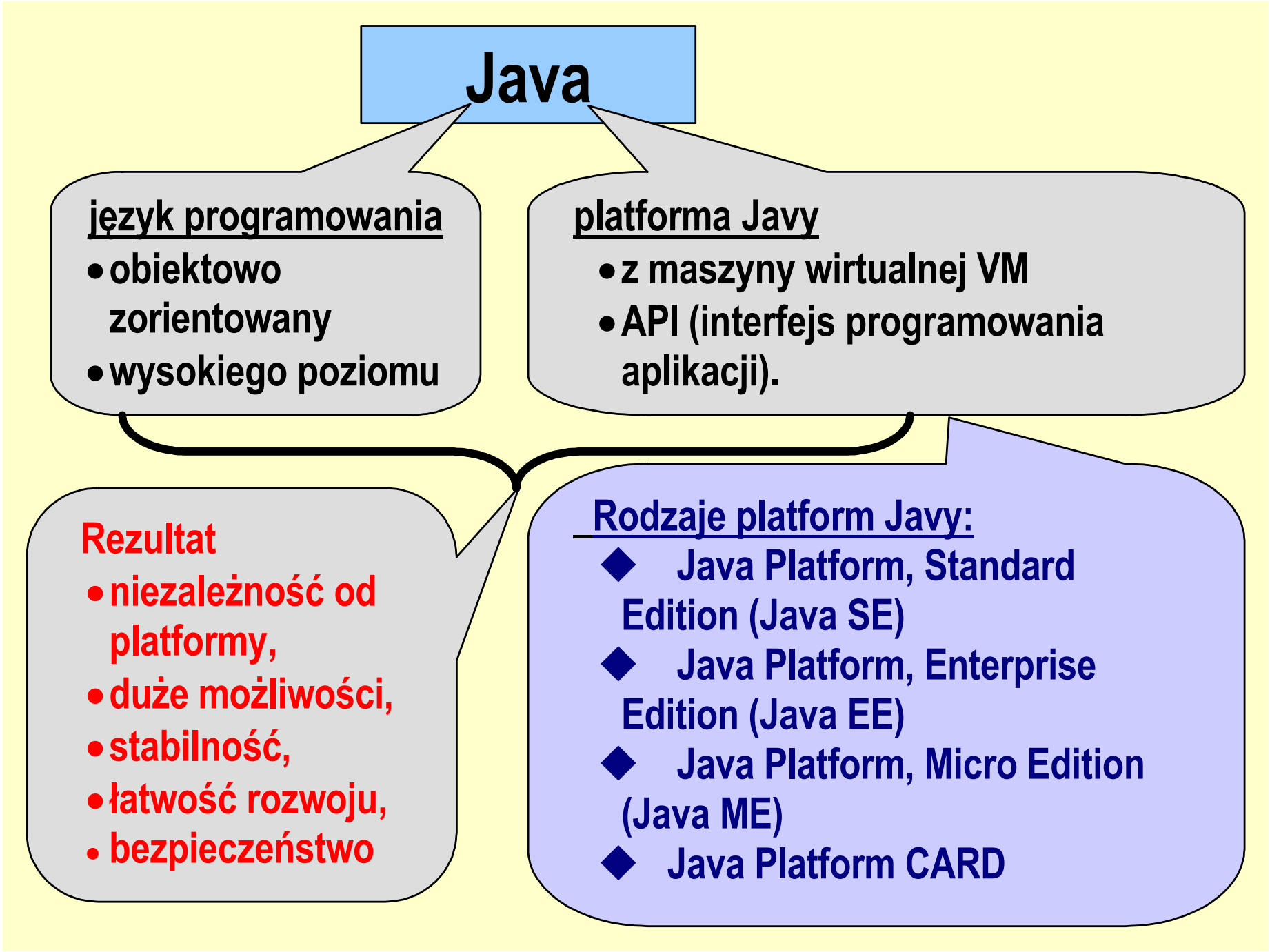
Cele laboratorium 2

Należy:

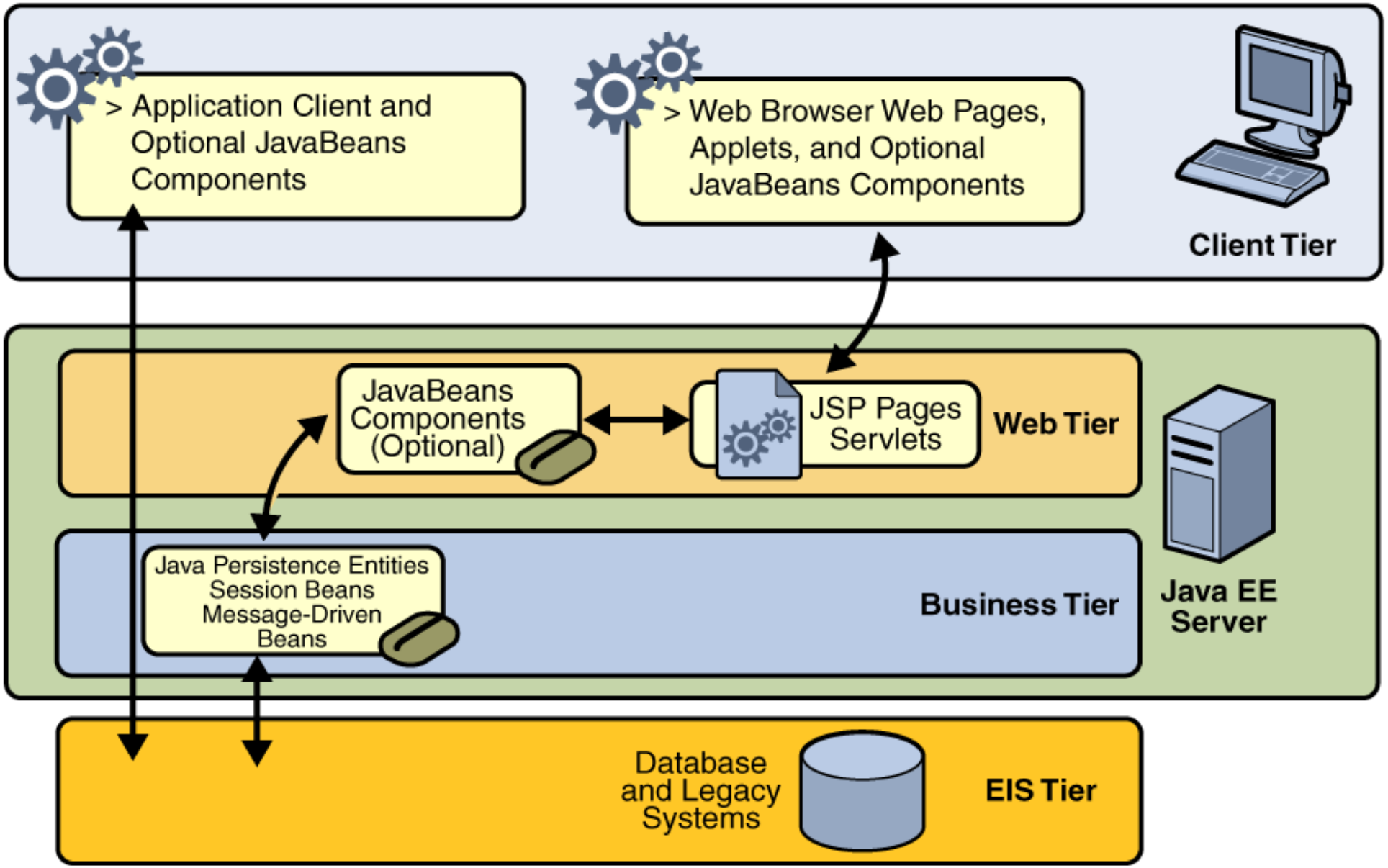
- wybrać projekt z podanej listy dostępnej za pomocą linku podanego w w laboratorium 1
- sformułować wymagania funkcjonalne i niefunkcjonalne dla wybranego projektu jako zadanie domowe. Zadanie domowe będzie stanowić podstawę do zaprojektowania przypadków użycia na kolejnych laboratorium.
- Wykonać projekt UML i wykonać prosty program stanowiący realizację 1-go etapu wykonania wybranego projektu. Instrukcja zawiera przykłady 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 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.
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)

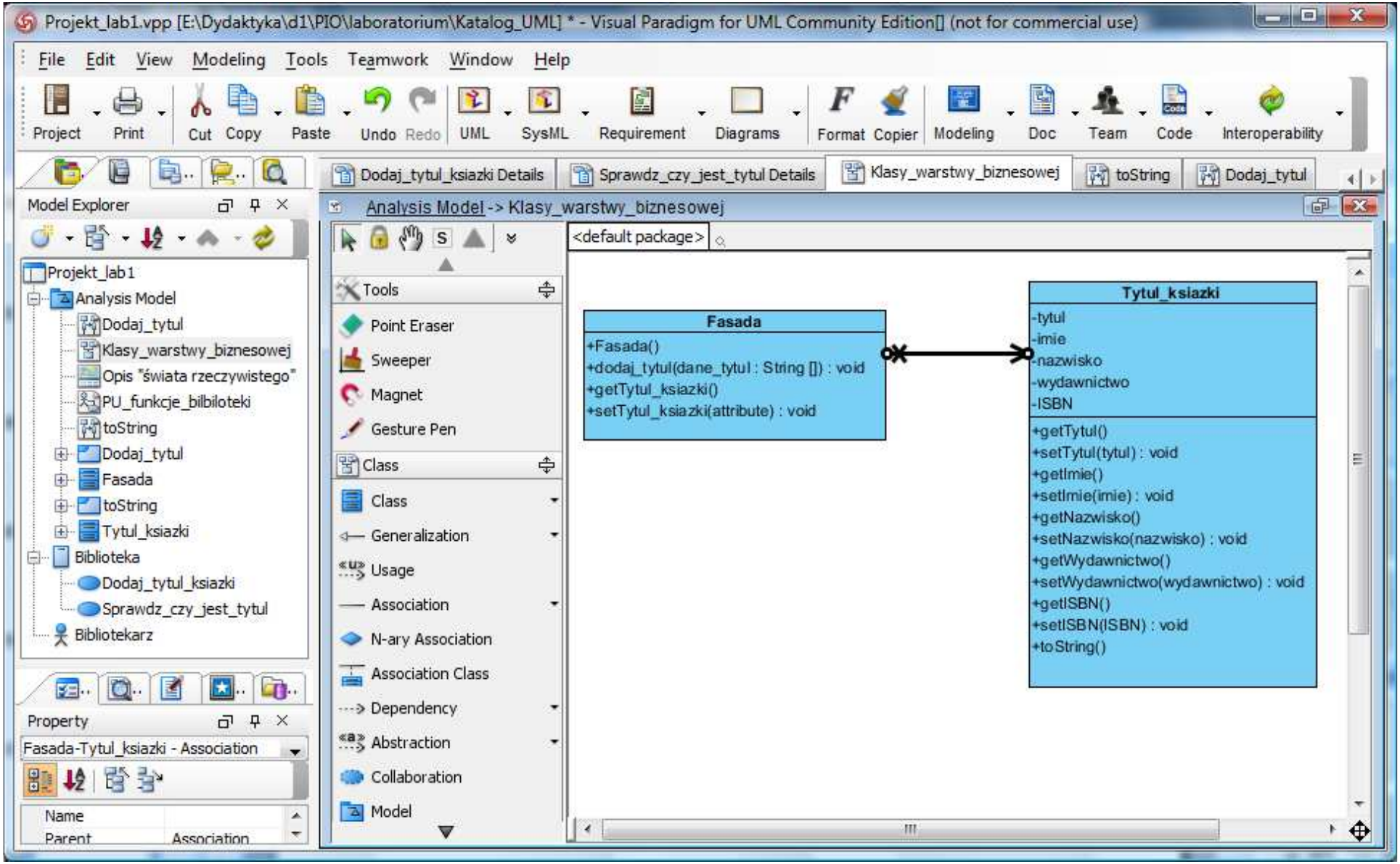


Materiał pomocniczy z lab2

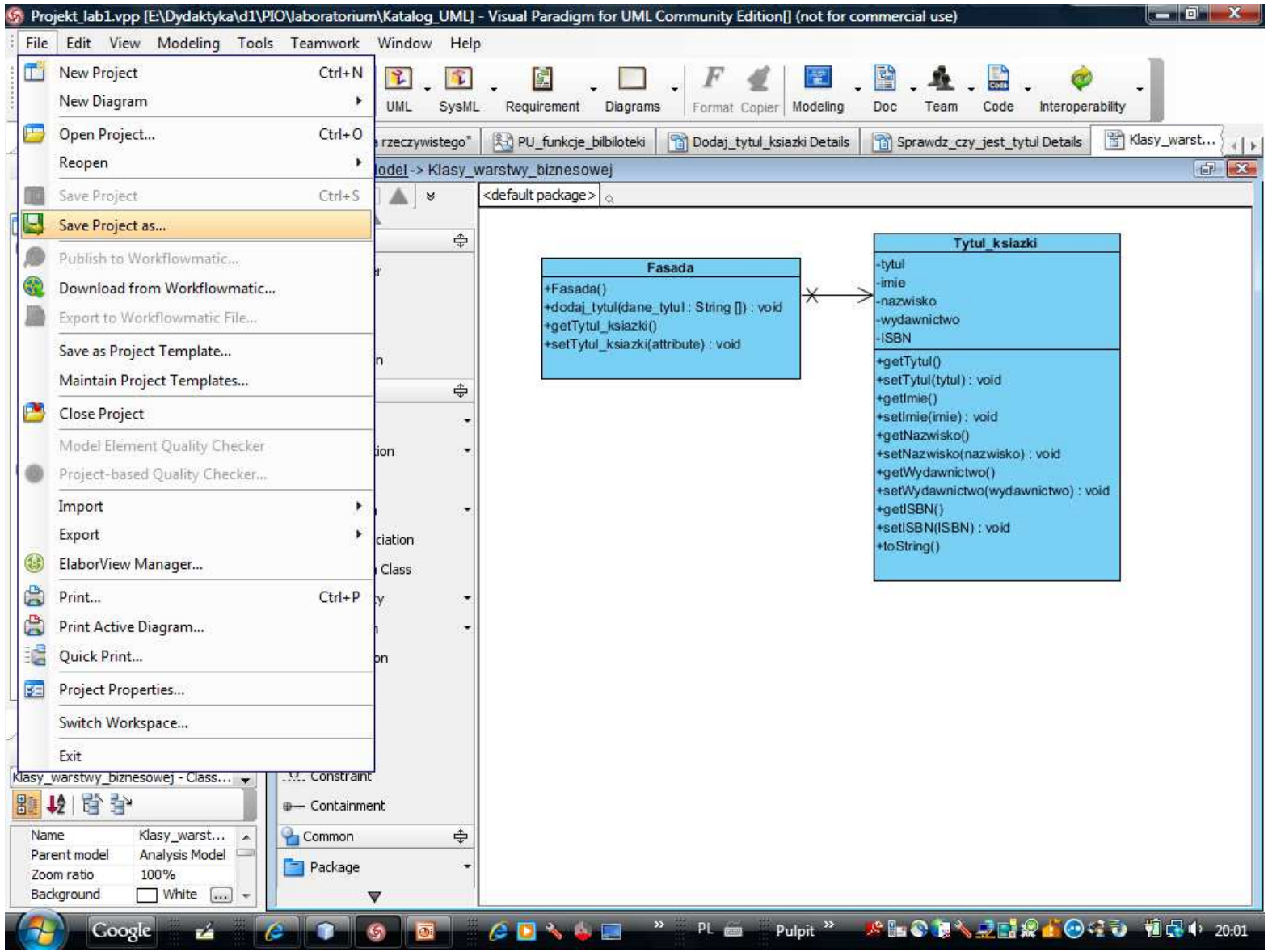
Relacja jeden do wiele

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

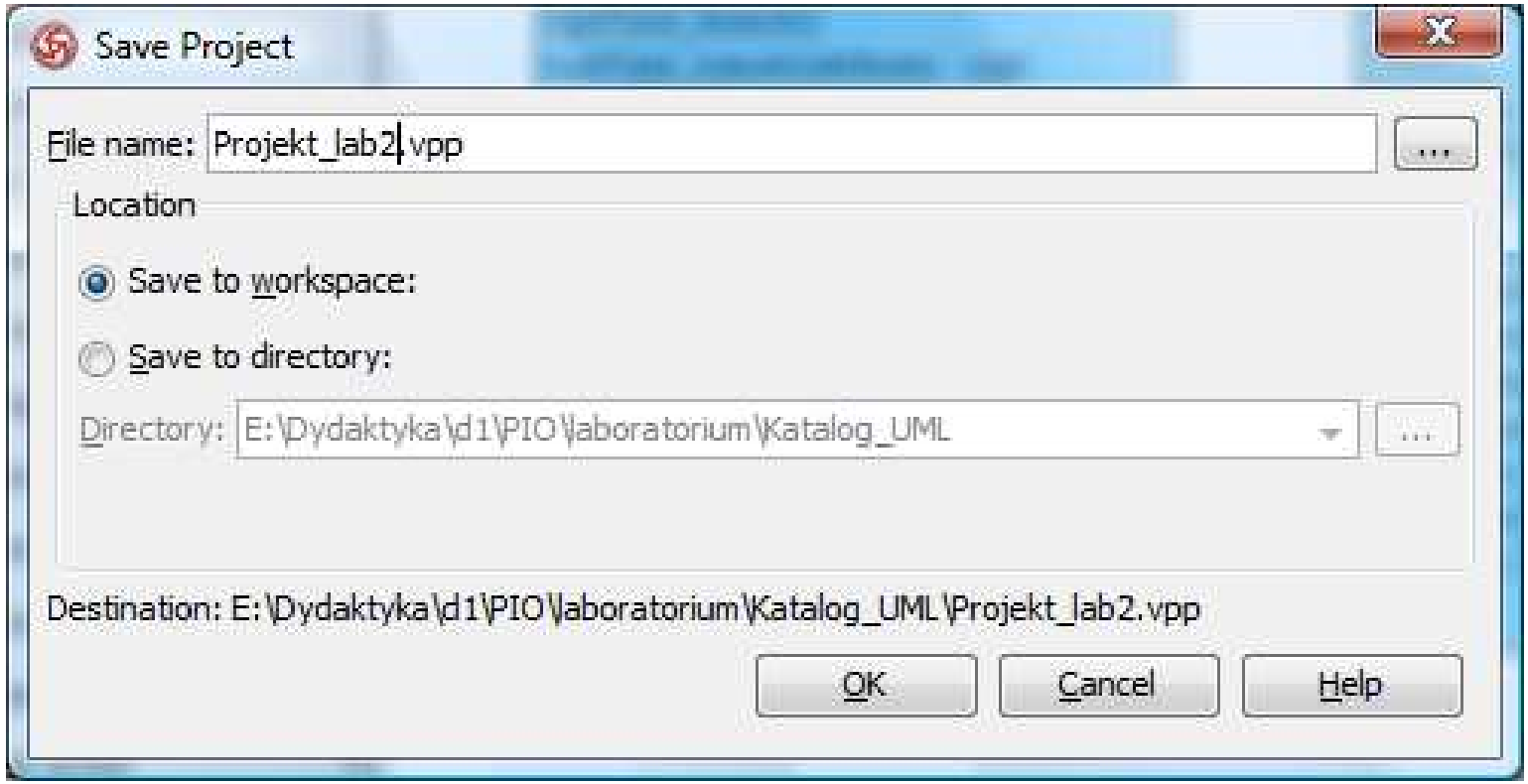
1.1. Należy kontynuować projekt z lab1



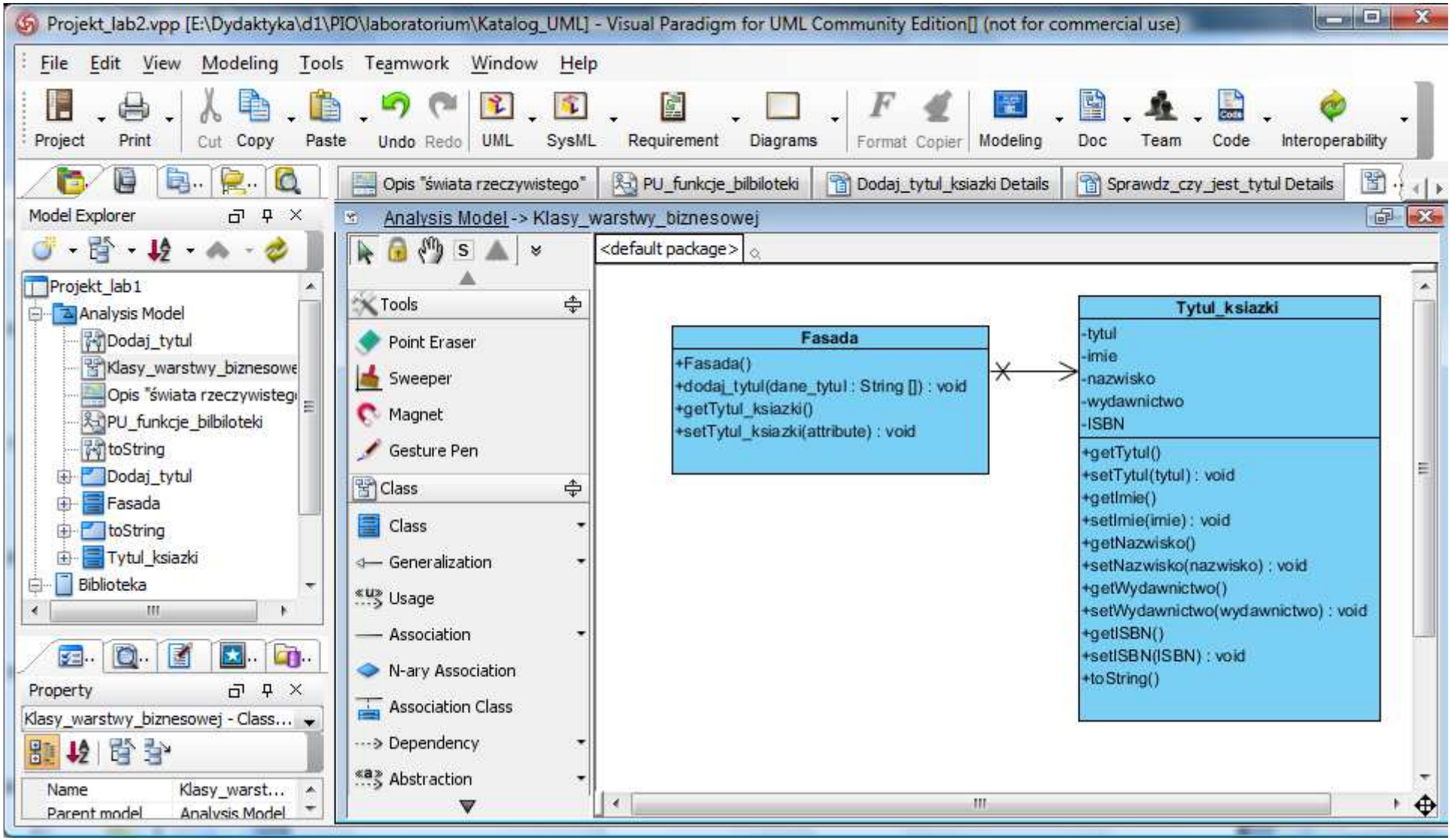
1.2. Należy utworzyć kopię projektu Projekt_lab1 za pomocą *File/Save Project as*



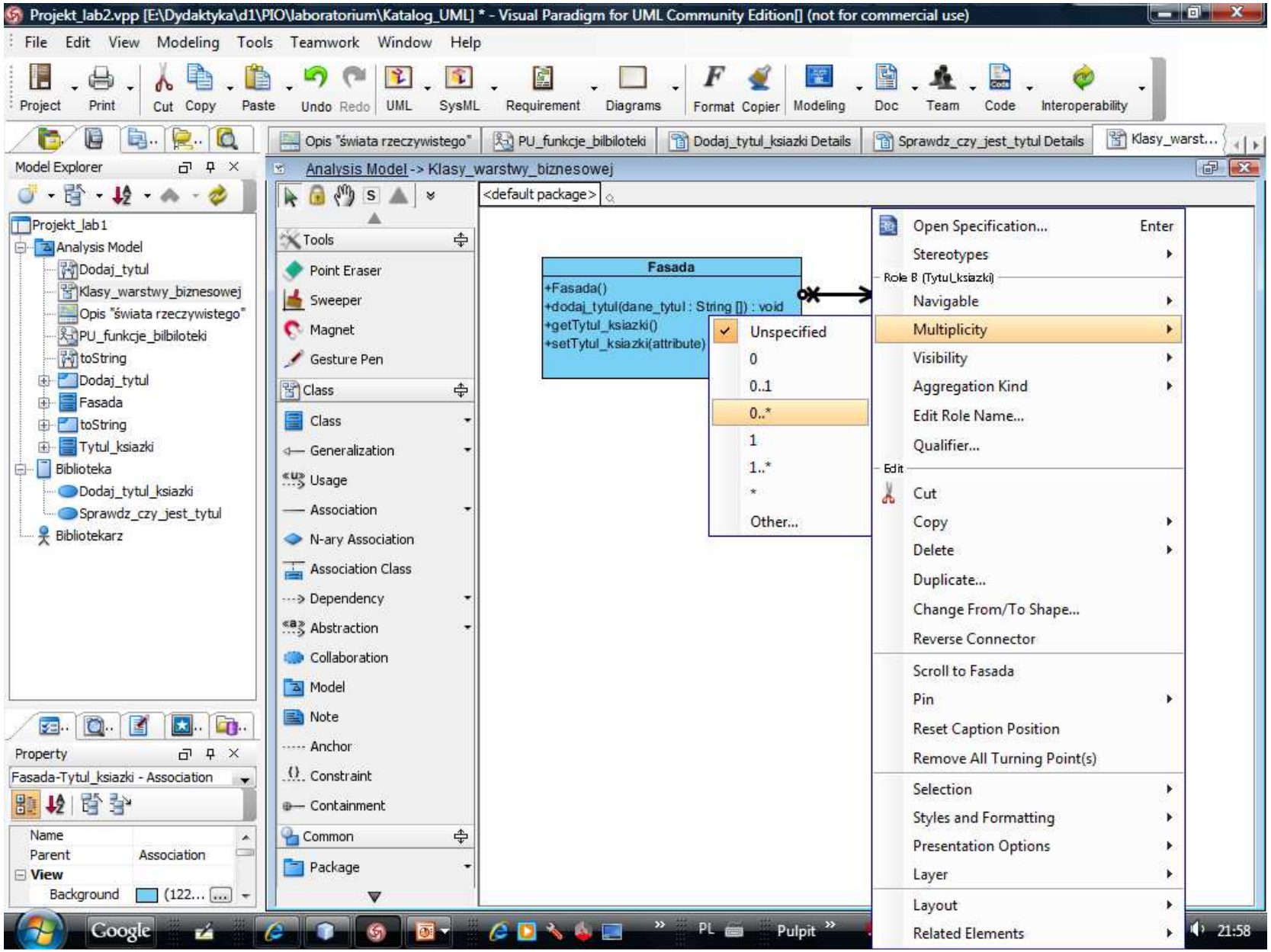
1.3. Utworzenie kopii o nazwie Projekt_lab2



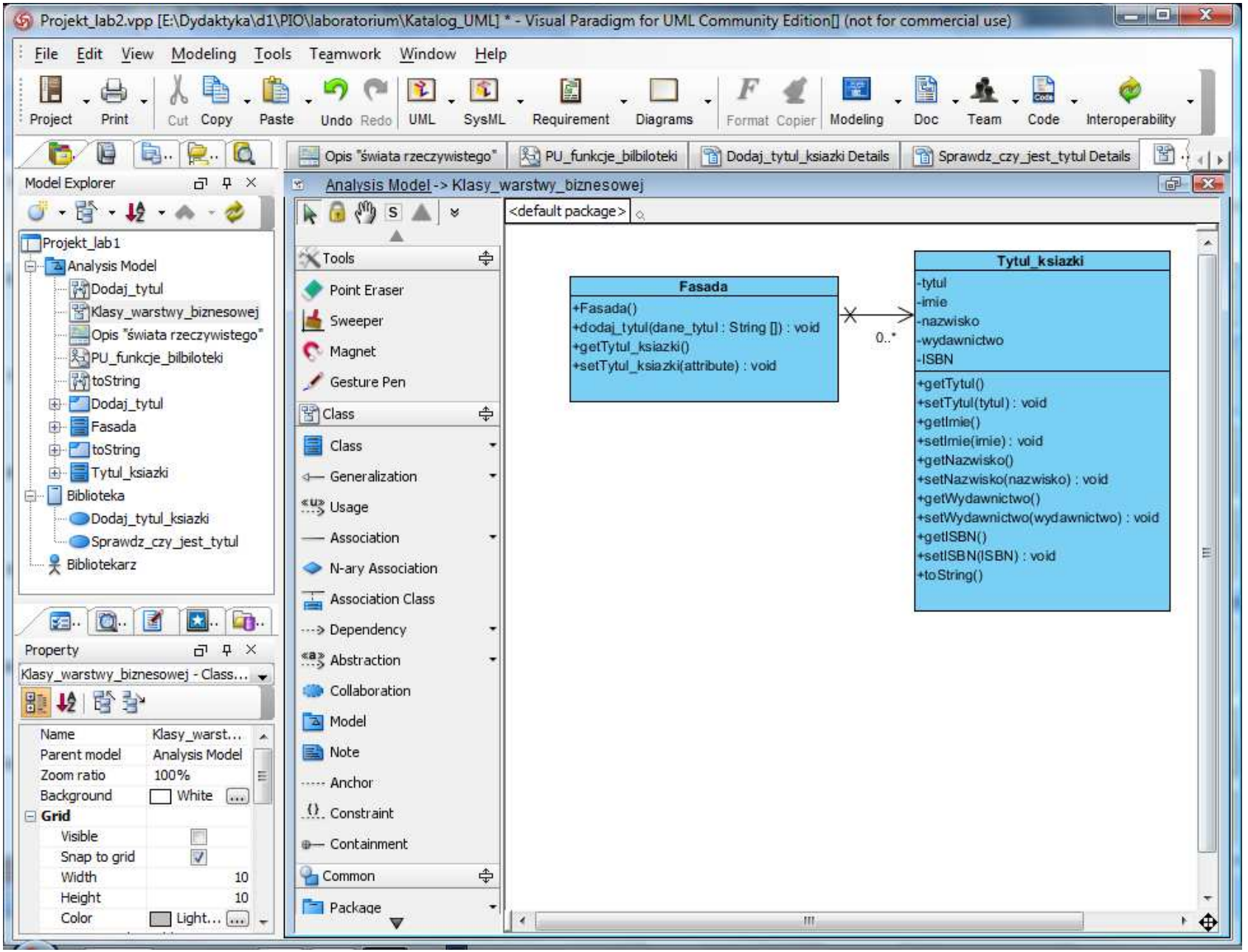
2.1. Zmiana relacji 1 : 1 między klasami **Fasada** i **Tytul_ksiazki** na 1 : 0..*



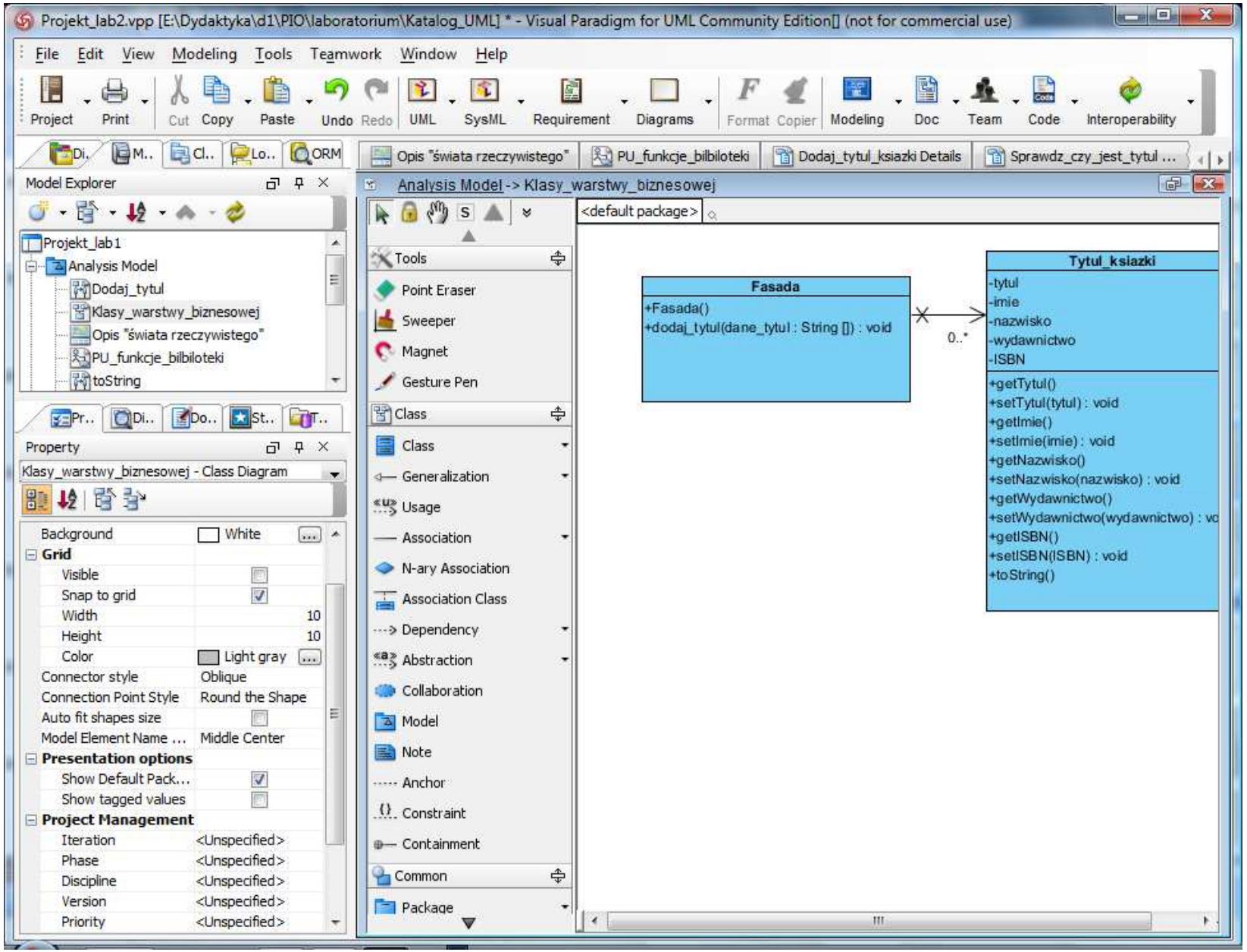
2.2. Relacja jeden do wiele – zmiana Multiplicity w relacji typu *Association* dla klasy Tytul_ksiazki na 1:0..* (jeden do wiele)



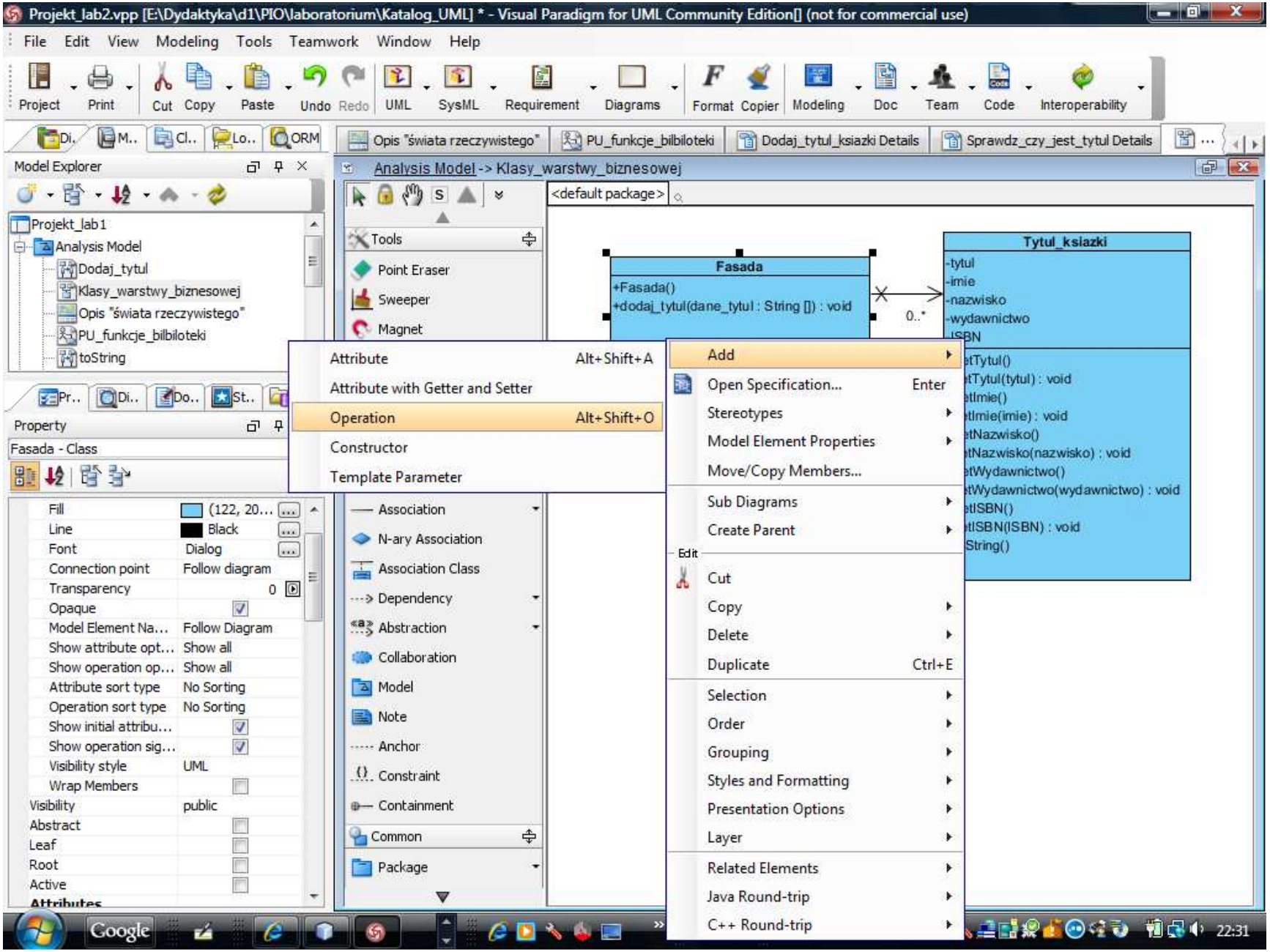
2.3. Rezultat zmian



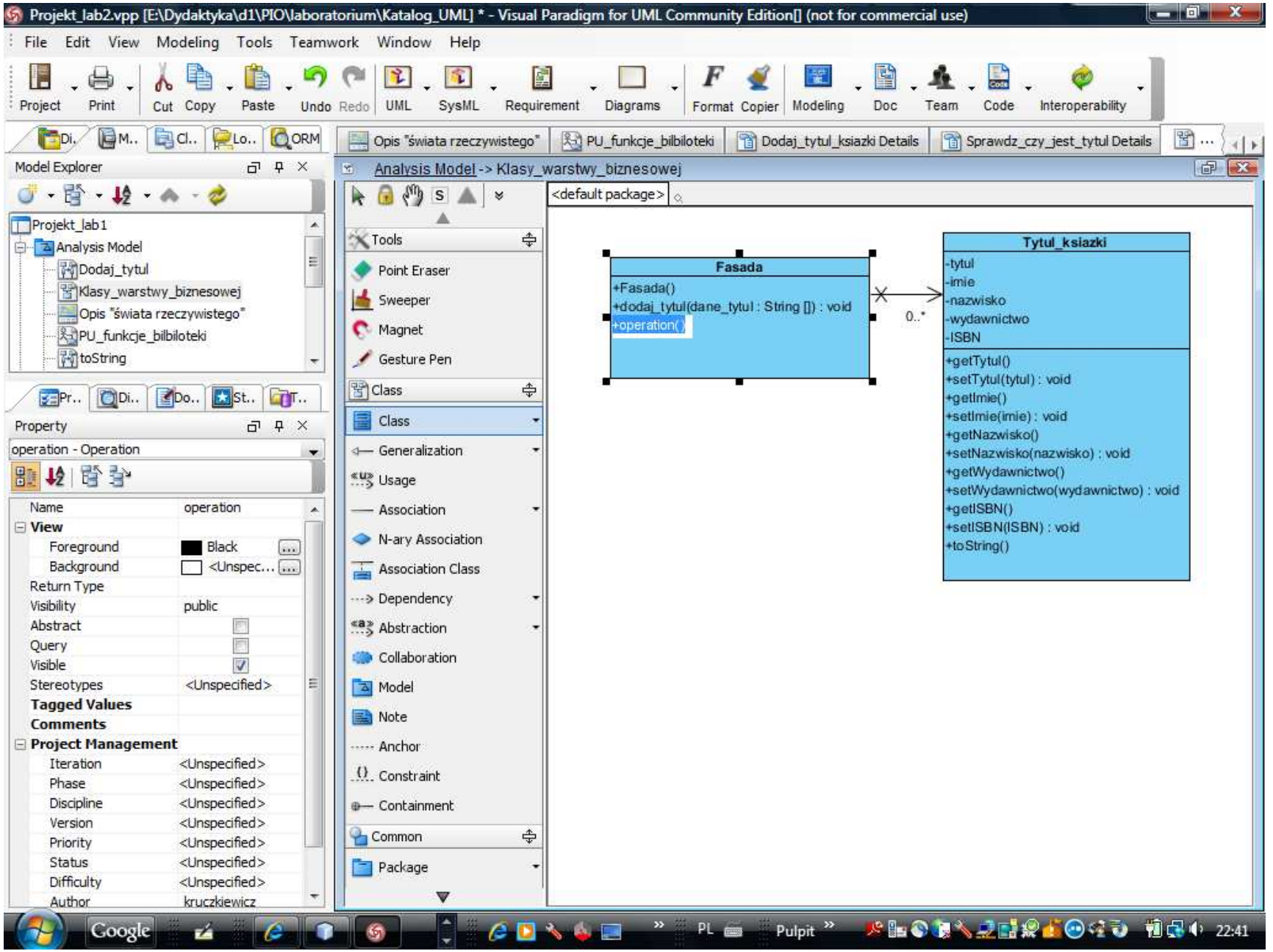
2.4. Zmiana metod typu set i get w klasie **Fasada** ze względu na zmianę typu atrybutu implementującego relację 1:0..* na kolekcje typu np.. ArrayList



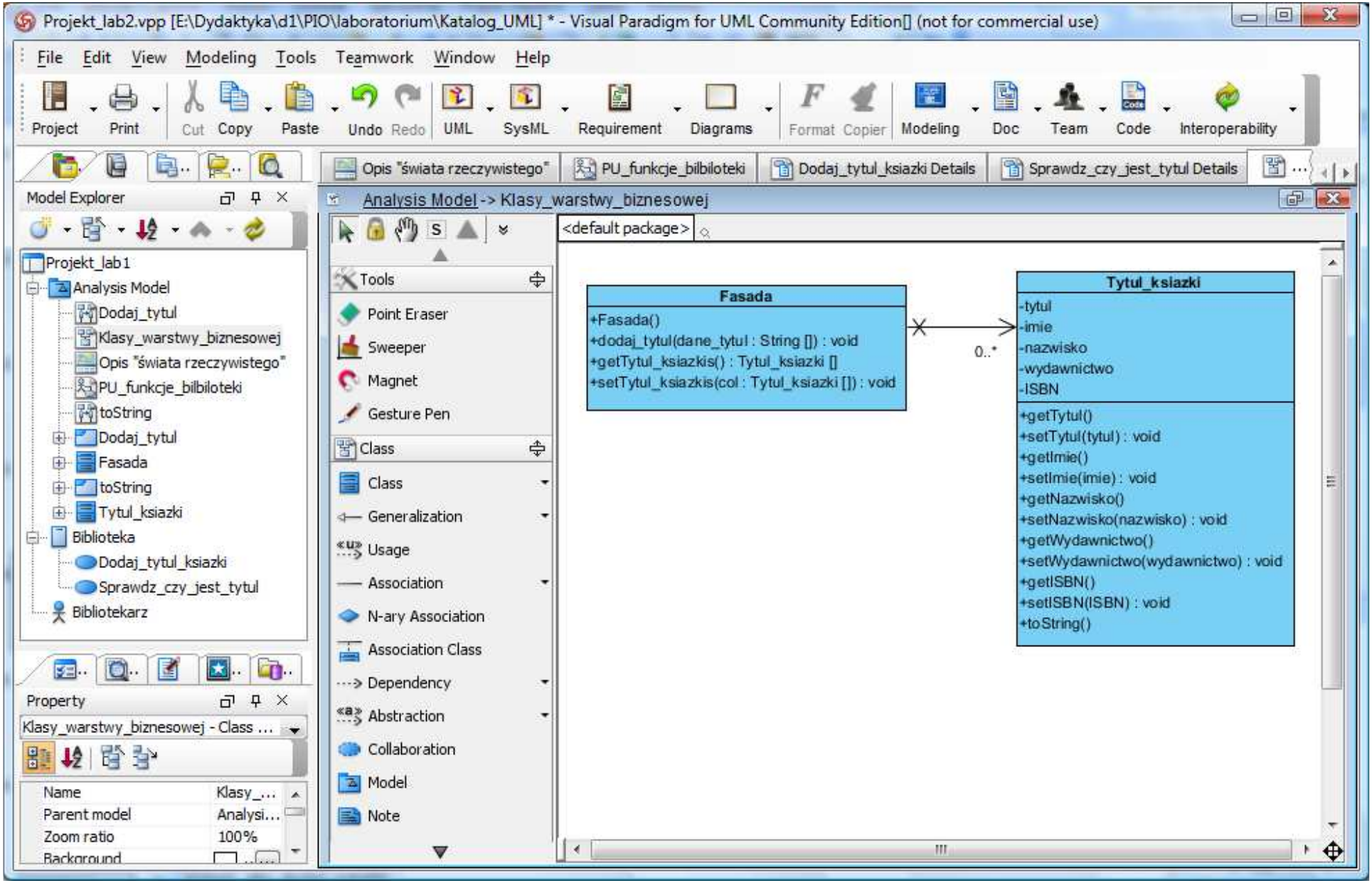
2.5.Dodawanie nowych metod typu set i get do klasy **Fasada**



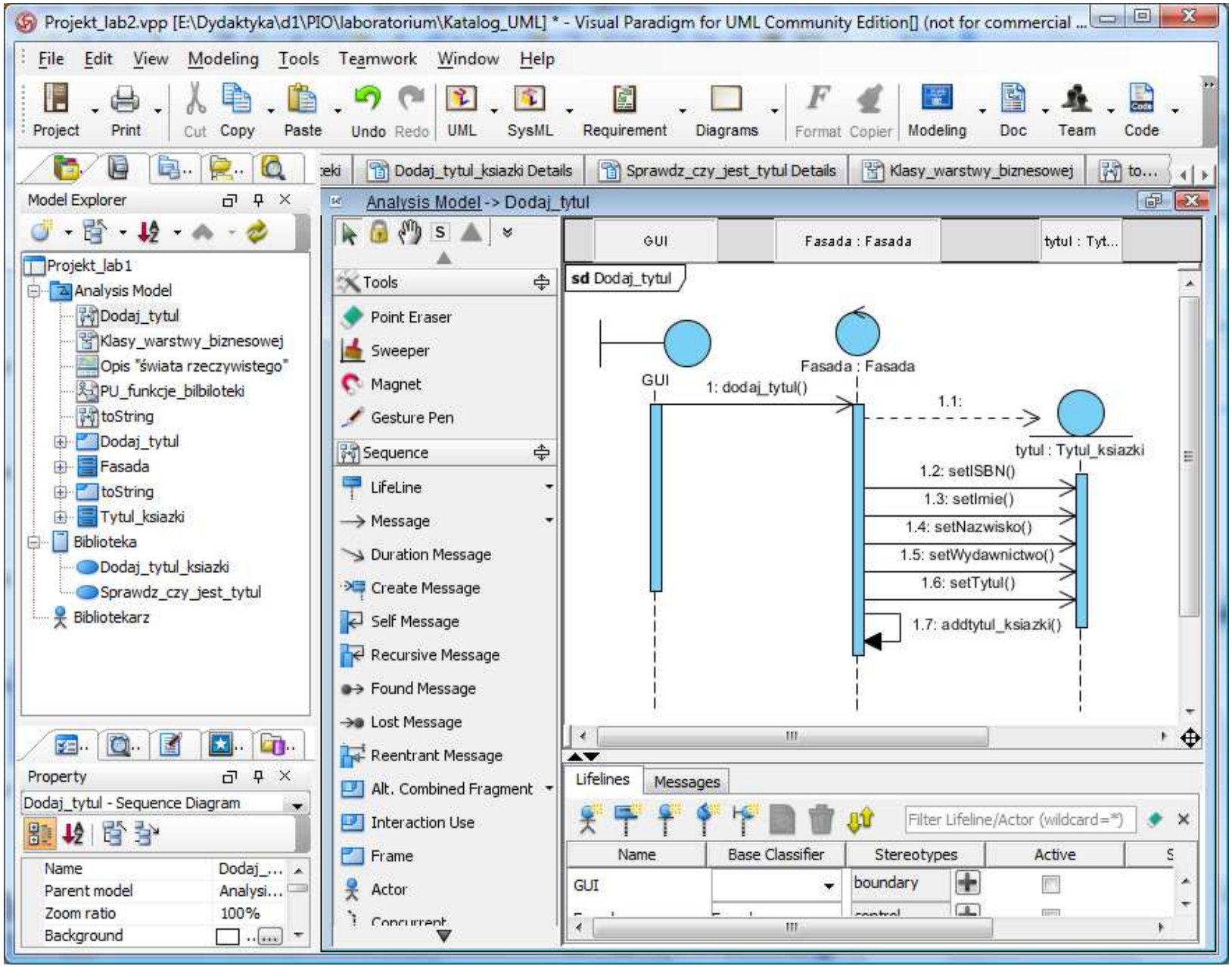
2.6.Dodawanie nowych metod typu set i get do klasy **Fasada**



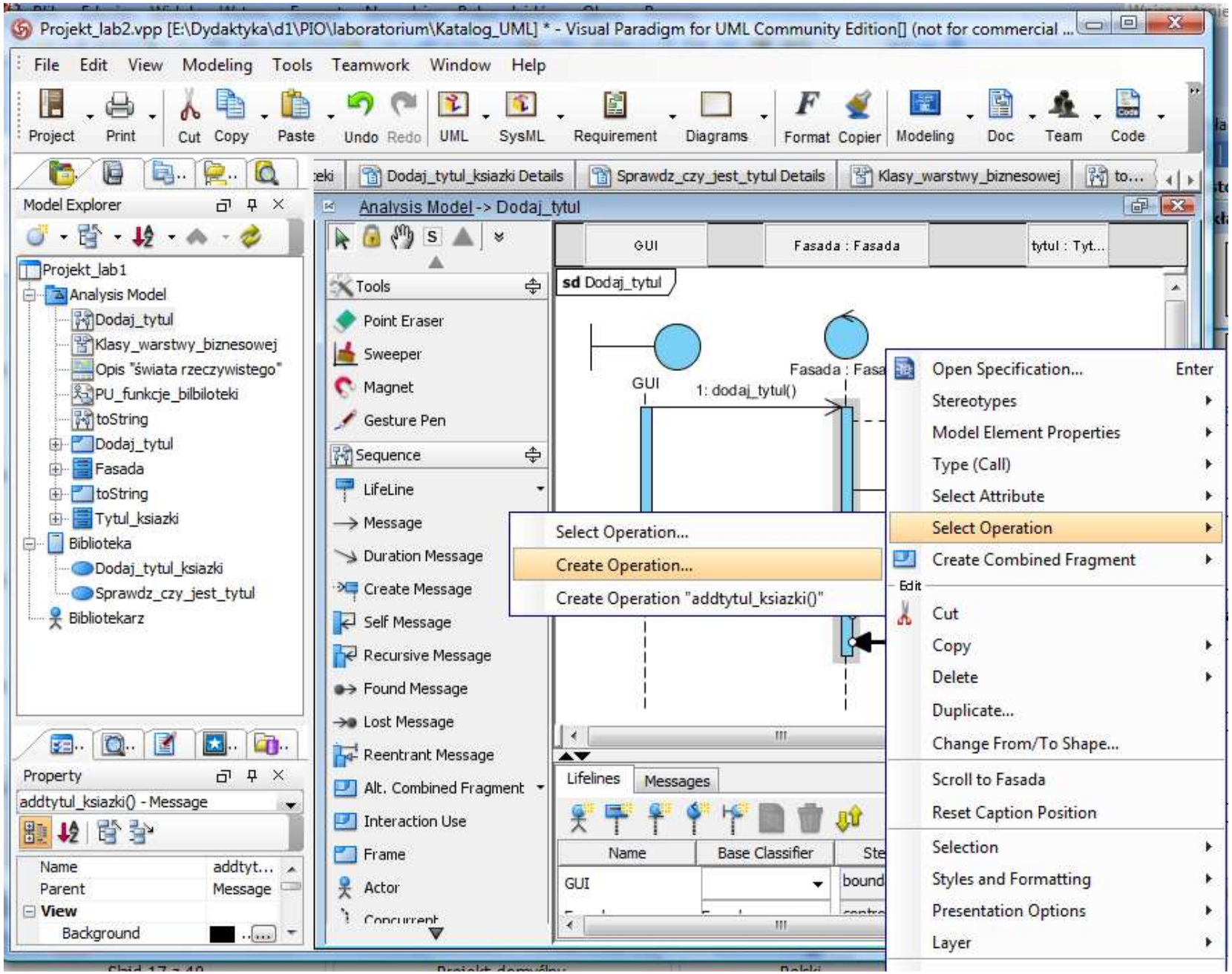
2.7.Dodano nowe metody typu set i get do klasy **Fasada**



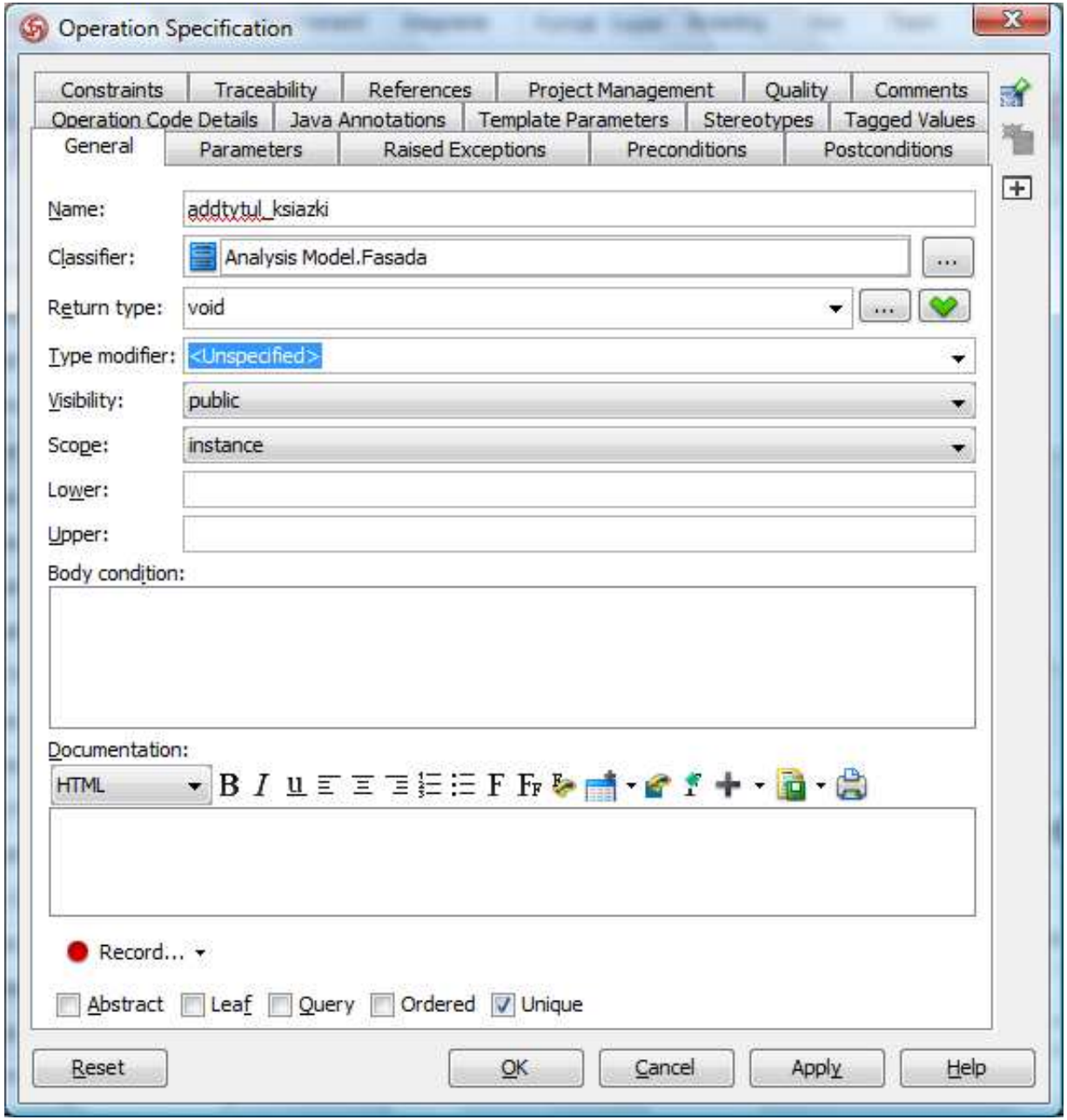
3.1. Dodanie do metody dodaj_tytul nowej metody addTytul_ksiazki, która realizuje sprawdzenie unikalności numeu ISBN wśród innych tytułów książek



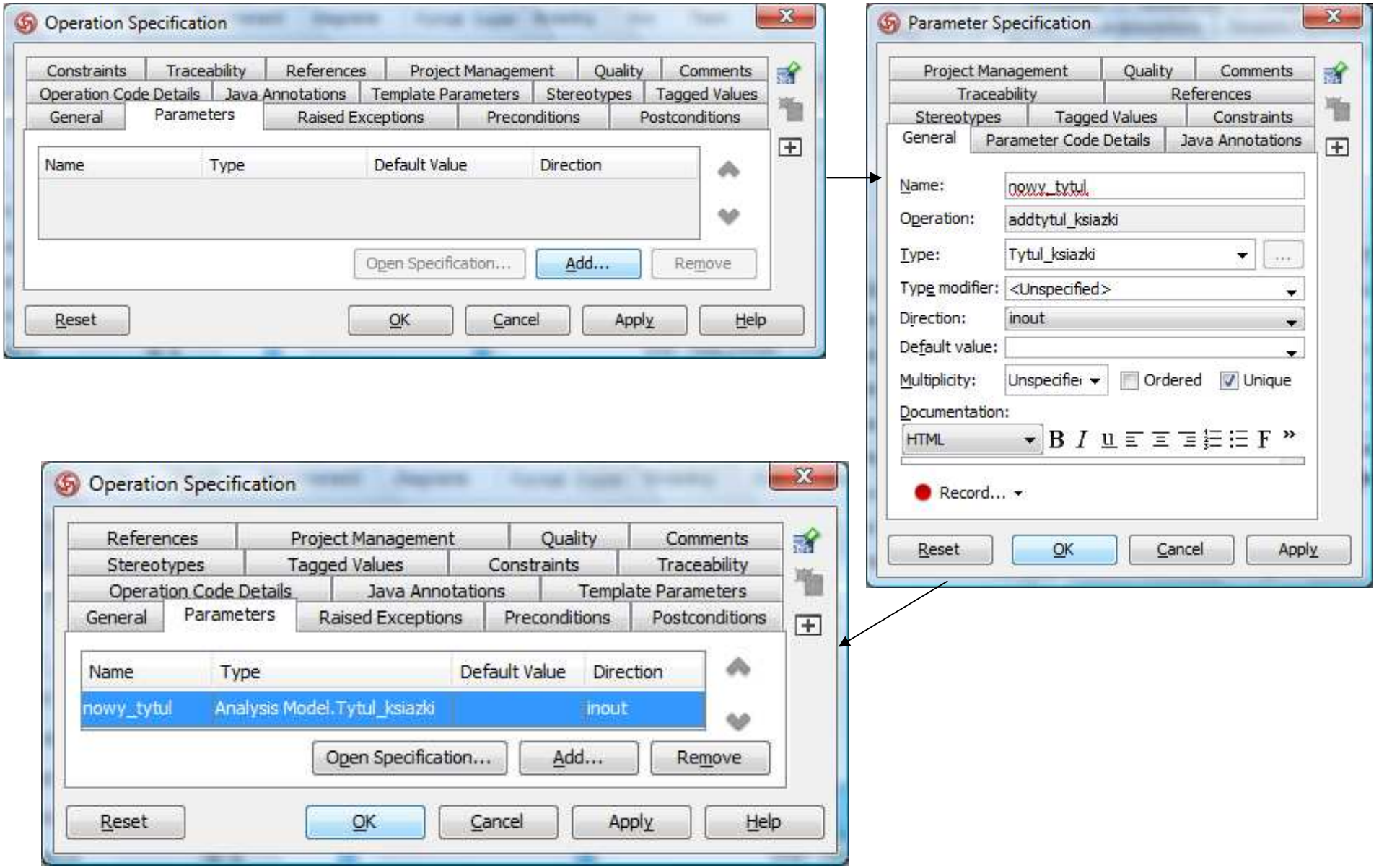
3.2. Tworzenie nowej metody **addtytul_ksiazki** w klasie **Fasada**



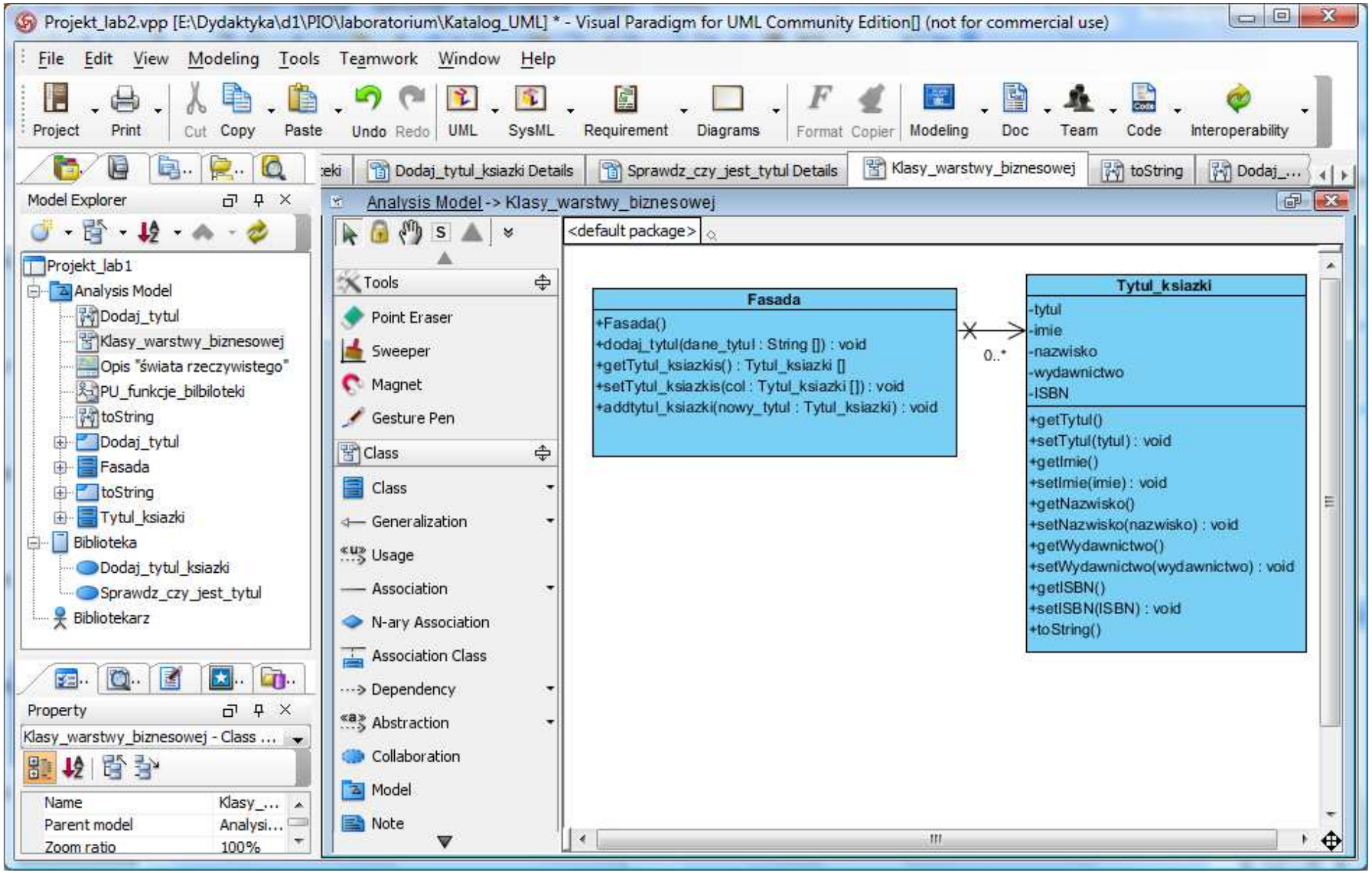
3.3. Tworzenie nowej metody **addtytul_ksiazki** w klasie **Fasada**



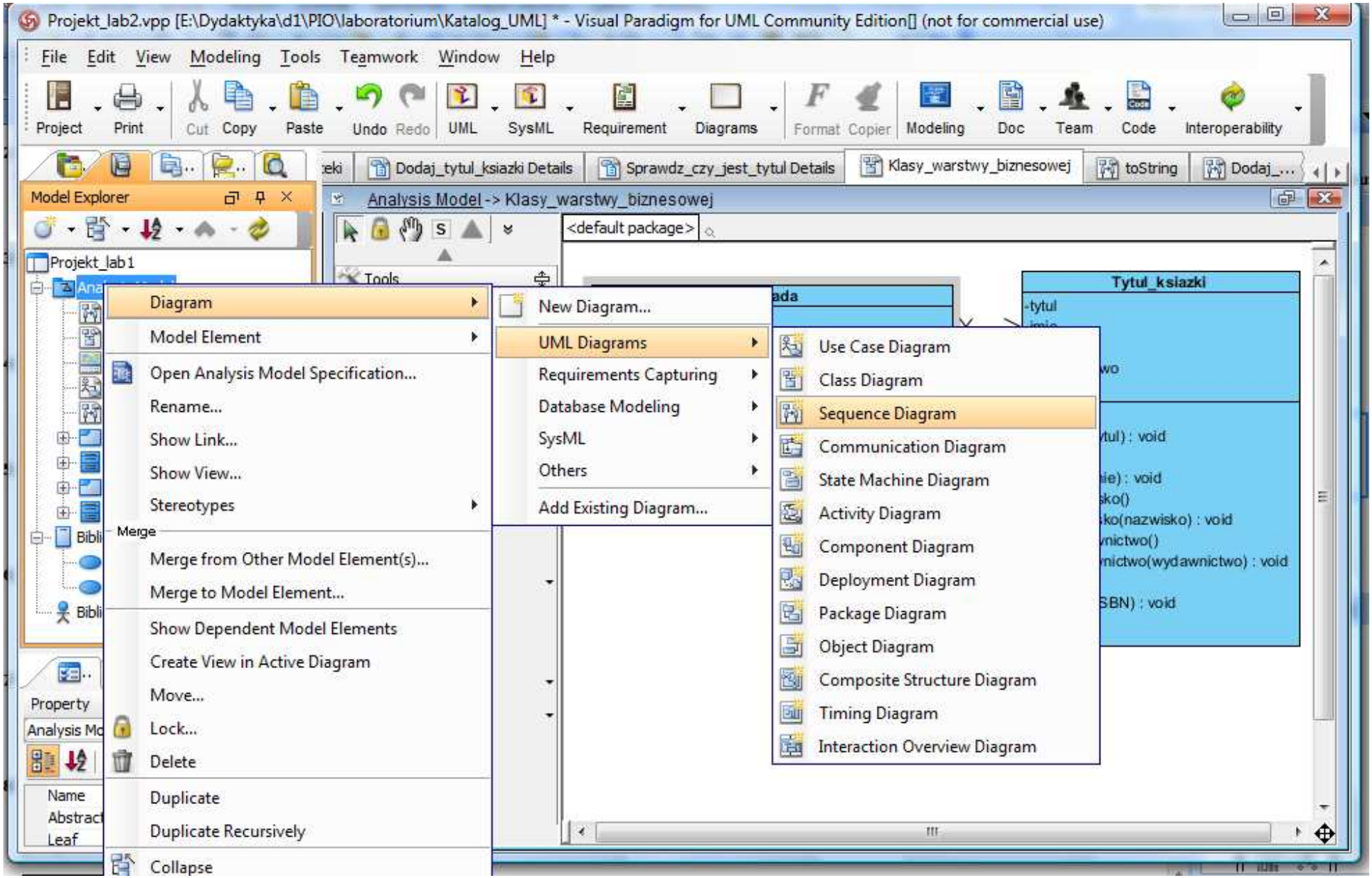
3.4. Definiowanie parametru przekazywanego do metody **addtytul_ksiazki** typu **Tytul_ksiazki**



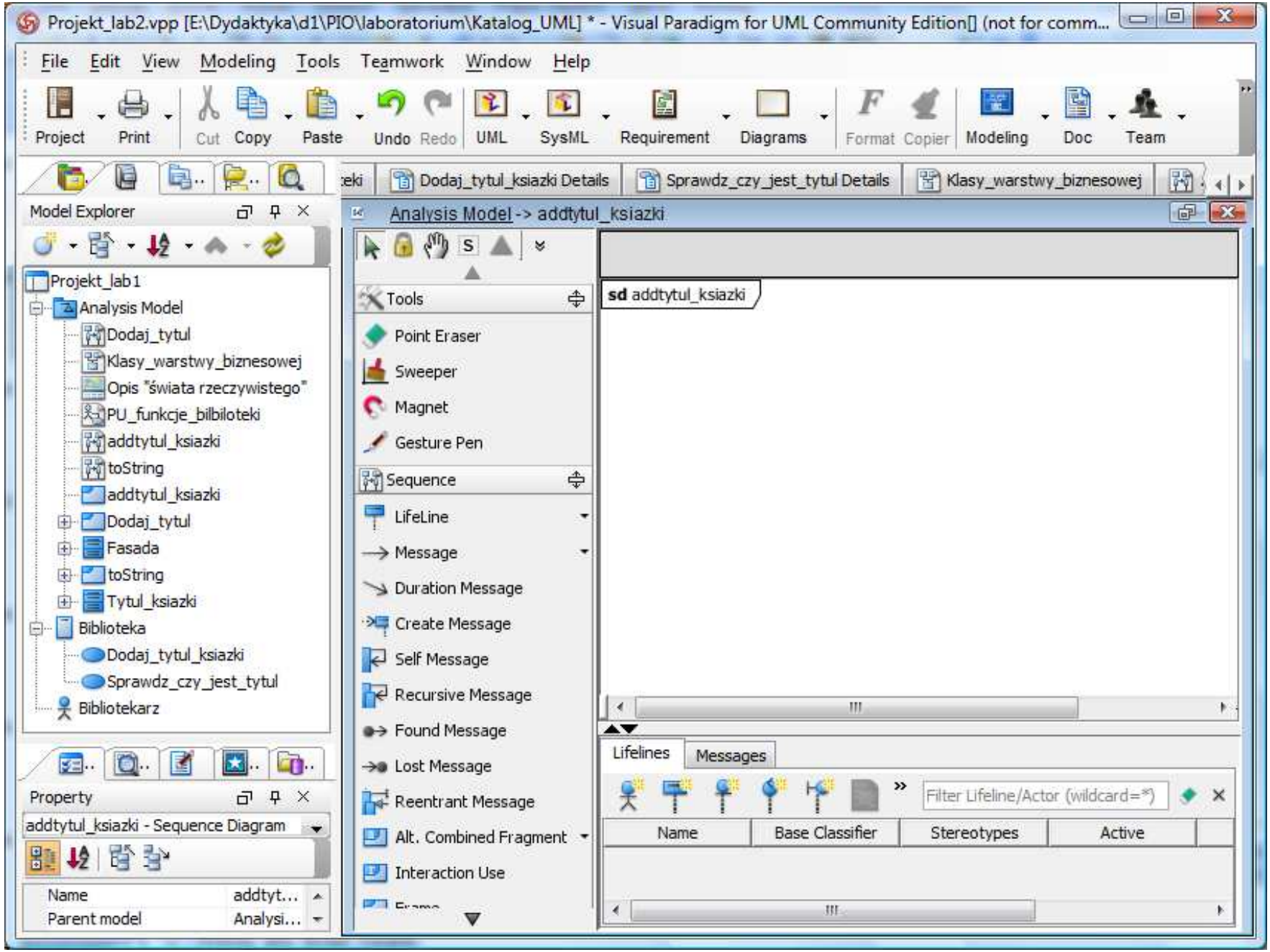
3.5. Widok diagramu klas po dodaniu metody **addtytul_ksiazki** do klasy **Fasada,**



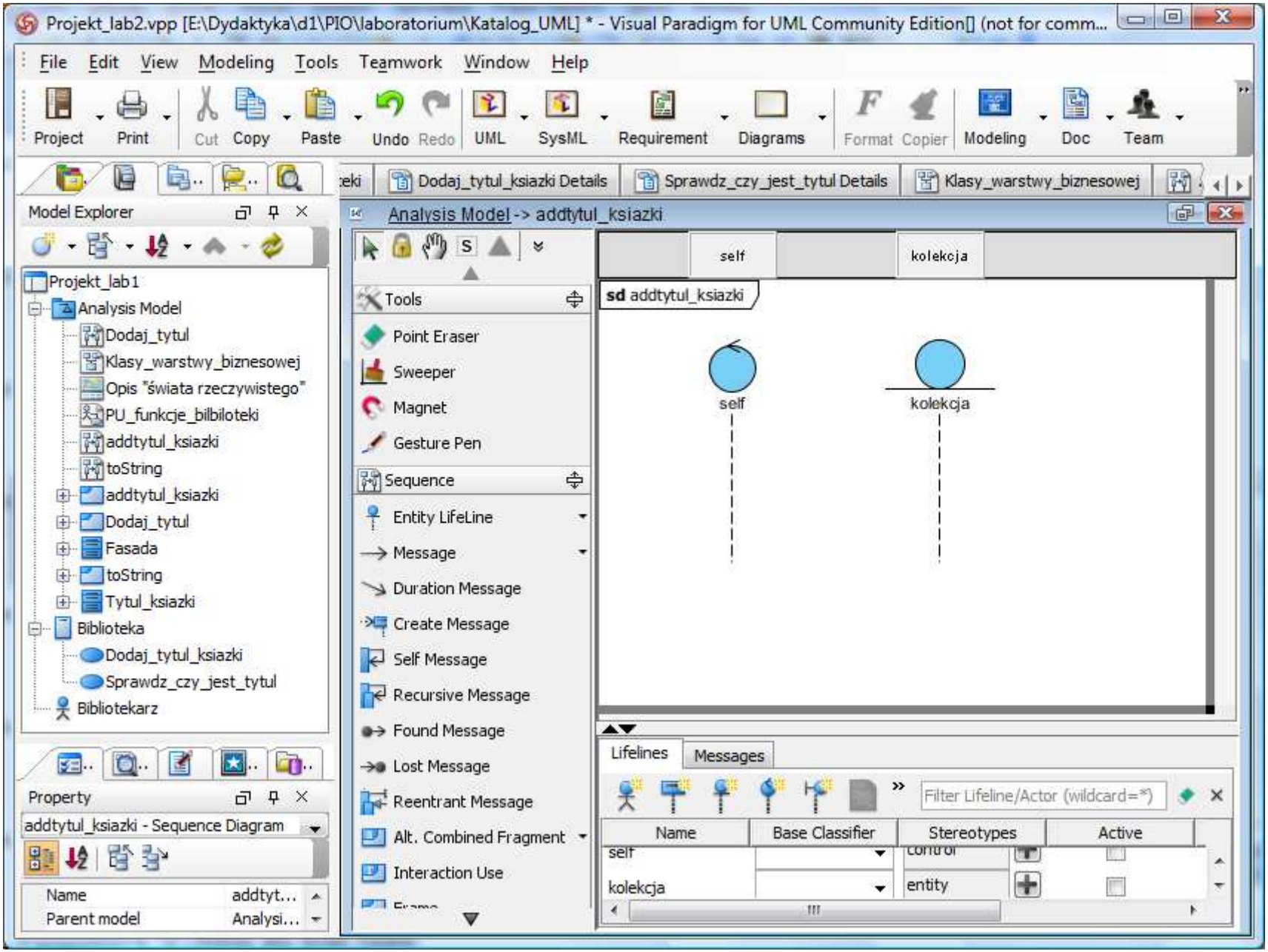
4.1. Dodanie nowego diagramu sekwencji reprezentujacego metodę **addtytul_ksiazki** w klasie **Fasada**



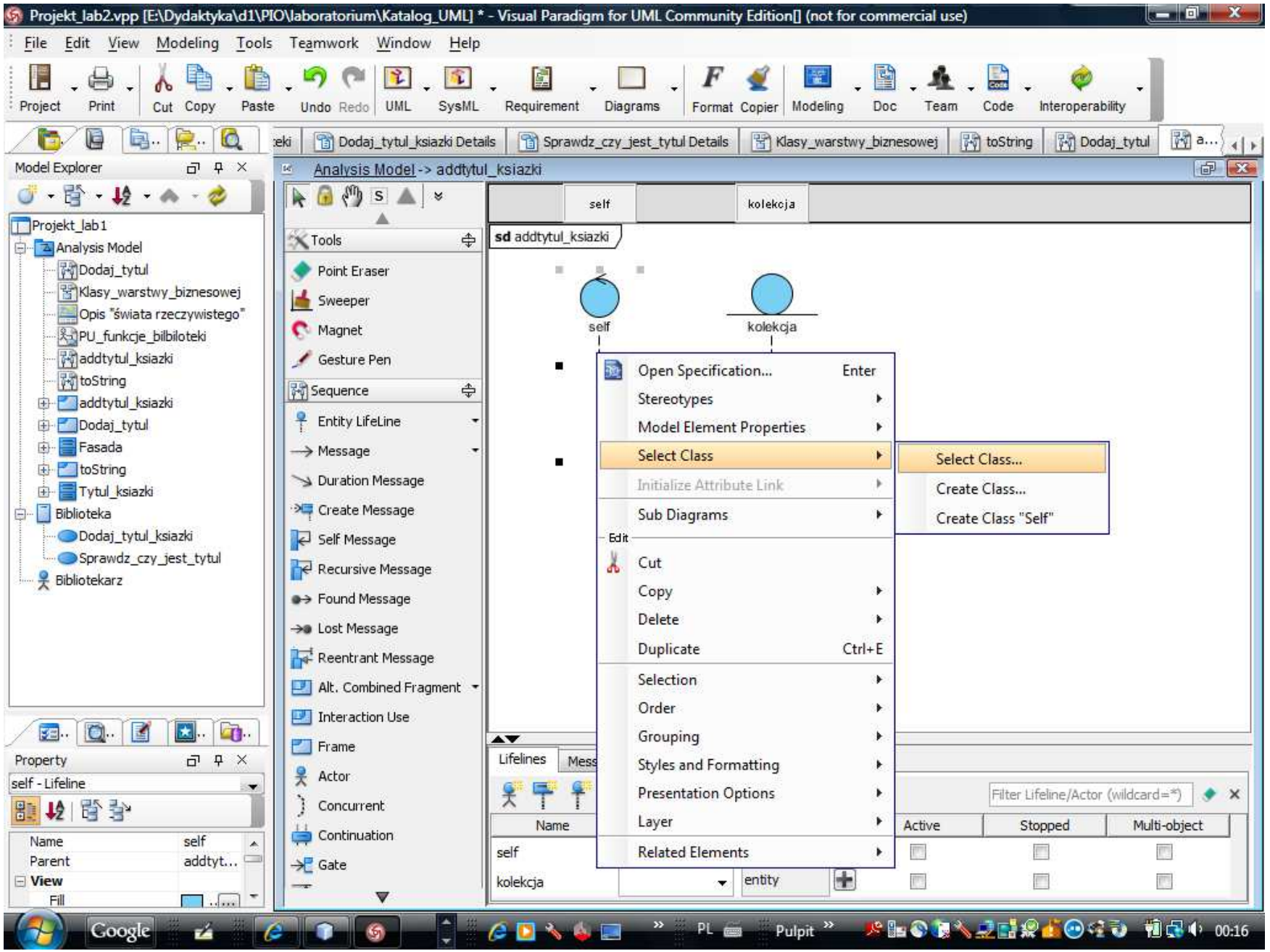
4.2.Nadanie nazwy diagramowi – addtytul_ksiazki



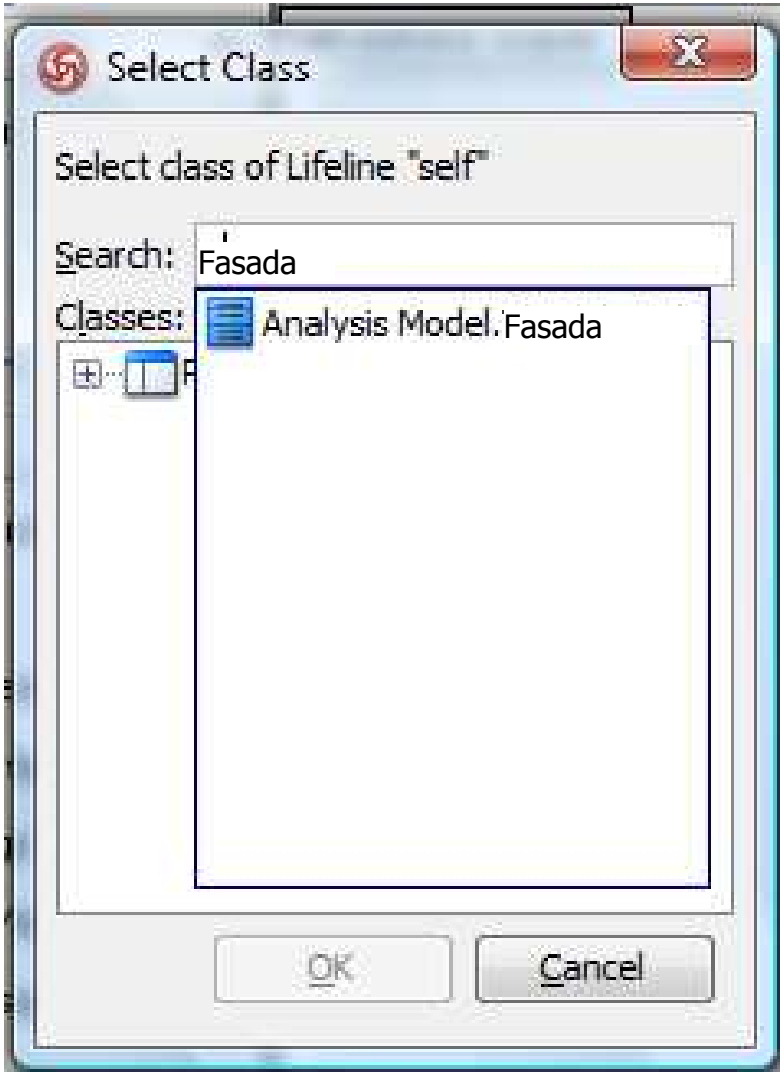
4.3. Dodanie dwóch linii życia typu *Entity Lifeline*

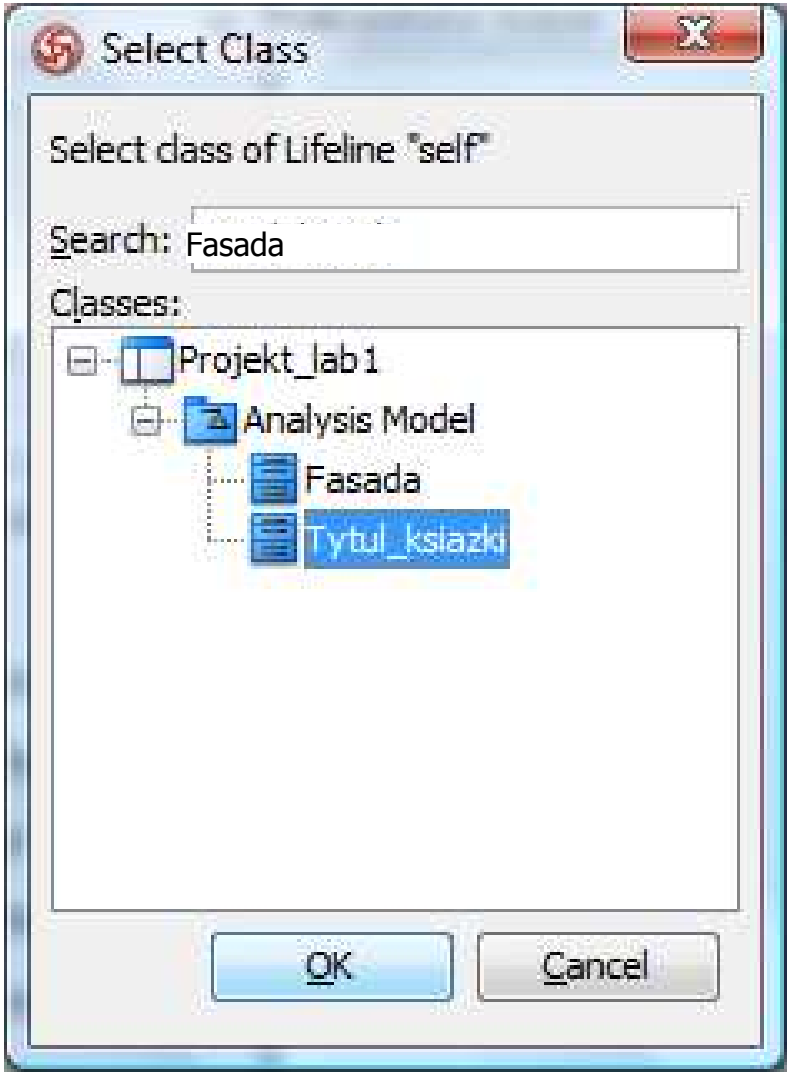


4.4. Powiązanie linii życia self z klasą **Fasada**

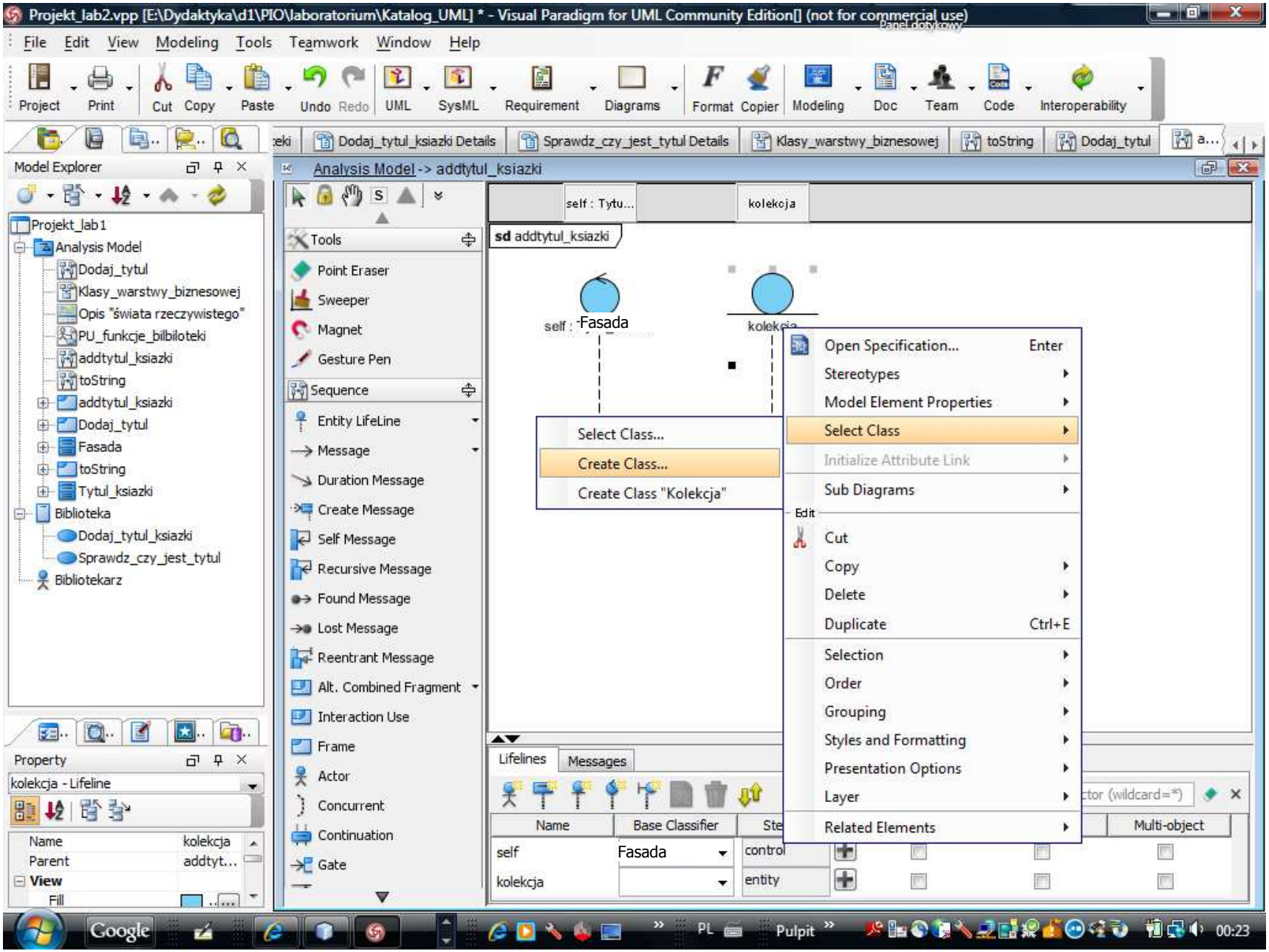


4.5. Dokonanie selekcji klasy **Fasada**

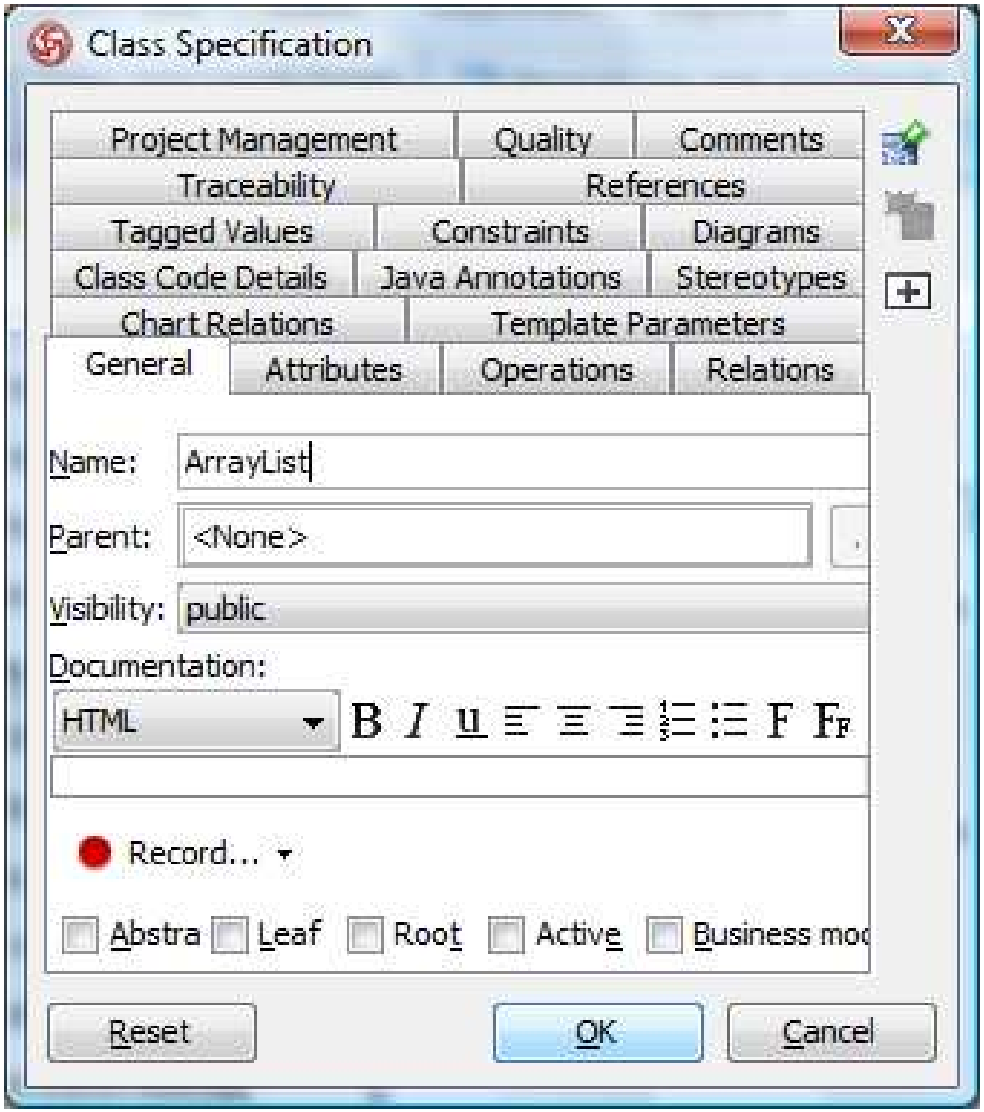




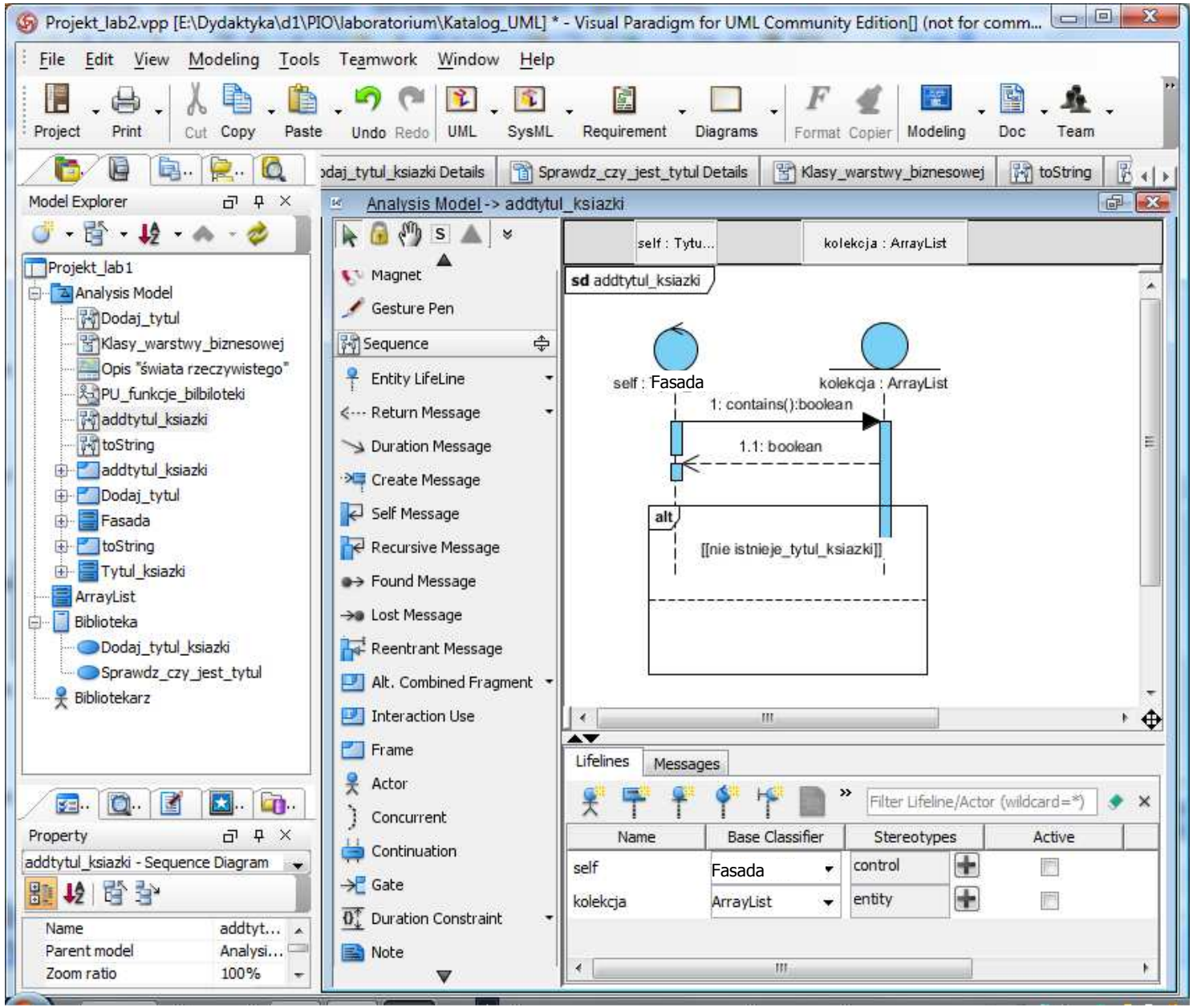
4.6. Powiązanie linii życia kolekcja z nową klasą pomocniczą **ArrayList** (odpowiednik klasy Javy)



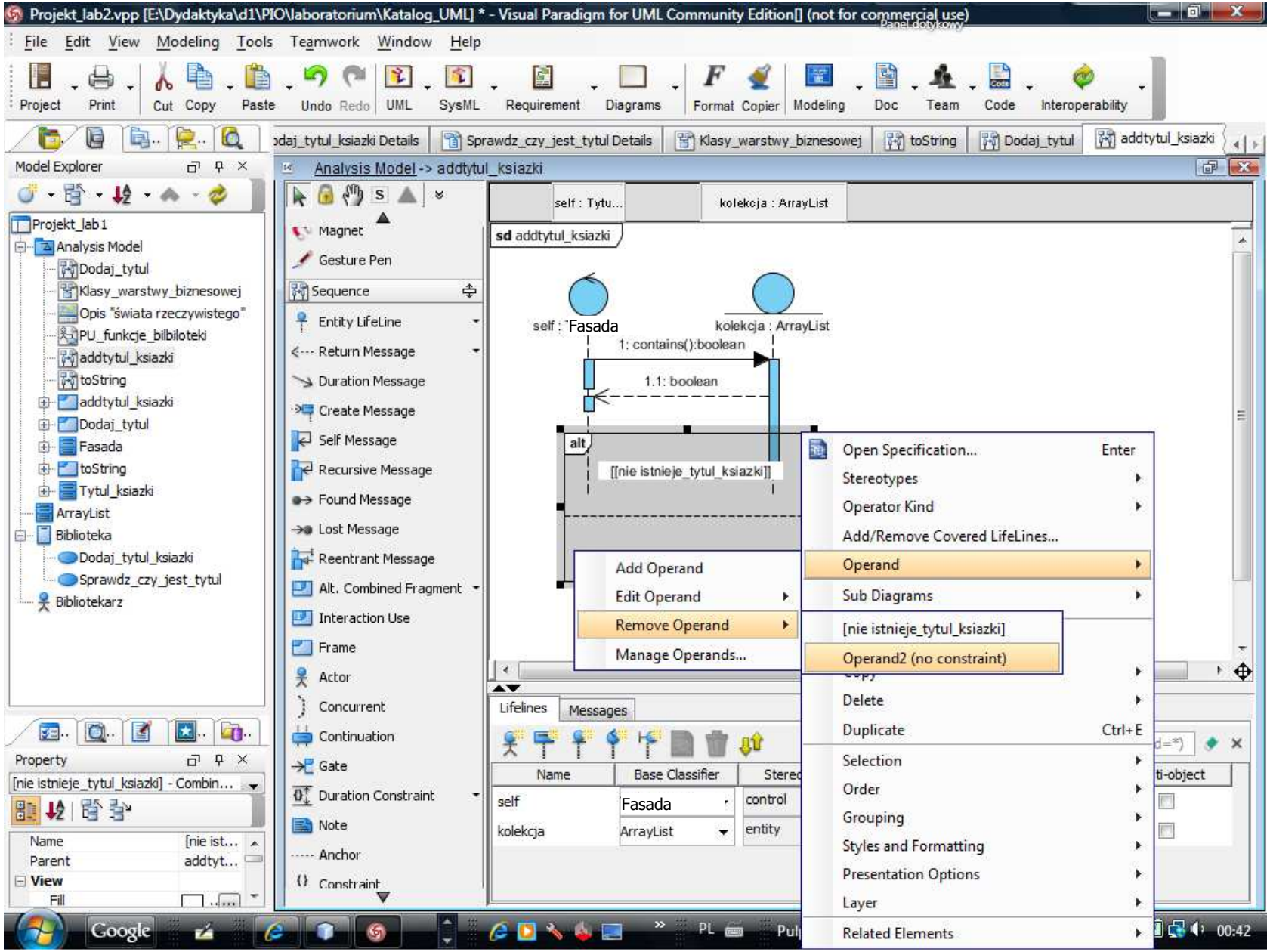
4.7. Utworzenie klasy pomocniczej typu **ArrayList**



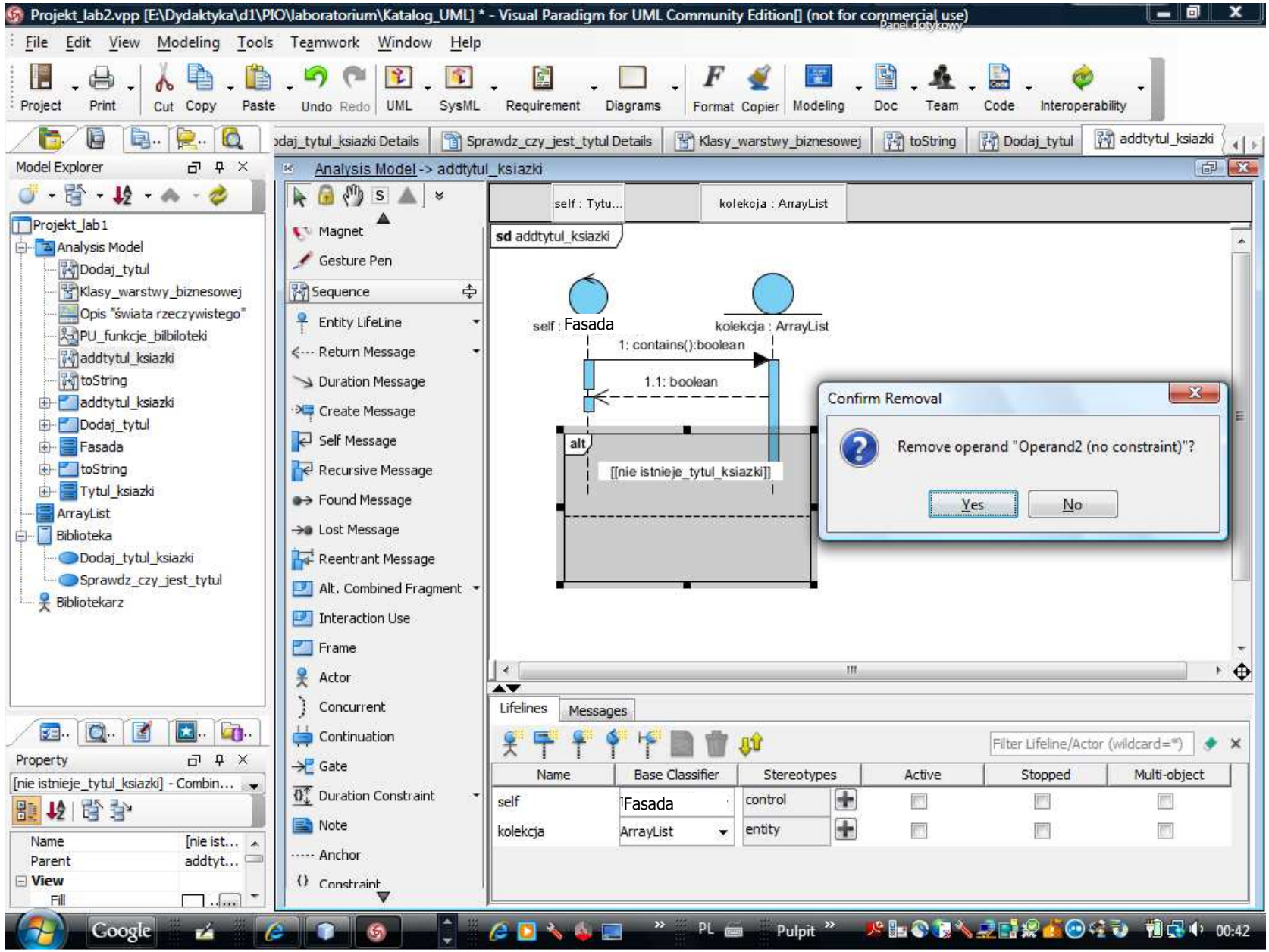
4.8. Dodanie bloku warunkowego Alt Combined Fragment z palety oraz edycja wyrażenia warunkowego bloku typu Alt Combined Fragment



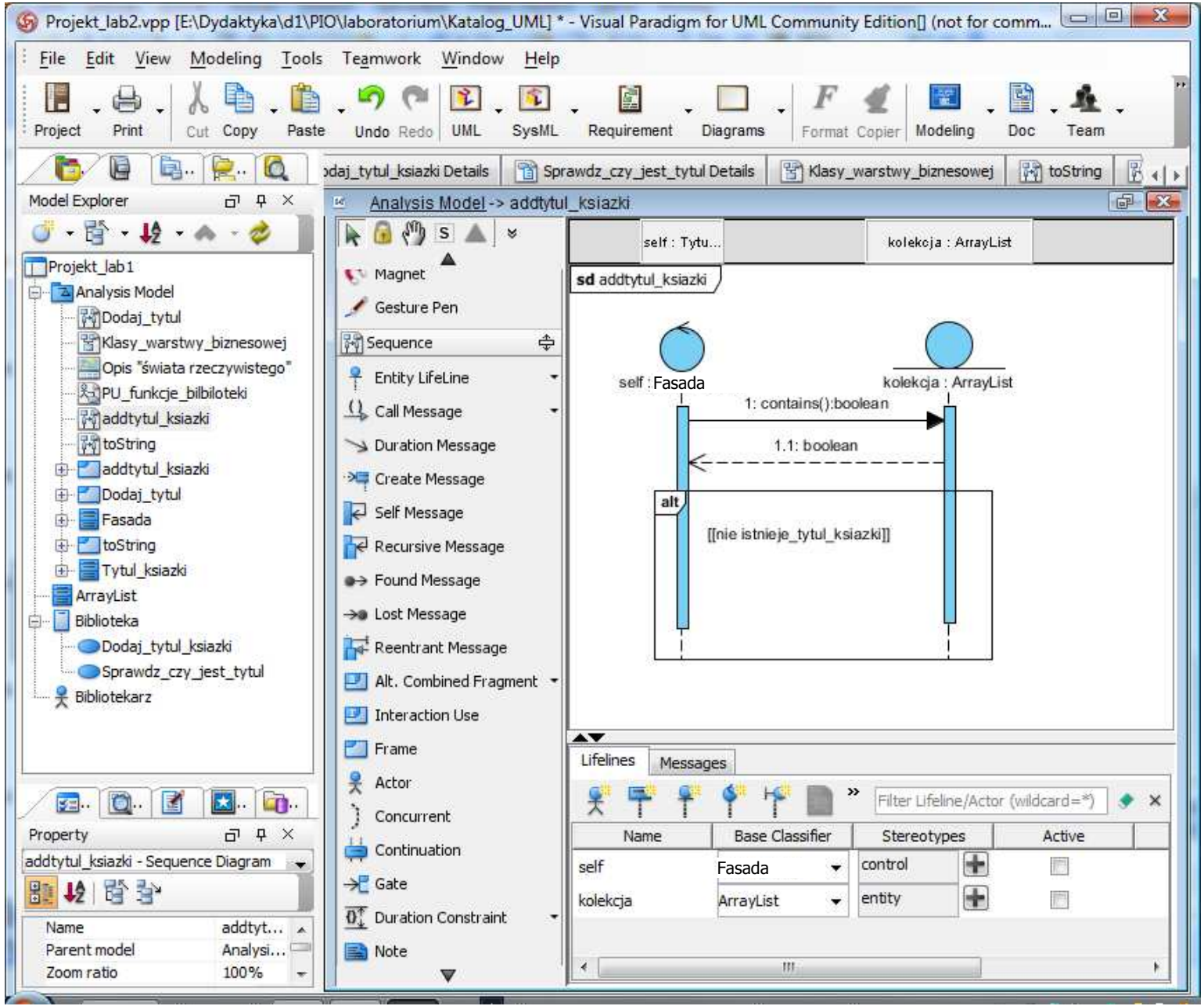
4.9. **Usunięcie** niepopotrzebnej linii reprezentującej dodatkowy operand elementu typu Alt Combined Fragment



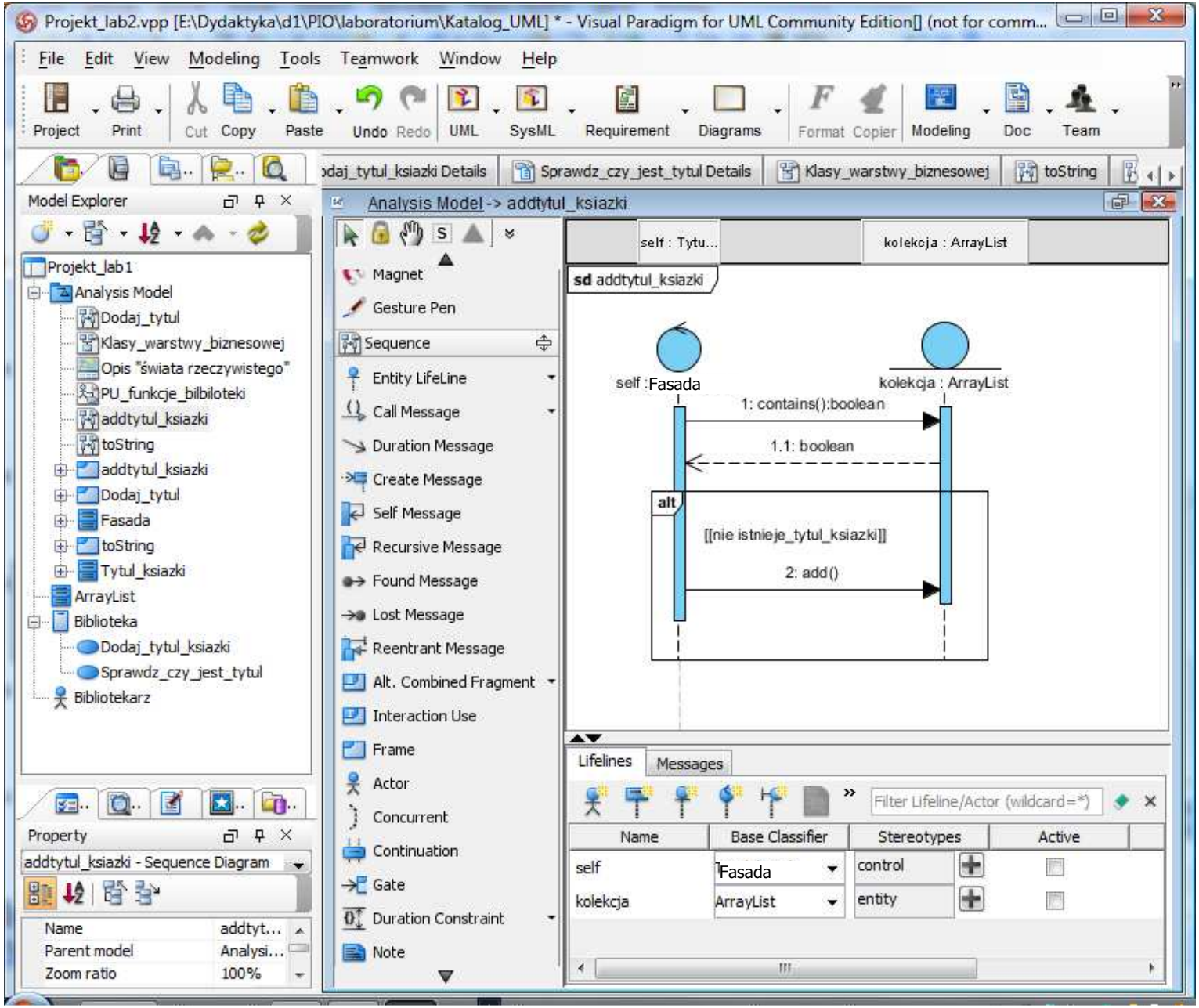
4.10. Usunięcie niepopotrzebnej linii reprezentującej dodatkowy operand elementu typu Alt Combined Fragment – potwierdzenie operacji usuwania



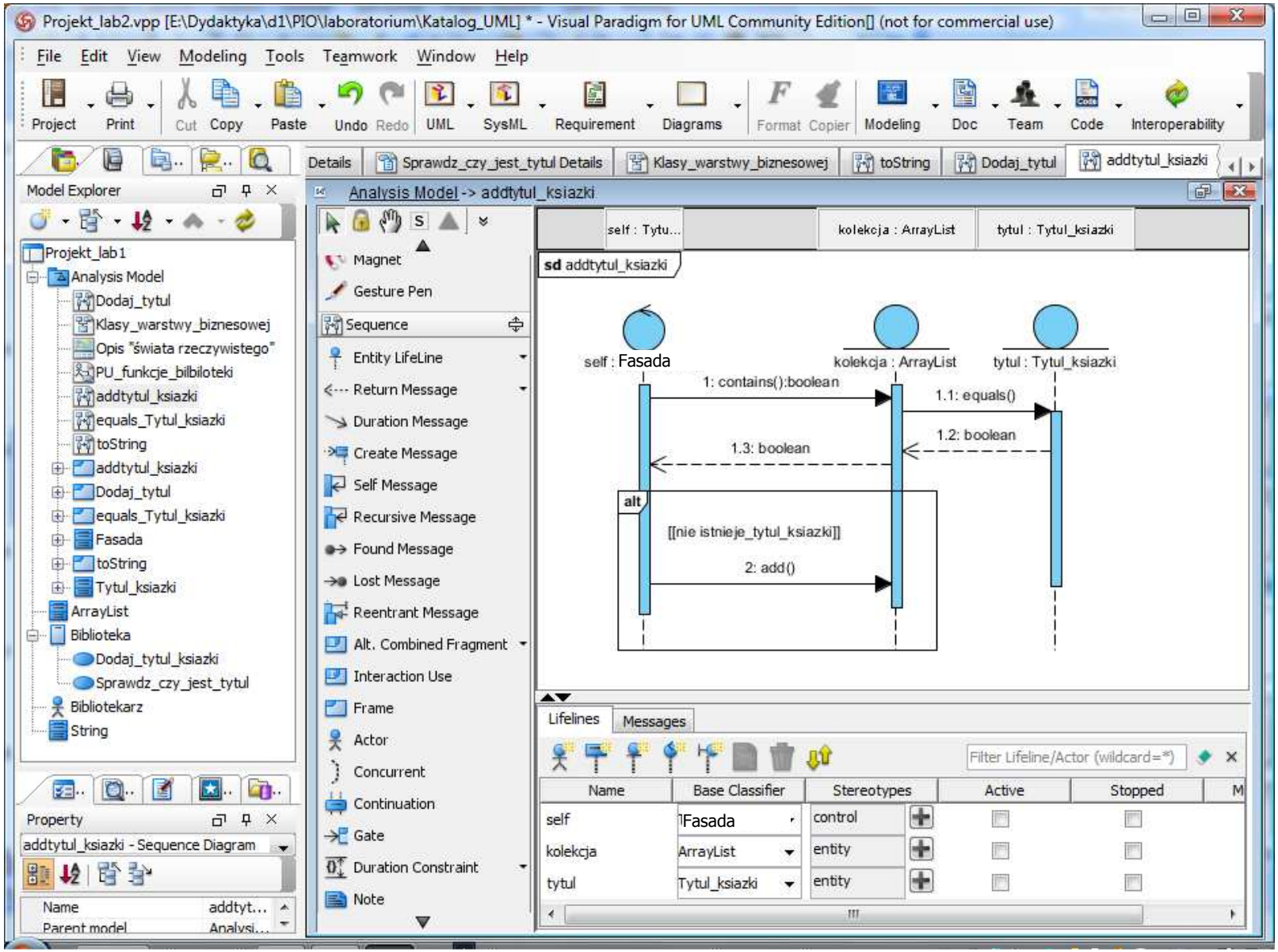
4.11. Zakończenie zmiany w bloku typu Alt Combined Fragment.



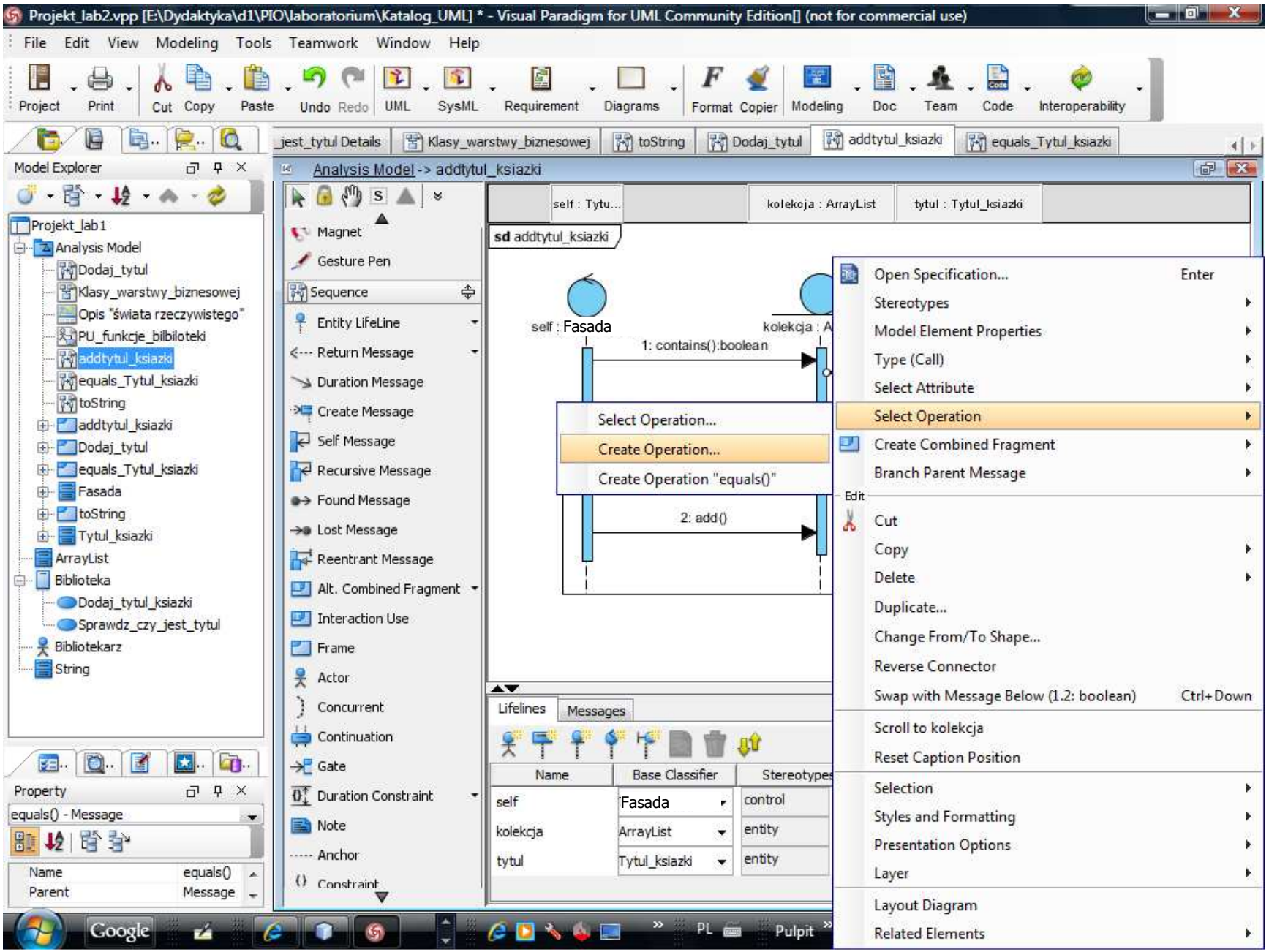
4.12. Dodanie do metody **dodaj_tytul** nowej metody addTytul_ksiazki metody add wywoływanej przez linię życia typu Fasada z linii życia typu ArrayList

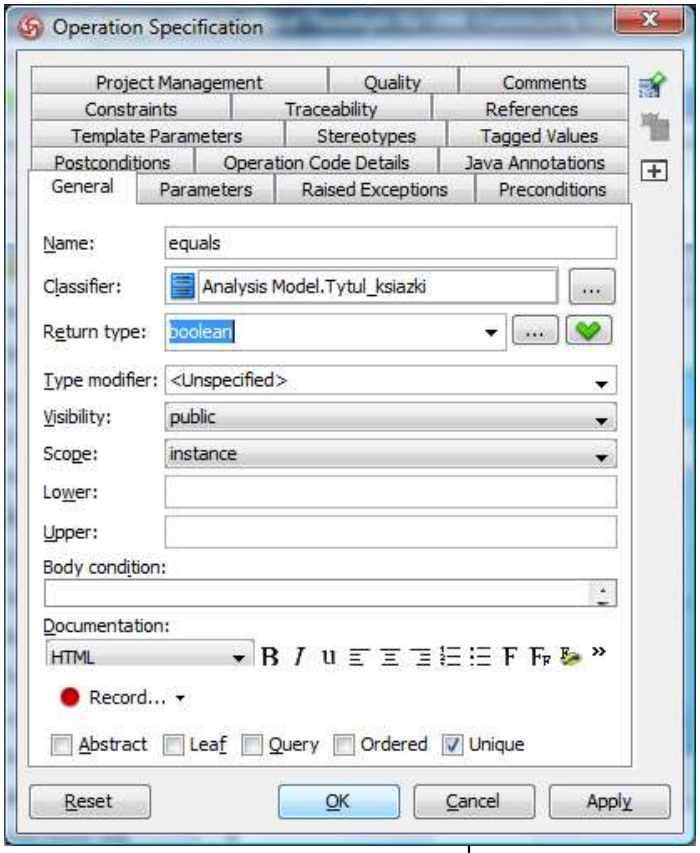


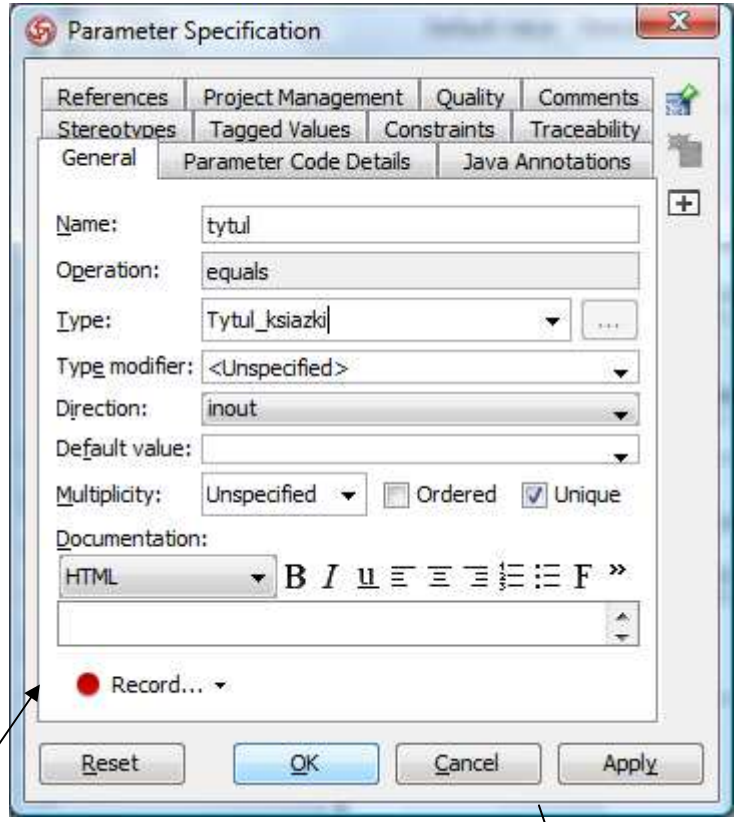
4.13. Uzupełniono scenariusz metody addtytul_ksiazki o wywołanie metody equals z klasy Tytul_ksiazki przesłaniającej metodę equals dziedziczoną od klasy Object – teraz w metodzie contains wywoływana jest metoda equals z klasy Tytul_ksiazki

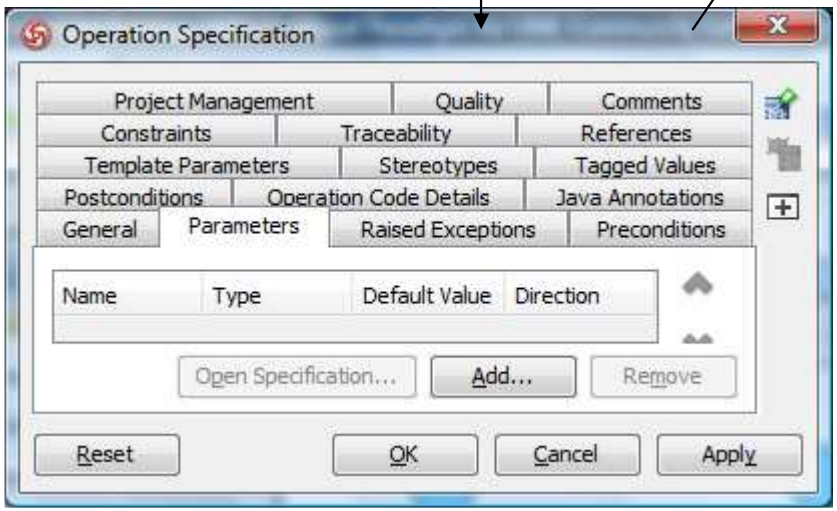


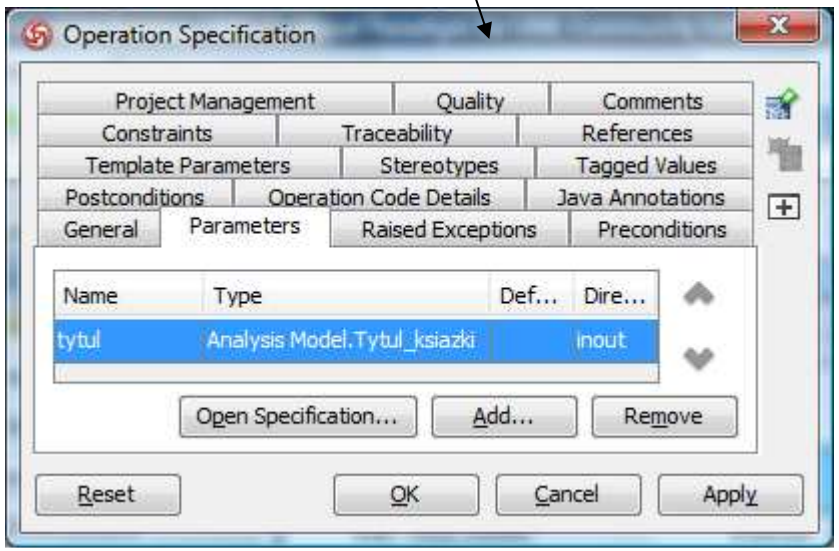
4.14. Zapisanie metody equals w klasie **Tytul_ksiazki**.





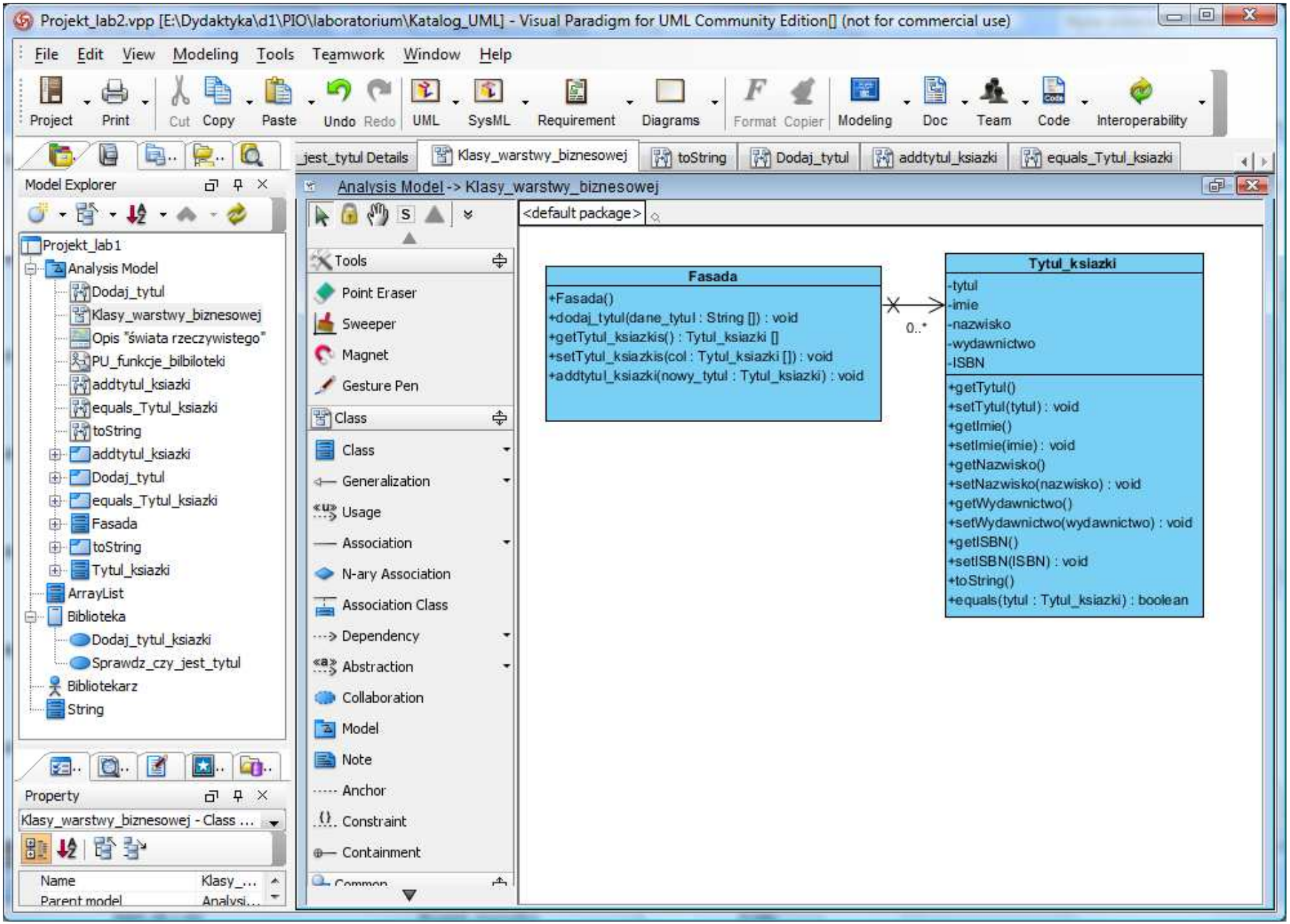




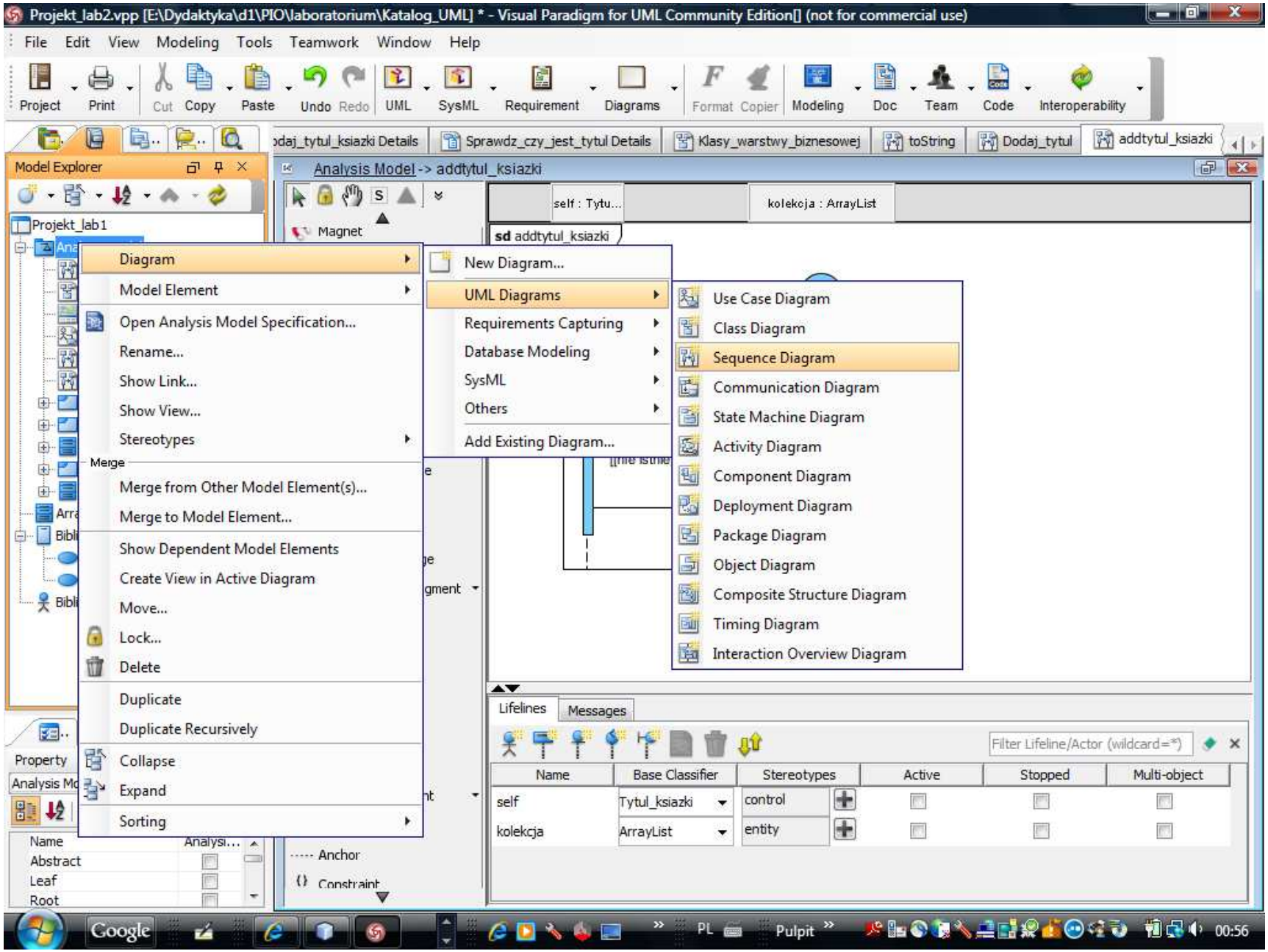


4.15
Definiowanie
metody equals
w klasie
Tytul_ksiazki

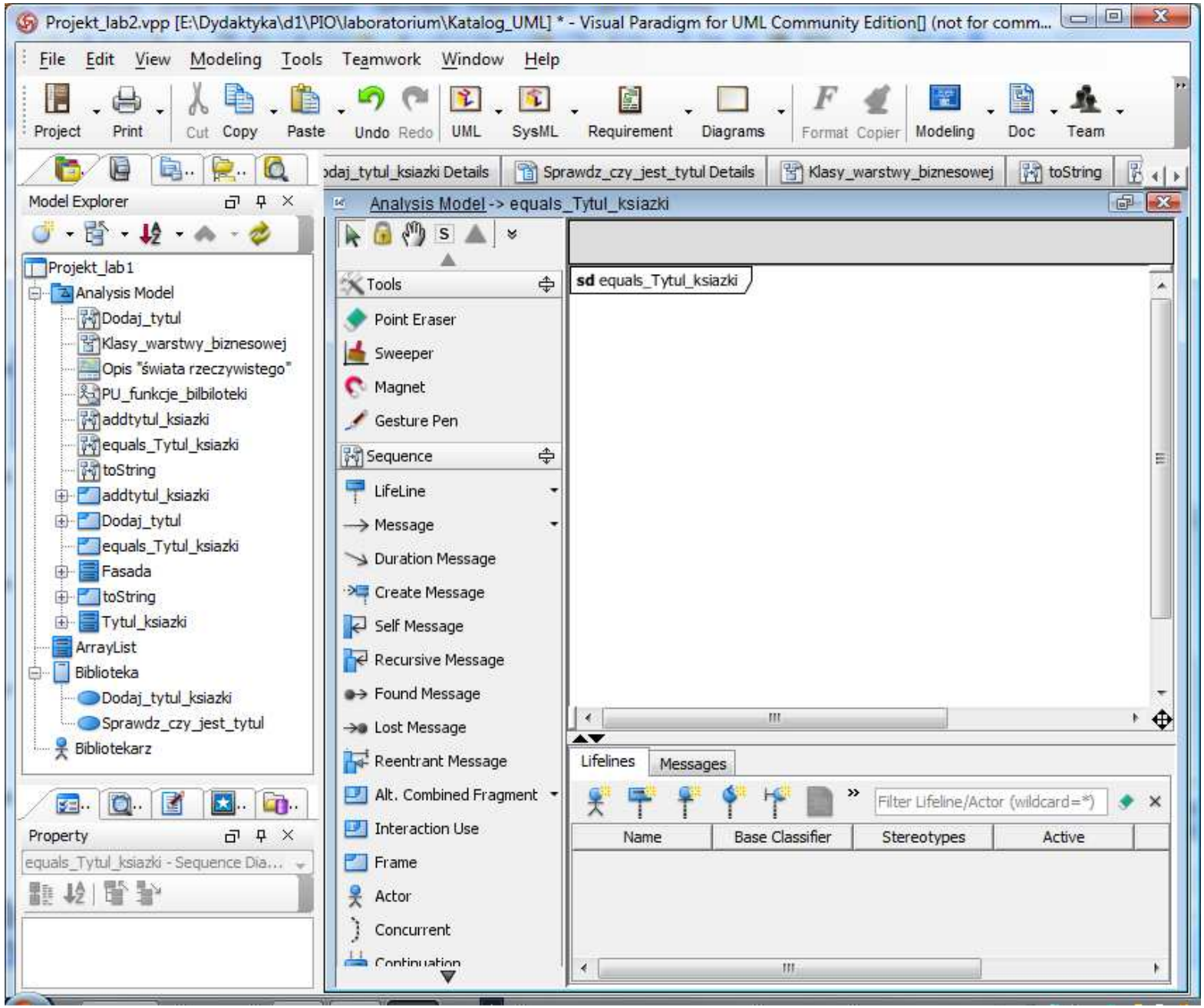
4.16. Nowe metody w klasach **Fasada** (typu get i set oraz addtytul_ksiazki) oraz **Tytul_ksiazki** (equals)



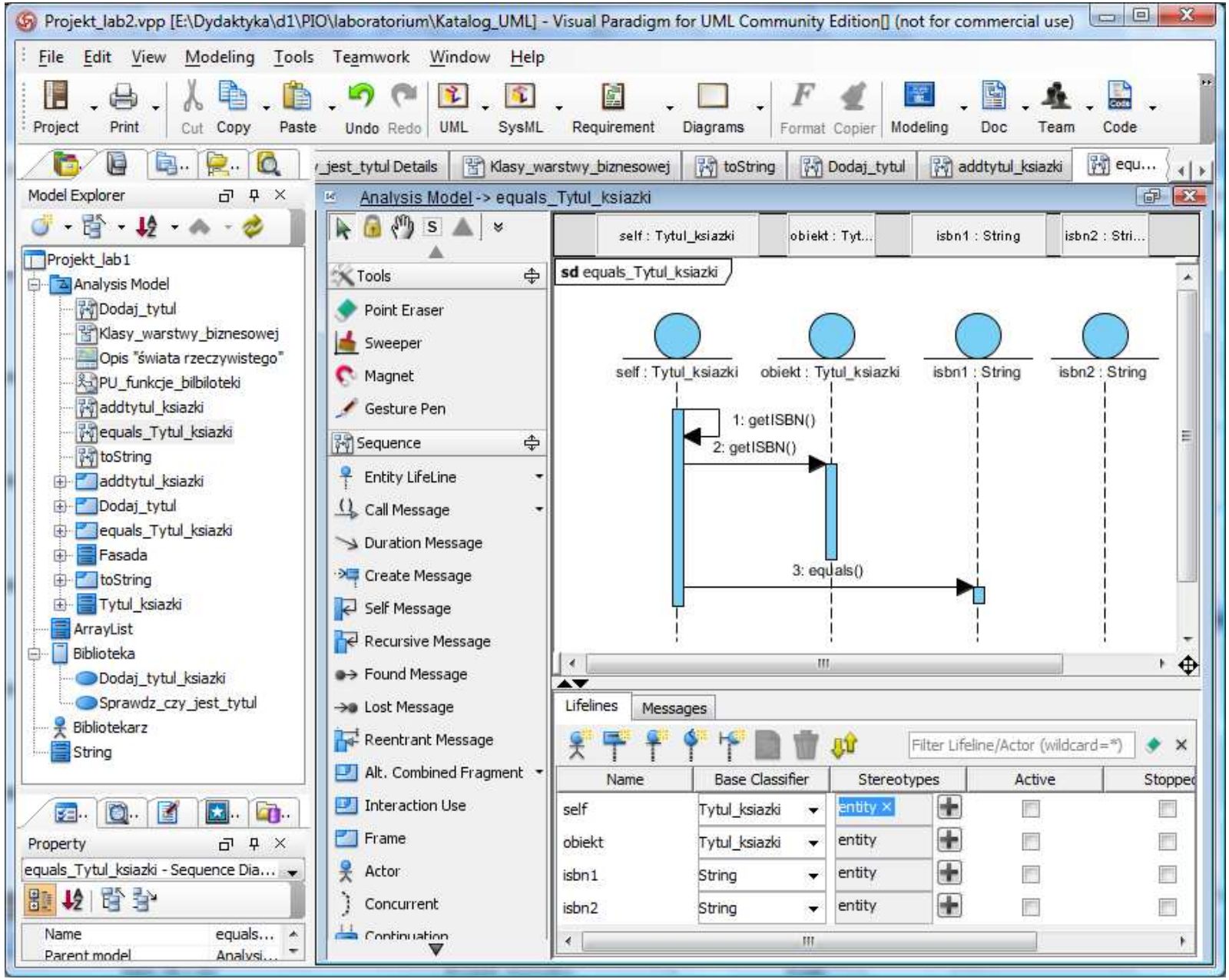
5.1. Utworzenie nowego diagramu sekwencji do zdefiniowania metody equals w klasie **Tytul_ksiazki**



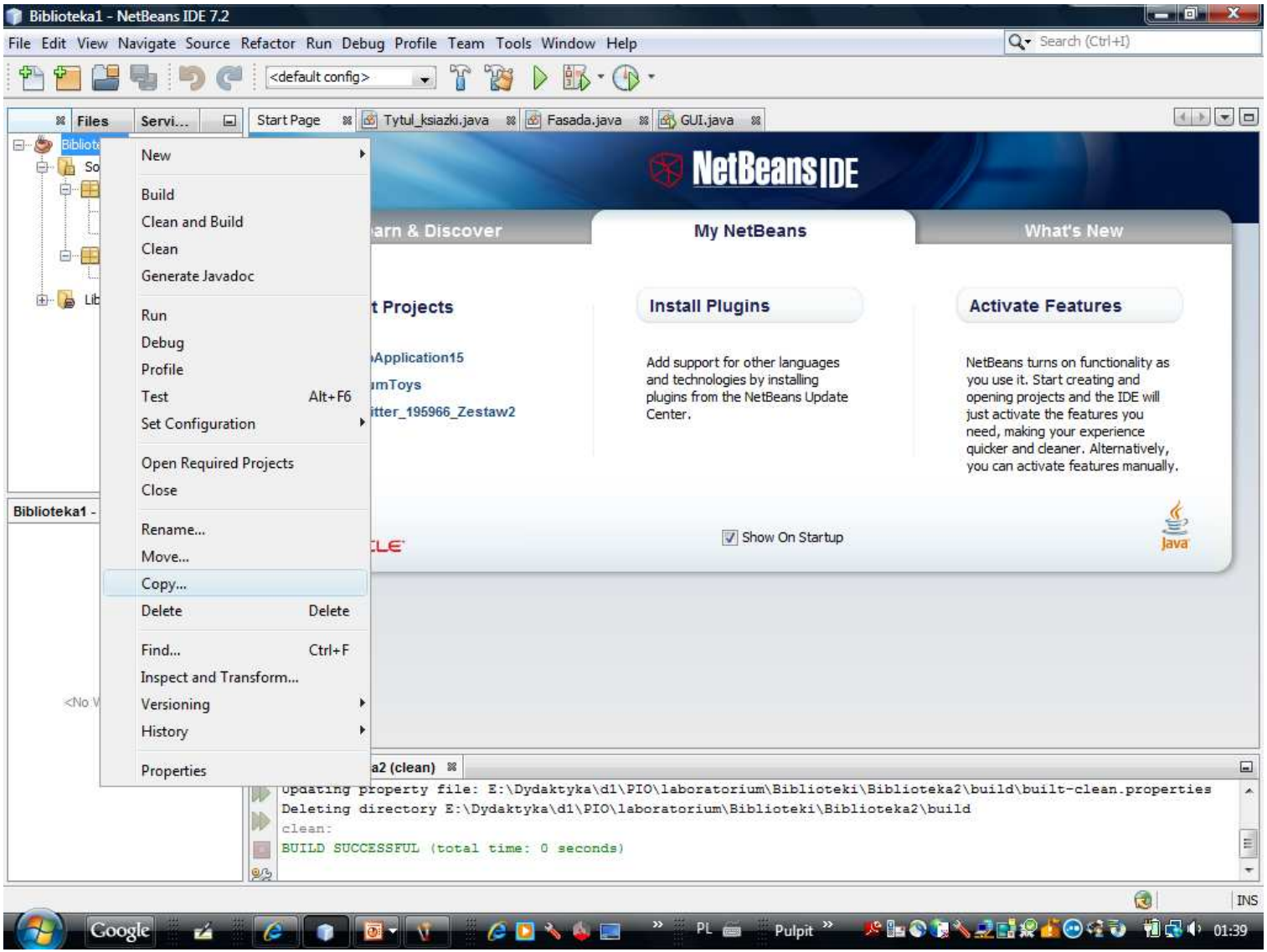
5.2. Nadanie nazwy diagramowi sekwencji



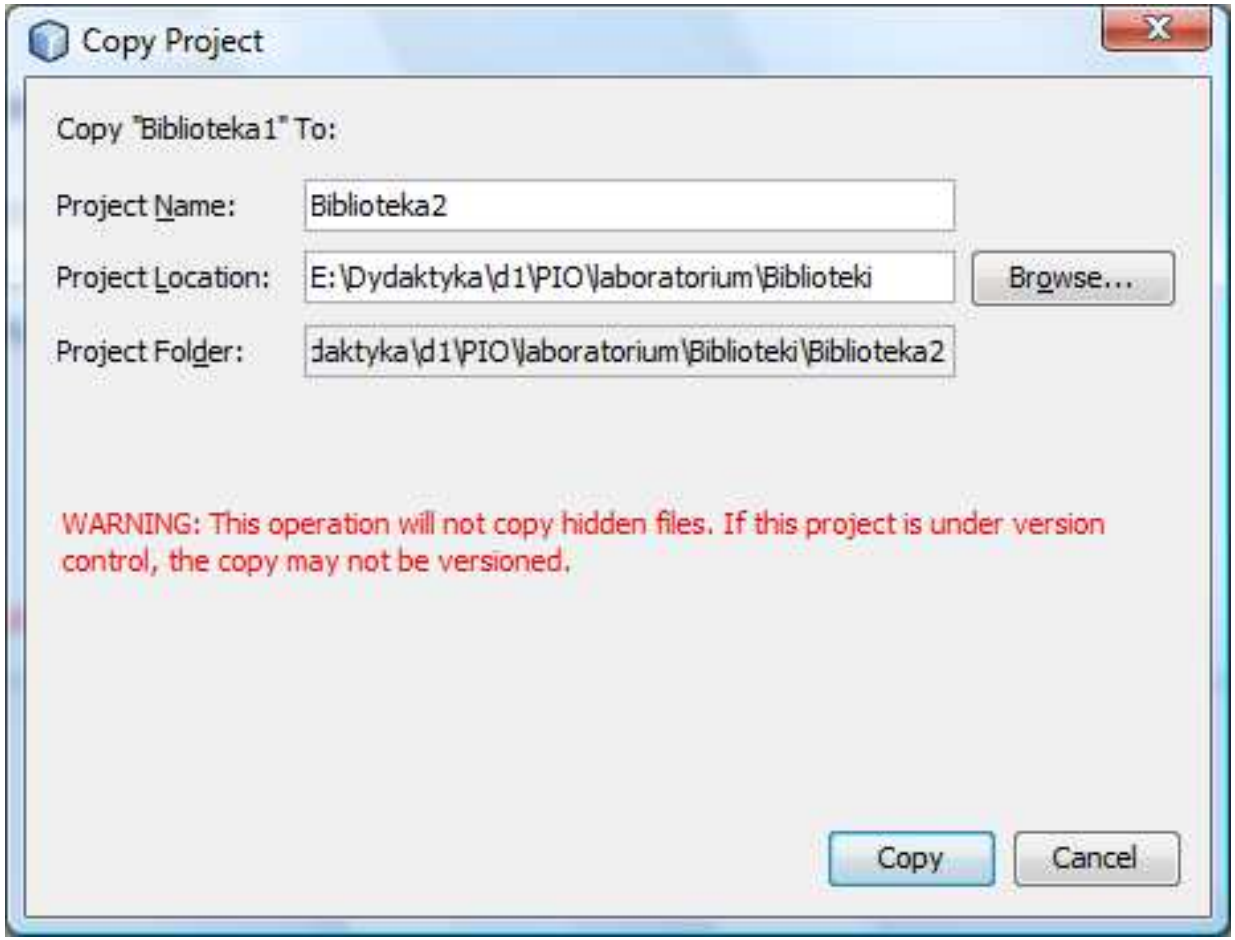
5.3. Wykonanie diagramu sekwencji reprezentującego metodę **equals** w klasie **Tytul_ksiazki**



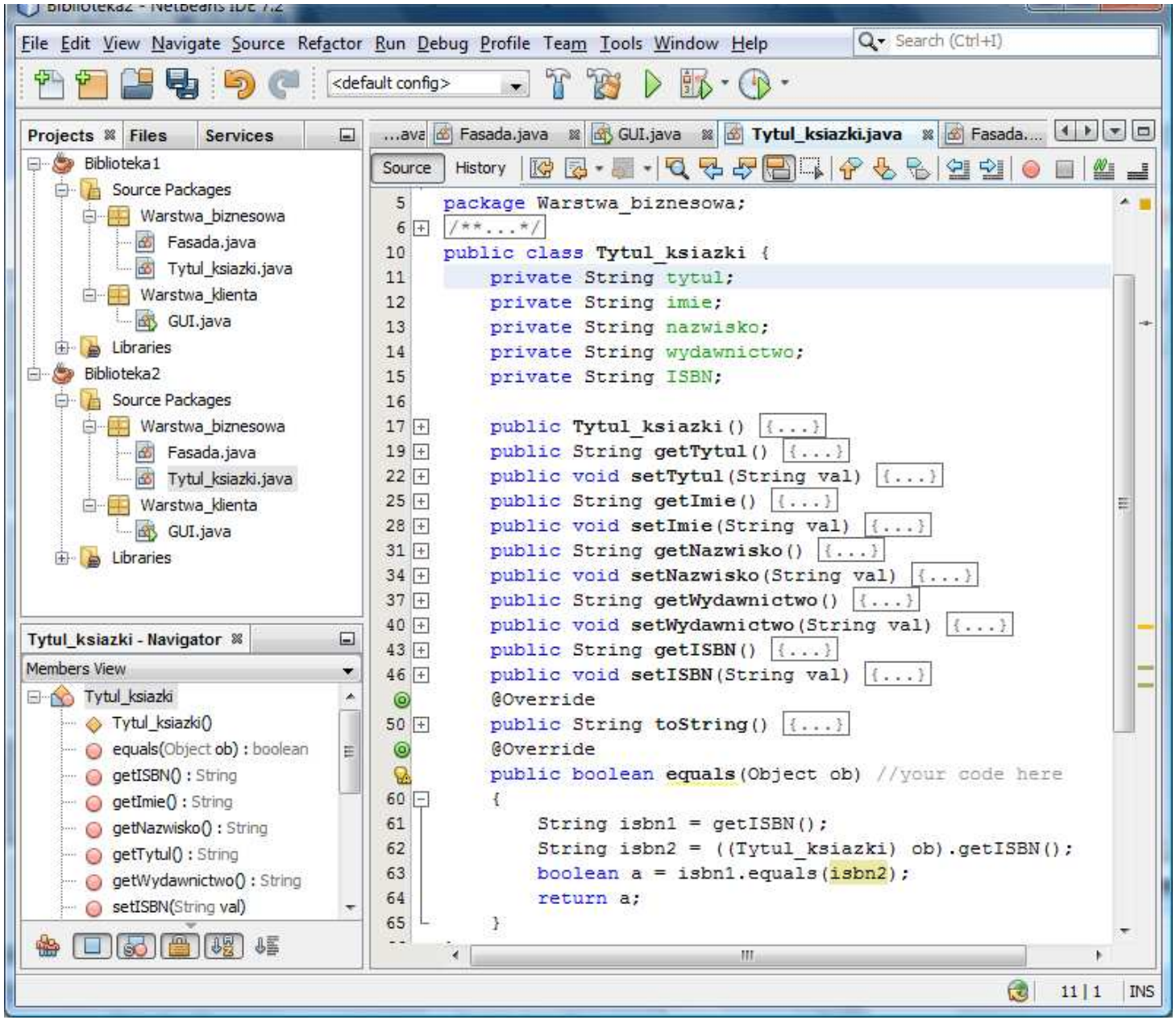
6.1. Modyfikacja kodu programu na podstawie diagramu klas i zmodyfikowanych diagramów sekwencji



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



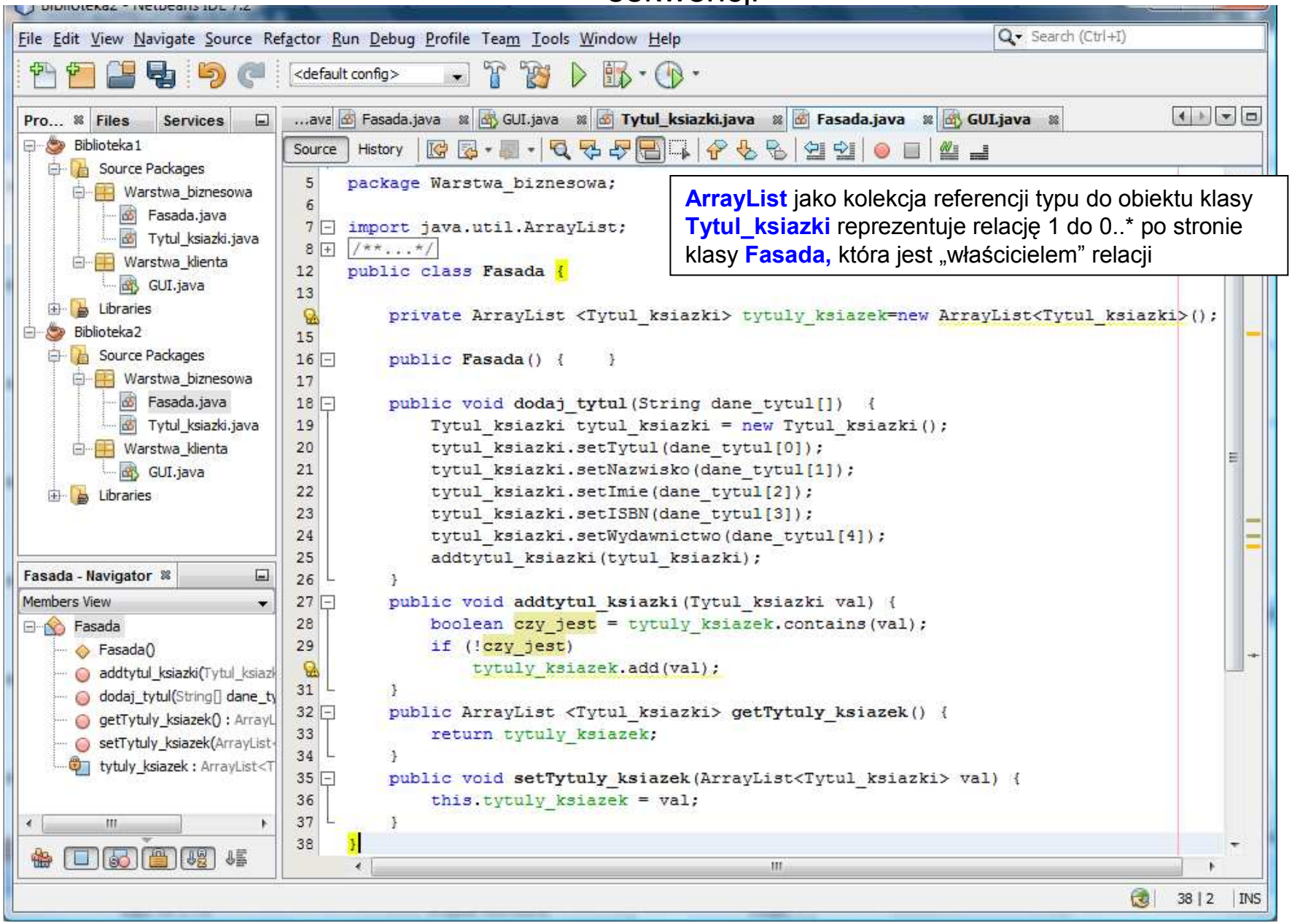
6.3. Kod klasy **Tytul Ksiazki**: metody **toString** i **equals** napisane na podstawie ich diagramów sekwenji



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_ksiazki zostały napisane na podstawie diagramów sekwencji



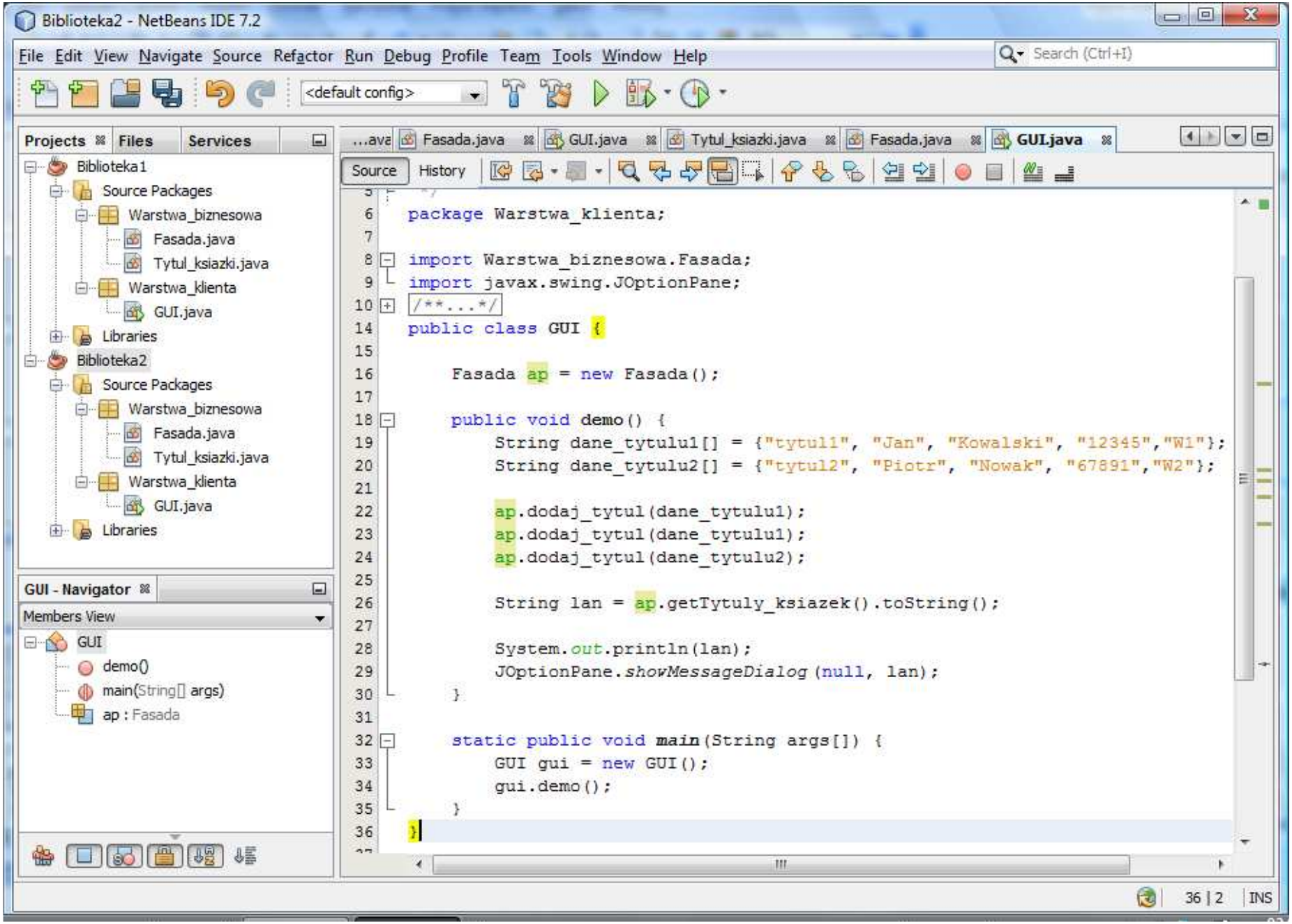
ArrayList jako kolekcja referencji typu do obiektu klasy **Tytul_ksiazki** reprezentuje relację 1 do 0..* po stronie klasy **Fasada,** która jest „właścicielem” relacji

Uzupełniono kod metody dodaj_tytul oraz dodano metodę addTytul_ksiazki

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

```
package Warstwa_biznesowa;

import java.util.ArrayList;
public class Fasada {

    private ArrayList <Tytul_ksiazki> tytuly_ksiazek=new ArrayList<Tytul_ksiazki>();

    public Fasada() {    }

    public void dodaj_tytul(String dane_tytul[])  {
        Tytul_ksiazki tytul_ksiazki = new Tytul_ksiazki();
        tytul_ksiazki.setTytul(dane_tytul[0]);
        tytul_ksiazki.setNazwisko(dane_tytul[1]);
        tytul_ksiazki.setImie(dane_tytul[2]);
        tytul_ksiazki.setISBN(dane_tytul[3]);
        tytul_ksiazki.setWydawnictwo(dane_tytul[4]);
        addtytul_ksiazki(tytul_ksiazki);
    }
    public void addtytul_ksiazki(Tytul_ksiazki val) {
        boolean czy_jest = tytuly_ksiazek.contains(val);
        if (!czy_jest)
            tytuly_ksiazek.add(val);
    }
    public ArrayList <Tytul_ksiazki> getTytuly_ksiazek()          {       return tytuly_ksiazek;    }
    public void setTytuly_ksiazek(ArrayList<Tytul_ksiazki> val)   {       this.tytuly_ksiazek = val;    }
}
```

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



```
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_ksiazki w klasie Fasada– każdy tytul może mieć inny ISBN dzięki metodzie equals zdefiniowanej w klasie Tytul_książki i wywoływanej w metodzie contains klasy typu ArrayList