

Instrukcja 1

Laboratorium z Podstaw Inżynierii Oprogramowania

Relacja 1 do 1..0– instrukcja z lab1

Cele laboratorium 1

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 projektu zgodnie z materiałem zawartym na slajdach 5-84. Jest to ćwiczenie, które pozwala poznać narzędzie Visual Paradigm 10 for UML 2.0 wykorzystane w ramach zajęć laboratoryjnych z przedmiotu Podstawy Inżynierii Oprogramowania.

Java

język programowania

- obiektowo zorientowany
- wysokiego poziomu

platforma Javy

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

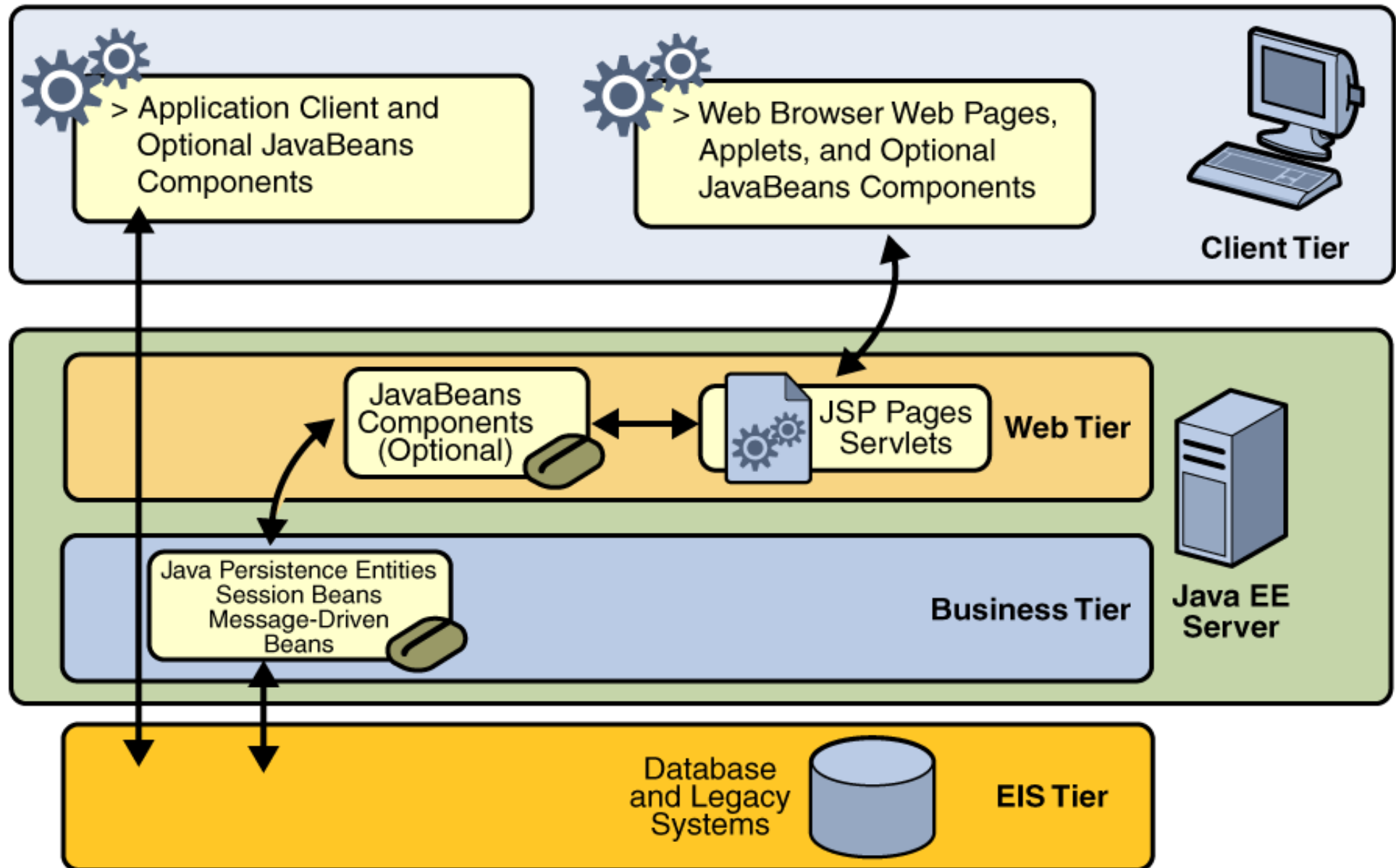
Rezultat

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

Rodzaje platform Javy:

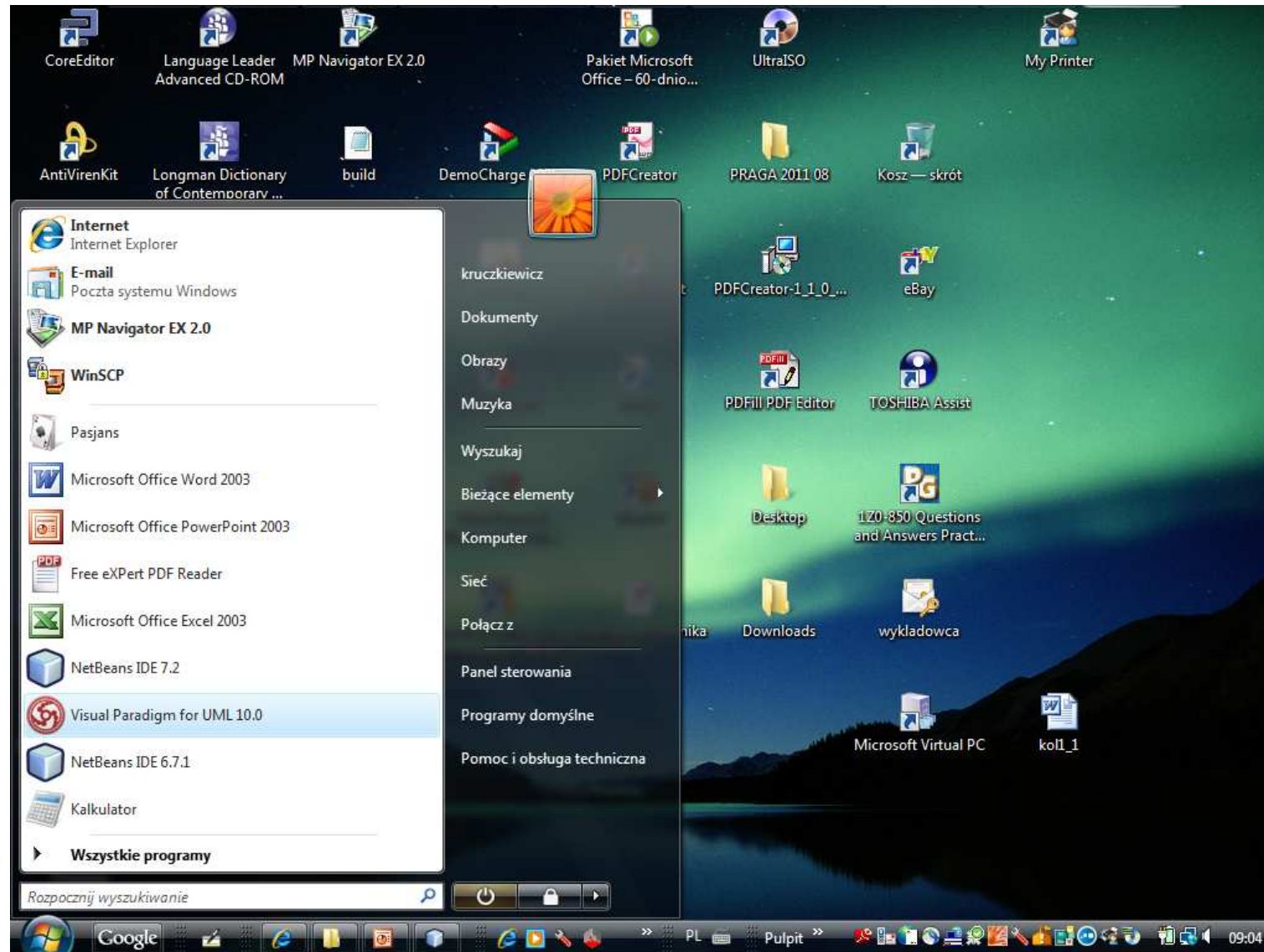
- ◆ Java Platform, Standard Edition (Java SE)
- ◆ Java Platform, Enterprise Edition (Java EE)
- ◆ Java Platform, Micro Edition (Java ME)
- ◆ Java Platform CARD

Warstwy aplikacji (Java EE)

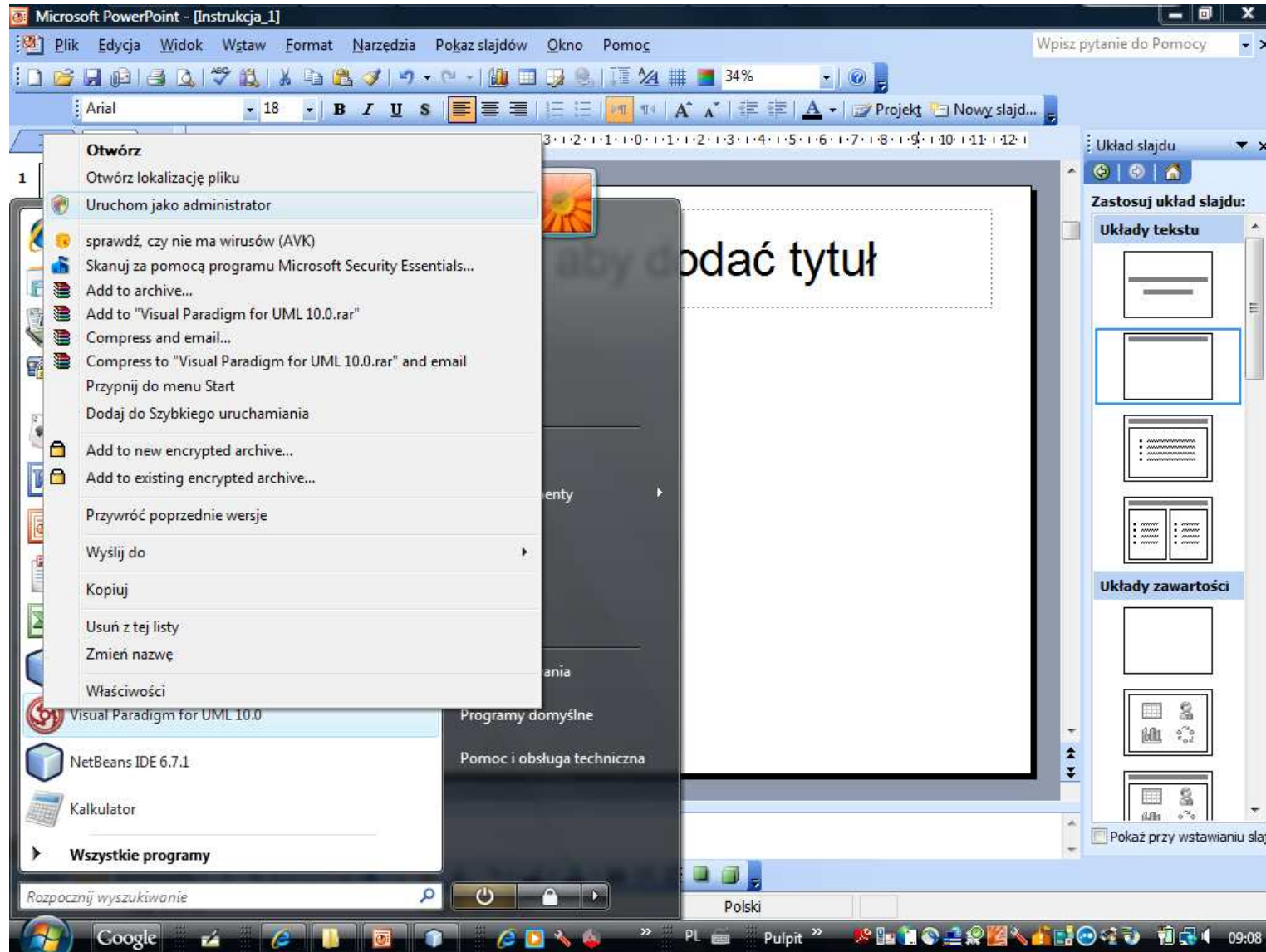


1.1. Uruchomienie programu programu *Visual Paradigm* z uprawnieniami administratora

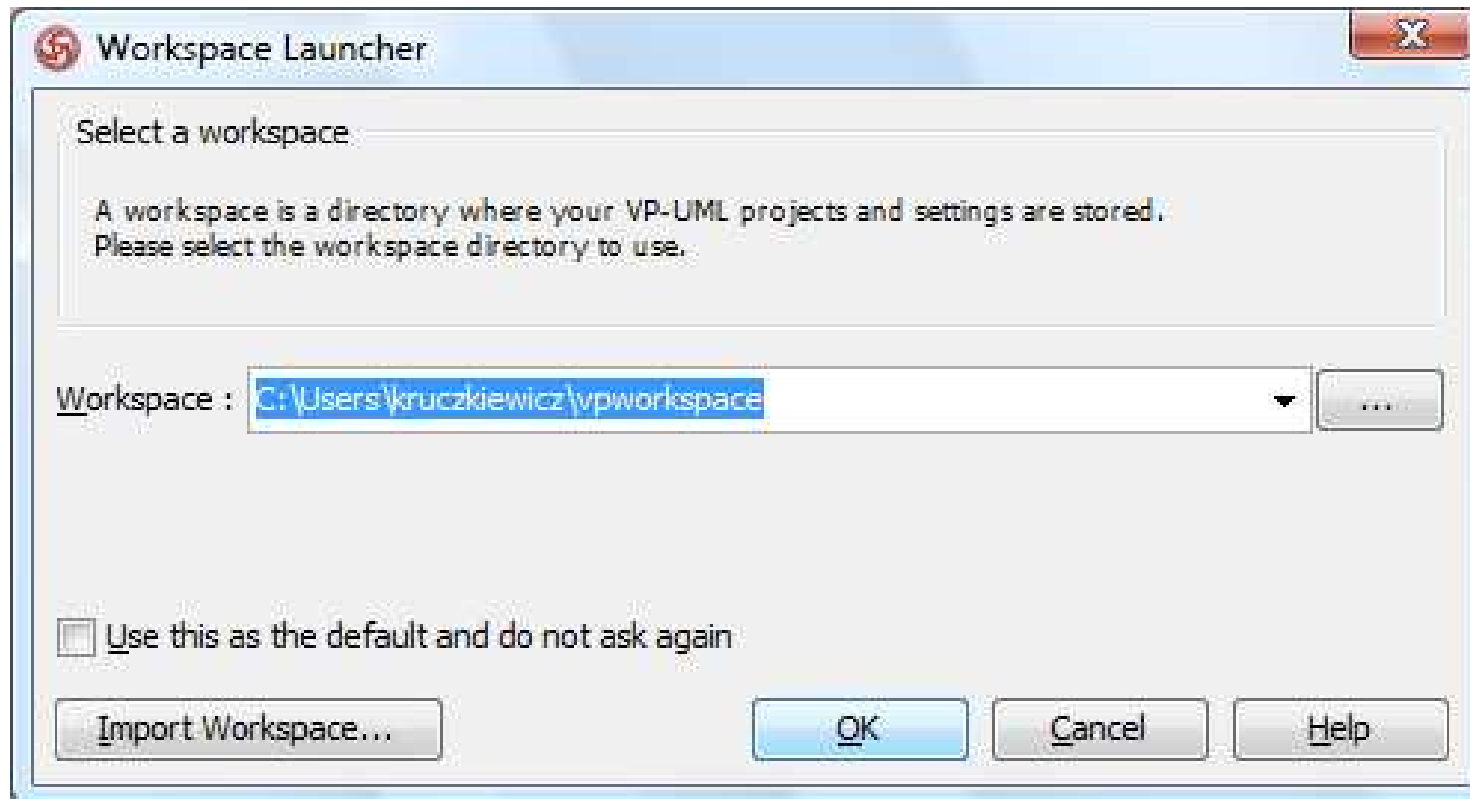
- kliknięcie prawym klawiszem myszy na nazwę programu



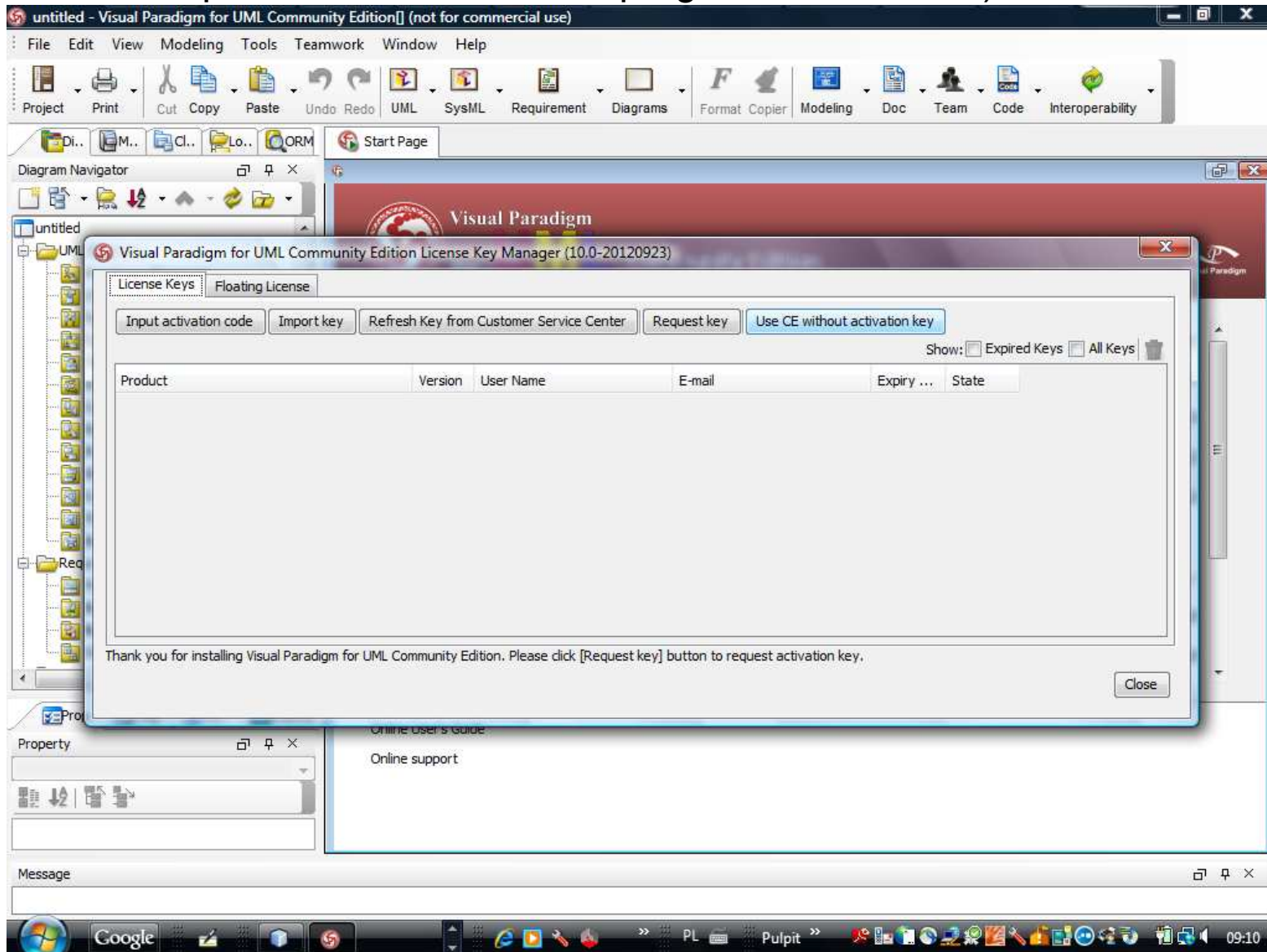
1.2. Uruchomienie programu programu *Visual Paradigm* z uprawnieniami administratora - wybór z listy opcji *Uruchom jako administrator*



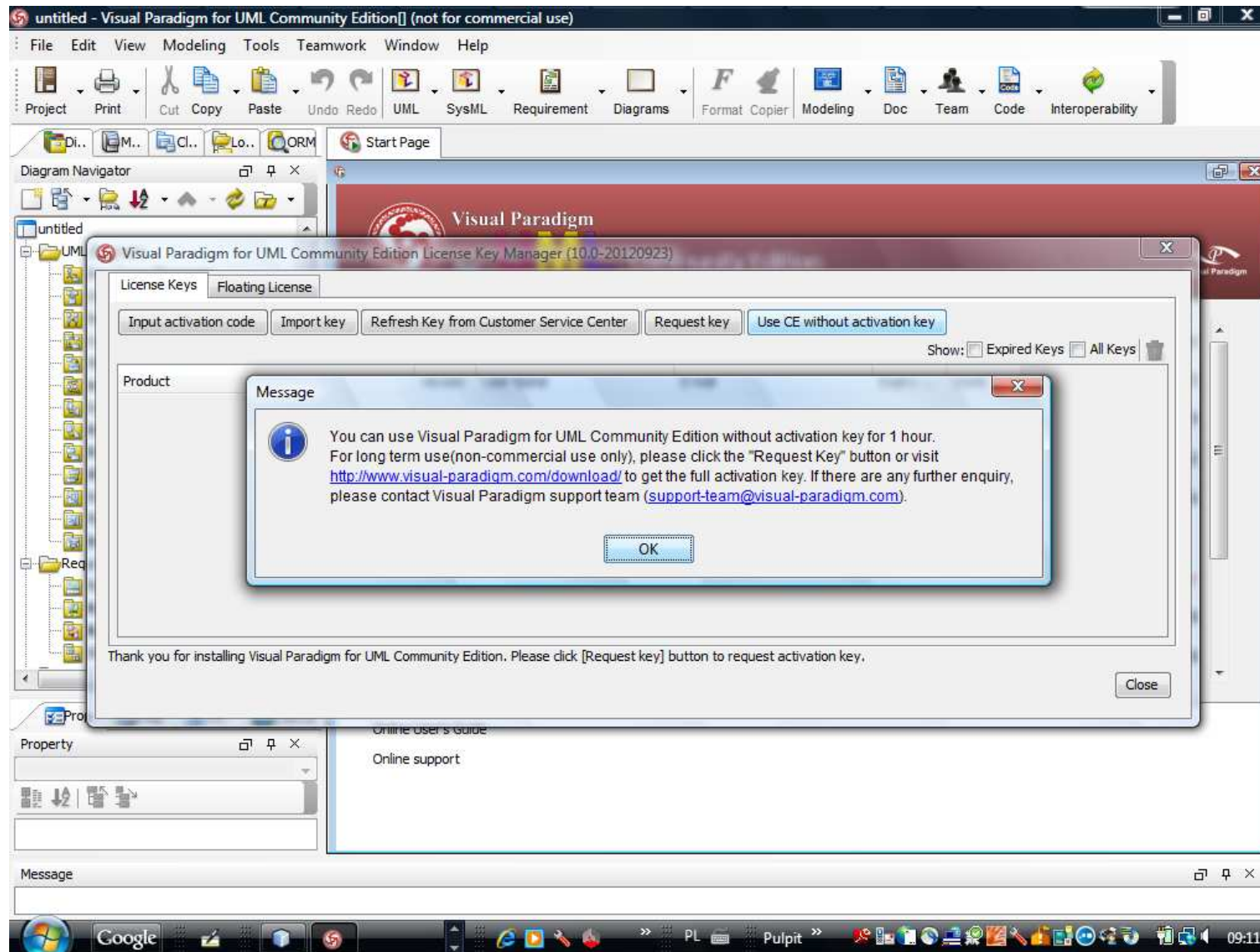
2. Wybór katalogu do tworzenia projektów



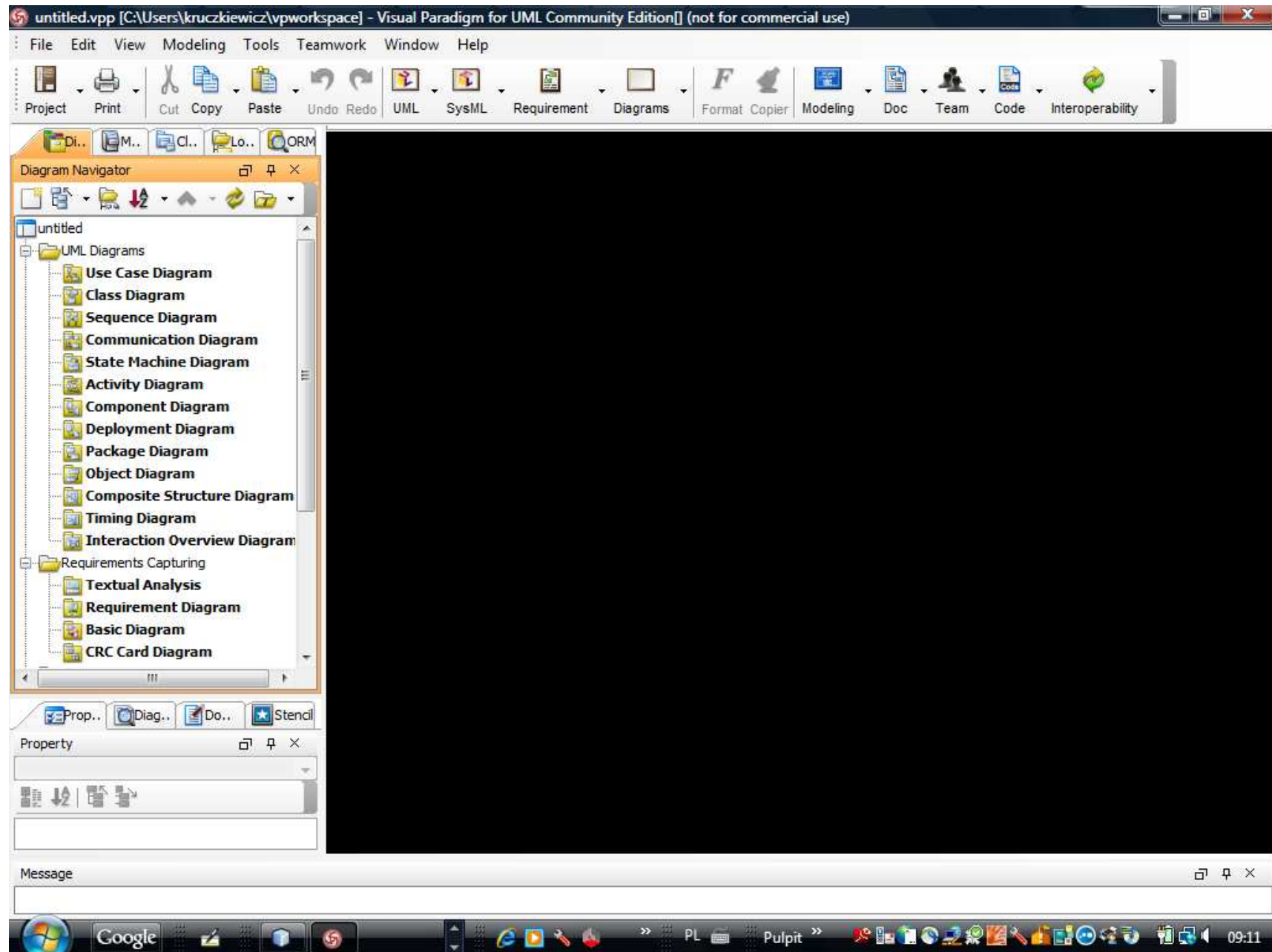
3.1. Uruchomienie wersji *Community* bez klucza (co godzinę automatycznie pojawia się to okienko w celu kolejnego potwierdzenia działania programu bez klucza)



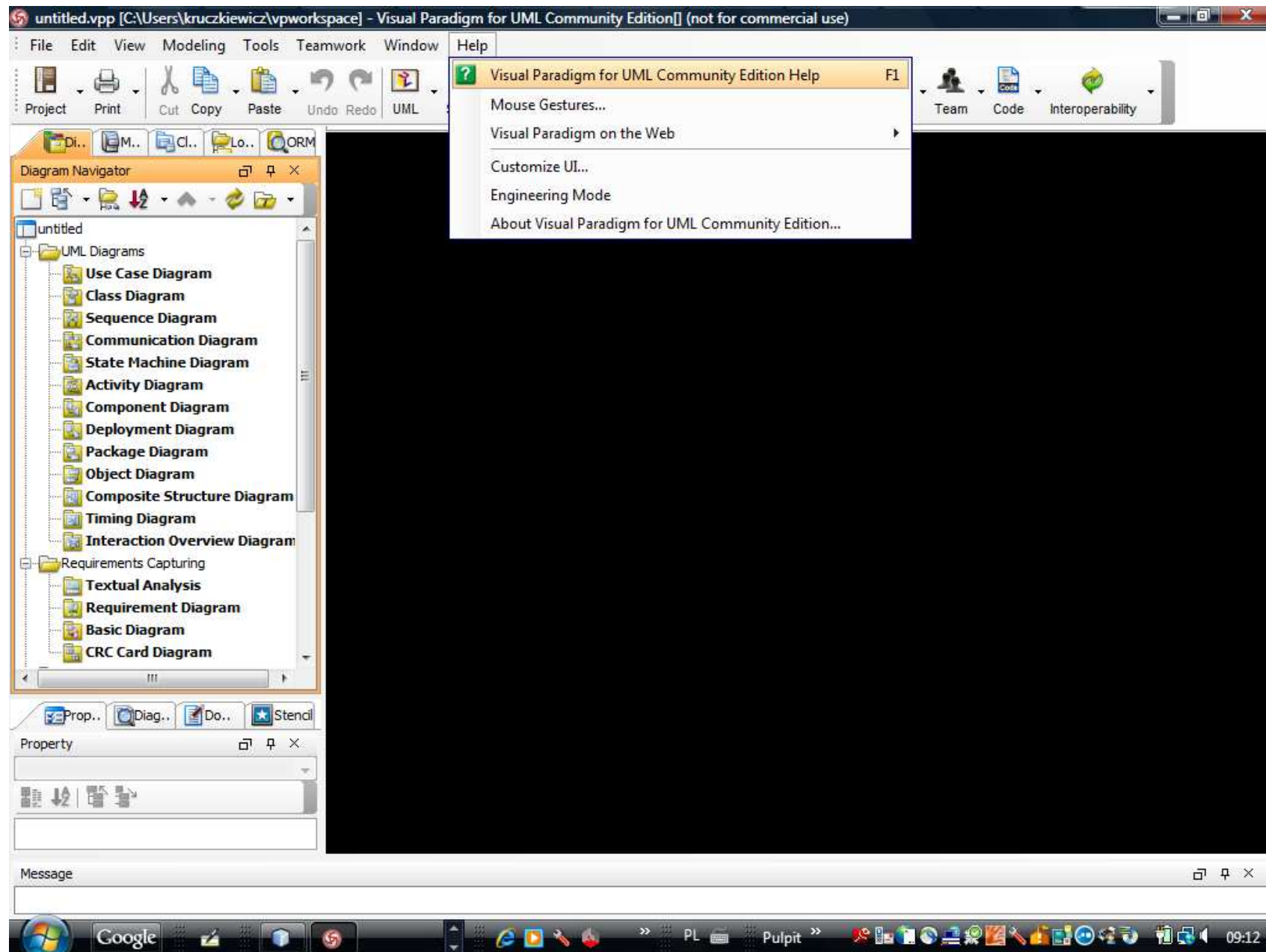
3.2. Uruchomienie wersji *Community* bez klucza (co godzinę automatycznie pojawia się to okienko w celu kolejnego potwierdzenia działania programu bez klucza)



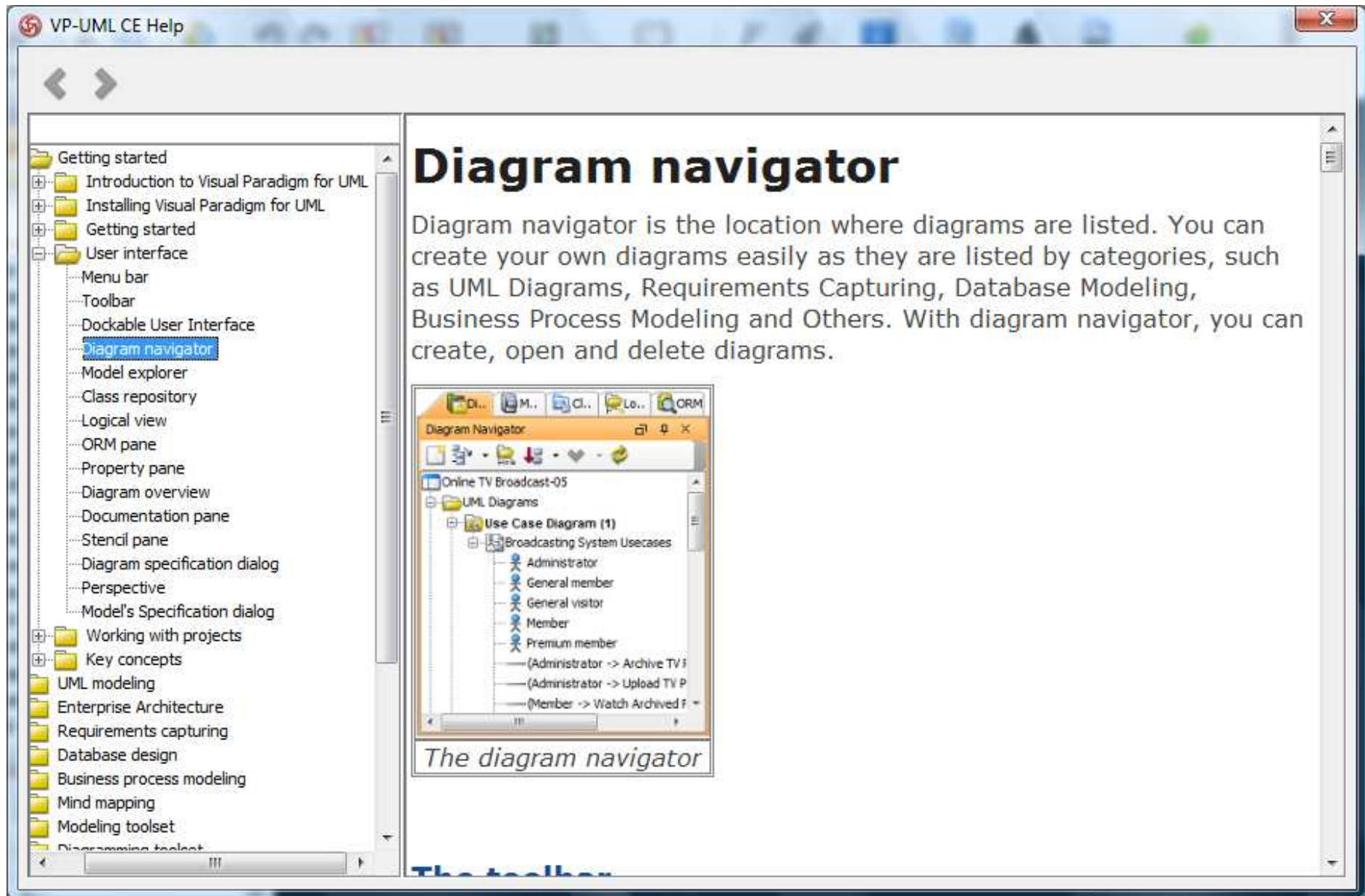
4. Start programu z pustym projektem bez nazwy



5. Ściągnięcie części *Help* ze strony producenta narzędzia



6. Przykładowe informacje dostarczone z części *Help*



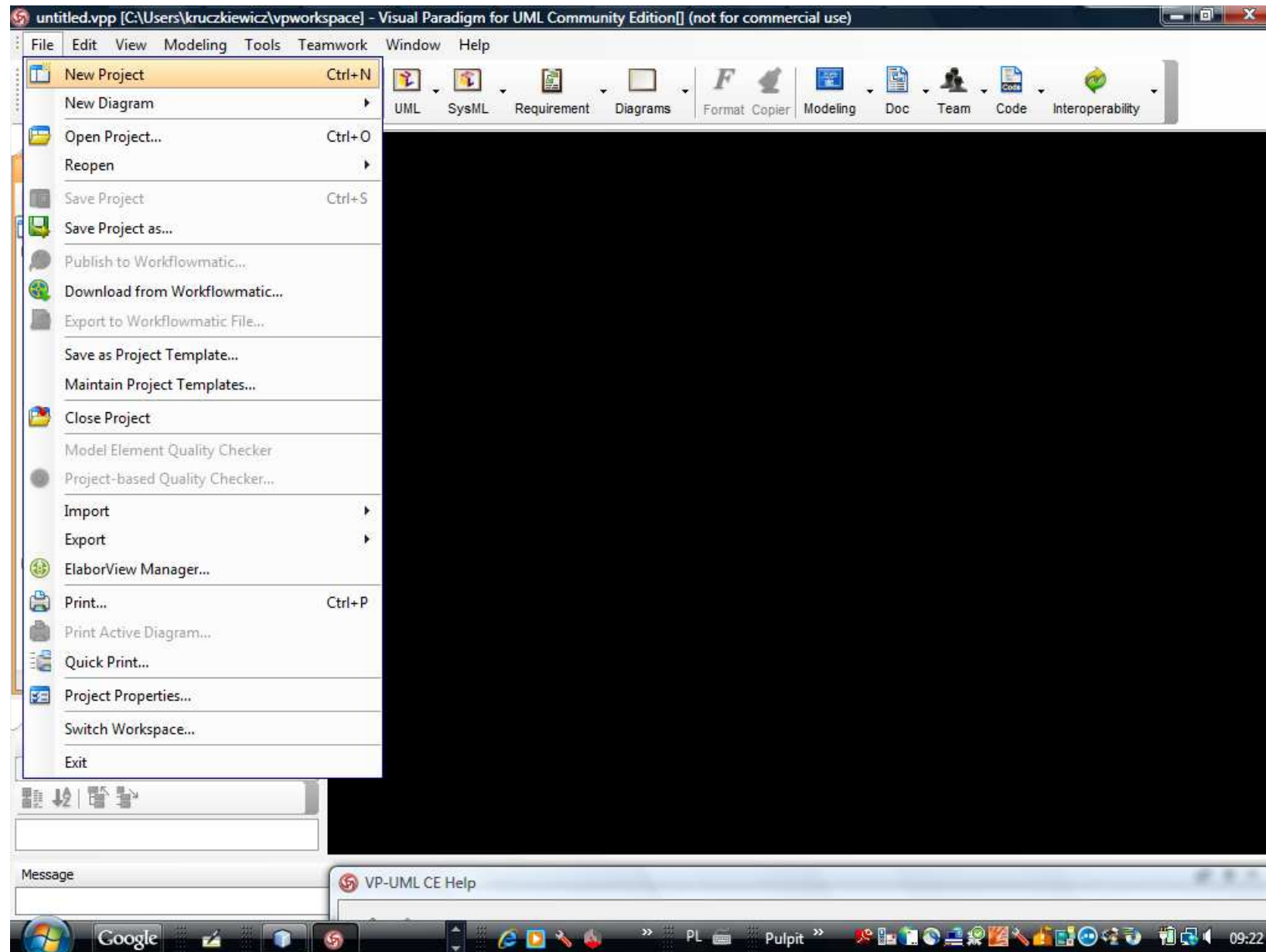
The screenshot shows a help window titled "VP-UML CE Help". On the left is a table of contents with "Diagram navigator" highlighted. The main content area has the heading "Diagram navigator" and a descriptive paragraph. Below the text is a screenshot of the "Diagram Navigator" tool window, which displays a tree view of a project named "Online TV Broadcast-05". Under "UML Diagrams", there is a "Use Case Diagram (1)" containing "Broadcasting System Usecases" with actors: Administrator, General member, General visitor, Member, and Premium member. Below the actors are use cases: "(Administrator -> Archive TV I", "(Administrator -> Upload TV P", and "(Member -> Watch Archived F".

Diagram navigator

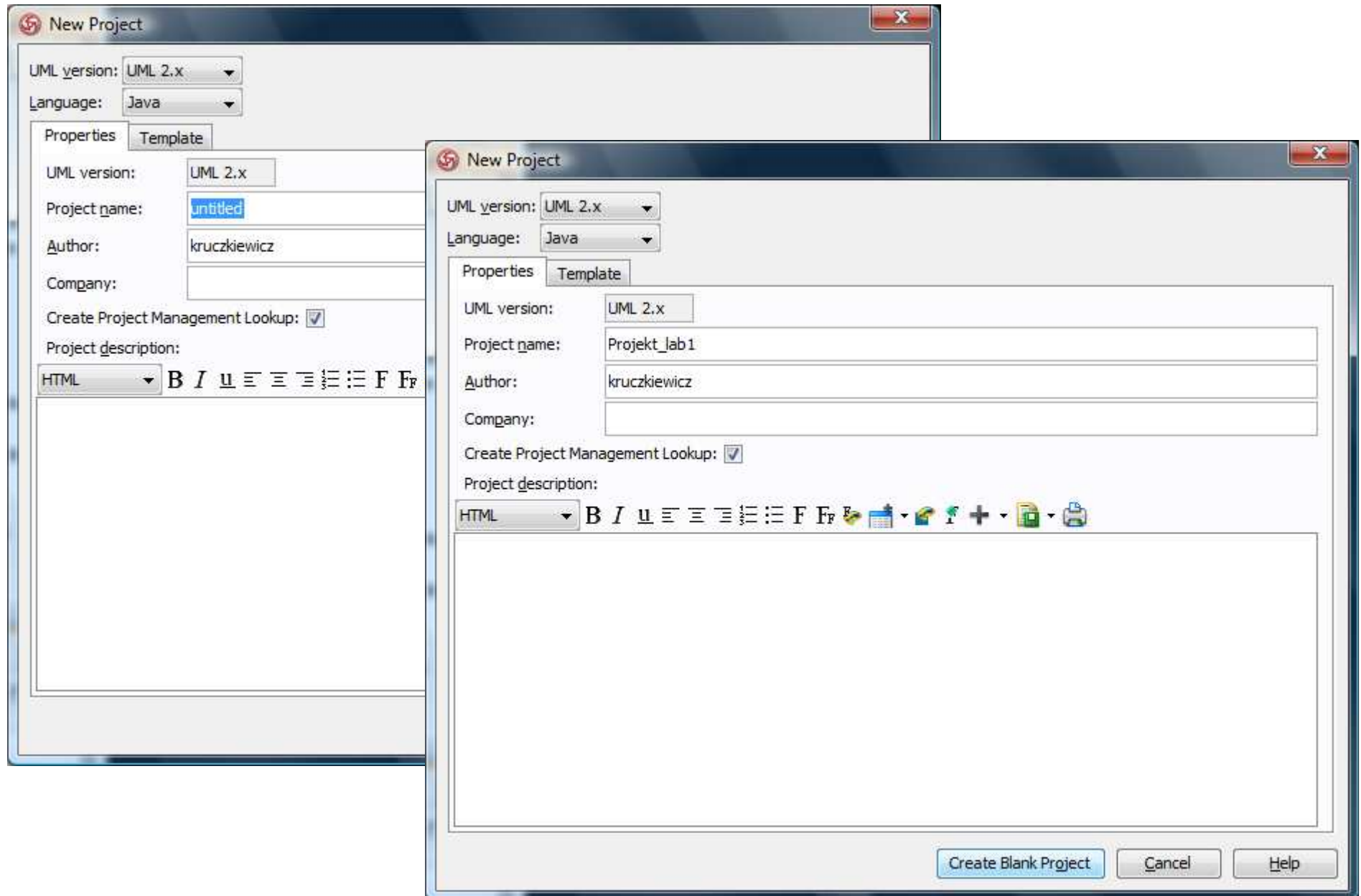
Diagram navigator is the location where diagrams are listed. You can create your own diagrams easily as they are listed by categories, such as UML Diagrams, Requirements Capturing, Database Modeling, Business Process Modeling and Others. With diagram navigator, you can create, open and delete diagrams.

The diagram navigator

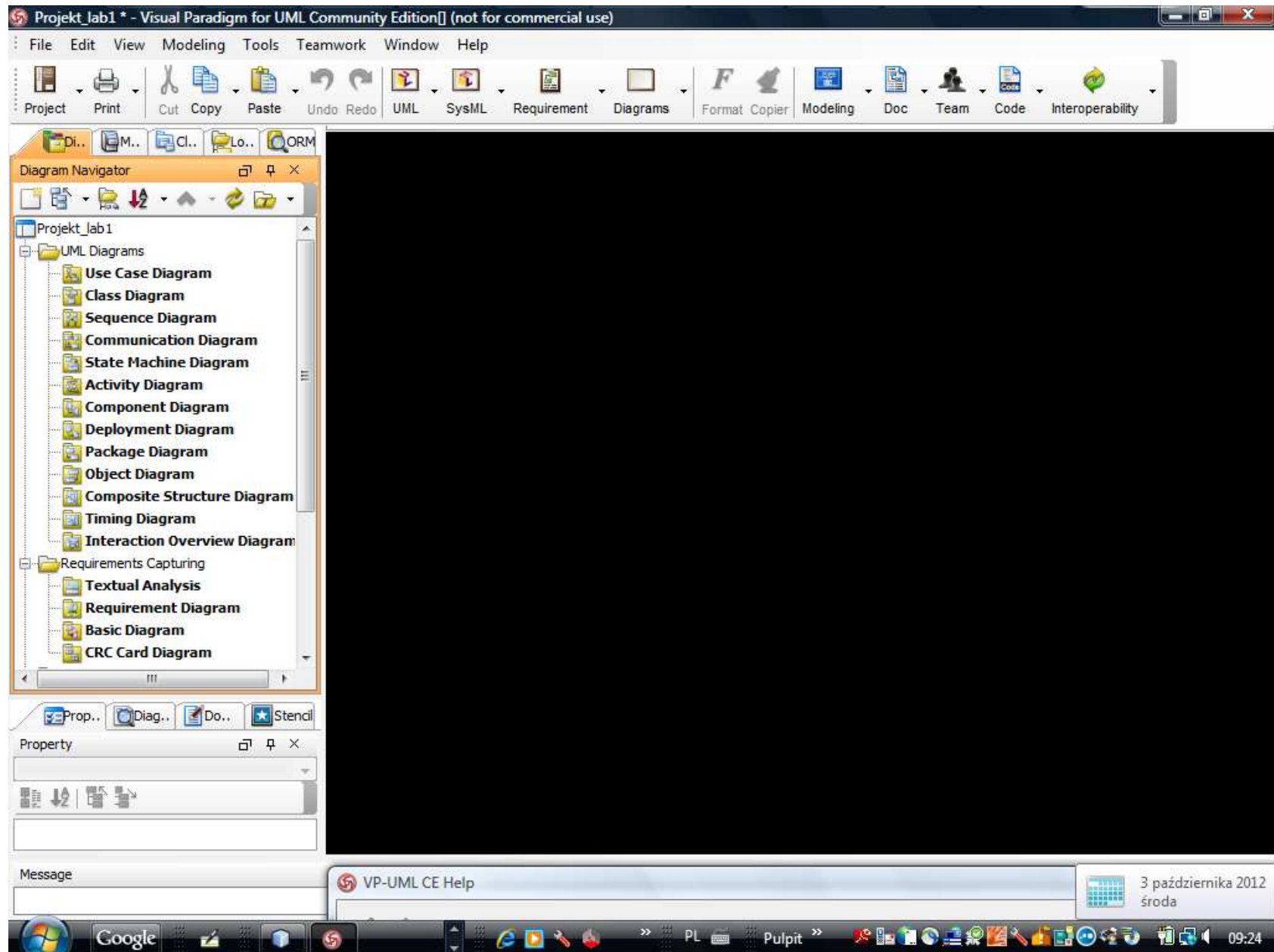
7.1. Utworzenie nowego projektu – *File/New Project*



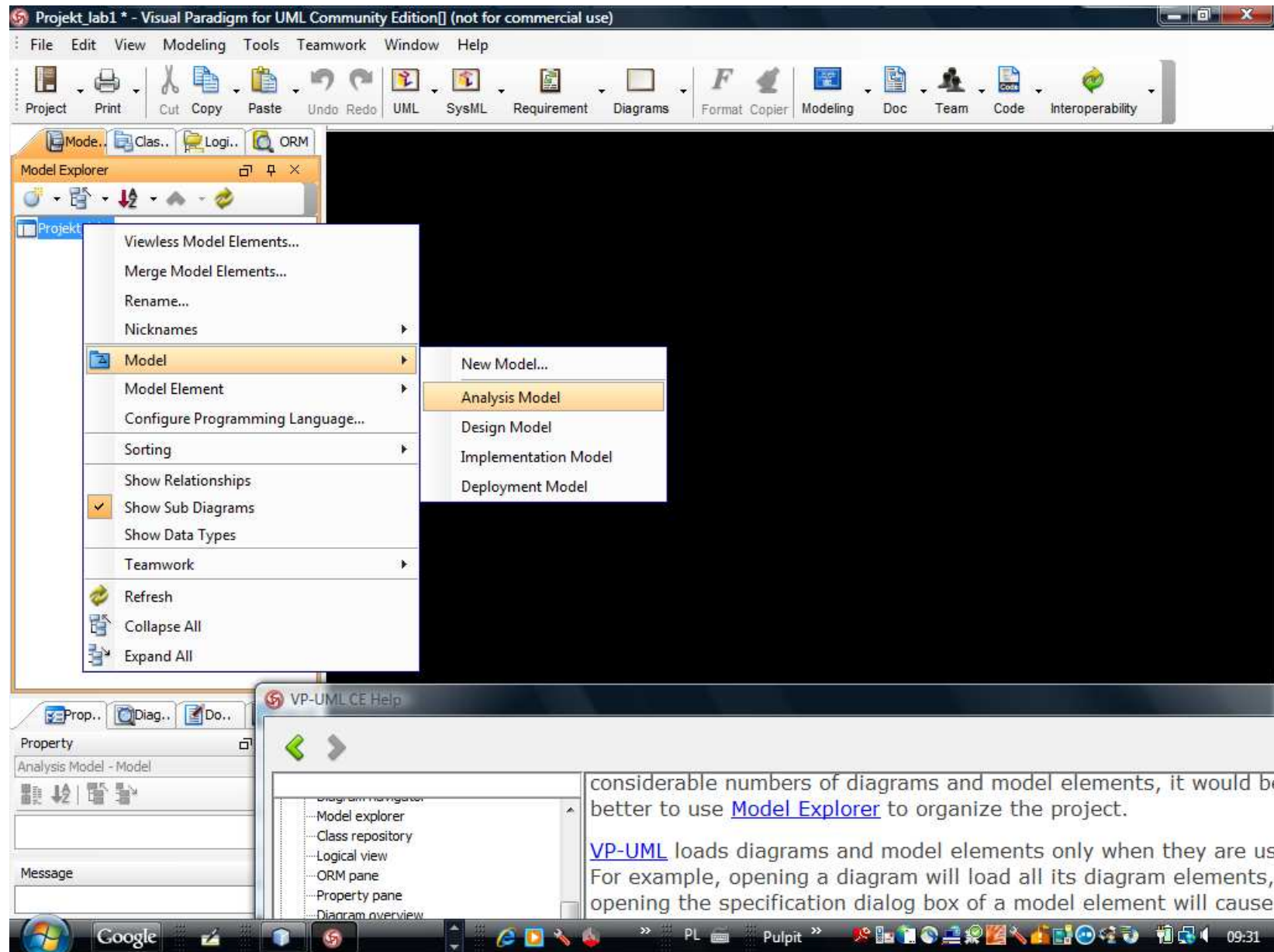
7.2. Utworzenie nowego projektu – nadanie nazwy nowemu projektowi w formularzu *New Project* w polu *Project name*



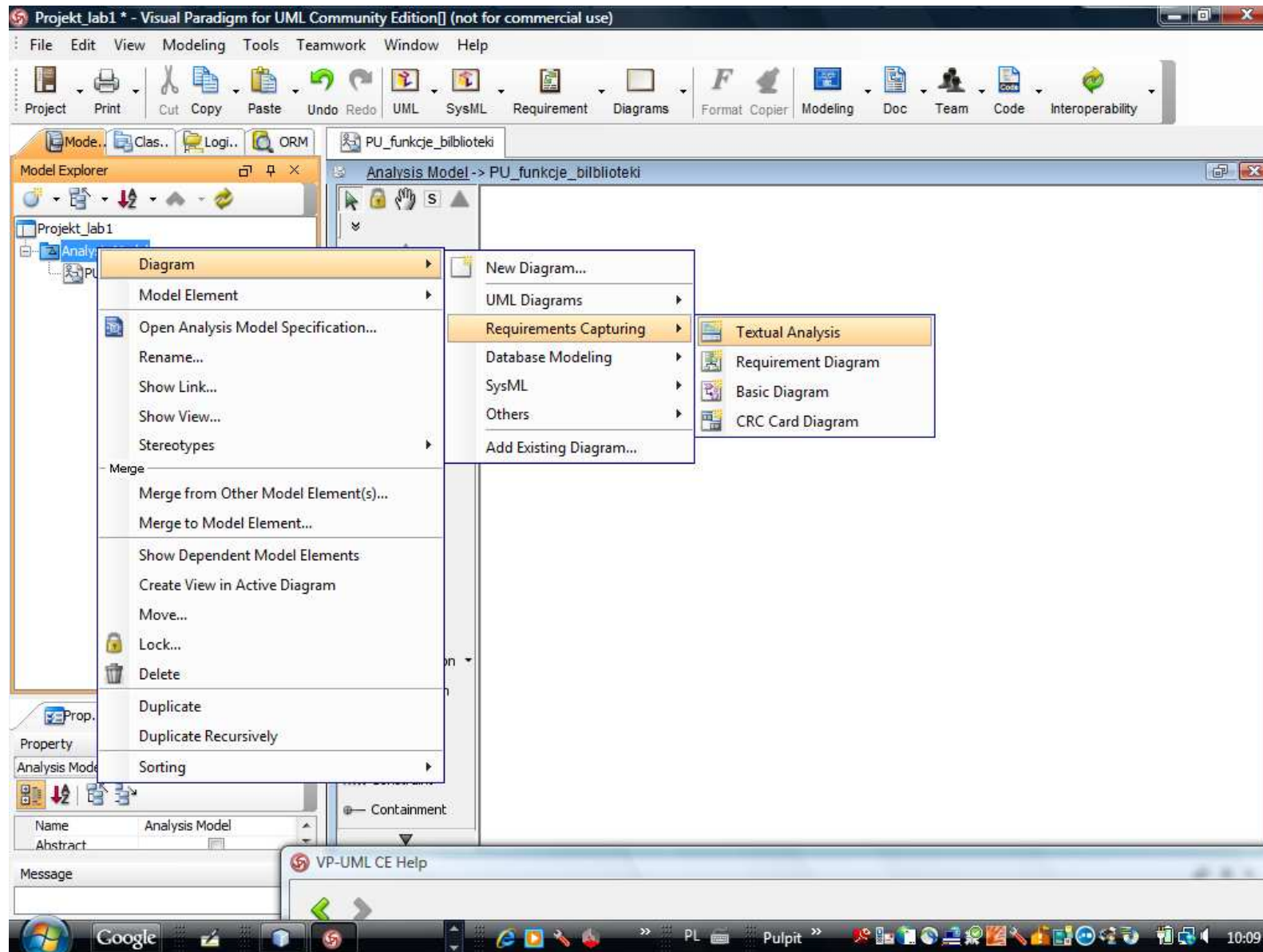
8. Widok nowego, pustego projektu – w stylu *Diagram Navigator*



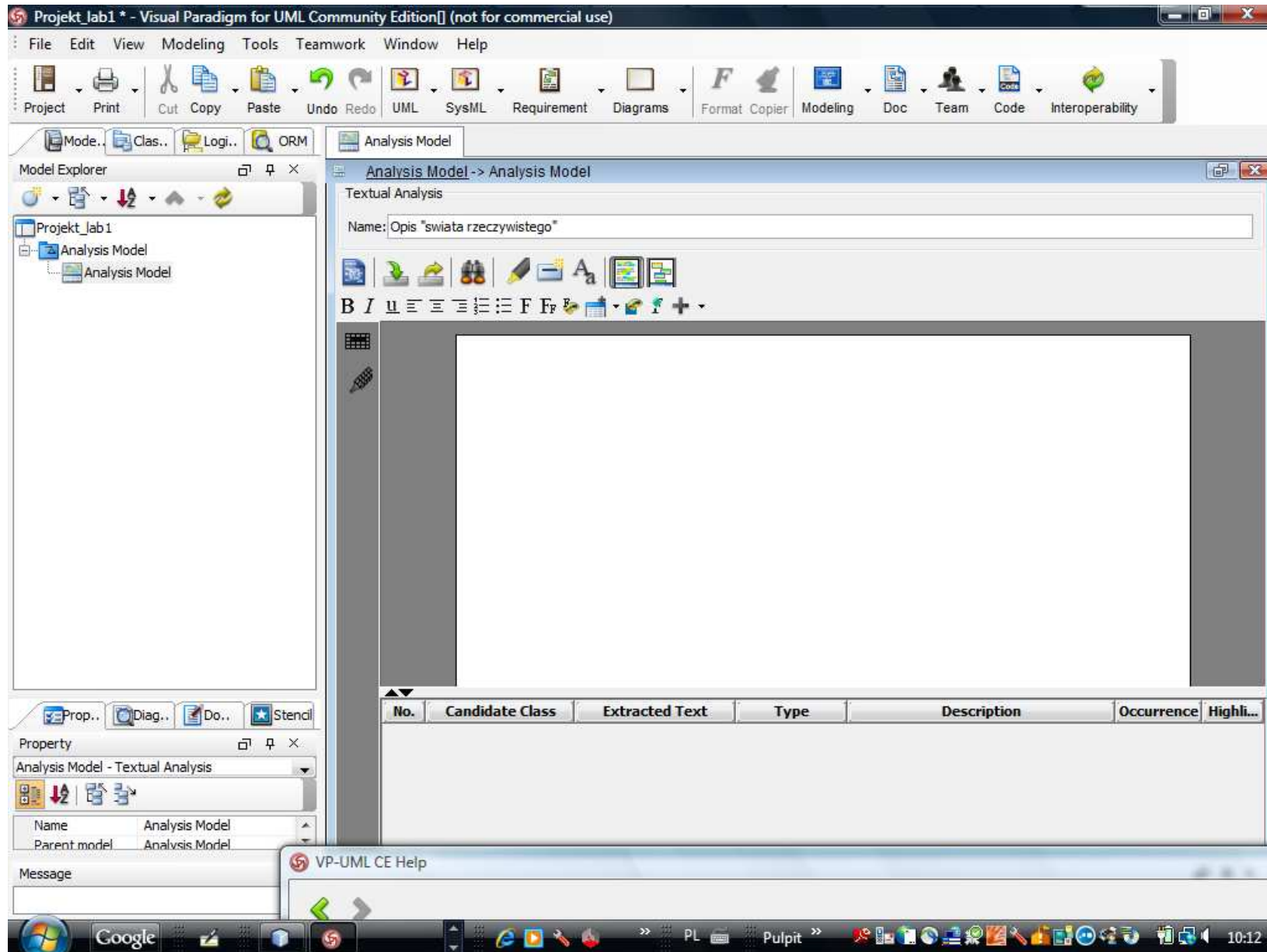
9. Utworzenie modelu analizy w projekcie – po kliknięciu prawym klawiszem na nazwę projektu należy wybrać z list: *Model/Analysis Model*



10.1. Dodanie diagramu reprezentującego opis „świata rzeczywistego”
– po kliknięciu prawym klawiszem myszy na nazwę modelu wybór z list:
Diagram/Requirements Caturing/Textual Analysis



10.2. Okno diagramu – należy nadać nazwę digramowi np. Opis „świata rzeczywistego” i należy wprowadzić tekst reprezentujący opis „świata rzeczywistego” projektu (następny slajd zawiera przykład opisu)



10.3. 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 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.

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 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.

10.4. Okno diagramu – wykonany opis

The screenshot shows the Visual Paradigm for UML Community Edition interface. The main window displays a textual analysis of a document titled "Opis zasobów ludzkich". The text is as follows:

Opis zasobów ludzkich.
Pracownik wydziału 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) informacje 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.

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

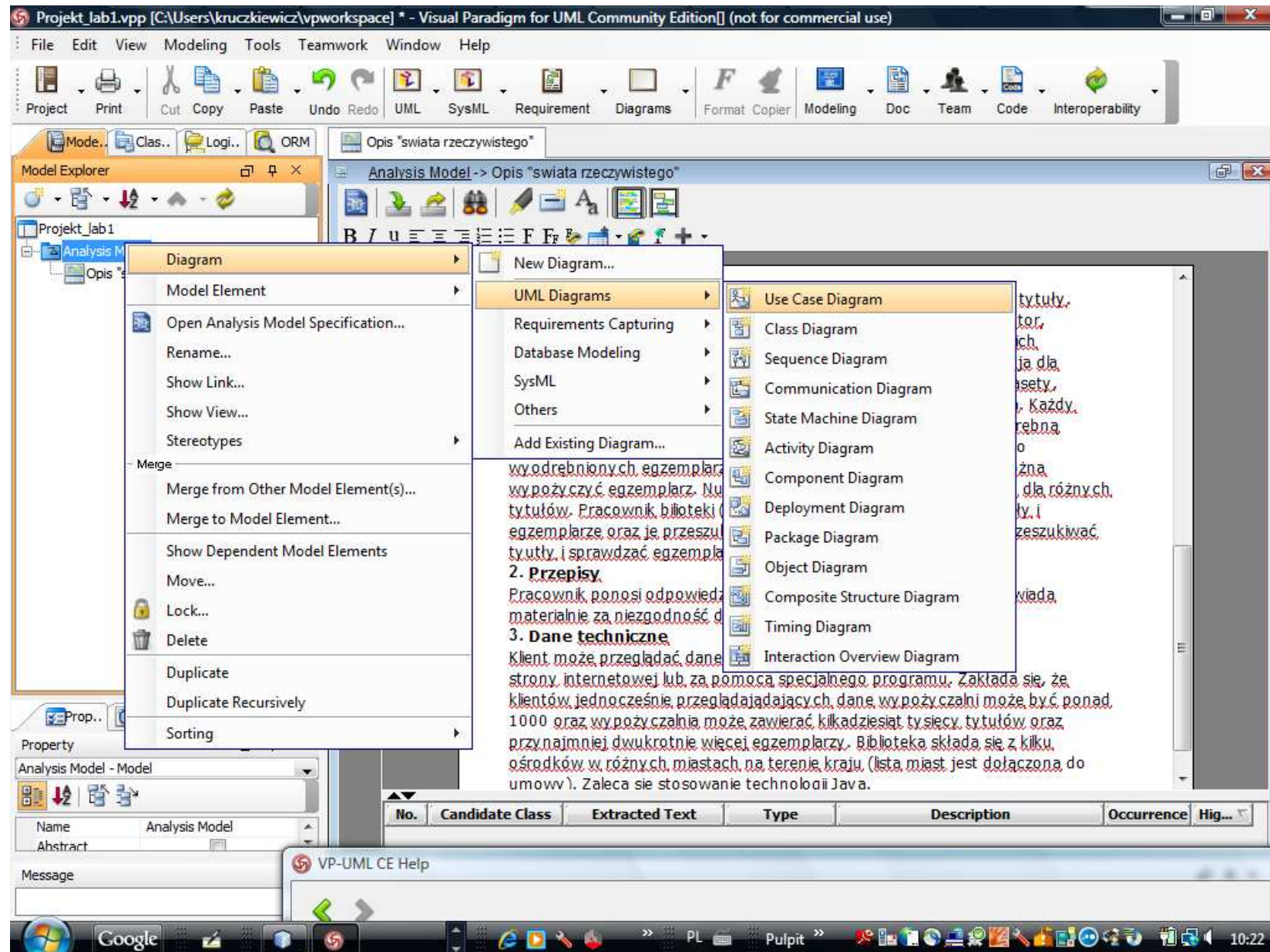
3. Dane techniczne.
Klient może przeglądać dane wypożyczalni za pośrednictwem strony internetowej lub za pomocą specjalnego programu. Zakłada się, że klientów jednocześnie przeglądających dane wypożyczalni może być ponad 1000 oraz wypożyczalnia może zawierać kilkadziesiąt tysięcy tytułów oraz przynajmniej 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.

At the bottom of the window, a table displays the results of the analysis:

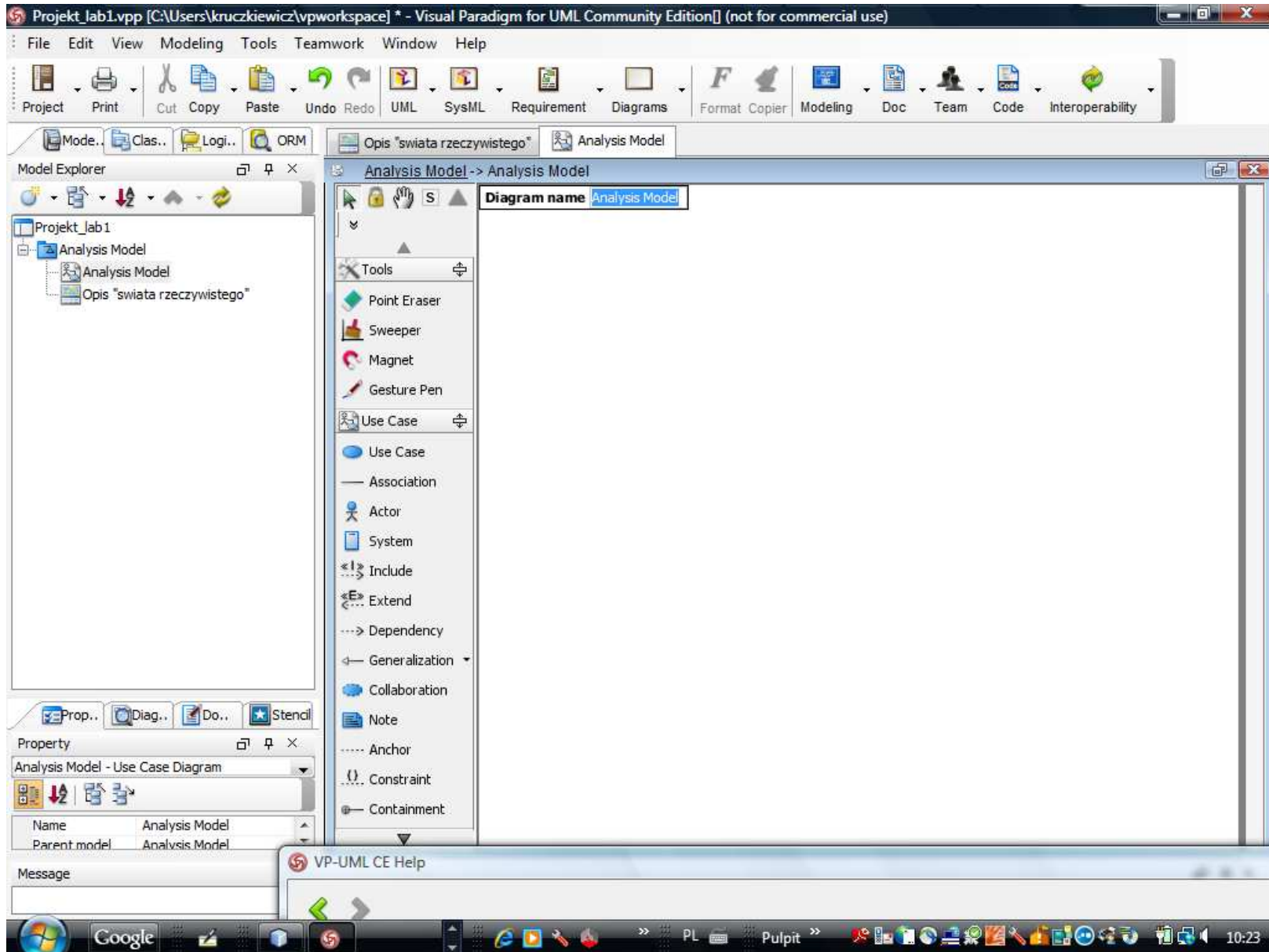
No.	Candidate Class	Extracted Text	Type	Description	Occurrence	Hig...
-----	-----------------	----------------	------	-------------	------------	--------

The Windows taskbar at the bottom shows the system clock at 10:19 and the language set to PL.

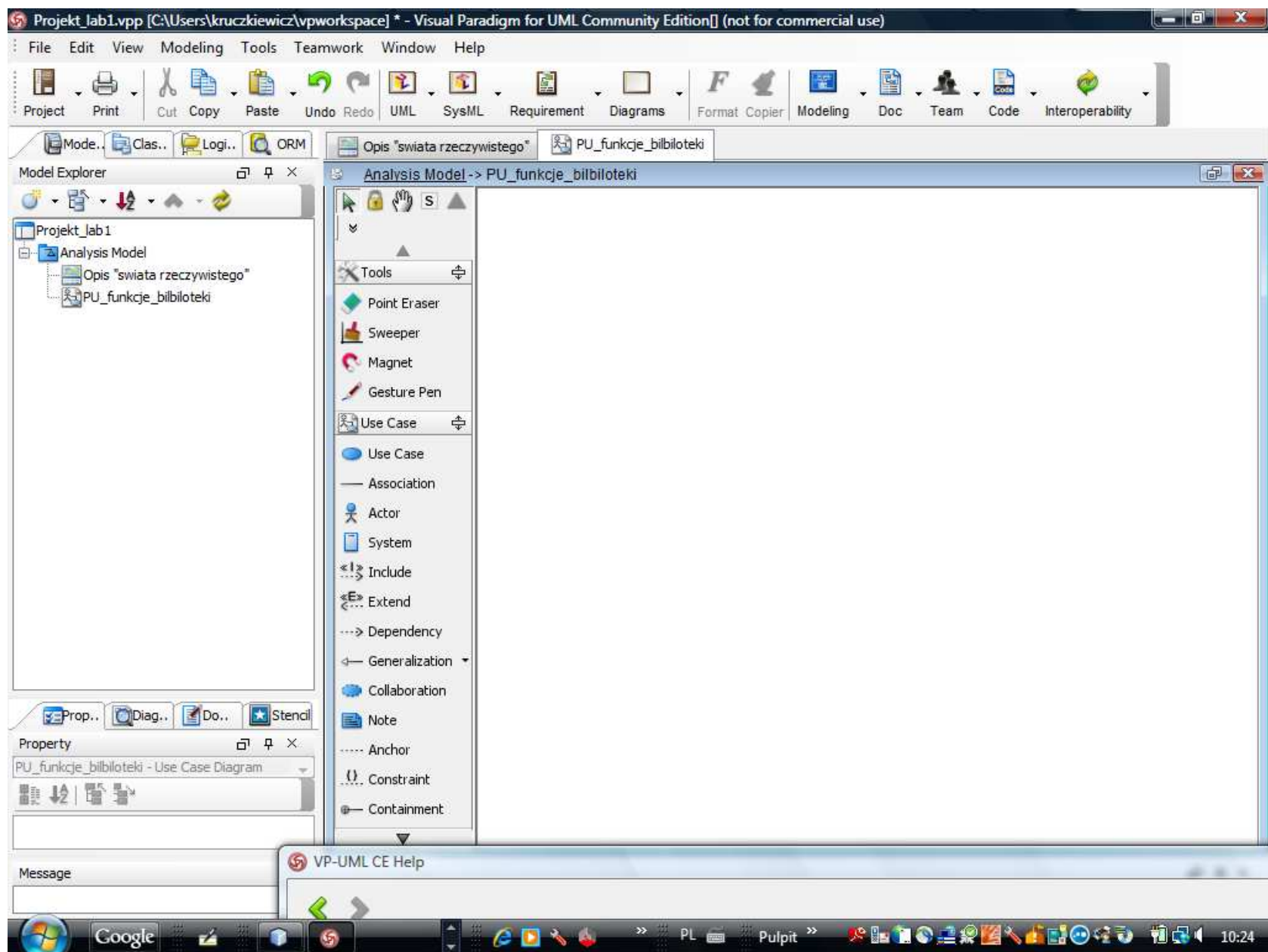
11.1. Wstawianie do projektu diagramu typu *Use Case* – po kliknięciu na nazwę modelu wybór z list: *Diagram/UML Diagram/ Use Case Diagram*



11.2. Wstawianie do projektu diagramu typu *Use Case* – nadanie nazwy diagramowi typu *Use Case*



11.3. Wstawianie do projektu diagramu typu *Use Case* – nadanie nazwy diagramowi typu *Use Case* np..**PU_funkcje_biblioteki**



12. 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 нефunkcjonalne

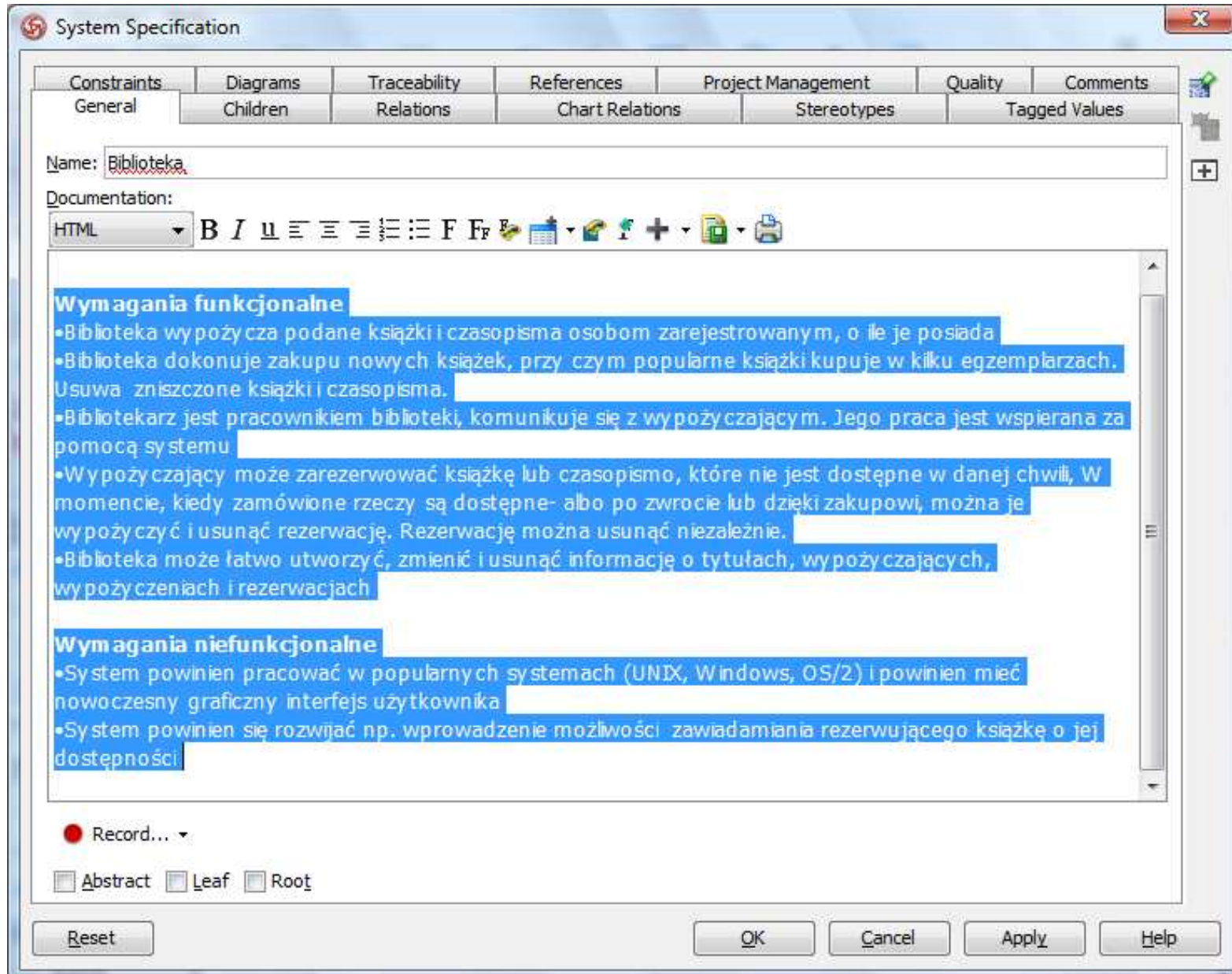
- System powinien pracować w popularnych systemach (UNIX, Windows, OS/2) i powinien mieć nowoczesny graficzny interfejs użytkownika
- System powinien się rozwijać np. wprowadzenie możliwości zawiadamiania rezerwującego książkę o jej dostępności

13. Wstawianie do projektu diagramu typu *Use Case* – definicja przypadków użycia specyfikujących wymagania stawiane aplikacji (p.12) w zakresie wstawiania zasobów biblioteki

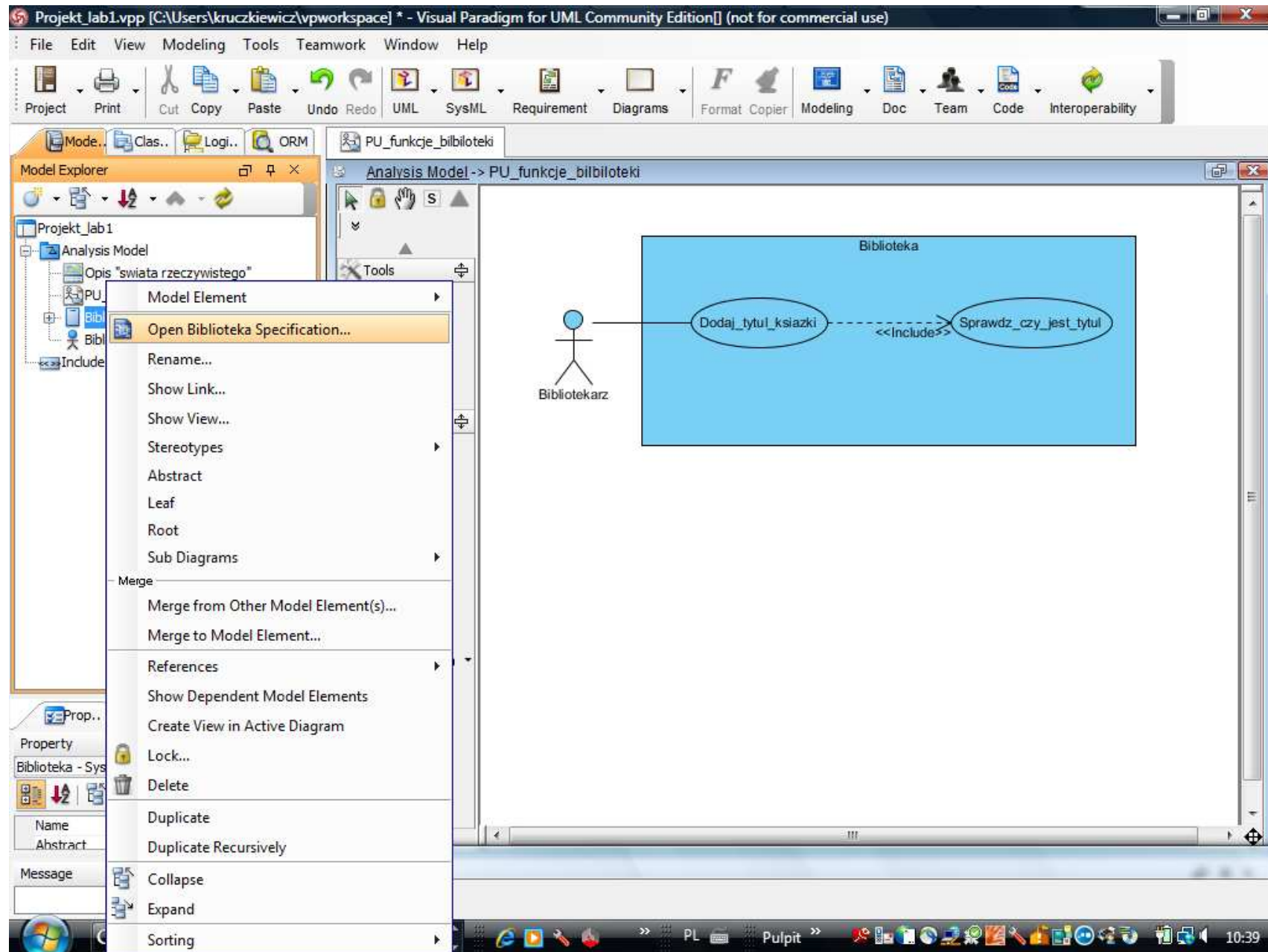
The screenshot displays the Visual Paradigm for UML Community Edition interface. The main workspace shows a Use Case Diagram for a system named 'Biblioteka'. An actor named 'Bibliotekarz' is connected to a use case 'Dodaj_tytul_książki'. This use case includes another use case 'Sprawdź_czy_jest_tytul' via an include relationship. The left sidebar contains the Model Explorer, Tools palette, and Property window. The Tools palette shows various UML elements like Use Case, Actor, System, Association, and Include. The Property window shows the name 'PU_funkcje_biblioteki' and the parent model 'Analysis.Model'.

Przeciągnięcie ikon **Actor**, **System**, **Use Case**, pobranych z palety lewym klawiszem myszy i upuszczenie na diagramie PU. Należy nadać im podane nazwy. Następnie, należy połączyć element **Actor** z PU **Dodaj_tytul_książki** relacją **Association**, przeciągając ją z palety (z lewej strony) lewym klawiszem myszy i następnie położyć ją na elemencie **Actor** i przeciągnąć na PU **Dodaj_tytul_książki**. Podobnie należy połączyć PUs relacją **<<Include>>**, przeciągając ją z palety - i następnie należy ją położyć na PU **Dodaj_tytul_książki** i przeciągnąć do PU **Sprawdź_czy_jest_tytul**.

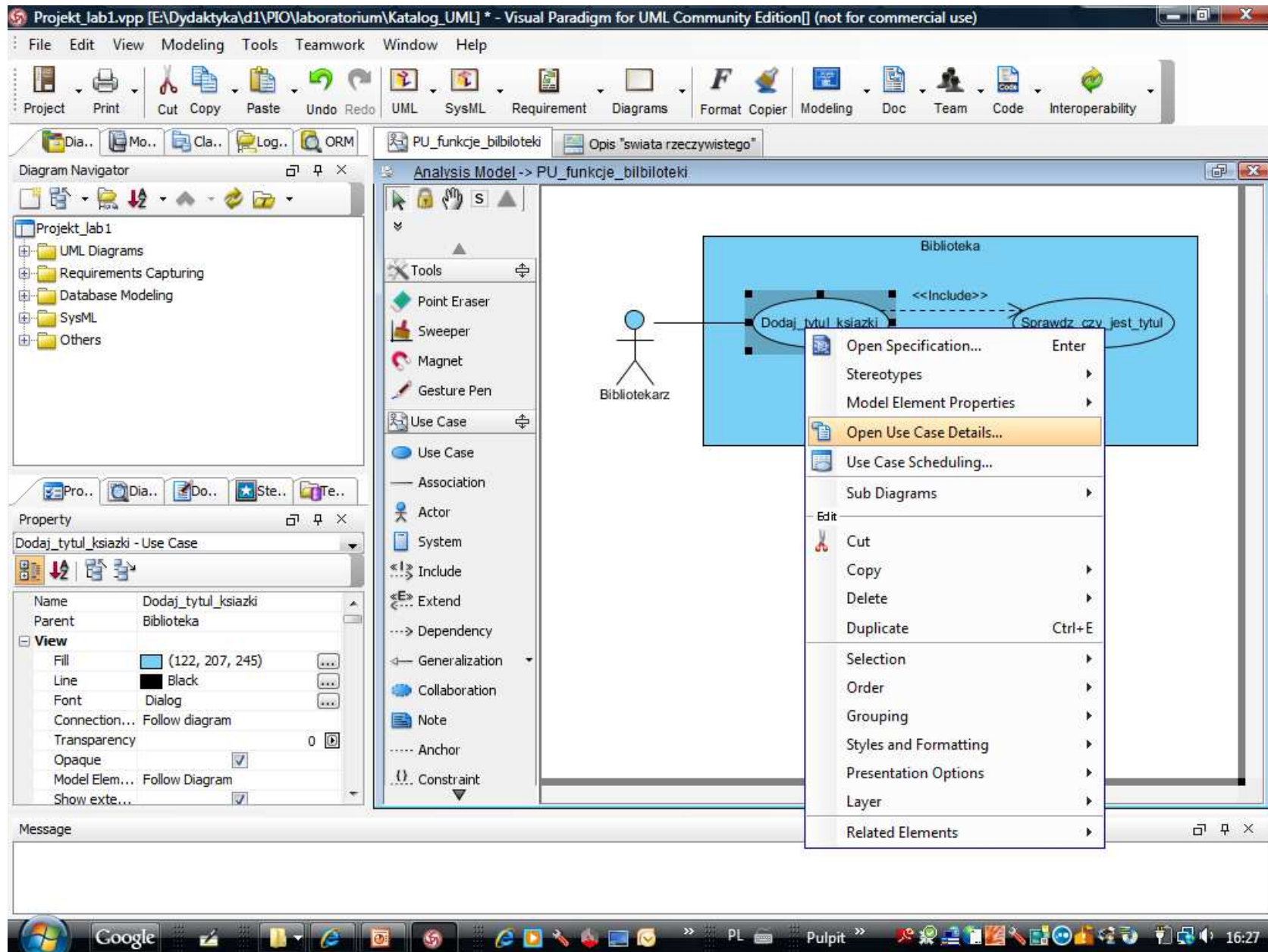
14.1. Po kliknięciu prawym klawiszem na element *System* (albo na diagramie albo w okienku *Model Explorer* – następny slajd), należy wybrać z listy *Open Specification*. Następnie, należy wprowadzić tekst wymagań stawianych aplikacji (p. 12)



14.2. Wybór okna specyfikacji elementu *System* po kliknięciu prawym klawiszem na nazwę *System* w okienku *Model Explorer*



15.1. Definiowanie elementów typu *Use Case* – po kliknięciu prawym klawiszem myszy na PU *Dodaj_tytul_ksiazki* należy dokonać wyboru z listy opcji *Open Use Case Details*



15.2. Nadanie wagi diagramowi – wybór wartości z listy w polu *Rank*

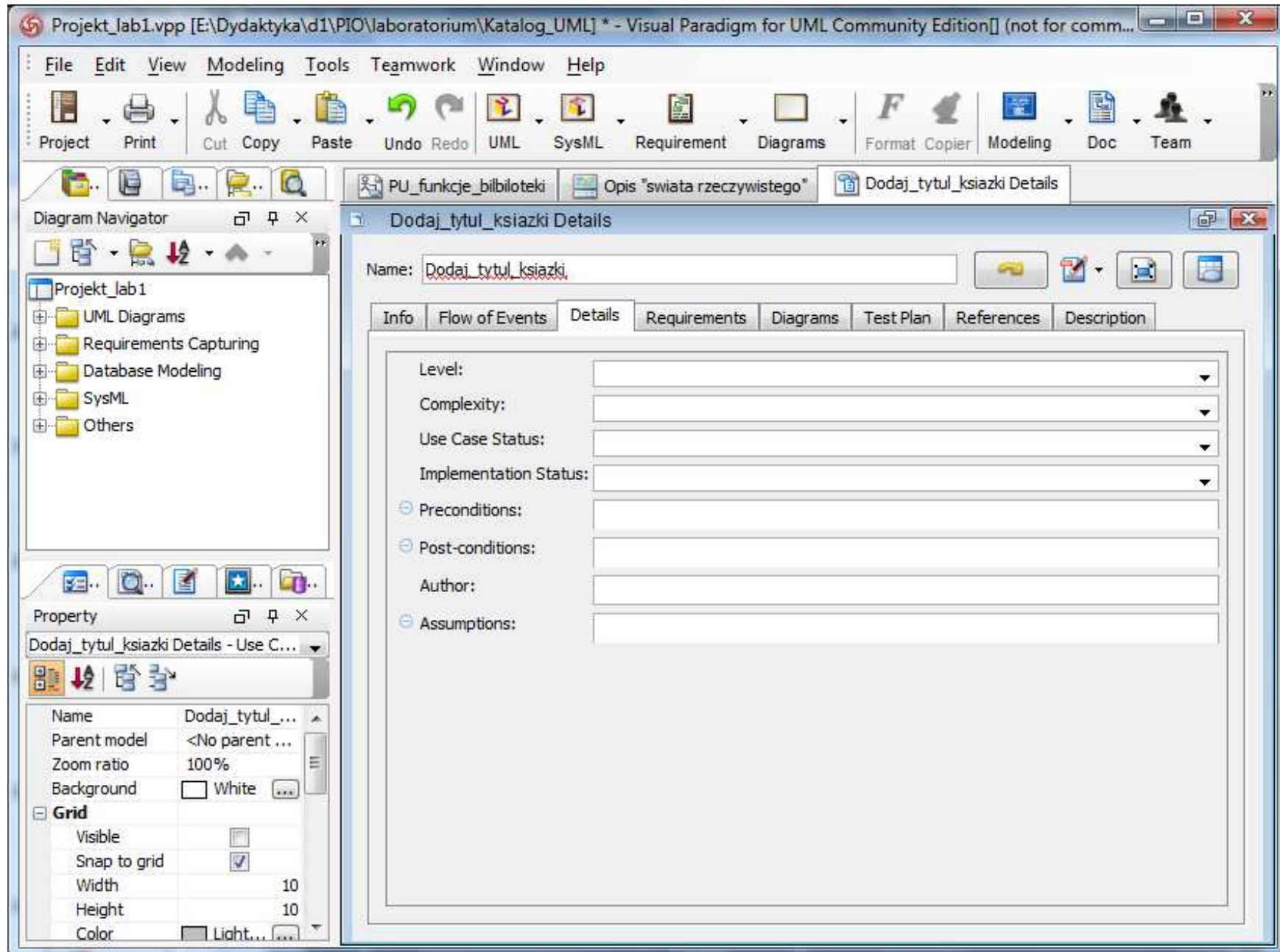
The screenshot displays the Visual Paradigm for UML Community Edition interface. The main window is titled "Projekt_lab1.vpp [E:\Dydaktyka\d1\PIO\laboratorium\Katalog_UML] * - Visual Paradigm for UML Community Edition[] (not for commercial use)". The menu bar includes File, Edit, View, Modeling, Tools, Teamwork, Window, and Help. The toolbar contains various icons for Project, Print, Cut, Copy, Paste, Undo, Redo, UML, SysML, Requirement, Diagrams, Format, Copier, Modeling, Doc, Team, Code, and Interoperability.

The Diagram Navigator on the left shows a project named "Projekt_Jab1" with subfolders for UML Diagrams, Requirements Capturing, Database Modeling, SysML, and Others. The Property window shows details for "Dodaj_tytul_ksiazki Details - Use Case Details", including Name, Parent model, Zoom ratio, Background, Grid settings, and Connector style.

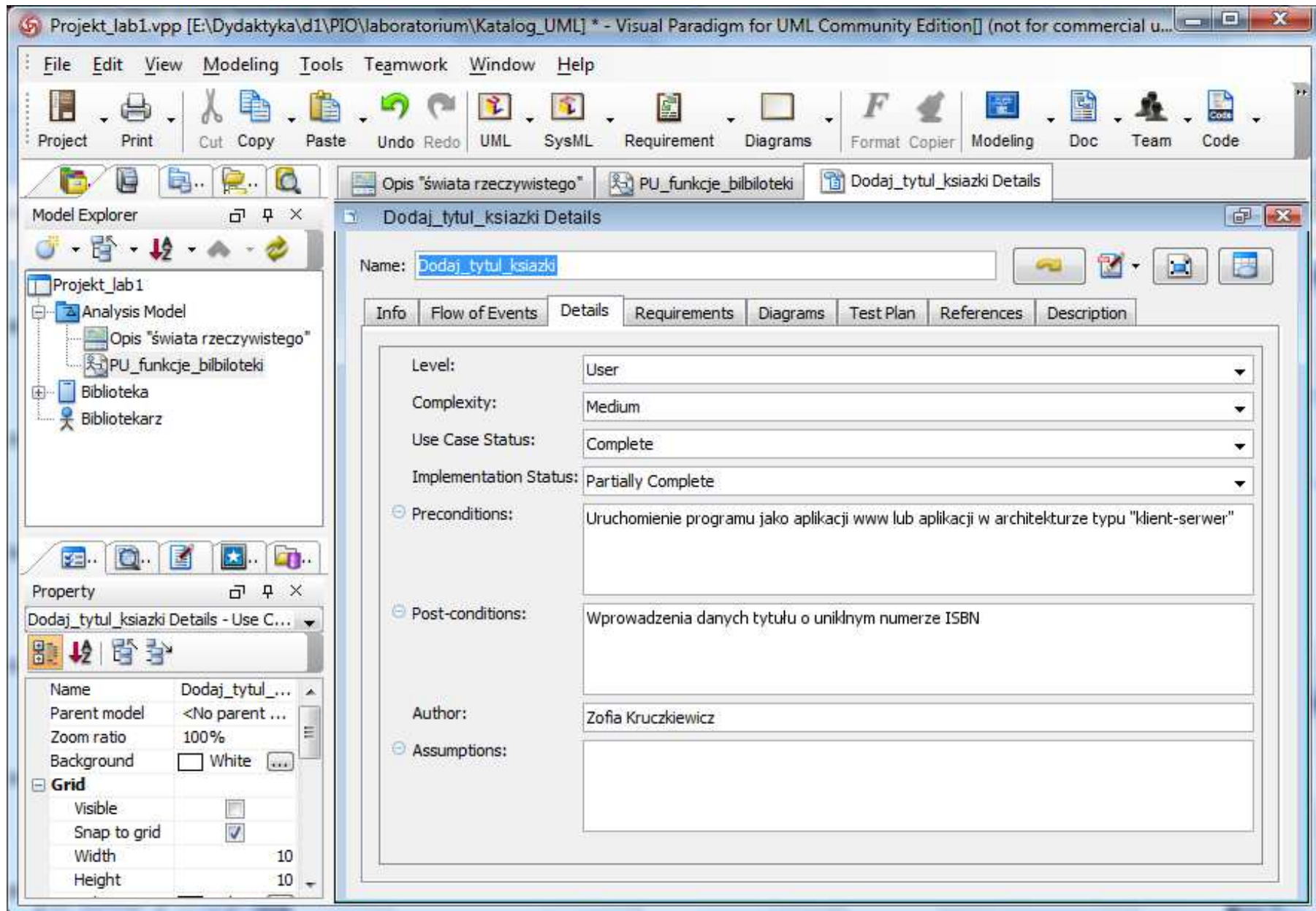
The main dialog box, "Dodaj_tytul_ksiazki Details", has a Name field containing "Dodaj_tytul_ksiazki.". The "Info" tab is active, showing a list of properties: Rank, Justification, Primary Actors, and Supporting Actors. The "Rank" dropdown menu is open, displaying options: <Unspecified>, Low, Medium (selected), and High. The "Documentation" section includes an HTML editor with a toolbar and a text area. At the bottom, there are checkboxes for "Abstract", "Leaf", and "Root".

The Windows taskbar at the bottom shows the Start button, Google search, and several application icons. The system tray on the right displays the language "PL", the name "Pulpit", and the time "16:28".

15.3. Wybór podformularza *Details* związanego z wybranym wcześniej PU



15.4. Wybór podformularza *Details* związanego z wybranym wcześniej PU – nadanie wartości poszczególnym polom formularza przez wybór z listy lub wprowadzenie tekstu

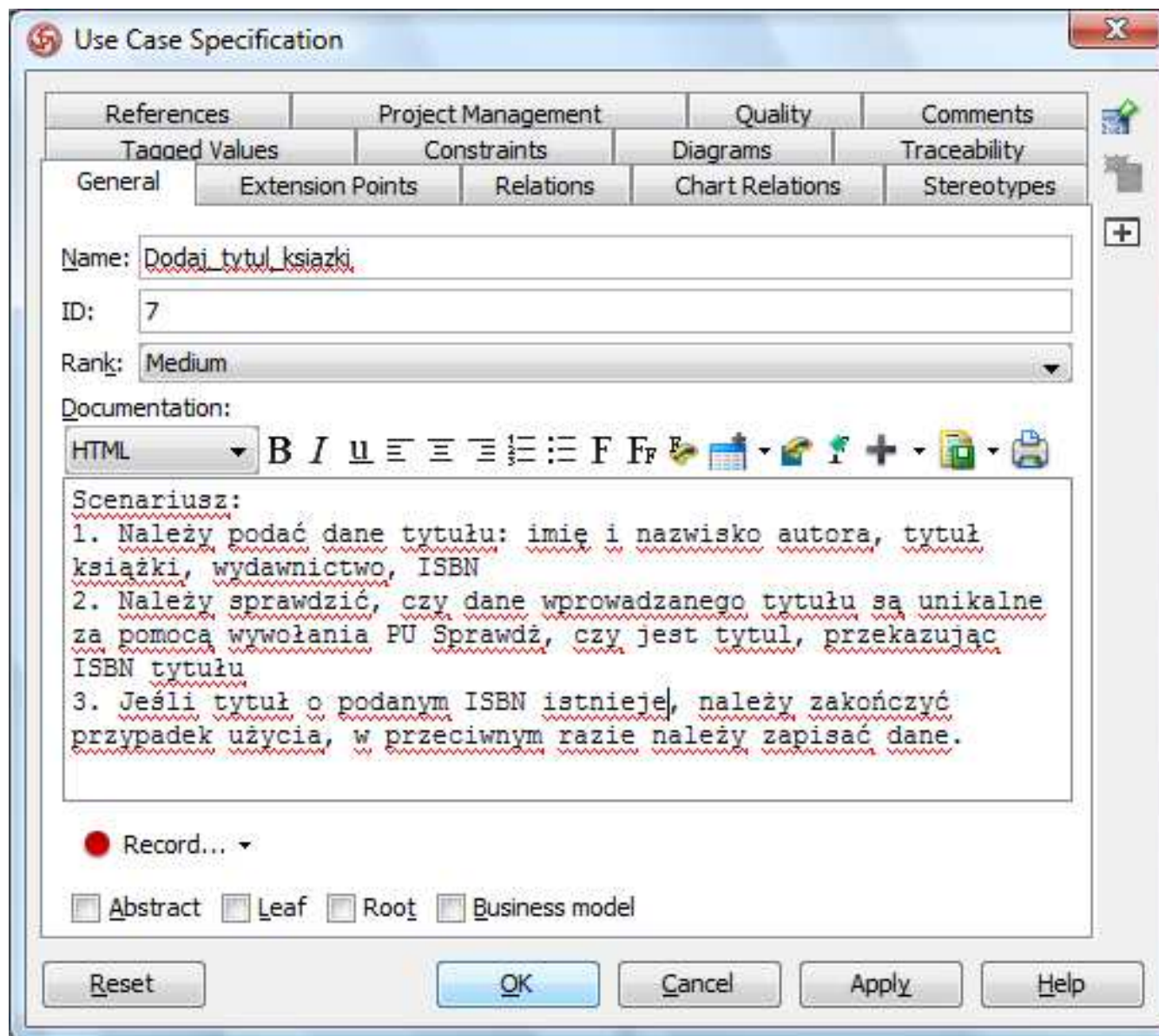


15.5. Wpisanie do podformularza *Info* w części *Documentation* scenariusza przypadku użycia *Dodaj_tytul*

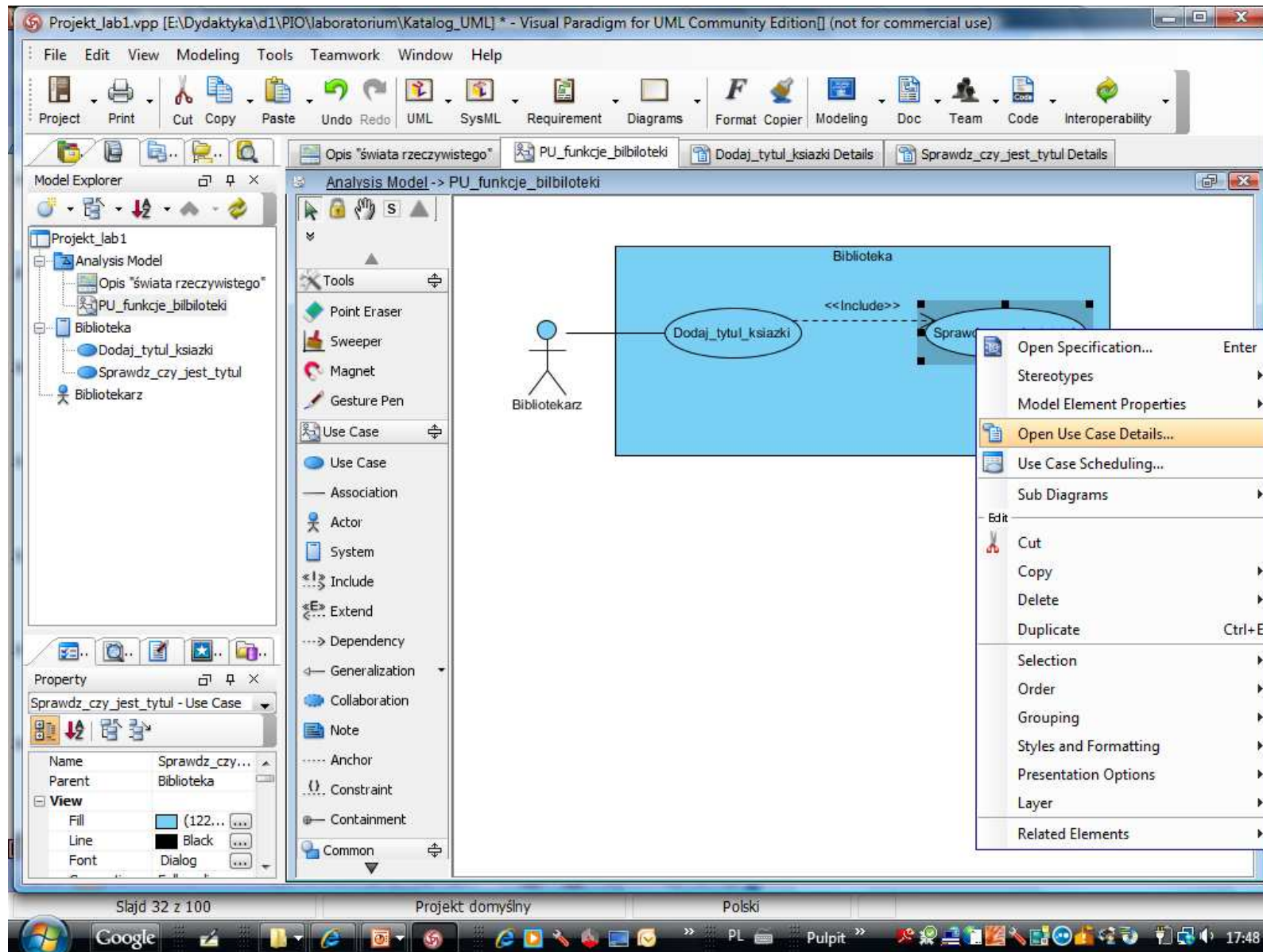
The screenshot displays the Visual Paradigm for UML Community Edition interface. The main window is titled 'Projekt_lab1.vpp [E:\Dydaktyka\d1\PIO\laboratorium\Katalog_UML] * - Visual Paradigm for UML Community Edition[] (not f...'. The menu bar includes File, Edit, View, Modeling, Tools, Teamwork, Window, and Help. The toolbar contains icons for Project, Print, Cut, Copy, Paste, Undo, Redo, UML, SysML, Requirement, Diagrams, Format Copier, Modeling, and Doc. The Model Explorer on the left shows a project structure with 'Projekt_lab1' containing 'Analysis Model', 'Opis "świata rzeczywistego"', 'PU_funkcje_biblioteki', 'Biblioteka', and 'Bibliotekarz'. The Property window shows details for 'Dodaj_tytul_książki Details - Use C...'. The main area displays the 'Dodaj_tytul_książki Details' form with the following fields and options:

- Name:
- Rank:
- Justification:
- Primary Actors:
- Supporting Actors:
- Documentation:
- Options: Abstract Leaf Root

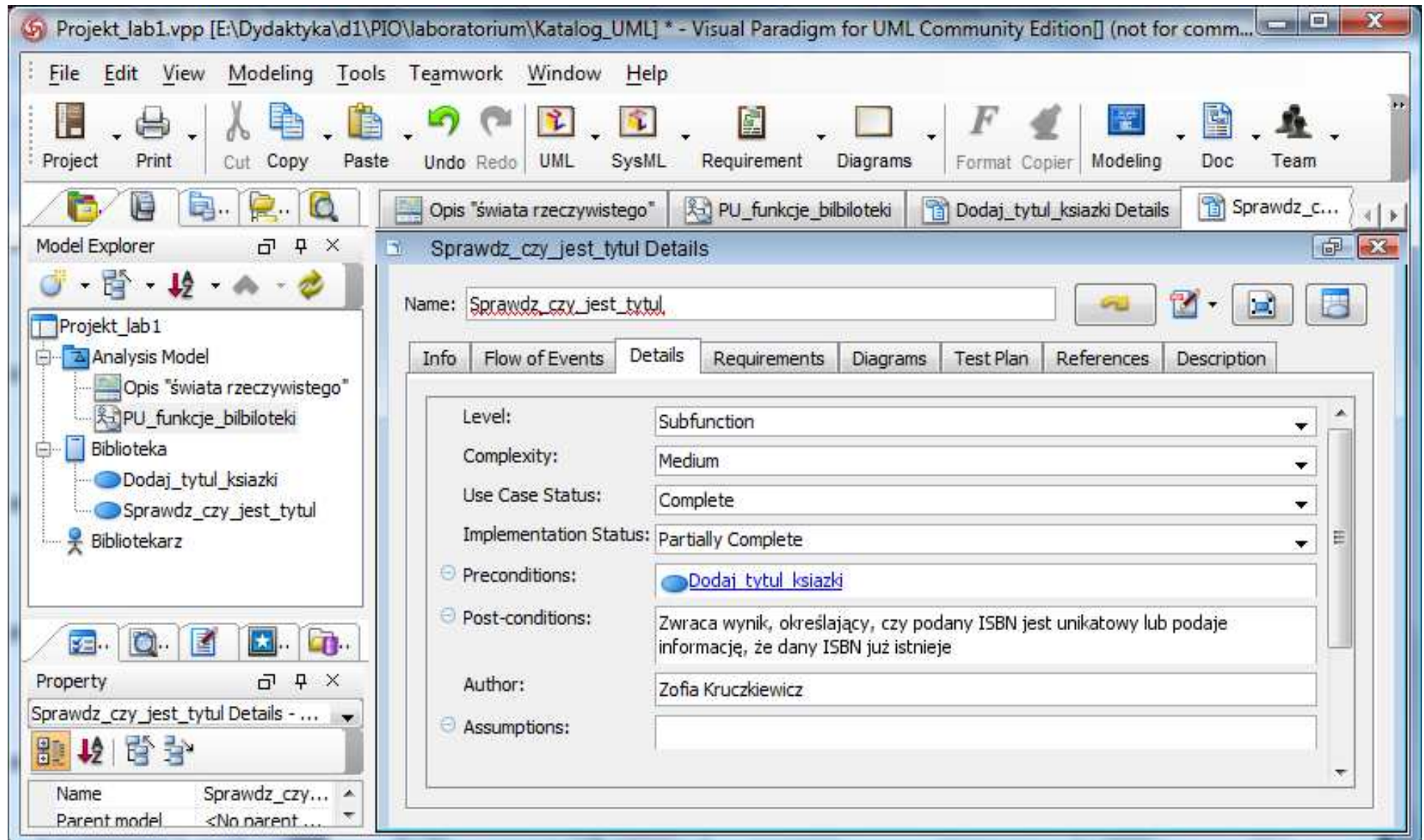
15.6. Po kliknięciu prawym klawiszem myszy na PU *Dodaj_tytul_ksiazki* należy wybrać z listy opcję *Open Specification* – powinien ukazać tekst wprowadzony wcześniej w podformularzu *Info* opcji *Open Use Case Details*



15.7. Definiowanie elementów typu Use Case – po kliknięciu prawym klawiszem myszy na PU *Sprawdz_czy_jest_tytul* należy dokonać wyboru z listy opcji *Open Use Case Details*



15.8. Wybór podformularza *Details* związanego z wybranym wcześniej PU – nadanie wartości poszczególnym polom formularza przez wybór z listy lub wprowadzenie tekstu



15.9. Wpisanie do podformularza *Info* w części *Documentation* scenariusza przypadku użycia PU *Sprawdz_czy_jest_tytul*

The screenshot shows the Visual Paradigm for UML Community Edition interface. The main window displays the 'Sprawdz_czy_jest_tytul Details' dialog box. The 'Info' tab is selected, showing the following fields:

- Name: Sprawdz_czy_jest_tytul
- Rank: Medium
- Justification: (empty)
- Primary Actors: (empty)
- Supporting Actors: (empty)

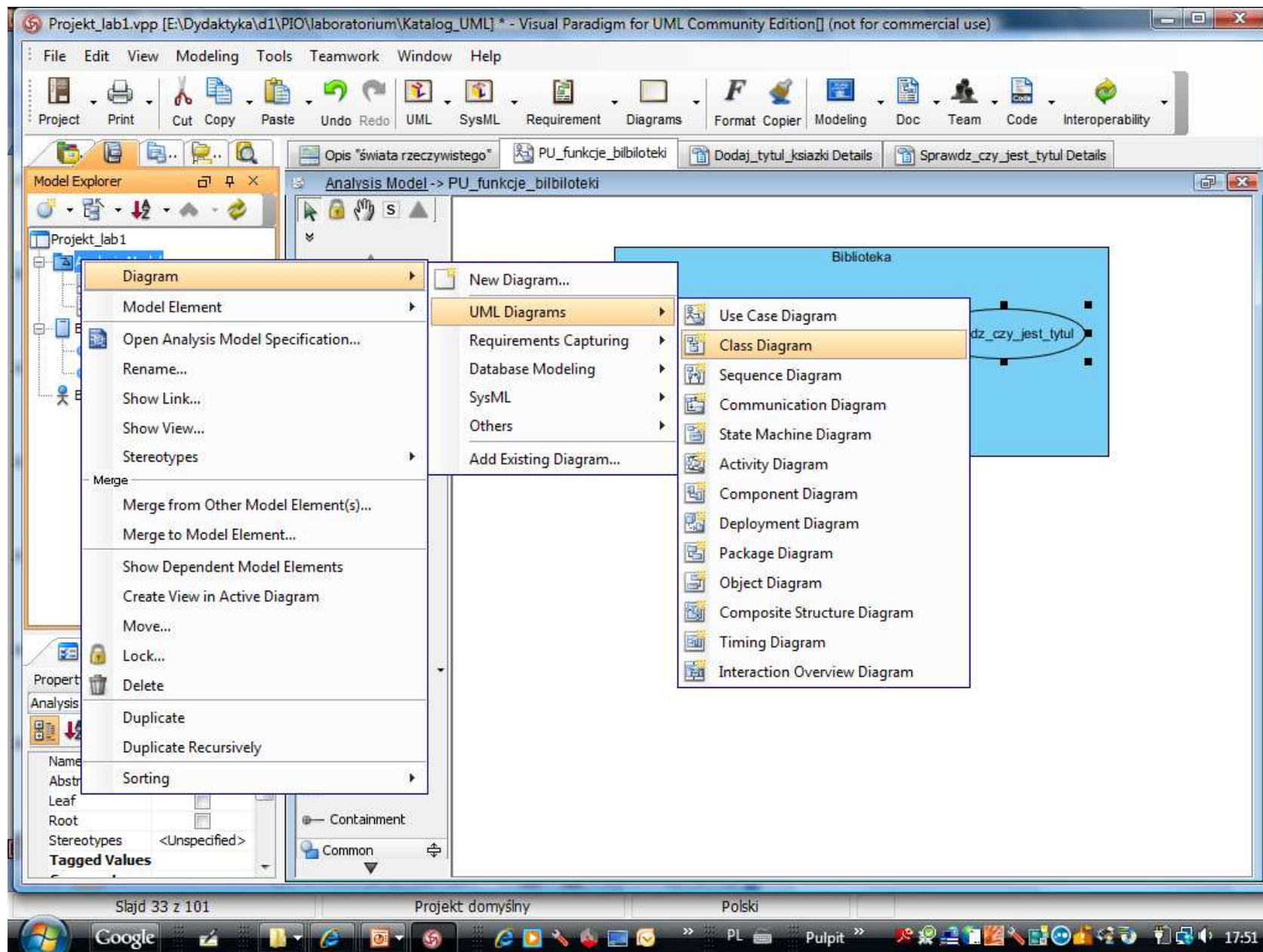
The Documentation section is visible, showing a rich text editor with a list of three items:

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 przejrzaniu tytułów książek, zwracany jest wynik negatywny

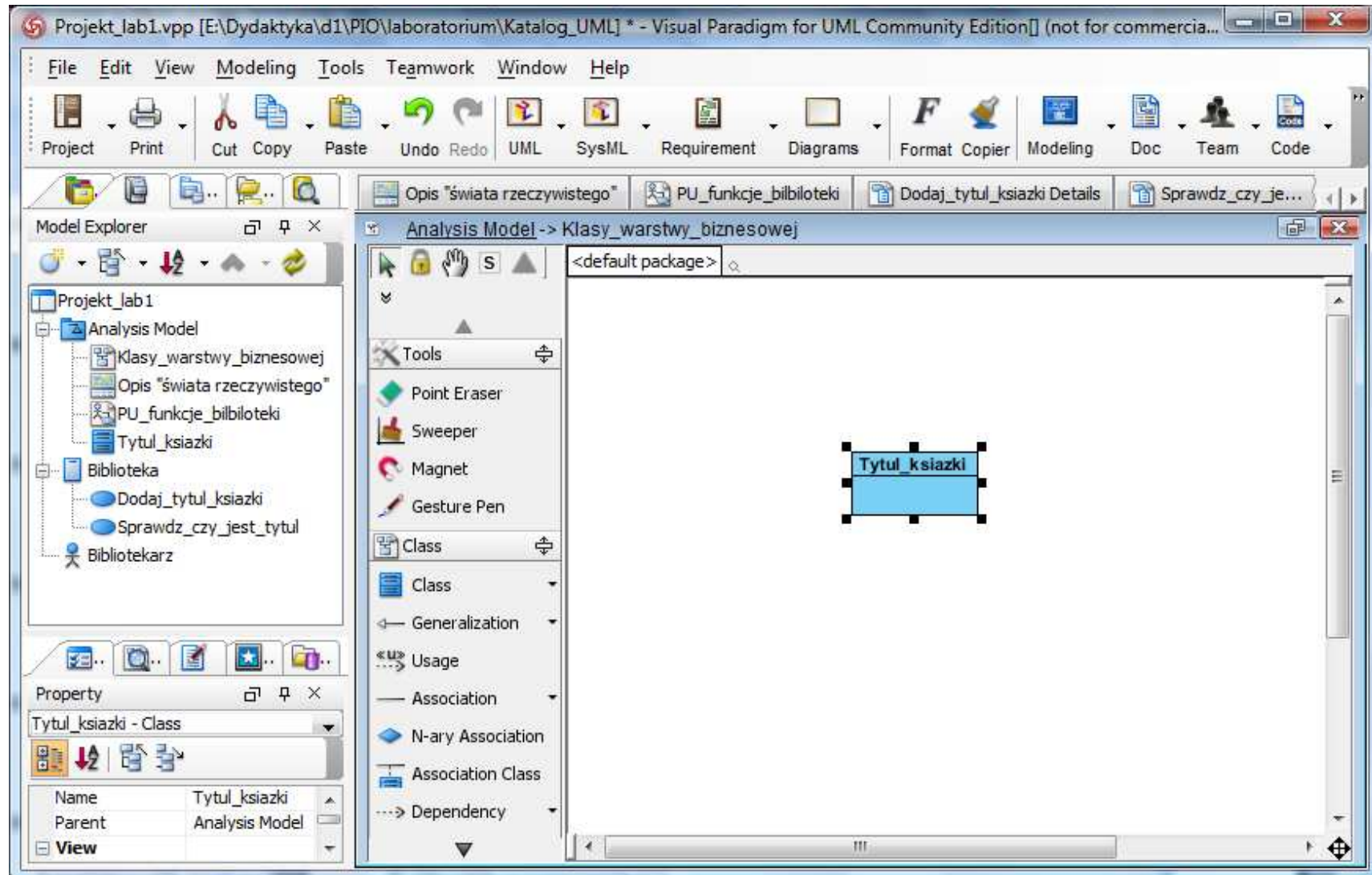
The Property window at the bottom left shows the following information:

Name	Sprawdz_czy...
Parent model	<No parent ...
Zoom ratio	100%

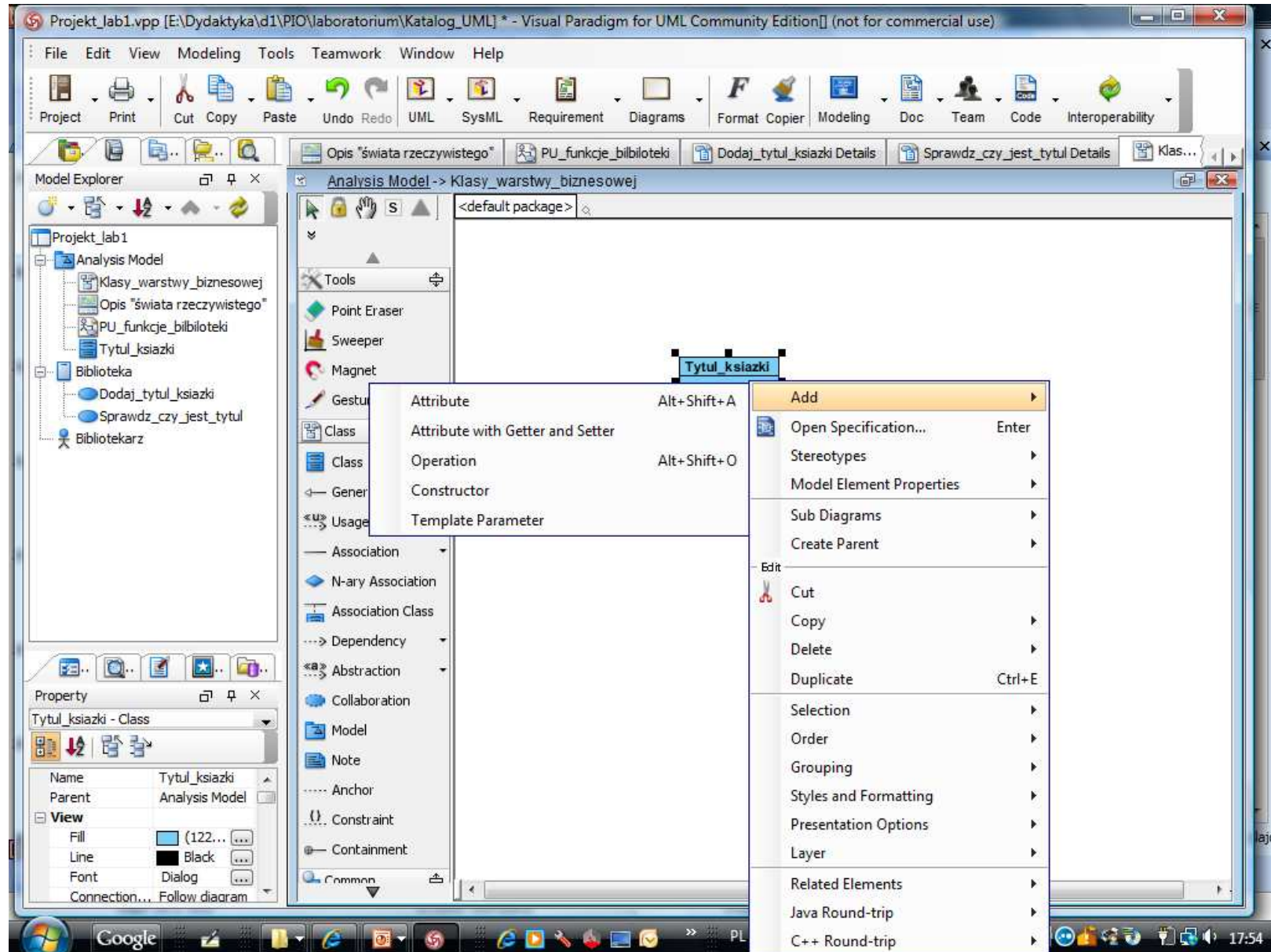
16. Dodanie diagramu klas do projektu – należy kliknąć prawym klawiszem na nazwę modelu w okienku *Model Explorer* i wybrać z listy *Diagram/UML Diagram/Class Diagram*



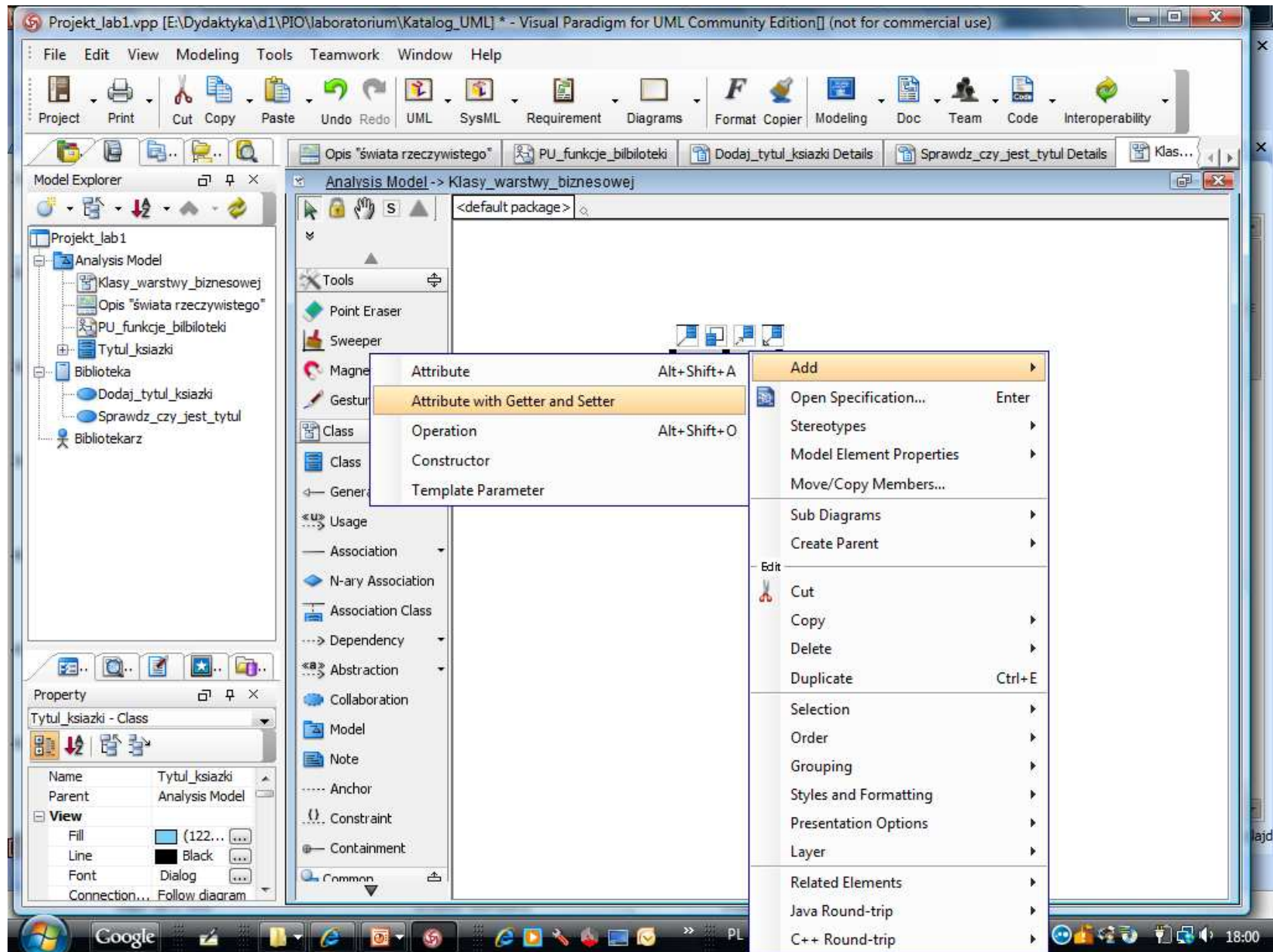
17.1. Po nadaniu nazwy diagramowi klas warstwy biznesowej jako **Klasy_warstwy_biznesowej** należy zdefiniować klasy zidentyfikowane na podstawie scenariuszy przypadków użycia. Pierwsza definiowana klasa zawiera dane tytułu książki – należy przeciągnąć ikonę klasy z palety lewym klawiszem myszy i położyć na diagramie i nadać jej nazwę **Tytul_książki**



17.2. Zdefiniowanie atrybutów i metod – po kliknięciu prawym klawiszem na klasę należy wybrać z listy pozycje *Attribute* do definiowania nowych atrybutów lub *Operation* do definiowania metod



17.3. Zdefiniowanie atrybutów i metod dostępu do atrybutów – po kliknięciu prawym klawiszem na klasę należy wybrać z listy pozycję *Attribute with Getter and Setter*



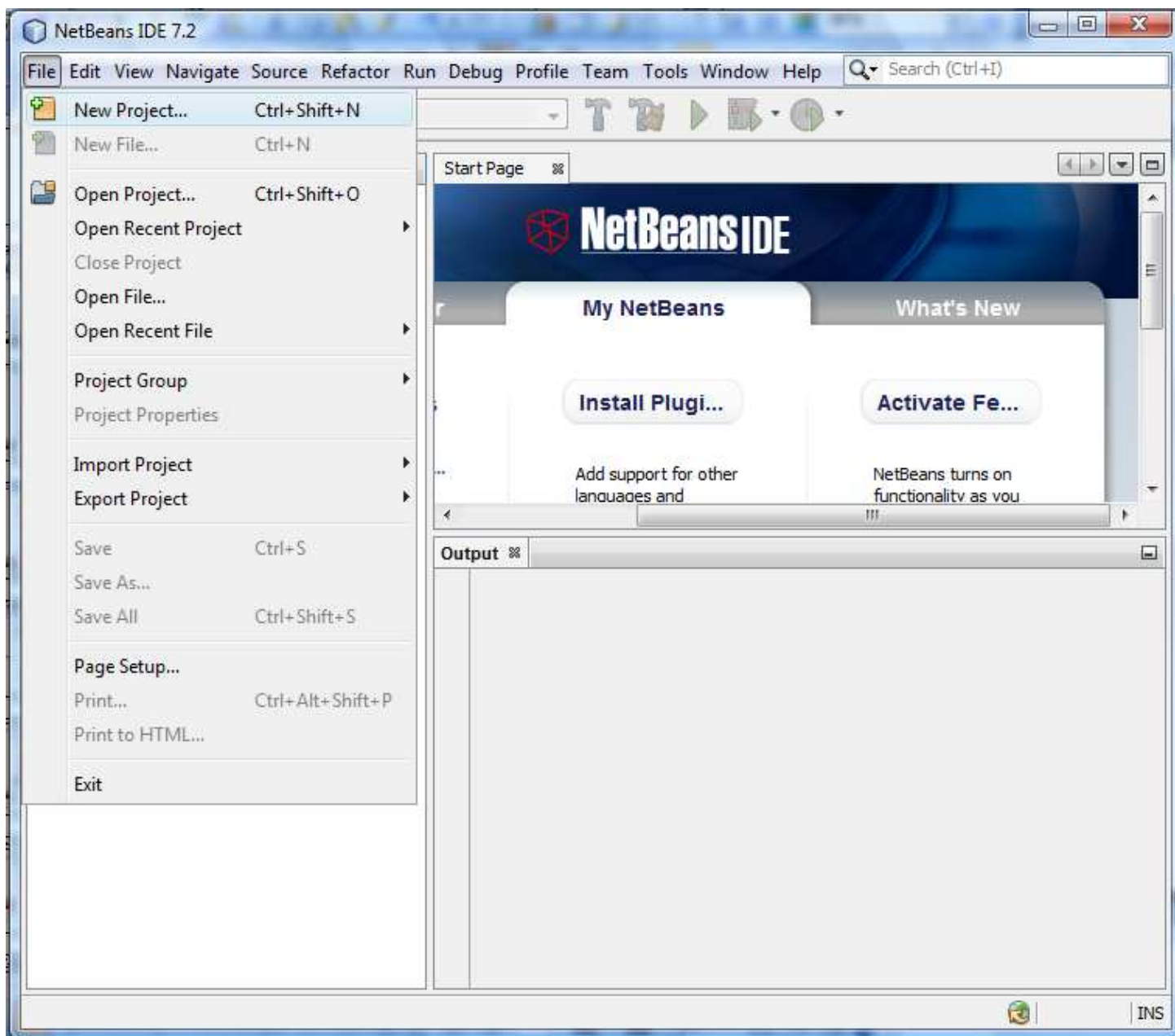
17.4. Dodano prywatne atrybuty i publiczne metody dostępu do atrybutów typu get i set klasie **Tytul_książki**

The screenshot displays the Visual Paradigm for UML interface. The main workspace shows a class diagram for the 'Klasy_warstwy_biznesowej' package. The class 'Tytul_książki' is highlighted, showing its internal structure:

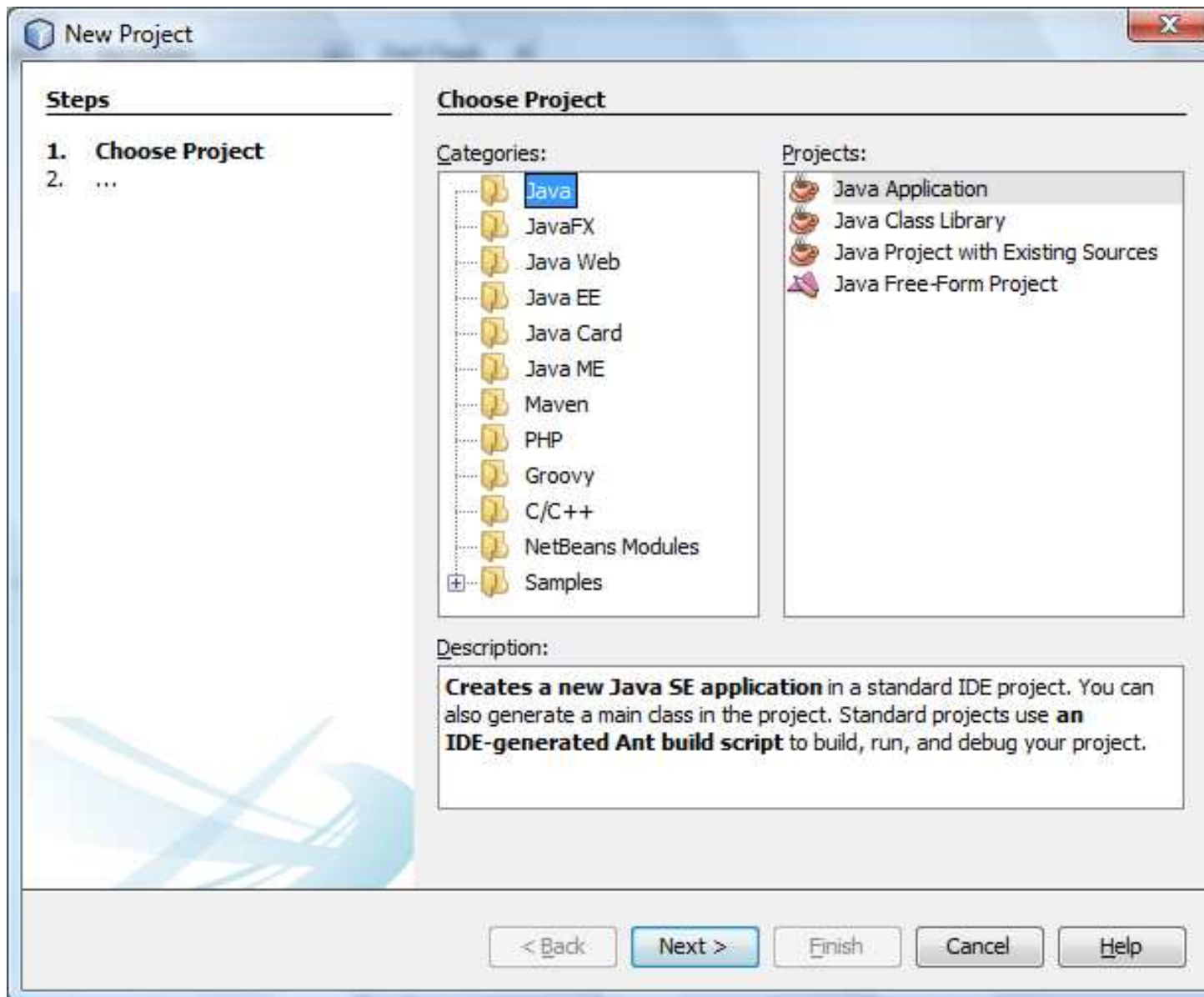
- Private attributes (indicated by a minus sign):
 - tytul
 - imie
 - nazwisko
 - wydawnictwo
 - ISBN
- Public methods (indicated by a plus sign):
 - +getTytul()
 - +setTytul(tytul) : void
 - +getImie()
 - +setImie(imie) : void
 - +getNazwisko()
 - +setNazwisko(nazwisko) : void
 - +getWydawnictwo()
 - +setWydawnictwo(wydawnictwo) : void
 - +getISBN()
 - +setISBN(ISBN) : void

The interface also shows the Model Explorer on the left, the Property window at the bottom, and the Tools palette in the center.

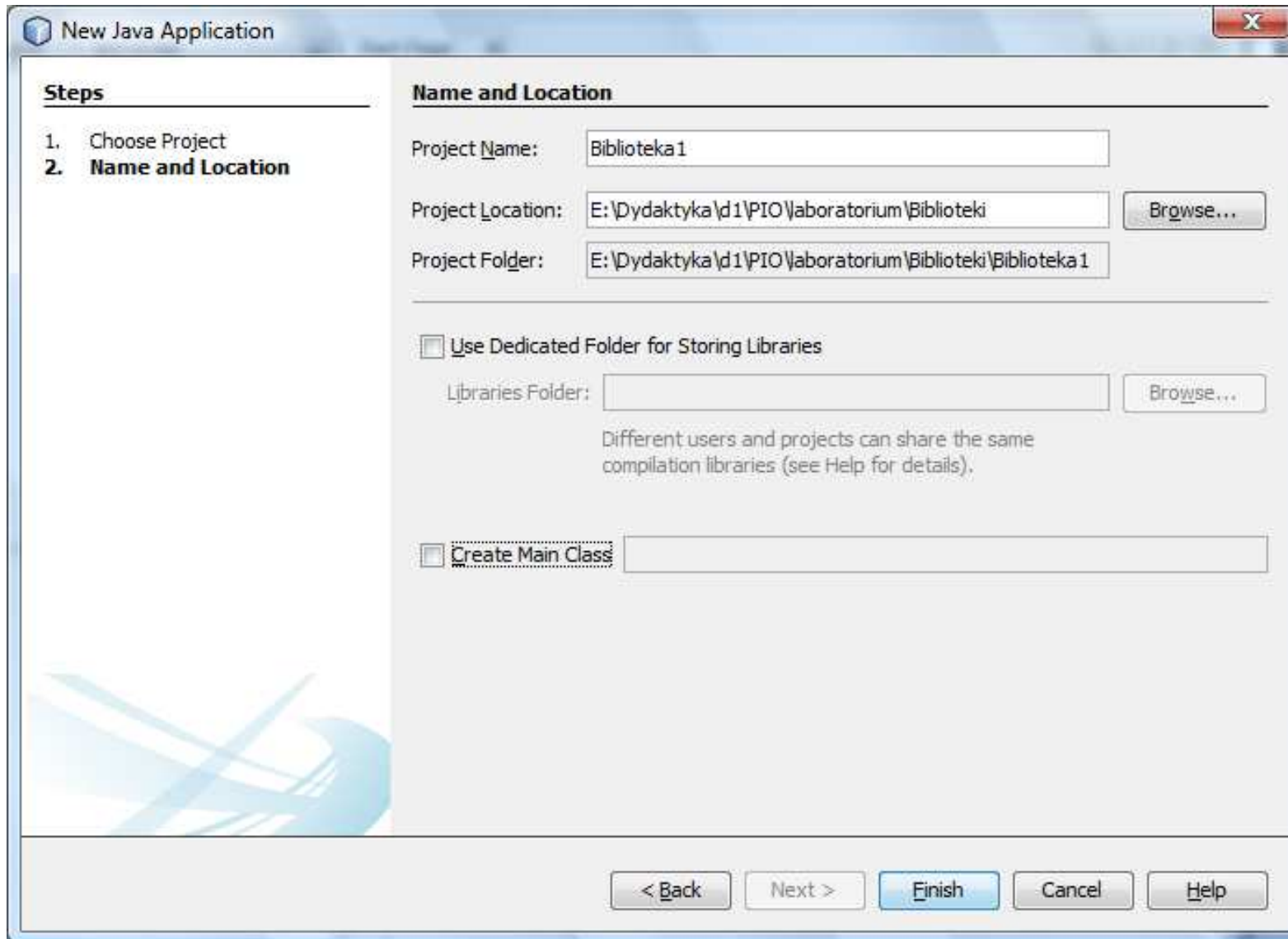
18.1. Wykonanie projektu typu aplikacja Javy w środowisku typu NetBeans – *File/New Project*



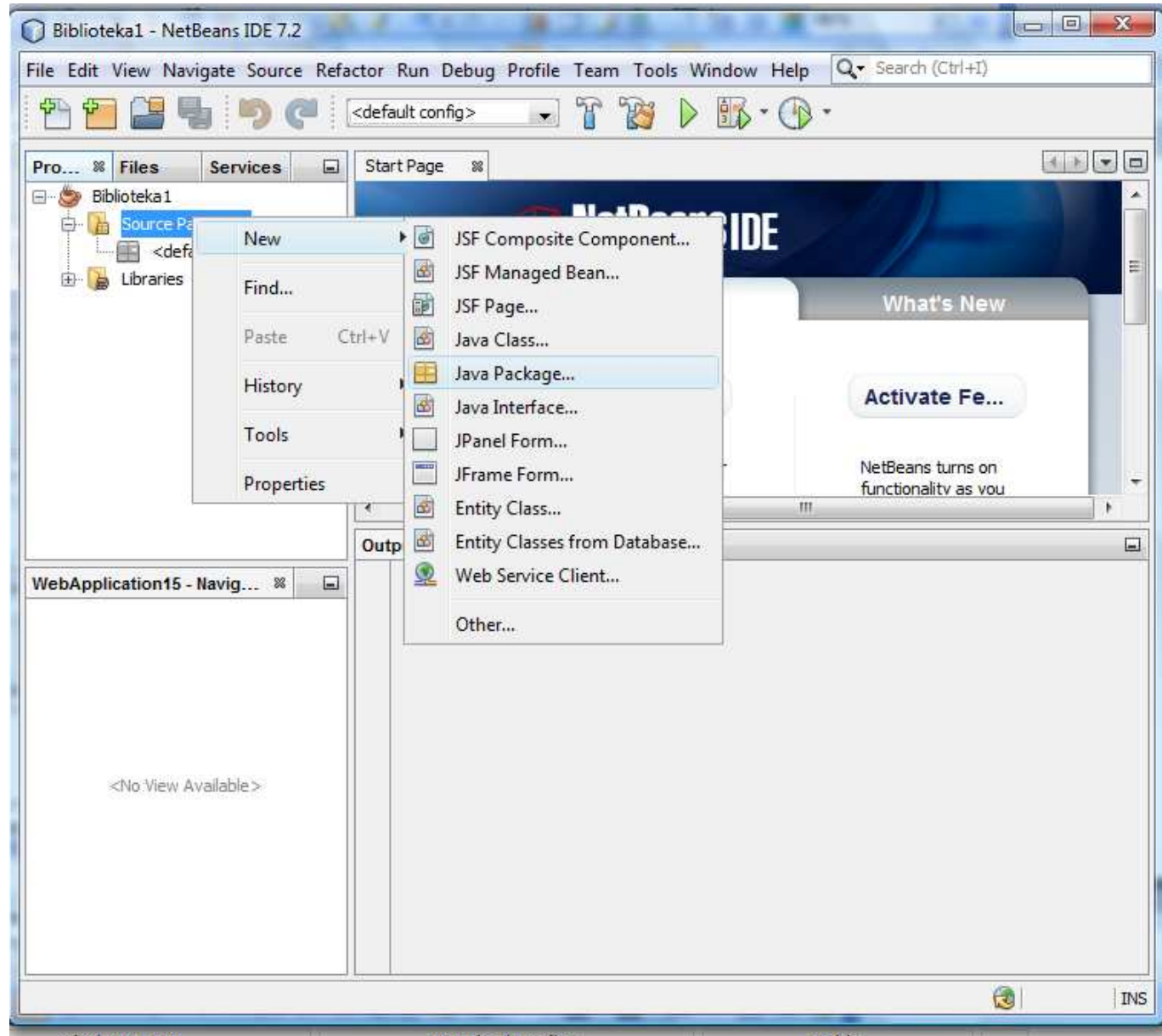
18.2. Wykonanie projektu typu *Java/ Java Application* – wybór w kolumnie *Categories* pozycji *Java* oraz pozycji *Java Application* w kolumnie *Projects*



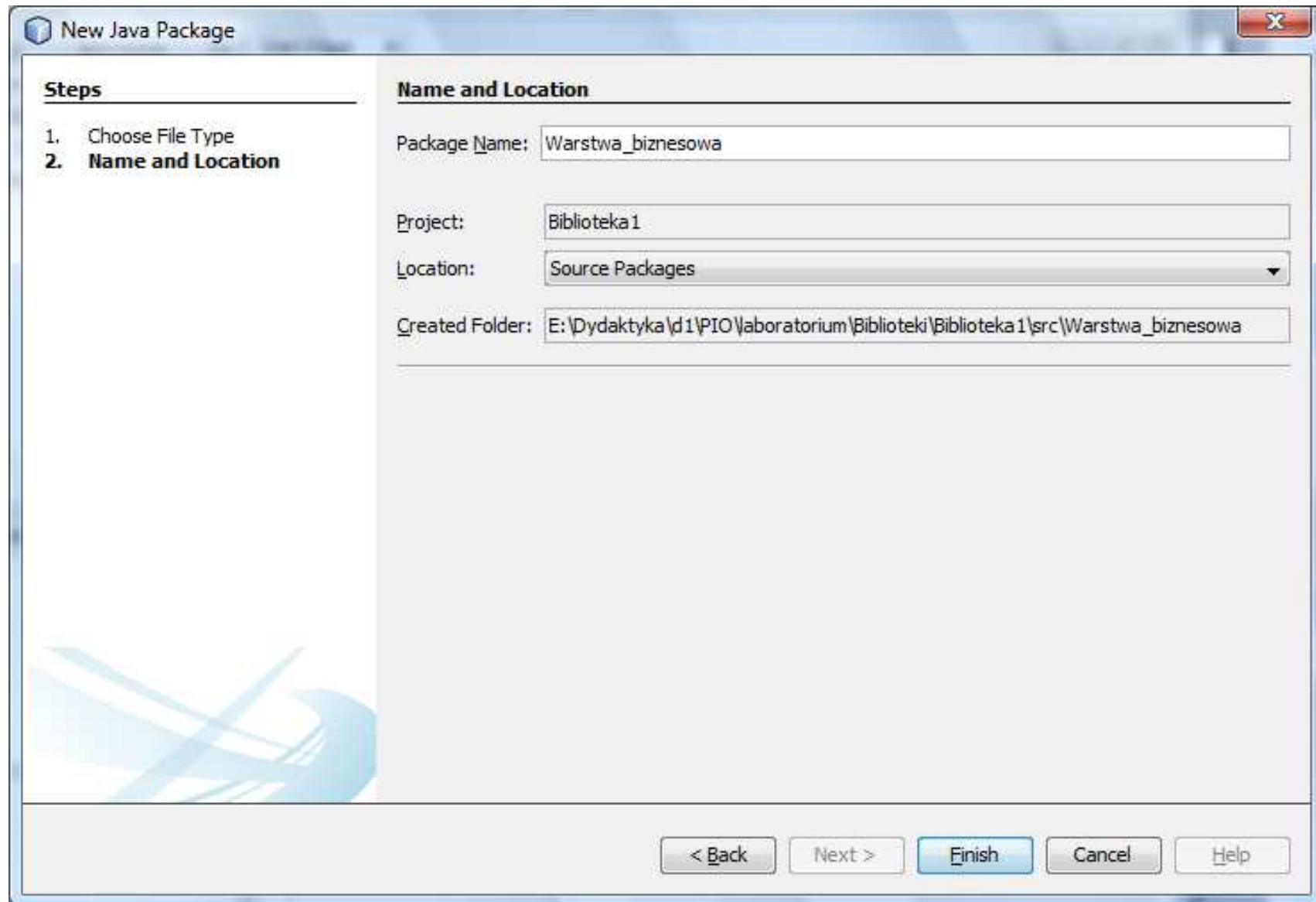
18.3. Wykonanie projektu typu *Java/ Java Application* – nadanie nazwy projektowi w polu *Project Name* oraz lokalizacji za pomocą klawiasza *Browse...* w polu *Project Location* bez ustawienia opcji *Create Main Class*



19.1. Wstawienie nowego pakietu do projektu – prawym klawiszem należy kliknąć na pozycję *Source Package* w okienku *Projects* i wybrać z listy pozycję *Java Package* (lub *Other*, jeśli nie ma takiej pozycji na liście)



19.2. Wstawienie nowego pakietu do projektu – nadanie nazwy pakietowi **Warstwa_biznesowa** w polu *Package Name*



New Java Package

Steps

1. Choose File Type
- 2. Name and Location**

Name and Location

Package Name: Warstwa_biznesowa

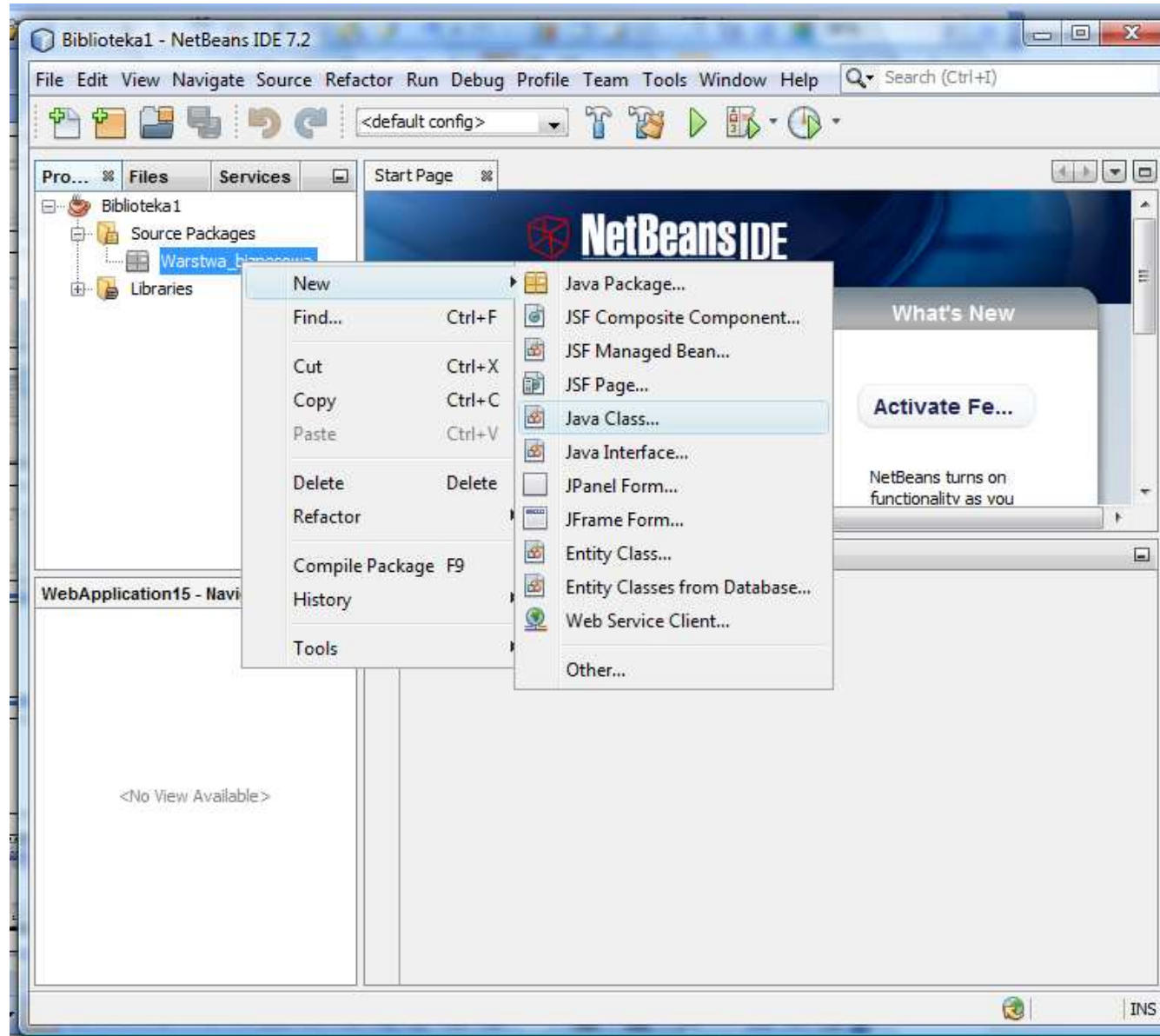
Project: Biblioteka 1

Location: Source Packages

Created Folder: E:\Dydaktyka\d1\PIO\laboratorium\Biblioteki\Biblioteka 1\src\Warstwa_biznesowa

< Back Next > **Finish** Cancel Help

20.1. Wstawienie do pakietu **Warstwa_biznesowa** nowej klasy – należy kliknąć prawym klawiszem myszy na nazwę pakietu i wybrać z listy pozycję *Java Class* (lub *Other*, jeśli nie ma takiej pozycji na liście)



20.2. Nadanie nazwy nowej klasie **Tytul_książki** w polu *Class Name*

New Java Class

Steps

1. Choose File Type
- 2. Name and Location**

Name and Location

Class Name: Tytul_książki

Project: Biblioteka 1

Location: Source Packages

Package: Warstwa_biznesowa

Created File: 'ka\d1\PIO\laboratorium\Biblioteki\Biblioteka 1\src\Warstwa_biznesowa\Tytul_książki.java

< Back Next > **Finish** Cancel Help

20.3. Zdefiniowanie kodu klasy **Tytul_książki** (kod klasy zawiera następuny slajd) na podstawie diagramu klas

The screenshot displays the NetBeans IDE 7.2 interface for a project named 'Biblioteka1'. The main editor window shows the source code for 'Tytul_książki.java'. The code defines a class with private attributes and public methods.

```
1  /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5  package Warstwa_biznesowa;
6
7  /**...*/
11 public class Tytul_książki {
12
13     private String tytul;
14     private String imie;
15     private String nazwisko;
16     private String wydawnictwo;
17     private String ISBN;
18
19     public Tytul_książki() {
20     }
21
22     public String getTytul() {
23         return tytul;
24     }
25 }
```

The left sidebar shows the project structure with 'Tytul_książki.java' selected under 'Warstwa_biznesowa'. The 'Members View' at the bottom left lists the class members:

- Tytul_książki()
- getISBN(): String
- getImie(): String
- getNazwisko(): String
- getTytul(): String
- getWydawnictwo(): String
- setISBN(String val)
- setImie(String val)
- setNazwisko(String val)

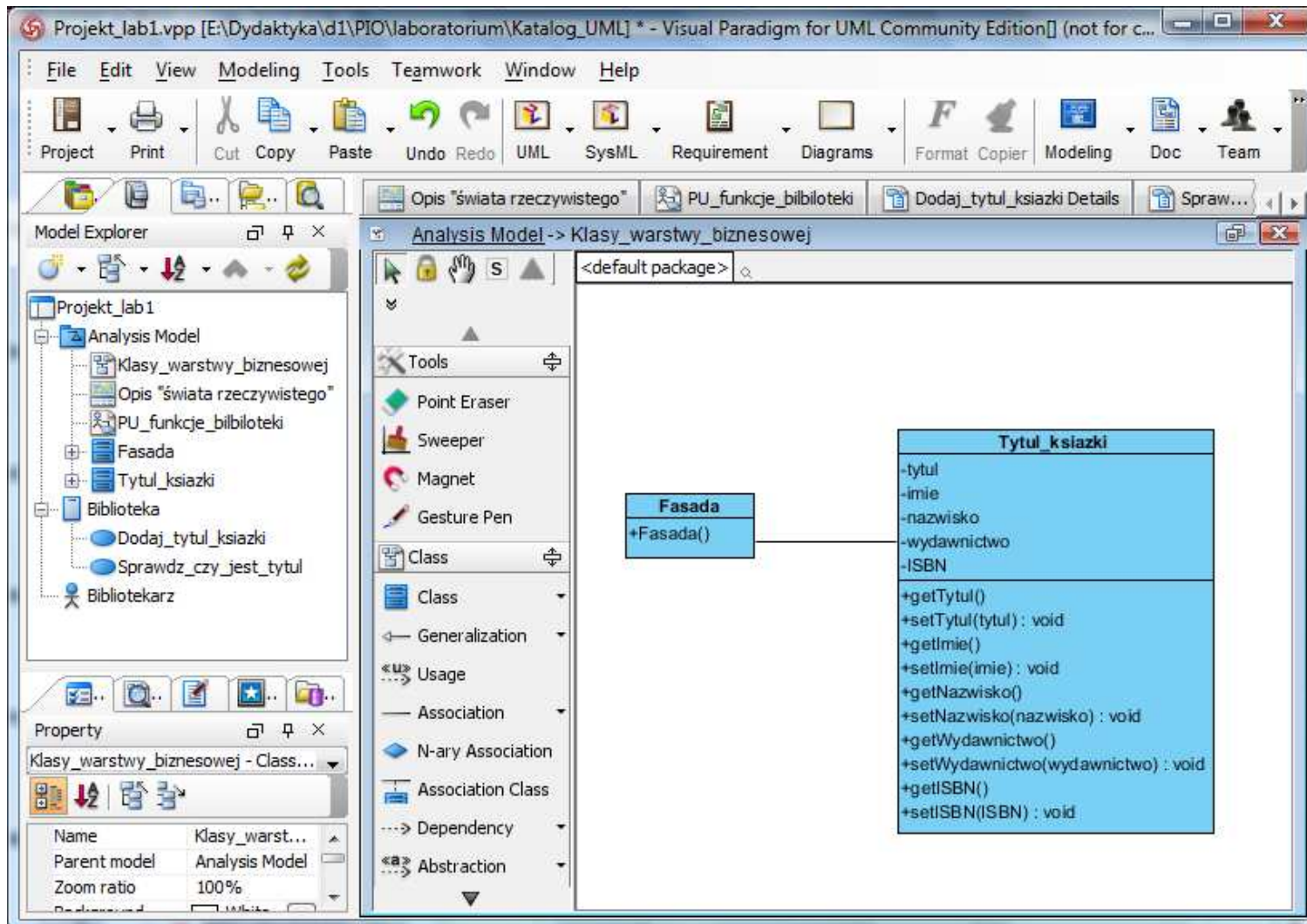
The bottom right corner of the IDE shows the 'Output' window and the status bar with page number 46 of 21 and the user 'INS'.

20.4, Kod klasy Tytul_ksiazki

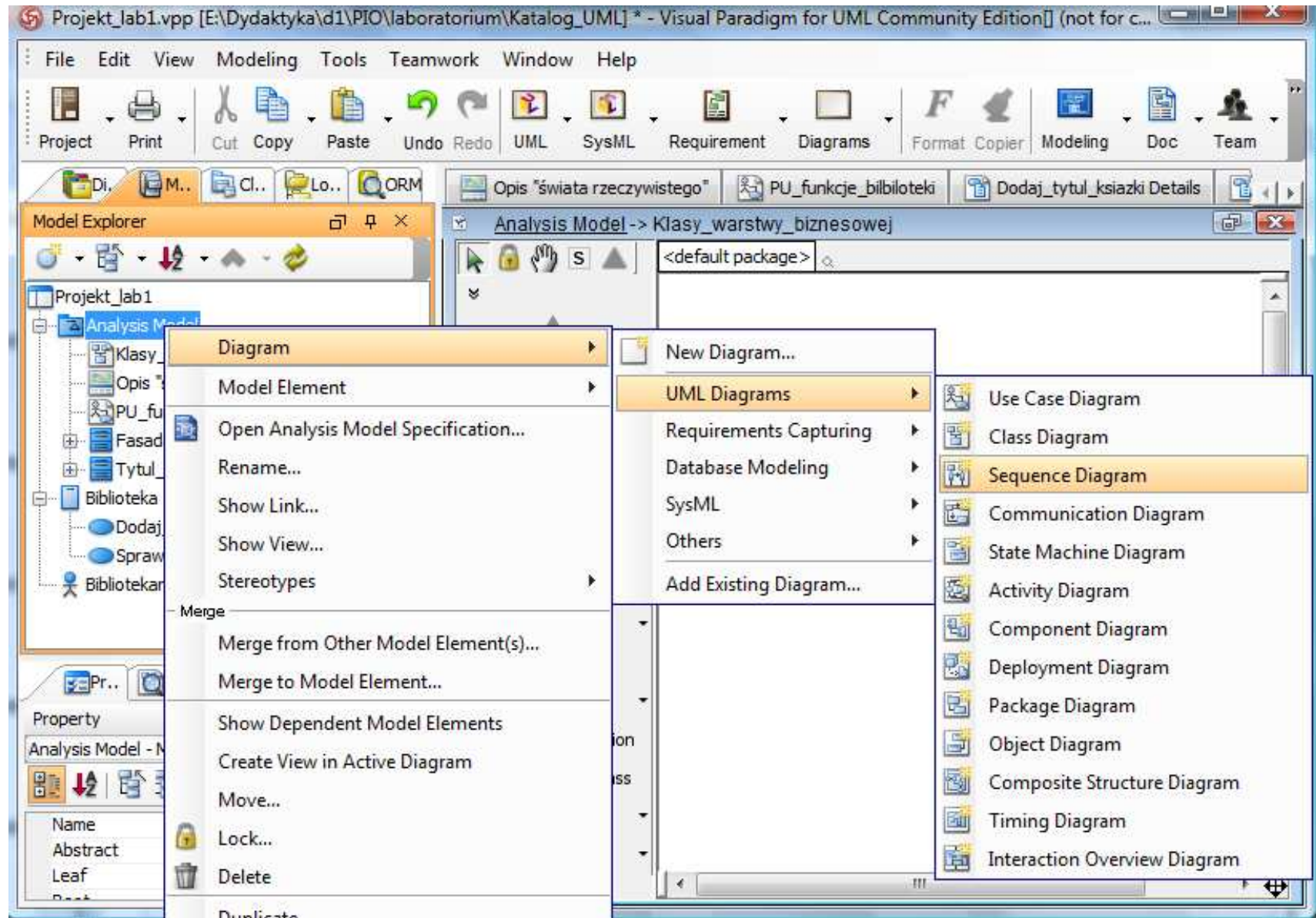
```
public class Tytul_ksiazki {  
    private String tytul;  
    private String imie;  
    private String nazwisko;  
    private String wydawnictwo;  
    private String ISBN;  
  
    public String getTytul()           { return tytul; }  
    public void setTytul(String val)  { this.tytul = val; }  
    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; }  
  
}
```

Relacja jeden do jeden

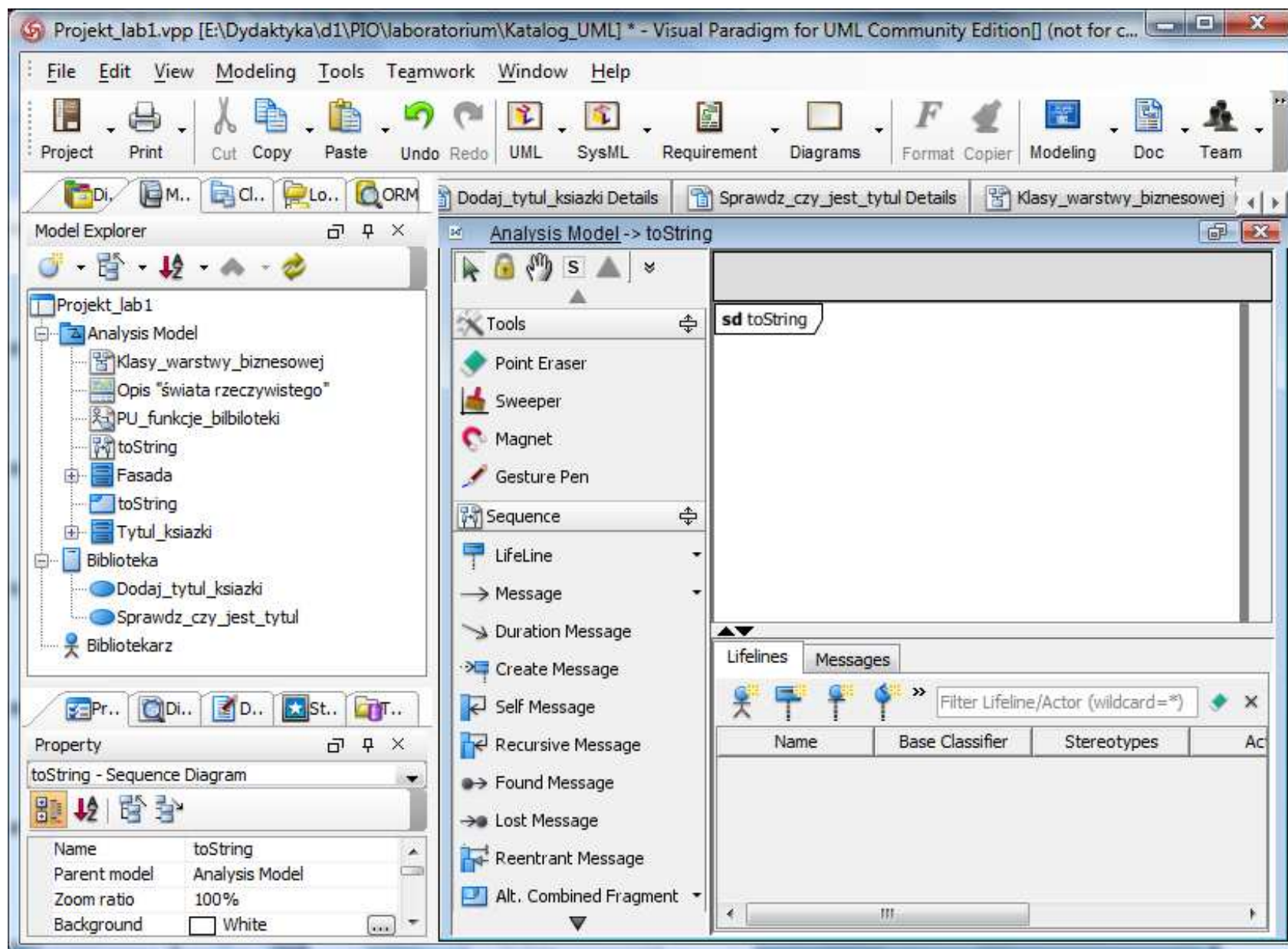
21. Wstawienie nowej klasy **Fasada** (podobnie jak klasę Tytuł_książki) powiązanej za pomocą relacji *Association* 1..0 z klasą typu **Tytuł_książki** – relację należy wybrać z palety z lewej strony lewym klawiszem myszy oraz położyć ją na klasie **Fasada** i przeciągnąć na klasę **Tytuł_książki**. Klasa ta reprezentuje wzorzec projektowy Fasada - będzie zastosowana do obsługi wywołań przypadków użycia przez warstwę interfejsu graficznego użytkownika.



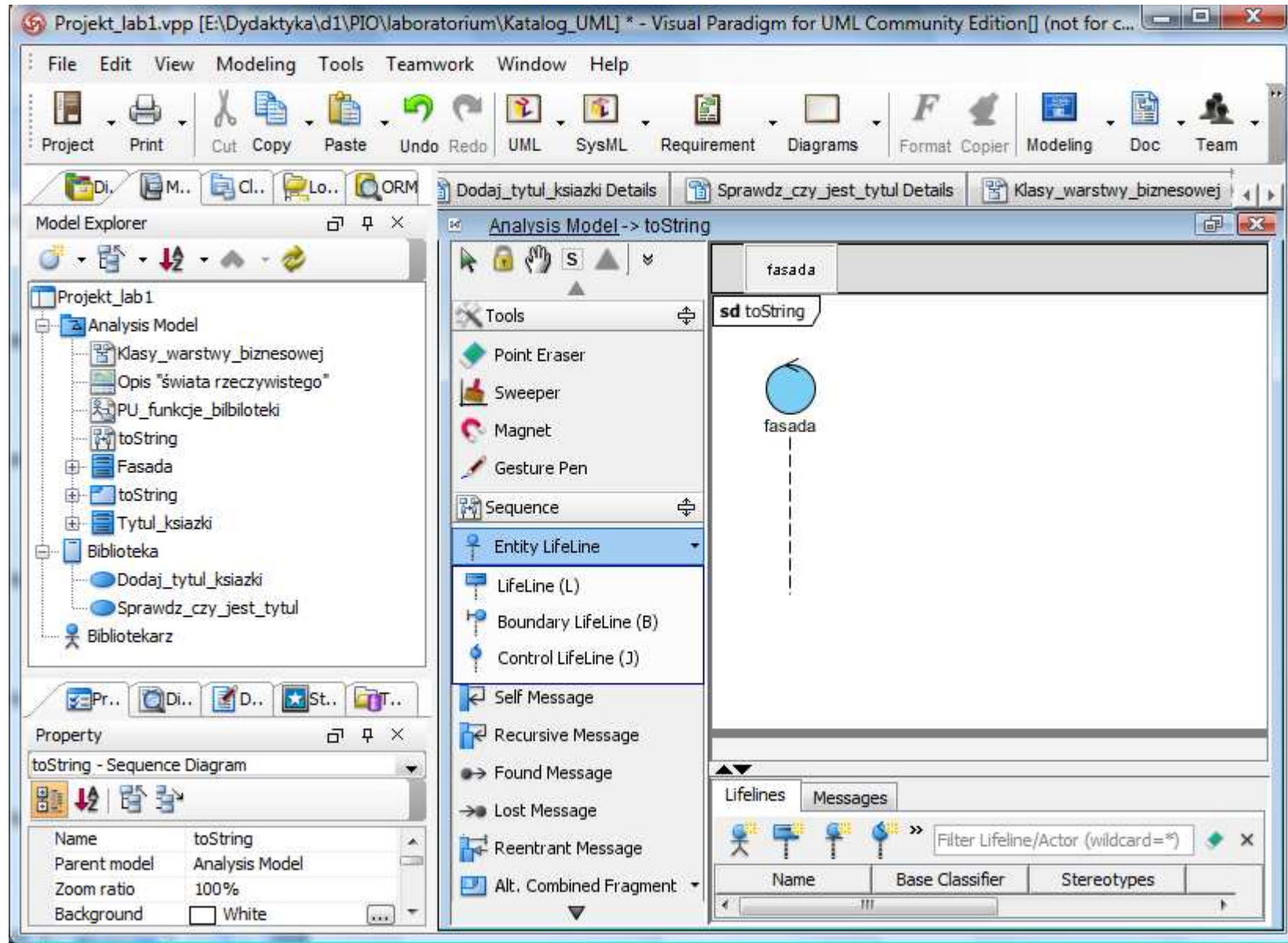
22.1. Wstawianie diagramu sekwencji – należy kliknąć prawym klawiszem myszy na nazwę modelu w okienku Model Explorer i wybrać z listy opcję *Diagram/UML Diagram/Sequence Diagram*



22.2. Należy nadać nazwę **toString** diagramowi sekwencji – diagram będzie zawierał definicję metody **toString** w klasie **Tytul_książki**



22.3. Należy z palety z lewej strony wybrać z listy Lifeline linię życia typu Control Lifeline

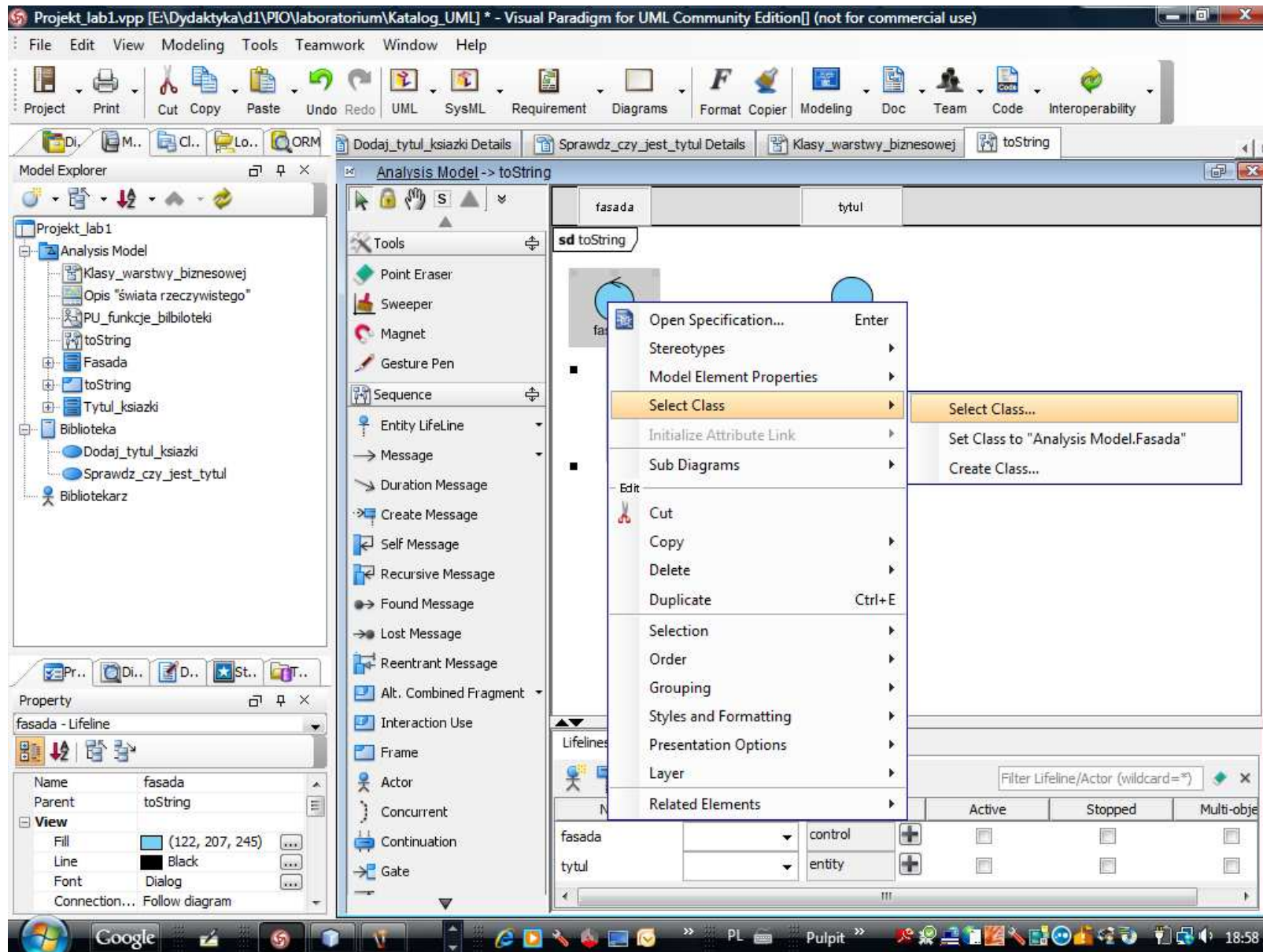


22.4. Należy z palety z lewej strony wybrać z listy Lifeline linię życia typu *Entity Lifeline* i nadać nazwę **tytul**

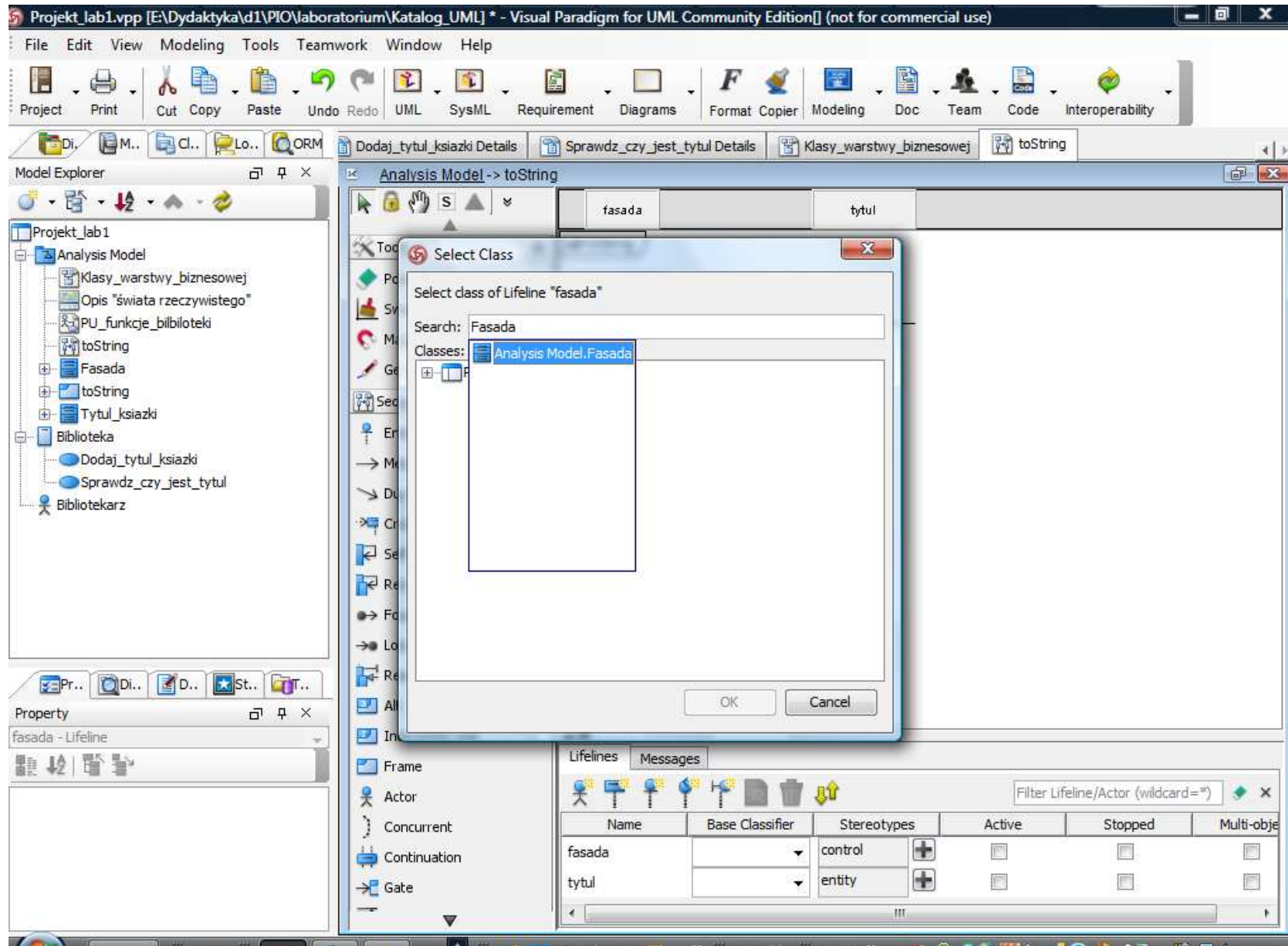
The screenshot displays the Visual Paradigm for UML Community Edition interface. The main workspace shows a sequence diagram titled "sd toString" with two lifelines: "fasada" and "tytul". The "tytul" lifeline is highlighted in blue, indicating it is the selected element. The left sidebar contains the Model Explorer, showing a project structure with "Projekt_lab1" and "Analysis Model" containing various elements like "Klasy_warstwy_biznesowej", "Opis 'swiata rzeczywistego'", "PU_funkcje_bililoteki", "toString", "Fasada", "Tytul_książki", "Biblioteka", "Dodaj_tytul_książki", "Sprawdz_czy_jest_tytul", and "Bibliotekarz". The bottom-left pane shows the Property window for the selected "fasada - Lifeline" element, with fields for Name (fasada), Parent (toString), and View (Fill: (122, 207, 245)). The bottom-right pane shows the Lifelines and Messages table, which lists the lifelines and their base classifiers and stereotypes.

Name	Base Classifier	Stereotypes	Ac
fasada		control	+
tytul		entity	+

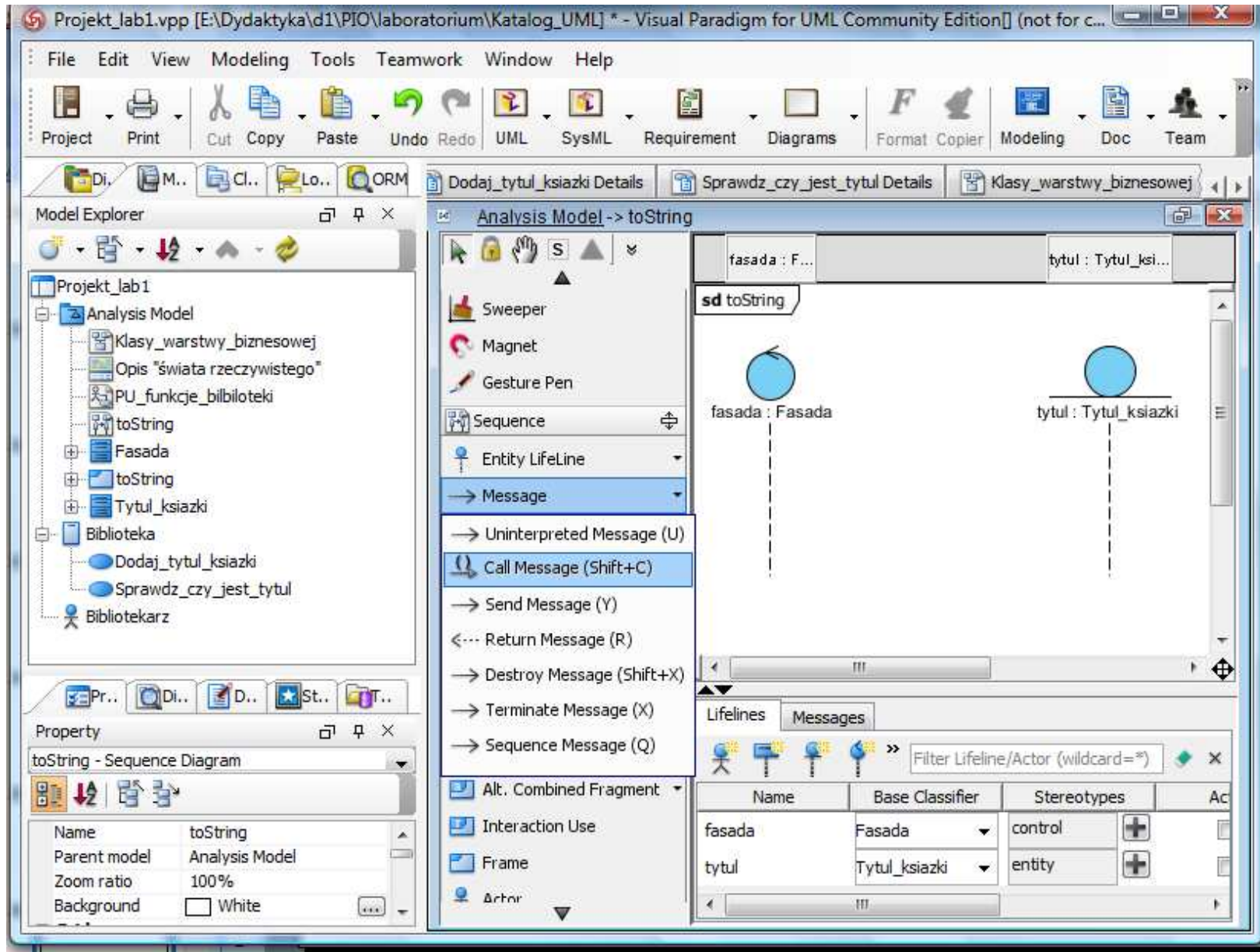
22.5. Należy linie życia obiektów powiązać z klasami z diagramu klas – po wybraniu linii życia **fasada** należy kliknąć prawym klawiszem myszy i wybrać z listy opcję *Select Class/Select Class*



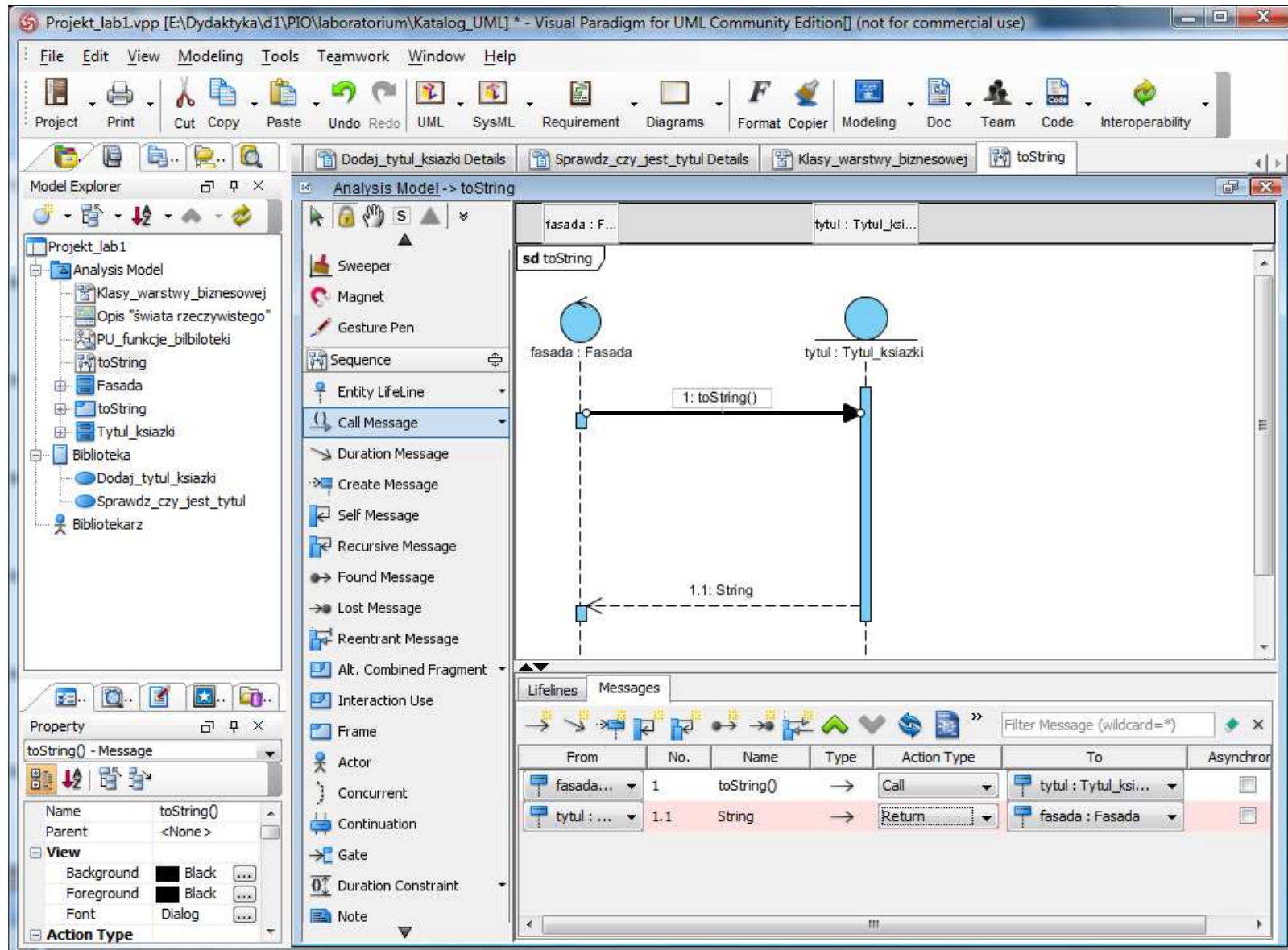
22.6. W formularzu *Select Class* należy w polu *Search* wpisać fragment nazwy klasy **Fasada**. W ukazanym okienku wybrać właściwą klasę i nacisnąć klawisz **OK**. Podobnie należy powiązać linię życia **tytul**.



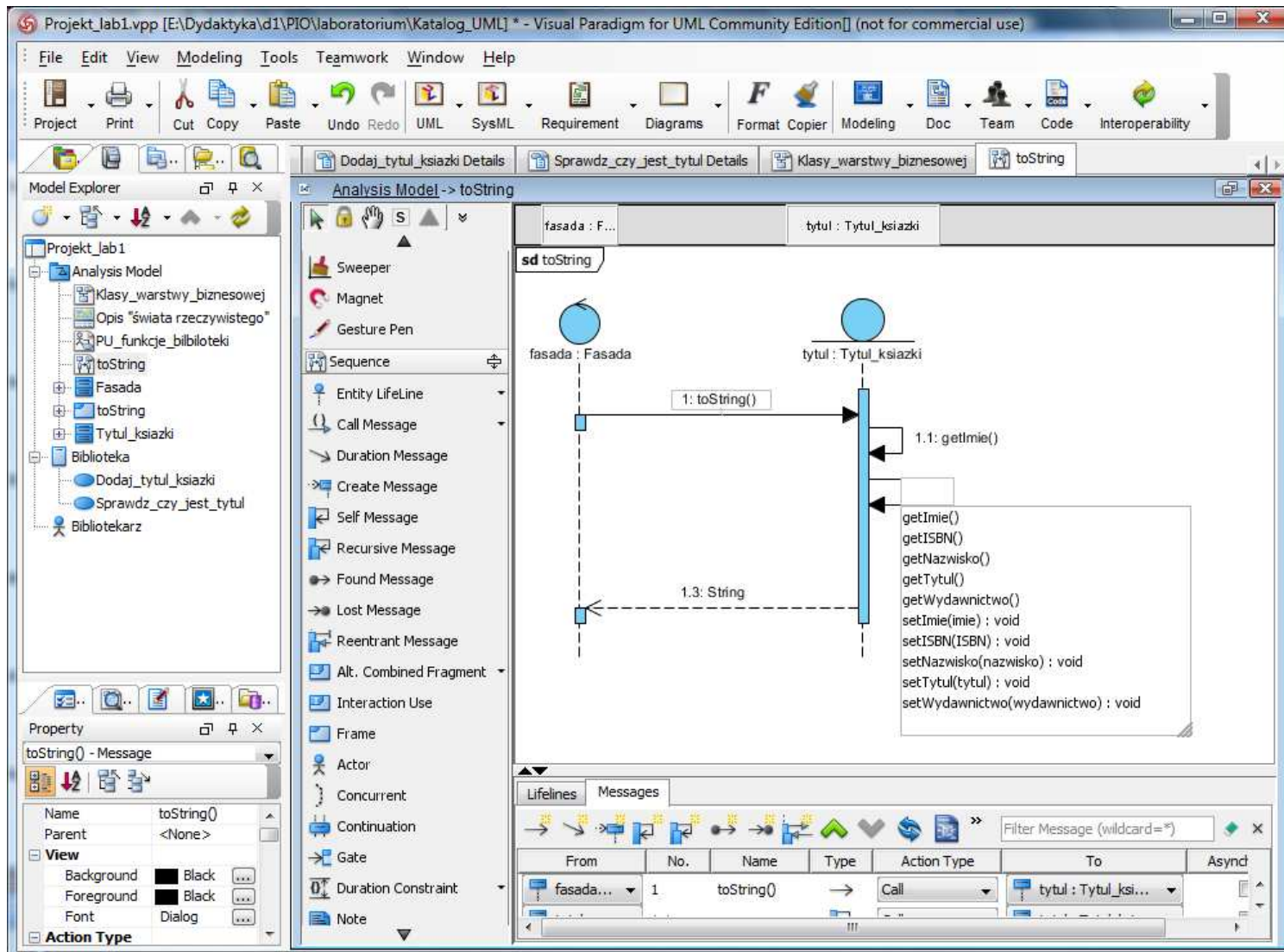
23.1. Należy wybrać z listy *Message* typ metody *Call Message* i przeciągnąć ją kładąc na linii życia **fasada** i przeciągając położyć na linii życia **tytul**. Podobnie należy zrobić z wiadomością typu *Return Message*.



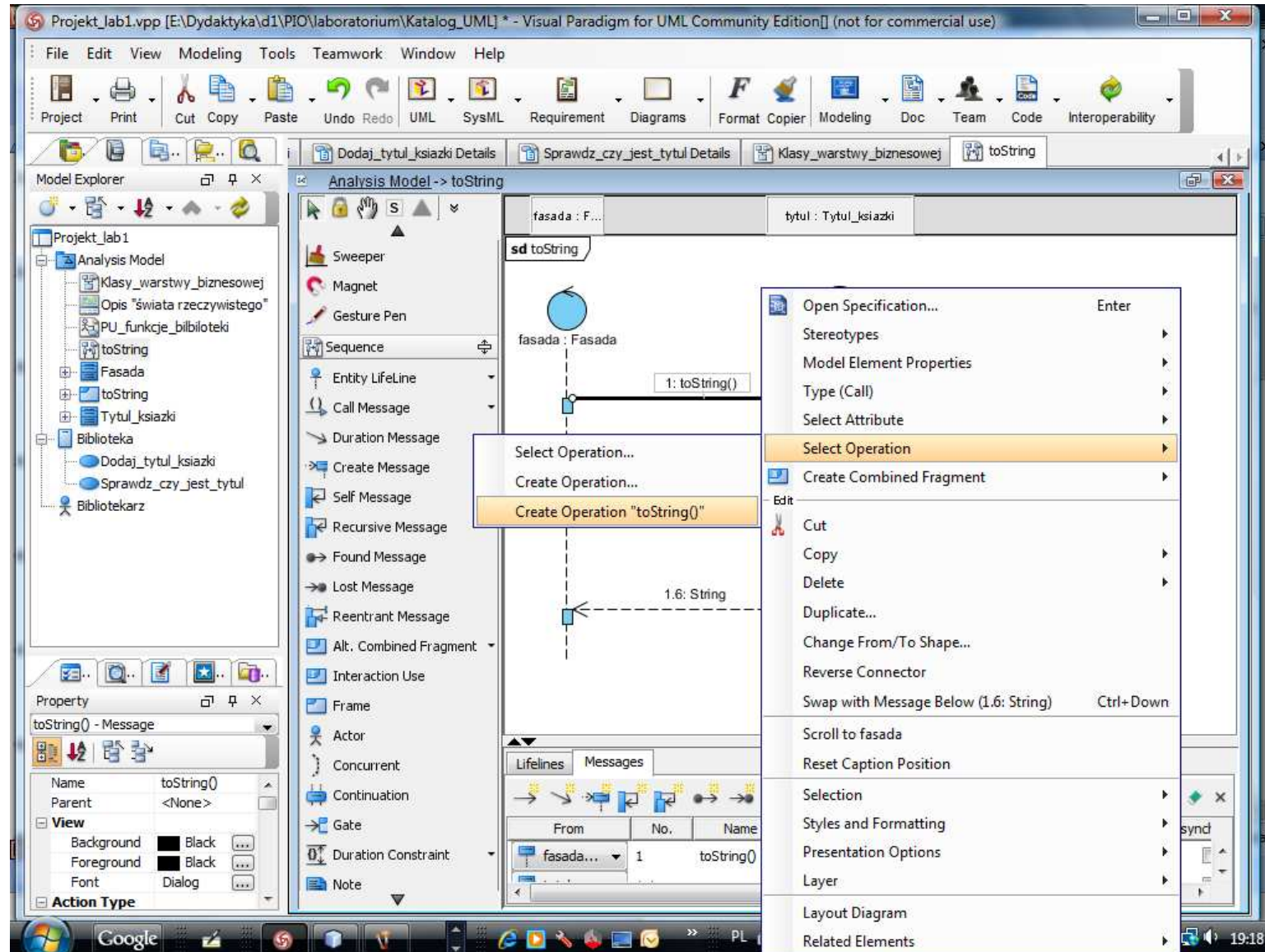
23.2. Należy wybrać z listy *Message* typ metody *Call Message* i przeciągnąć ją kładąc na linii życia **fasada** i przeciągając położyć na linii życia **tytul**. Podobnie należy zrobić z wiadomością typu *Return Message*, która powinna wychodzić z linii życia **tytul** i wchodzić do linii życia **fasada**.



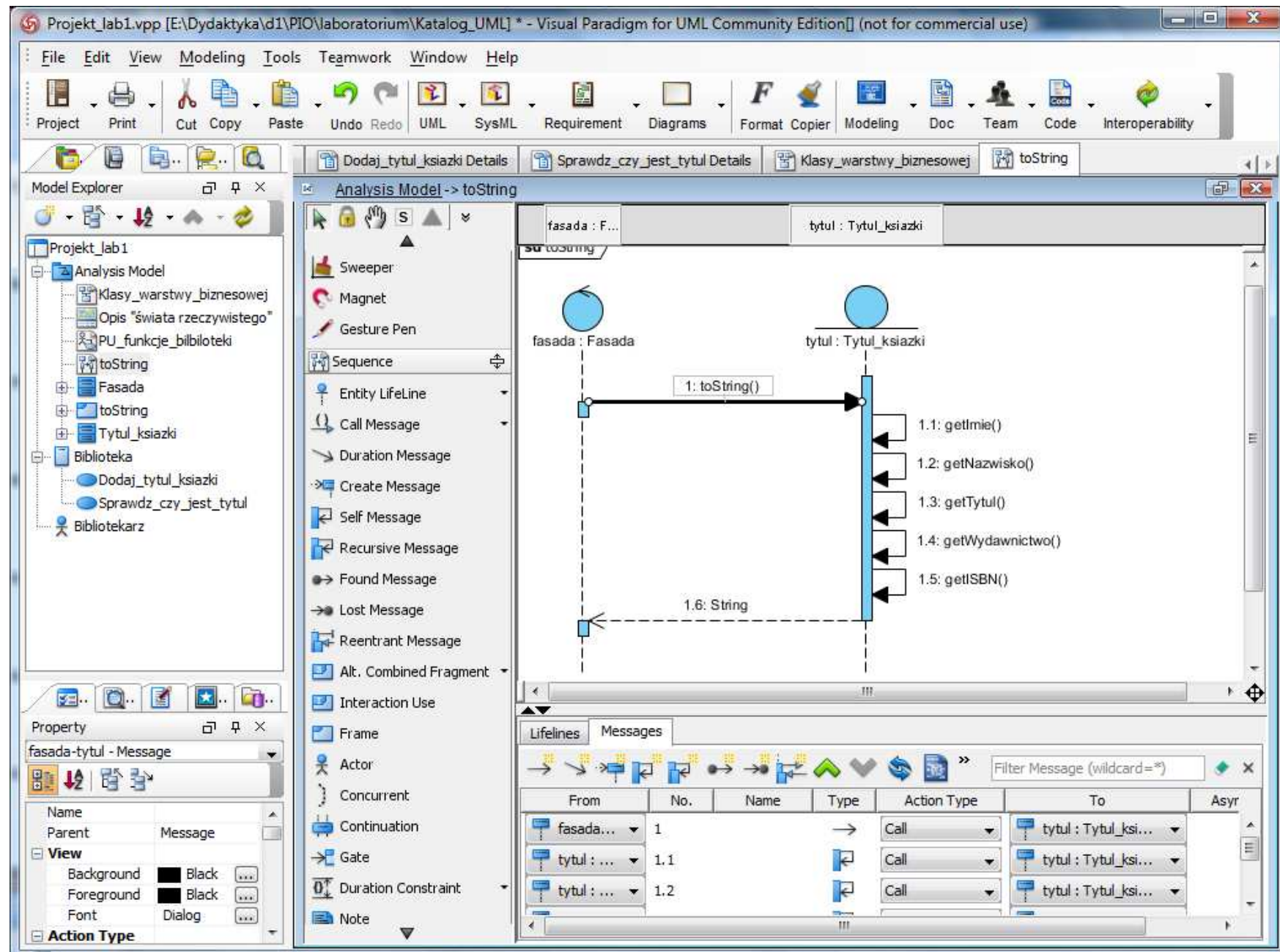
23.3. Należy zdefiniować ciało metody **toString** w klasie **Tytul_książki** za pomocą wiadomości *Self Message*, przeciąganych z palety z lewej strony. Podczas wstawiania wiadomości pokazuje się lista metod klasy **Tytul_książki** zdefiniowanych podczas tworzenia diagramu klas. Należy dokonać wyboru właściwej metody typu `get` z listy metod klasy **Tytul_książki**.



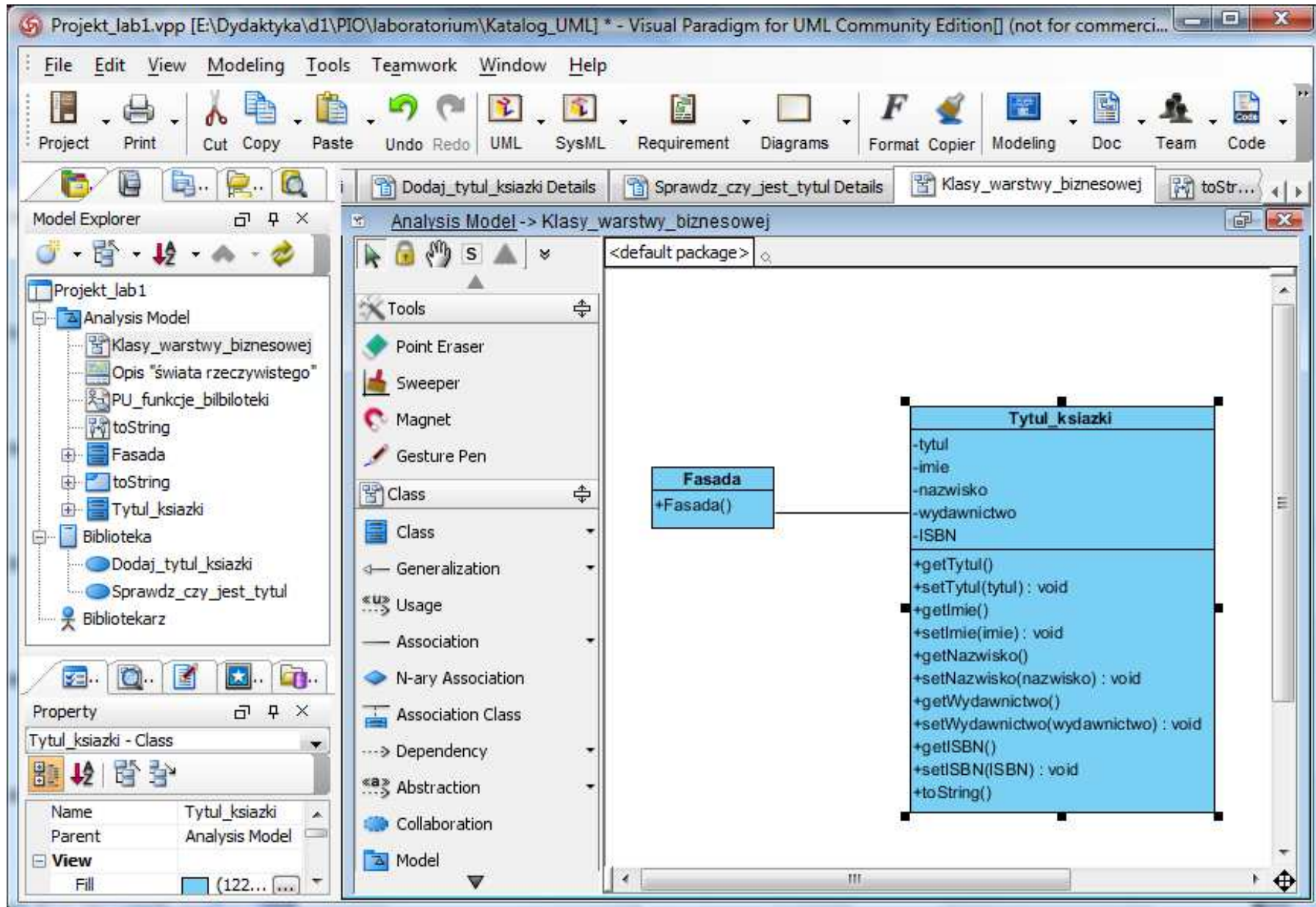
23.4. Włączenie zdefiniowanej metody do klasy **Tytuł_ksiazki** – po wybraniu wiadomości o nazwie **toString** klikając prawym klawiszem myszy należy wybrać z list pozycje *Select Operation/Select Operation „toString()”*



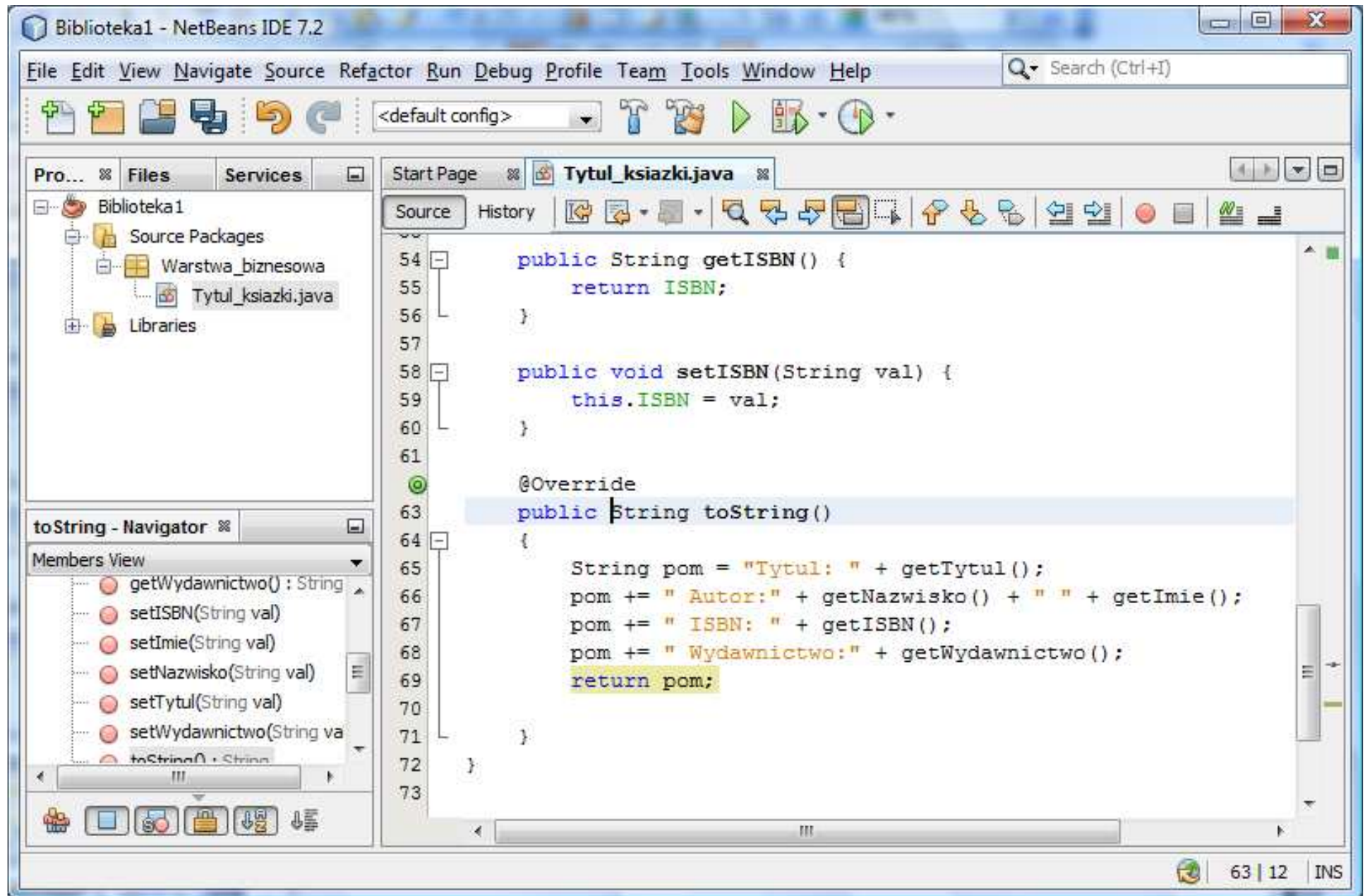
23.5. Rezultat wykonania scenariusza metody *toString* w klasie **Tytul_książki**.



23.6. Metoda **toString** zdefiniowana na diagramie sekwencji pojawiła się w klasie **Tytul_książki**



23.7. Definicja kodu metody `toString` w klasie `Tytul_książki` zdefiniowana na podstawie diagramu sekwencji (p. 23.5)

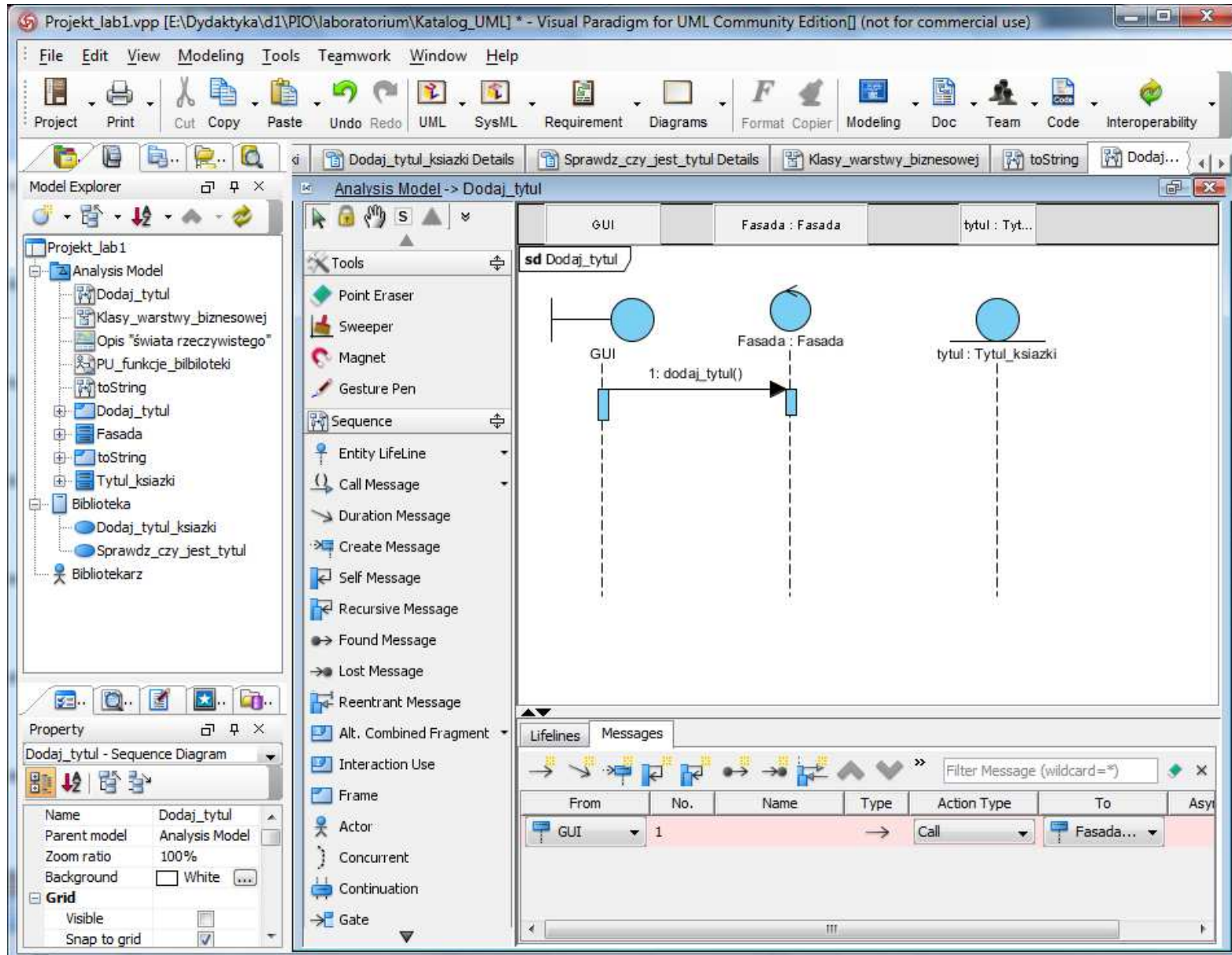


The screenshot displays the NetBeans IDE 7.2 interface. The main editor window shows the source code of the `Tytul_książki.java` file. The code defines three methods: `getISBN()`, `setISBN(String val)`, and `toString()`. The `toString()` method is highlighted, showing its implementation which concatenates the title, author name, and publisher information into a single string.

```
54 public String getISBN() {
55     return ISBN;
56 }
57
58 public void setISBN(String val) {
59     this.ISBN = val;
60 }
61
62 @Override
63 public String toString()
64 {
65     String pom = "Tytul: " + getTytul();
66     pom += " Autor:" + getNazwisko() + " " + getImie();
67     pom += " ISBN: " + getISBN();
68     pom += " Wydawnictwo:" + getWydawnictwo();
69     return pom;
70 }
71 }
72 }
73 }
```

The IDE interface includes a menu bar (File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help), a toolbar with various icons, and a project explorer on the left showing the project structure: Biblioteka1 > Source Packages > Warstwa_biznesowa > Tytul_książki.java. A 'Members View' window is also visible, listing the methods of the class.

24.1. Dodanie nowego diagramu sekwencji o nazwie **Dodaj_tytul** reprezentującego definicję przypadku użycia **Dodaj_tytul**.



24.2. Należy wiadomość **Dodaj_tytul** z klasy przypisać do klasy **Fasada** – należy kliknąć prawym klawiszem myszy na strzałkę reprezentującą metodę i wybrać z listy opcje *Select Operation/Create Operation*

The screenshot shows the Visual Paradigm for UML interface. The main window displays a sequence diagram titled "sd Dodaj_tytul". The diagram has 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 message arrow, with "Select Operation" and "Create Operation..." highlighted. The "Messages" table at the bottom of the diagram is as follows:

From	No.	N
GUI	1	
Fasada...	1.1	
Fasada...	1.2	
Fasada...	1.3	
Fasada...	1.4	
Fasada...	1.5	
Fasada...	1.6	

The "Property" window shows the "GUI-Fasada - Message" with the following details:

- Name: GUI-Fasada - Message
- Parent: Message
- View: Background (Black), Foreground (Black), Font (Dialog)
- Action Type: (empty)

24.3. Na formularzu *Operation Specification* należy wypełnić następujące pola: *Name* (nazwa metody), *Classifier* (Nazwę klasy), *Return type* (wynik zwracany przez metodę), *Visibility* (określanie zasięgu metody). Następnie należy zdefiniować listę parametrów metody wybierając kolejny formularz w zakładce *Parameters*

The screenshot shows the 'Operation Specification' dialog box with the following fields and values:

- Name:** dodaj_tytul
- Classifier:** Analysis Model.Fasada
- Return type:** void
- Type modifier:** <Unspecified>
- Visibility:** public
- Scope:** instance
- Lower:** (empty)
- Upper:** (empty)
- Body condition:** (empty text area)
- Documentation:** (empty text area with a rich text editor toolbar)
- Record...** (dropdown menu)
- Abstract:**
- Leaf:**
- Query:**
- Ordered:**
- Unique:**

Buttons at the bottom: Reset, OK, Cancel, Apply, Help.

24.5. Pełna definicja metody **dodaj_tytul** w klasie **Fasada** - linia życia **tytul** typu **Tytul_książki** została zdefiniowana jako *Create Message* przez przeciągnięcie na linię życia typu *Entity Lifeline* ikony *Create Message* z palety z lewej strony.

The screenshot displays the Visual Paradigm for UML interface. The main window shows a sequence diagram titled 'sd Dodaj_tytul' with three lifelines: GUI, Fasada : Fasada, and tytul : Tyt... (representing Tytul_książki). The GUI lifeline sends a call message '1: dodaj_tytul()' to the Fasada lifeline. The Fasada lifeline then sends six call messages to the tytul lifeline: '1.1: setISBN()', '1.2: setImie()', '1.3: setNazwisko()', '1.4: setWydawnictwo()', '1.5: setWydawnictwo()', and '1.6: setTytul()'. The tytul lifeline is created via a 'Create Message' icon from the left palette.

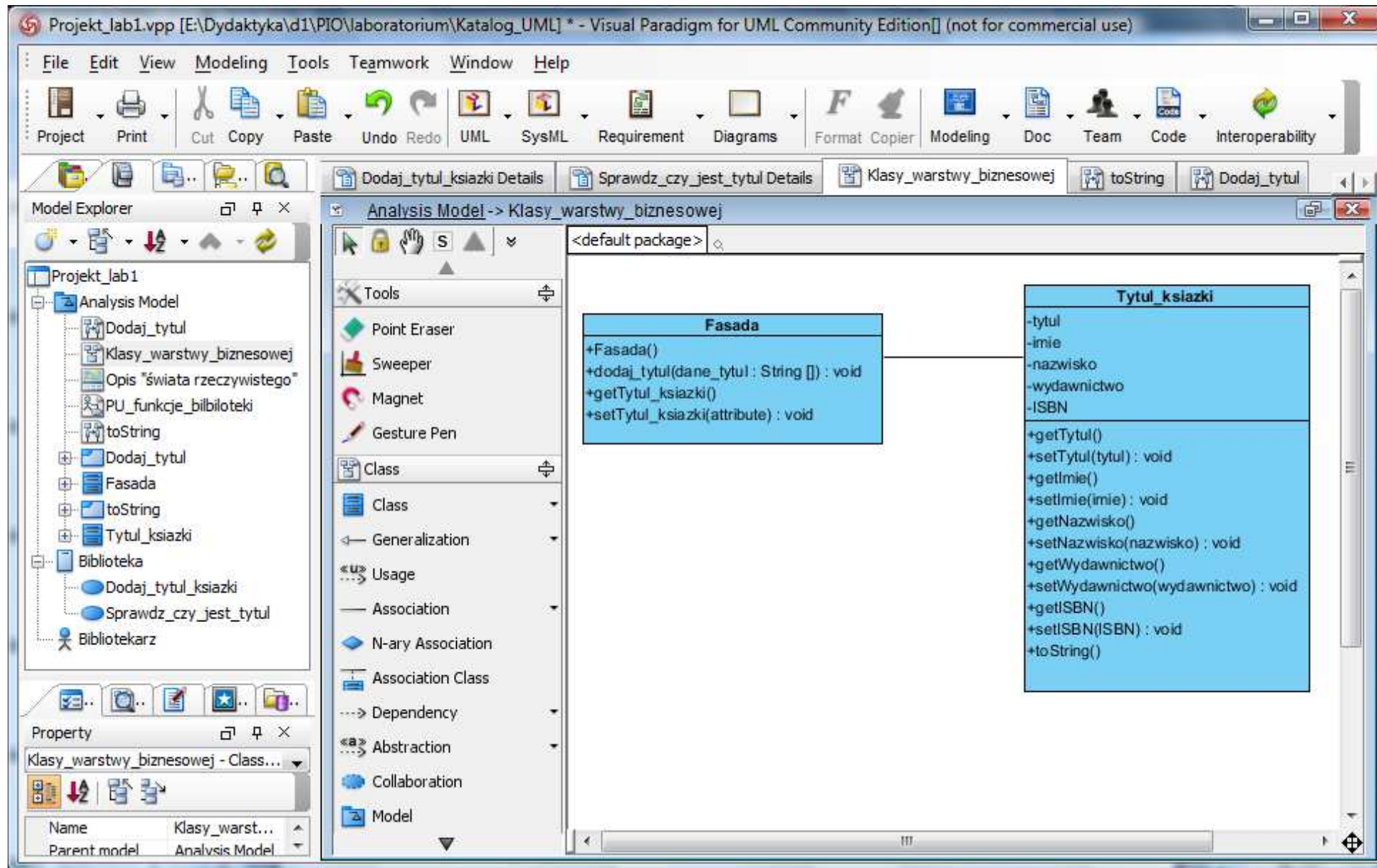
Below the diagram is a table showing the message details:

From	No.	Name	Type	Action Type	To	Asynchronous
GUI	1	dodaj_tytul()	→	Call	Fasada...	<input checked="" type="checkbox"/>
Fasada...	1.1	setISBN()	→	Call	tytul : ...	<input checked="" type="checkbox"/>
Fasada...	1.2	setImie()	→	Call	tytul : ...	<input checked="" type="checkbox"/>
Fasada...	1.3	setNazwisko()	→	Call	tytul : ...	<input checked="" type="checkbox"/>
Fasada...	1.4	setWydawnictwo()	→	Call	tytul : ...	<input checked="" type="checkbox"/>
Fasada...	1.5	setWydawnictwo()	→	Call	tytul : ...	<input checked="" type="checkbox"/>
Fasada...	1.6	setTytul()	→	Call	tytul : ...	<input checked="" type="checkbox"/>

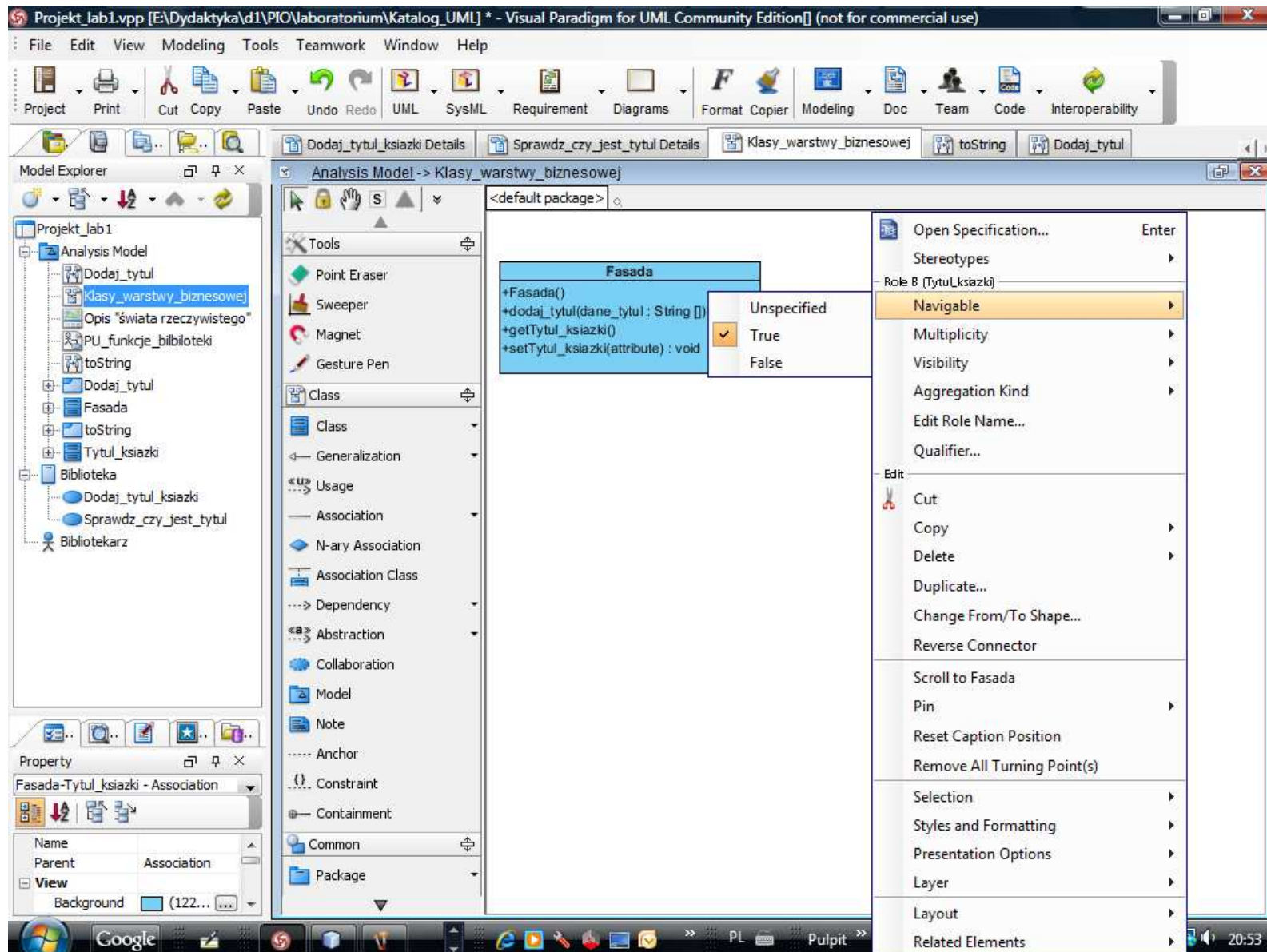
24.6. Uwidocznienie nowej metody **dodaj_tytul** z listą parametrów w klasie **Fasada**, zdefiniowanej na diagramie sekwencji z p.

The screenshot shows the Visual Paradigm for UML interface. The main window displays a class diagram with two classes: **Fasada** and **Tytul_książki**. The **Fasada** class has two methods: `+Fasada()` and `+dodaj_tytul(dane_tytul : String []) : void`. The **Tytul_książki** class has private attributes: `-tytul`, `-imie`, `-nazwisko`, `-wydawnictwo`, and `-ISBN`. It also has several public 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. The interface includes a menu bar, a toolbar, a Model Explorer on the left, and a Property window at the bottom.

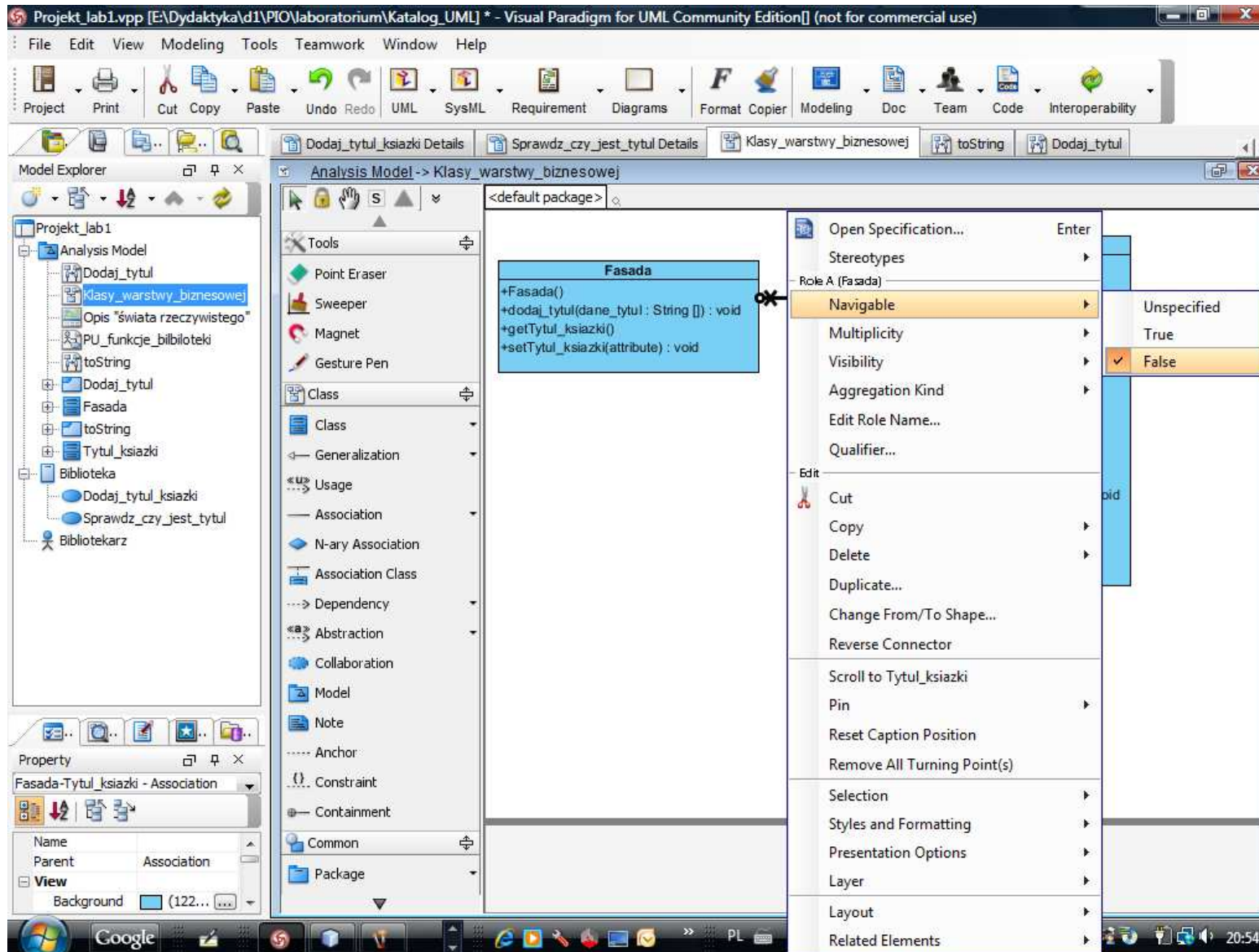
25.1. Dodanie dwóch operacji typu **get** i **set**, reprezentujących implementację relacji *Association* w klasie **Fasada** (obiekt typu **Fasada** będzie miał dostęp do jednego obiektu typu **Tytul_książki**)



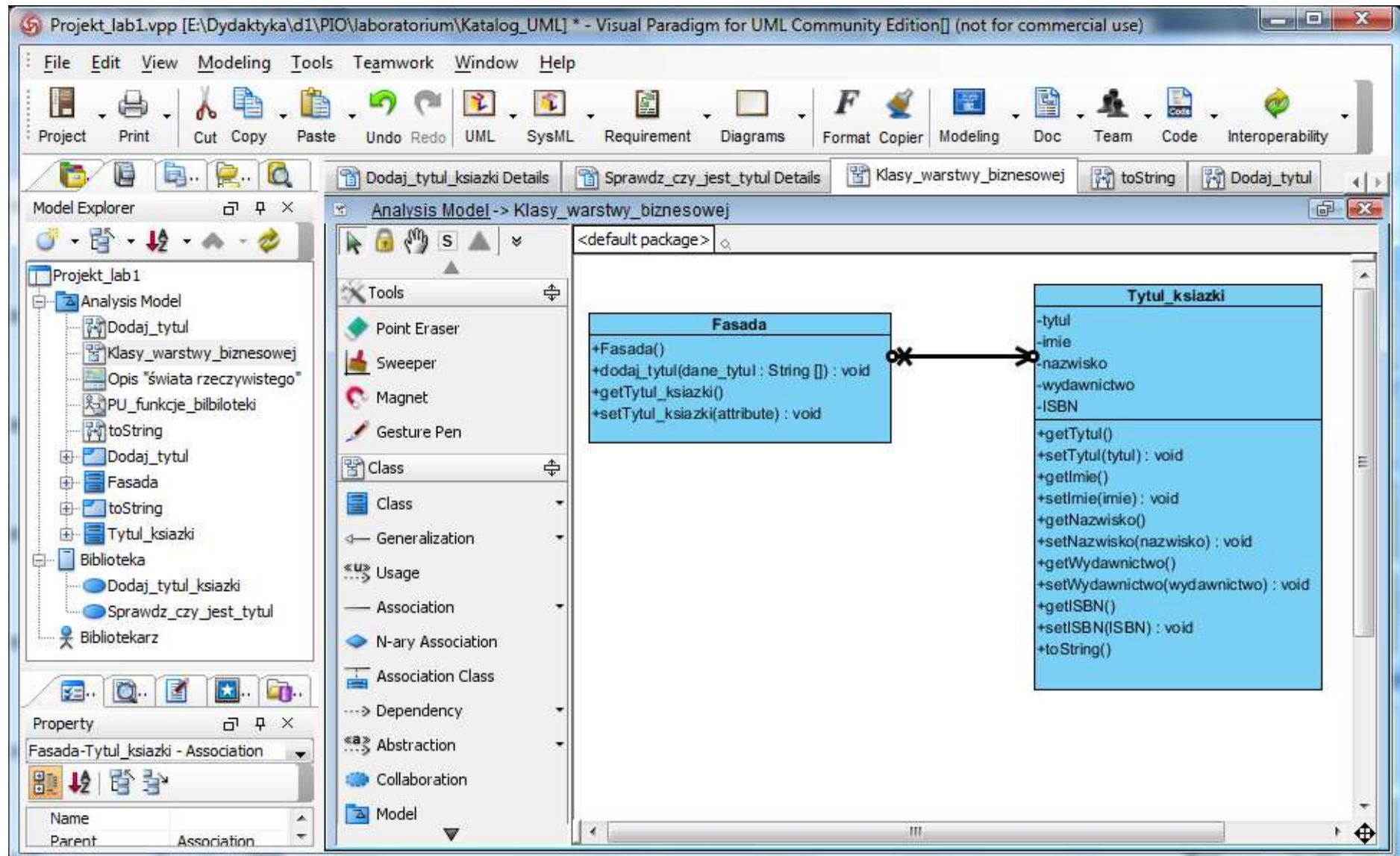
25.2. Definicja powiązania jednokierukowego typu *Association* – należy kliknąć prawym klawiszem na koniec relacji powiązanej z klasą **Tytuł_książki** i wybrać z listy opcje *Navigable/True*



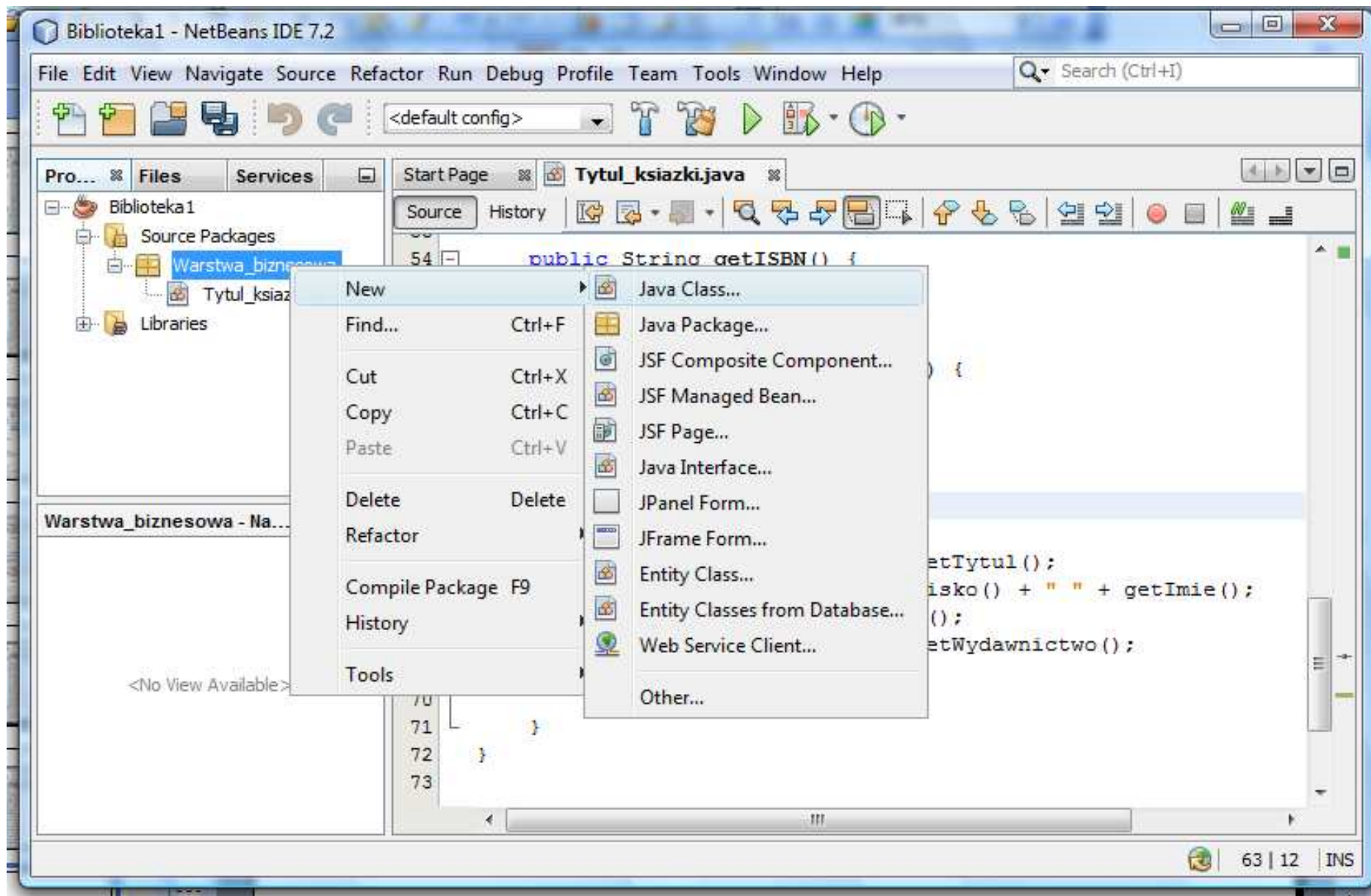
25.3. Definicja powiązania jednokierukowego typu *Association* – należy kliknąć prawym klawiszem na koniec relacji powiązanej z klasą **Fasada** i wybrać z listy opcje *Navigable/False*



25.4. Uzyskano jednokierunkową relację typu *Association* – oznacza ona, że obiekt typu **Fasada** zawiera referencję do obiektu typu **Tytul_książki**, natomiast obiekt typu **Tytul_książki** nie ma dostępu do obiektu klasy **Fasada** (definicja relacji jednokierunkowej typu 1..0..1)



26.1. Uzupelnienie kodu projektu **Java Application** zawierajacego implementacje diagramow sekwencji **dodaj_tytul** – nalezy dodac nowa klase typu **Fasada** (podobnie jak **Tytul_książki**)



26.2. Nadanie nazwy nowej klasie w polu *Class Name*

New Java Class

Steps

1. Choose File Type
- 2. Name and Location**

Name and Location

Class Name: Fasada

Project: Biblioteka 1

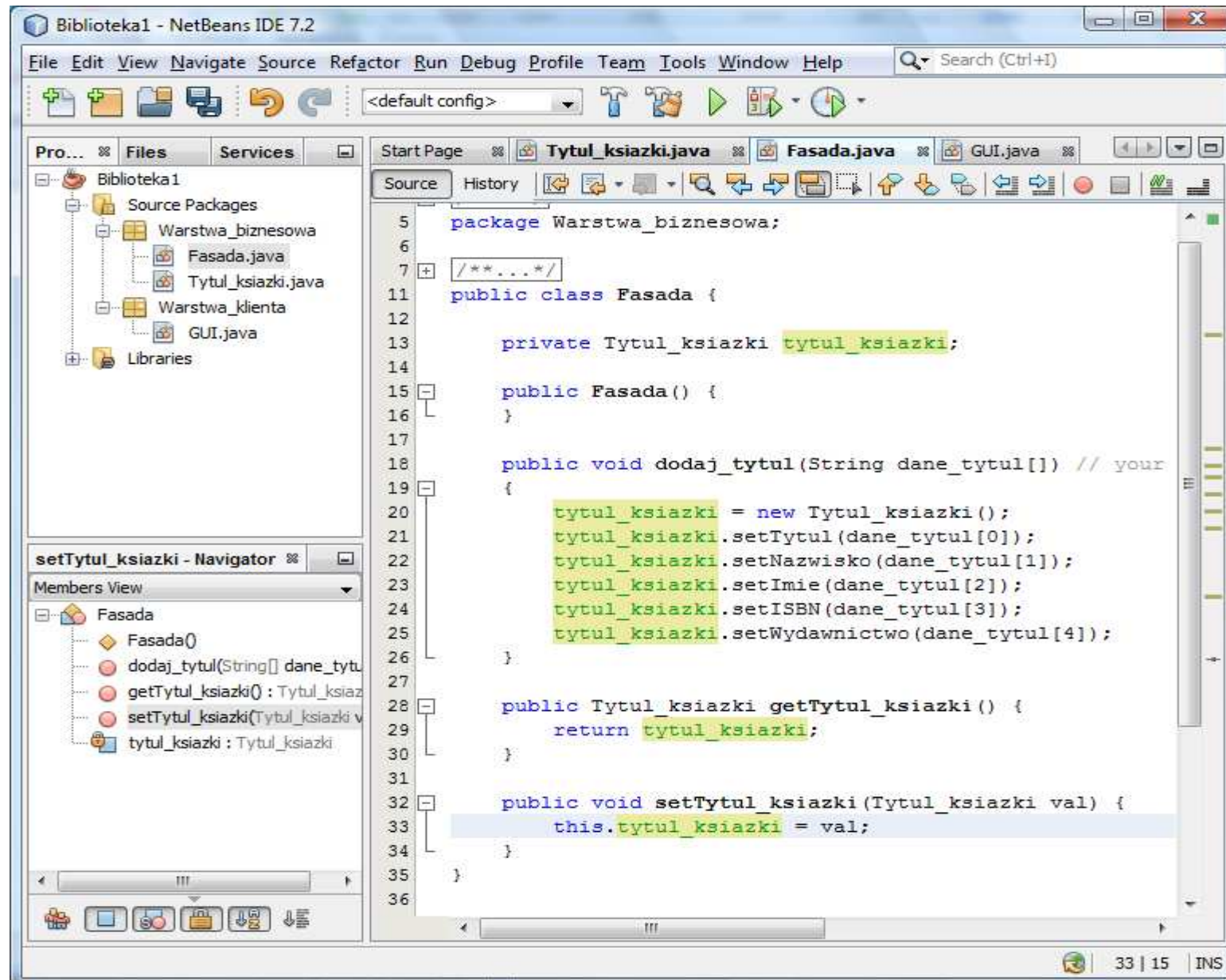
Location: Source Packages

Package: Warstwa_biznesowa

Created File: E:\Dydaktyka\d1\PIO\laboratorium\Biblioteki\Biblioteka 1\src\Warstwa_biznesowa\Fasada.java

< Back Next > **Finish** Cancel Help

26.3. Dodanie kodu metody **dodaj_tytul** do klasy **Fasada** wg diagramu z p.24.5 oraz metody typu set i get dla rerefencji typu **Tytul_książki**



26.4. Kod klasy Fasada

```
package Warstwa_biznesowa;

public class Fasada {
    private Tytul_książki tytul_książki;
    public Fasada() {
    }
    public void dodaj_tytul(String dane_tytul[]) // your code here
    {
        tytul_książki = new Tytul_książki(); //implementacja Create Message
        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]);
    }
    public Tytul_książki getTytul_książki() {
        return tytul_książki;
    }
    public void setTytul_książki(Tytul_książki val) {
        this.tytul_książki = val;
    }
}
```

Referencja do obiektu klasy Tytul_książki reprezentuje relację 1 do 1 po stronie klasy Fasada, która jest „właścicielem” relacji

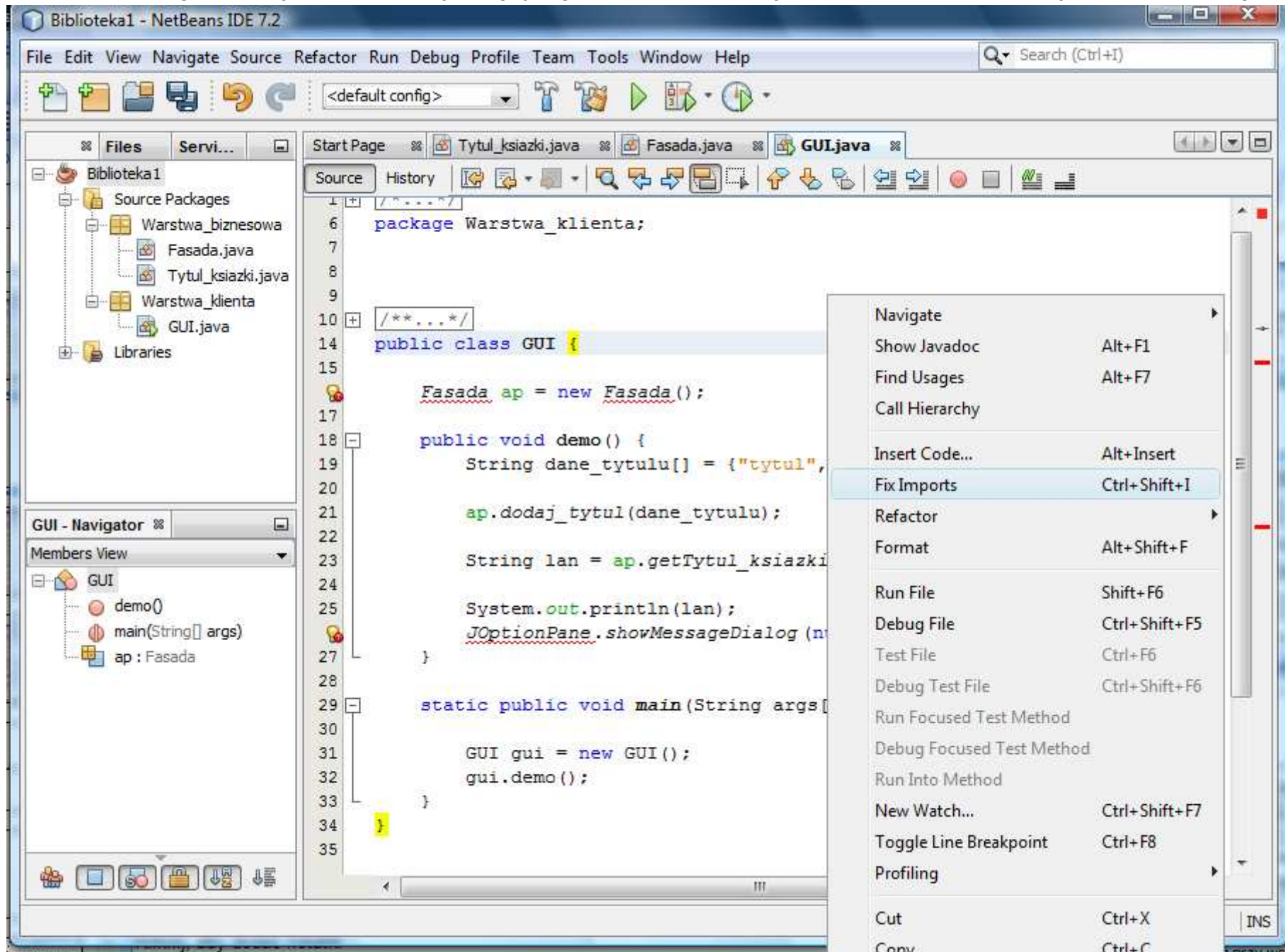
27.1. Kod klasy **GUI** reprezentujący warstwę klienta (interfejs graficzny użytkownika) z pakietu **Warstwa_klienta**

The screenshot displays the NetBeans IDE 7.2 interface for a project named 'Biblioteka1'. The main editor window shows the source code for the `GUI.java` class, which is part of the `Warstwa_klienta` package. The code includes a package declaration, a class definition, a constructor, a `demo()` method, and a `main()` method. The `demo()` method interacts with a `Fasada` object to add a book title and display the result. The `main()` method creates an instance of the `GUI` class and calls its `demo()` method.

```
1  /**...*/
2
3
4
5
6  package Warstwa_klienta;
7
8
9
10 /**...*/
11
12
13
14 public class GUI {
15
16
17     Fasada ap = new Fasada ();
18
19     public void demo() {
20         String dane_tytulu[] = {"tytul", "Jan", "Kowalski", "12345", "W1"};
21
22         ap.dodaj_tytul(dane_tytulu);
23
24         String lan = ap.getTytul_książki().toString();
25
26         System.out.println(lan);
27         JOptionPane.showMessageDialog (null, lan);
28     }
29
30     static public void main(String args[]) {
31
32         GUI gui = new GUI();
33         gui.demo();
34     }
35 }
```

The IDE interface also shows a 'Files' view on the left with a tree structure of the project, including 'Source Packages' like 'Warstwa_biznesowa' and 'Warstwa_klienta', and 'Libraries'. A 'GUI - Navigator' window at the bottom left shows the 'Members View' for the `GUI` class, listing methods `demo()` and `main(String[] args)`, and a field `ap : Fasada`. The status bar at the bottom right indicates '8 | 1 | INS'.

27.2. Definicja klasy **GUI** korzystającej z metod klasy **Fasada** z warstwy biznesowej



The screenshot displays the NetBeans IDE 7.2 interface for a project named "Biblioteka1". The "Files" view on the left shows the project structure with source packages "Warstwa_biznesowa" and "Warstwa_klienta", and files "Fasada.java", "Tytul_książki.java", and "GUI.java". The "GUI - Navigator" view shows the members of the GUI class: demo(), main(String[] args), and ap: Fasada.

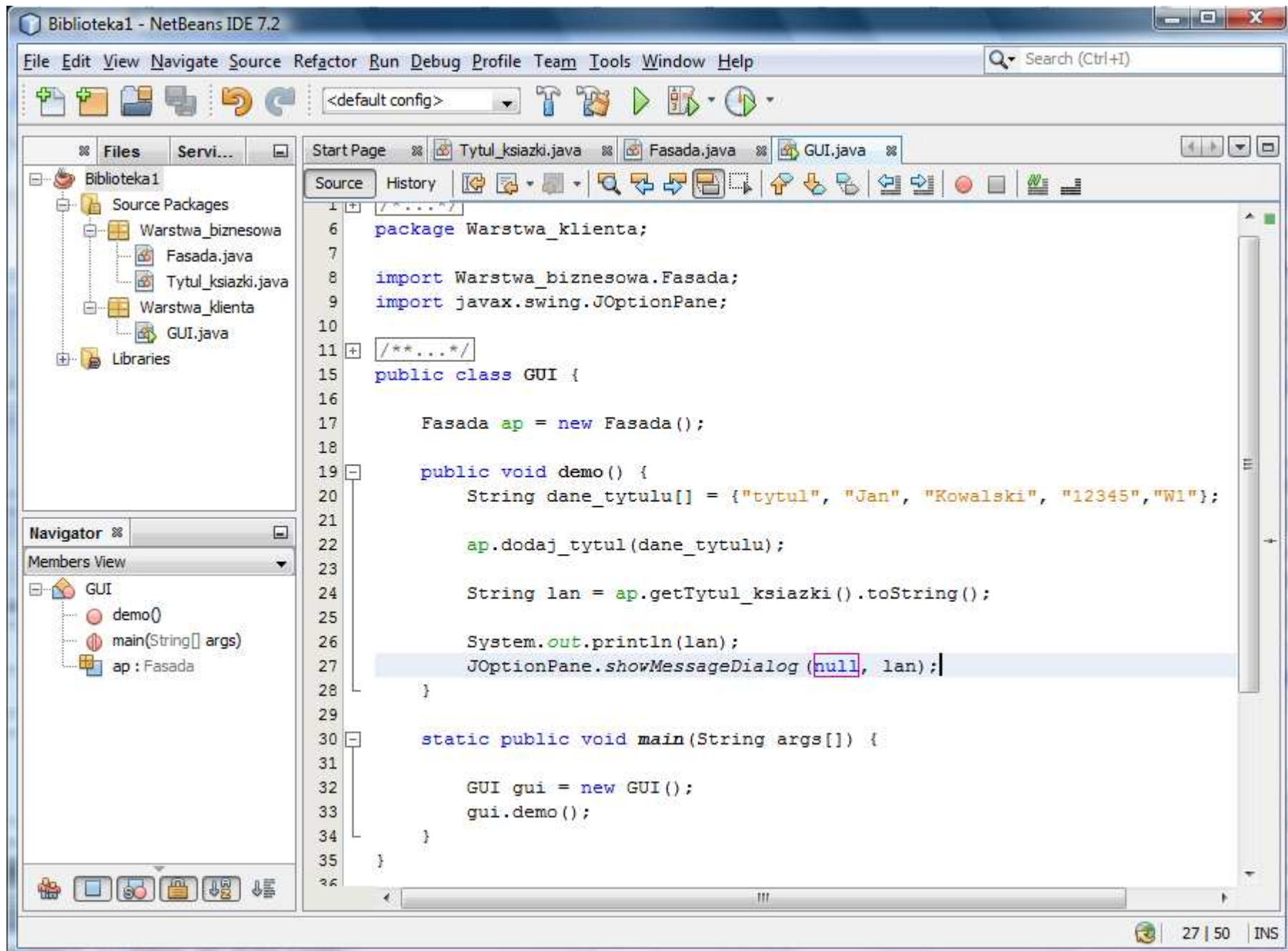
The main editor window shows the source code for the GUI class in the package "Warstwa_klienta". The code is as follows:

```
1 package Warstwa_klienta;
2
3 /**...*/
4 public class GUI {
5
6     Fasada ap = new Fasada();
7
8     public void demo() {
9         String dane_tytulu[] = {"tytul",
10
11         ap.dodaj_tytul(dane_tytulu);
12
13         String lan = ap.getTytul_książki();
14
15         System.out.println(lan);
16         JOptionPane.showMessageDialog(null, lan);
17     }
18
19     static public void main(String args[]) {
20
21         GUI gui = new GUI();
22         gui.demo();
23     }
24 }
```

A context menu is open over the GUI class definition, listing various actions and their keyboard shortcuts:

- Navigate
- Show Javadoc (Alt+F1)
- Find Usages (Alt+F7)
- Call Hierarchy
- Insert Code... (Alt+Insert)
- Fix Imports (Ctrl+Shift+I)
- Refactor
- Format (Alt+Shift+F)
- Run File (Shift+F6)
- Debug File (Ctrl+Shift+F5)
- Test File (Ctrl+F6)
- Debug Test File (Ctrl+Shift+F6)
- Run Focused Test Method
- Debug Focused Test Method
- Run Into Method
- New Watch... (Ctrl+Shift+F7)
- Toggle Line Breakpoint (Ctrl+F8)
- Profiling
- Cut (Ctrl+X)
- Copy (Ctrl+C)

27.3. Kod klasy GUI po uzupełnieniu importów pakietu **Warstwa_biznesowa**



The screenshot displays the NetBeans IDE 7.2 interface. The main editor window shows the source code for the `GUI.java` file. The code is as follows:

```
1  /*****/
2
3  package Warstwa_klienta;
4
5
6  import Warstwa_biznesowa.Fasada;
7  import javax.swing.JOptionPane;
8
9  /*****/
10
11 public class GUI {
12
13     Fasada ap = new Fasada ();
14
15     public void demo() {
16         String dane_tytulu[] = {"tytul", "Jan", "Kowalski", "12345", "W1"};
17
18         ap.dodaj_tytul(dane_tytulu);
19
20         String lan = ap.getTytul_książki().toString();
21
22         System.out.println(lan);
23         JOptionPane.showMessageDialog (null, lan);
24     }
25
26     static public void main(String args[]) {
27
28         GUI gui = new GUI ();
29         gui.demo ();
30     }
31 }
32
```

The IDE interface includes a menu bar (File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help), a toolbar, and a project explorer on the left showing the project structure: Biblioteka1 > Source Packages > Warstwa_biznesowa (Fasada.java, Tytul_książki.java) > Warstwa_klienta (GUI.java). The Navigator window at the bottom left shows the class structure for GUI, including methods `demo()` and `main(String[] args)`, and a field `ap : Fasada`. The status bar at the bottom right shows the page number 27 of 50 and the text INS.

27.4. Kod klasy **GUI** korzystającej z metod klasy **Fasada**

```
package Warstwa_klienta;
```

```
import Warstwa_biznesowa.Fasada;
```

```
import javax.swing.JOptionPane;
```

```
public class GUI {
```

```
    Fasada ap = new Fasada();
```

```
    public void demo() {
```

```
        String dane_tytulu[] = {"tytul", "Jan", "Kowalski", "12345","W1"};
```

```
        ap.dodaj_tytul(dane_tytulu);
```

```
        String lan = ap.getTytul_ksiazki().toString();
```

```
        System.out.println(lan);
```

```
        JOptionPane.showMessageDialog(null, lan);
```

```
    }
```

```
    static public void main(String args[]) {
```

```
        GUI gui = new GUI();
```

```
        gui.demo();
```

```
    }
```

```
}
```

28. Uruchomienie programu – metoda **demo** z klasy GUI pozwala przetestować metody klasy **Fasada** i **Tytuł_książki**

