# Paradygmaty programowania

## Paradygmat programowania

Paradygmat programowania to wzorzec programowania szczególnie ceniony w danym okresie rozwoju informatyki lub dla konkretnych zastosowań.

# Paradygmaty programowania

## Paradygmat programowania

Paradygmat programowania to wzorzec programowania szczególnie ceniony w danym okresie rozwoju informatyki lub dla konkretnych zastosowań.

Przykłady paradygmatów:

- programowanie imperatywne
- programowanie strukturalne
- programowanie obiektowe
- programowanie funkcyjne
- programowanie zdarzeniowe
- ...

# Paradygmaty programowania

## Paradygmat programowania

Paradygmat programowania to wzorzec programowania szczególnie ceniony w danym okresie rozwoju informatyki lub dla konkretnych zastosowań.

Przykłady paradygmatów:

- programowanie imperatywne
- programowanie strukturalne
- programowanie obiektowe
- programowanie funkcyjne
- programowanie zdarzeniowe
- ...

Paradygmaty ułatwiają zrozumienie kodu i dowodzenie jego poprawności.

# Paradygmaty programowania

## Paradygmat programowania

Paradygmat programowania to wzorzec programowania szczególnie ceniony w danym okresie rozwoju informatyki lub dla konkretnych zastosowań.

Przykłady paradygmatów:

- programowanie imperatywne
- programowanie strukturalne
- programowanie obiektowe
- programowanie funkcyjne
- programowanie zdarzeniowe
- ...

Paradygmaty ułatwiają zrozumienie kodu i dowodzenie jego poprawności.

# Obiektowy model programowania

Programowanie obiektowe jest jednym z najpopularniejszych obecnie paradygmatów programowania.
Rzeczywistość jest modelowana jako zbiór obiektów z przypisanymi do nich możliwymi akcjami (metodami).

# Obiektowy model programowania

Programowanie obiektowe jest jednym z najpopularniejszych obecnie paradygmatów programowania.
Rzeczywistość jest modelowana jako zbiór obiektów z przypisanymi do nich możliwymi akcjami (metodami).
Najpopularniejsze języki obiektowe:

- C++
- C#
- Java
- Smalltalk
- PHP

Wiele współczesnych języków wspiera obiektowość.

# Główna idea programowania obiektowego

Programowanie obiektowe opiera się na podstawowej idei:

# Główna idea programowania obiektowego

Programowanie obiektowe opiera się na podstawowej idei:

1. W celu wykonania pewnego zadania znajdujemy kogoś (agenta), któremu je zlecamy.

# Główna idea programowania obiektowego

Programowanie obiektowe opiera się na podstawowej idei:

1. W celu wykonania pewnego zadania znajdujemy kogoś (agenta), któremu je zlecamy.
2. Agent ten jest odpowiedzialny za dane zlecenie i jego zadaniem jest udzielenie nam odpowiedzi.

# Główna idea programowania obiektowego

Programowanie obiektowe opiera się na podstawowej idei:

1. W celu wykonania pewnego zadania znajdujemy kogoś (agenta), któremu je zlecamy.
2. Agent ten jest odpowiedzialny za dane zlecenie i jego zadaniem jest udzielenie nam odpowiedzi.
3. Agent sam wybiera metodę realizacji zlecenia (może np. zlecić jego wykonanie innemu agentowi).

# Główna idea programowania obiektowego

Programowanie obiektowe opiera się na podstawowej idei:

1. W celu wykonania pewnego zadania znajdujemy kogoś (agenta), któremu je zlecamy.
2. Agent ten jest odpowiedzialny za dane zlecenie i jego zadaniem jest udzielenie nam odpowiedzi.
3. Agent sam wybiera metodę realizacji zlecenia (może np. zlecić jego wykonanie innemu agentowi).
4. Nie interesuje nas sposób wykonania zlecenia, a jedynie jego efekt.

# Postulaty programowania obiektowego

1. Hermetyzacja.
2. Dziedziczenie i polimorfizm.
3. Abstrakcja.

# Postulaty programowania obiektowego

## Hermetyzacja

Obiekt udostępnia swój interfejs – metody wejściowe i wyjściowe. Nie jest możliwa modyfikacja stanu obiektu w inny sposób. Zapewnia to większą kontrolę nad obiektem i ułatwia unikanie błędów.

# Postulaty programowania obiektowego

## Dziedziczenie i polimorfizm

Obiekty mogą dziedziczyć cechy i metody – klasa potomna (pochodna) jest szczególnym przypadkiem klasy rodzicielskiej (bazowej), zachowuje jej cechy i dodatkowo posiada swoje własne.

# Postulaty programowania obiektowego

## Dziedziczenie i polimorfizm

Obiekty mogą dziedziczyć cechy i metody – klasa potomna (pochodna) jest szczególnym przypadkiem klasy rodzicielskiej (bazowej), zachowuje jej cechy i dodatkowo posiada swoje własne.

**Przykład:** Klasa `Kwadrat` dziedziczy po klasie `Prostokąt` – każdy kwadrat jest prostokątem i ma wszystkie jego właściwości, a ponadto ma też właściwości charakterystyczne dla kwadratów (np. można weń wpisać okrąg).

# Postulaty programowania obiektowego

## Abstrakcja

Obiekty są abstrakcyjnymi wykonawcami – mogą realizować zlecenia w różny sposób, a zleceniodawca nie musi być wcale tego świadomy – interesuje go tylko wynik zlecenia. Jest to cecha ściśle związana z dziedziczeniem.

# Postulaty programowania obiektowego

## Abstrakcja

Obiekty są abstrakcyjnymi wykonawcami – mogą realizować zlecenia w różny sposób, a zleceniodawca nie musi być wcale tego świadomy – interesuje go tylko wynik zlecenia. Jest to cecha ściśle związana z dziedziczeniem.

**Przykład:** Klasa `AbstrakcyjnySorter` posiada klasy dziedziczne `QuickSorter` i `MergeSorter`. Obiekt wywołujący metodę sortowania traktuje obiekt jako `AbstrakcyjnySorter`, nie wnikając, jaki algorytm sortujący jest faktycznie używany.

# Diagramy klas UML

Diagramy klas są najpopularniejszymi diagramami UML.

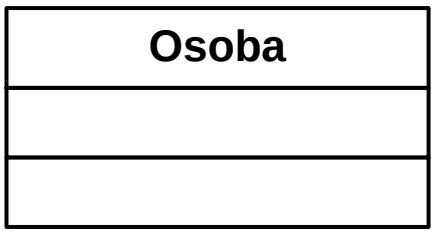
- Pokazują wewnętrzną strukturę systemu.
- Wspierają systemu modelowane obiektowo.
- Mogą posiadać różny poziom szczegółowości.
- Każda informacja na diagramie może zostać pominięta. Z braku informacji na diagramie nie można wysnuwać wniosków dotyczących obecności bądź nieobecności danego elementu w systemie.

# Diagramy klas UML

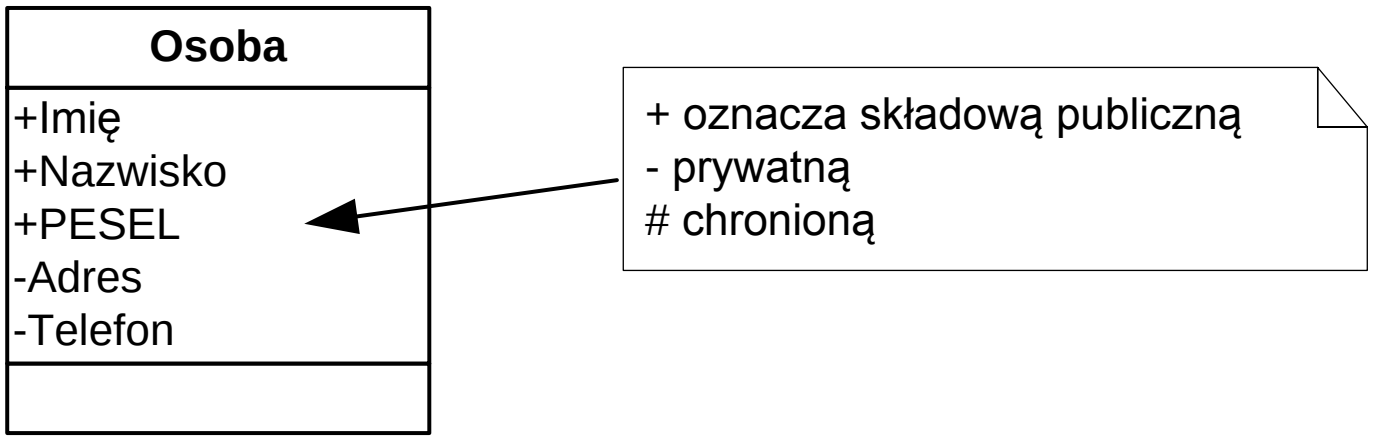
Diagramy klas są najpopularniejszymi diagramami UML.

- Pokazują wewnętrzną strukturę systemu.
- Wspierają systemu modelowane obiektowo.
- Mogą posiadać różny poziom szczegółowości.
- Każda informacja na diagramie może zostać pominięta. Z braku informacji na diagramie nie można wysnuwać wniosków dotyczących obecności bądź nieobecności danego elementu w systemie.
  Ale: Poziom szczegółowości diagramu musi odpowiadać wymaganiom danego dokumentu.

# Diagram klas UML

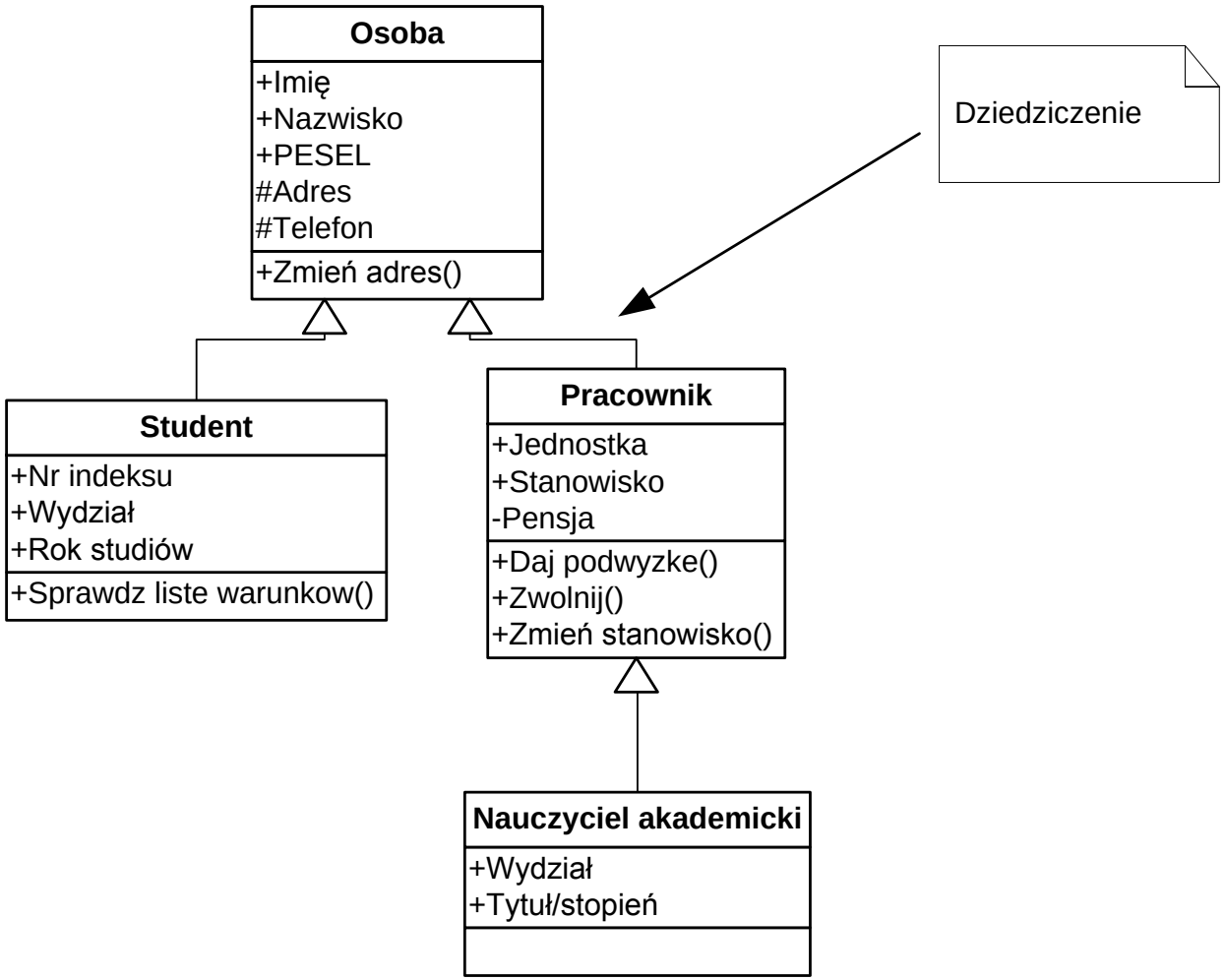


# Diagram klas UML

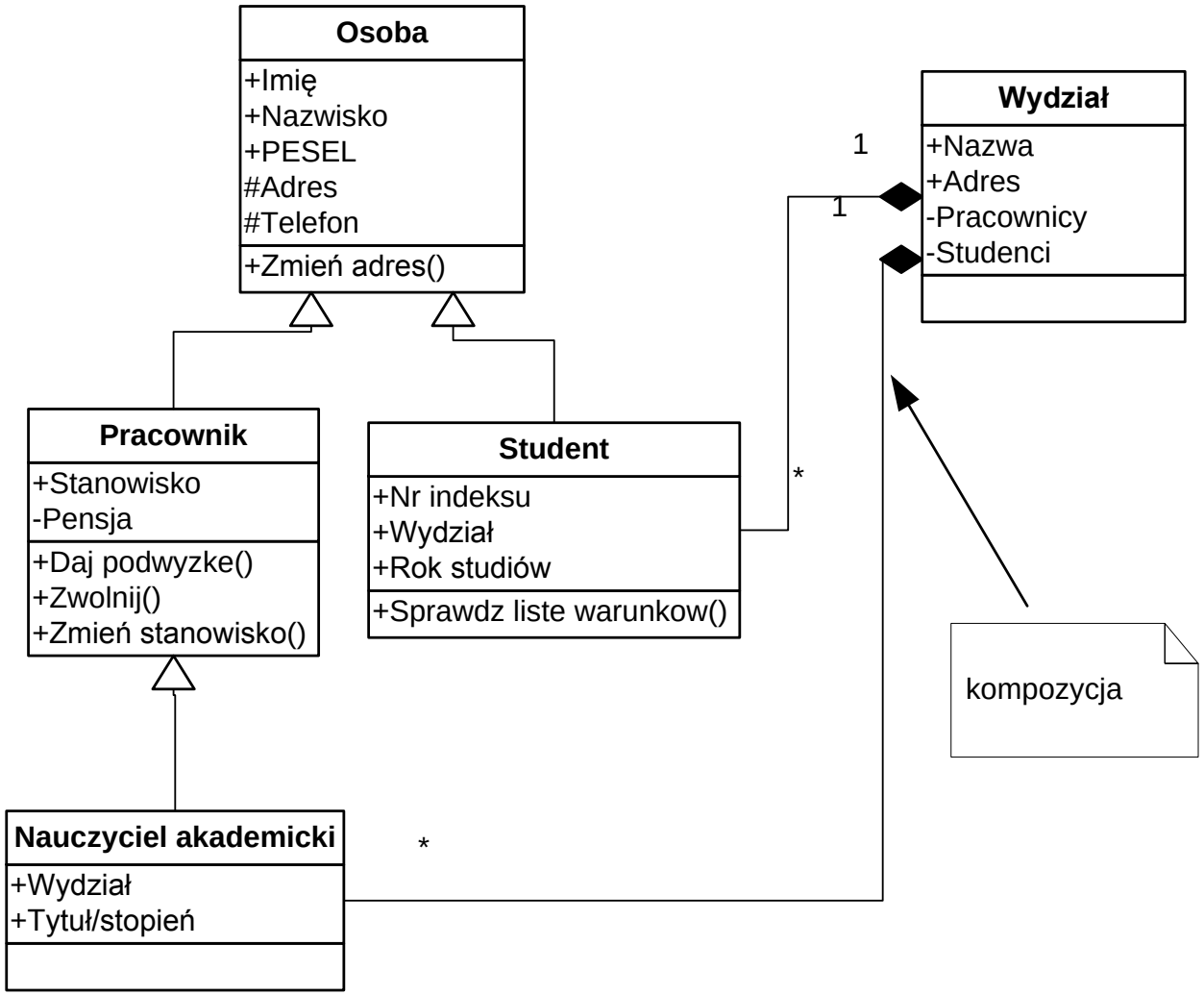


# Diagram klas UML

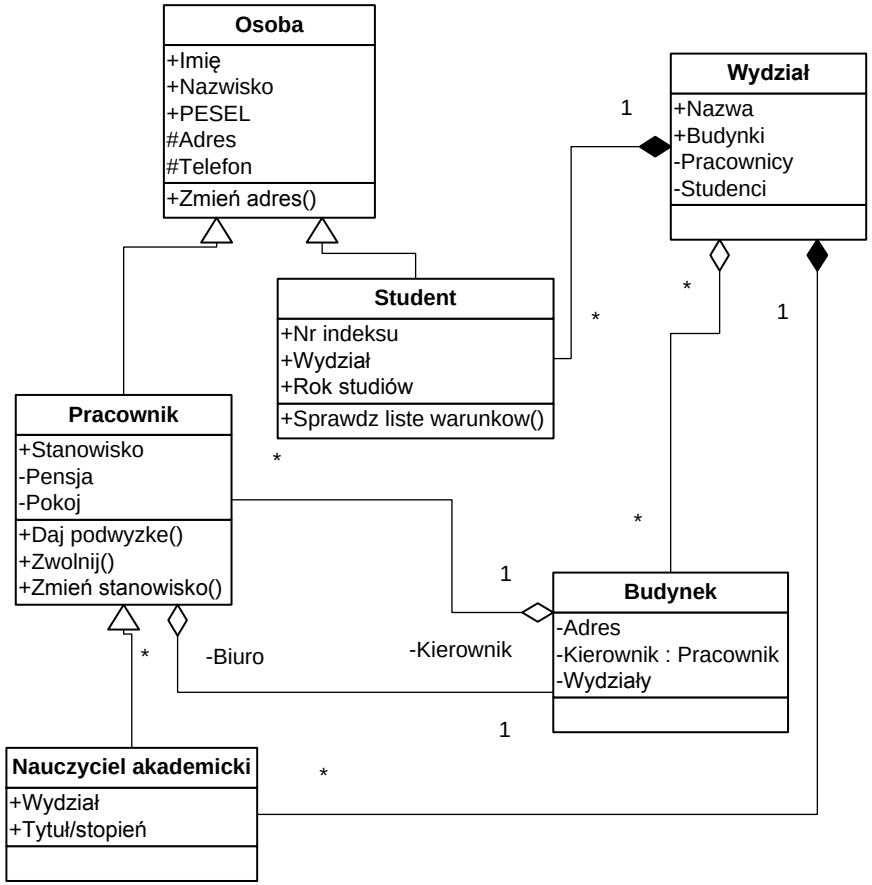
# Diagram klas UML



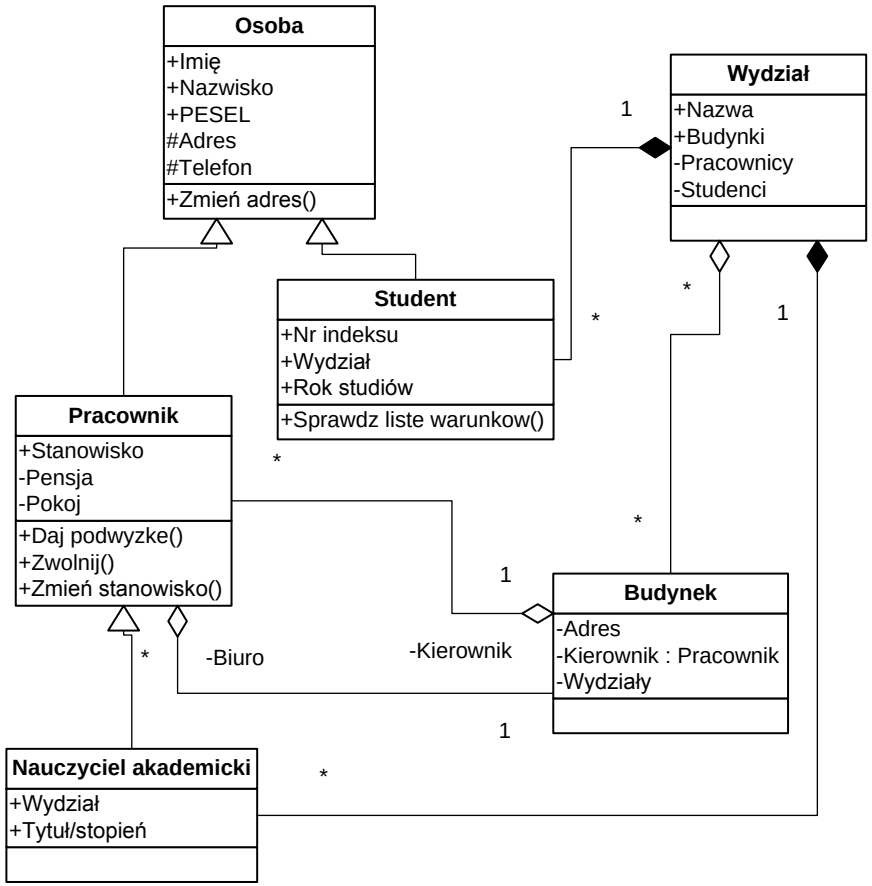
# Diagram klas UML



# Diagram klas UML



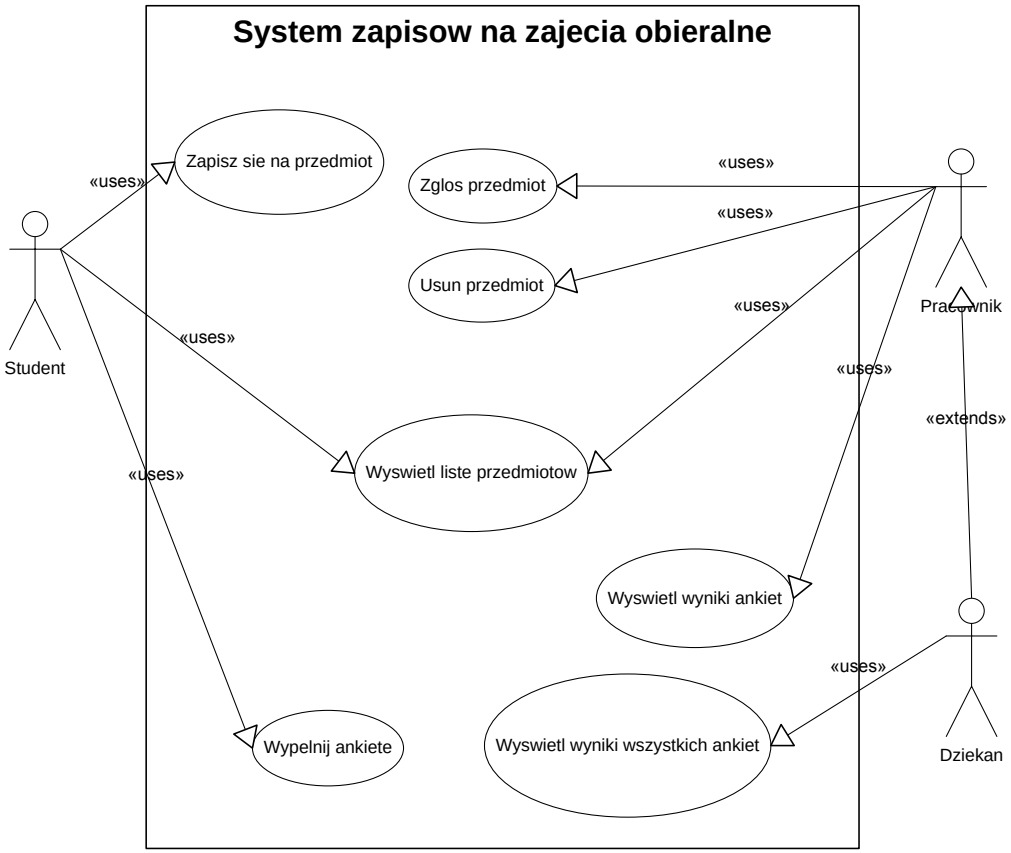
# Diagram klas UML




i tak dalej...

# Diagram klas UML

Zaprojektuj diagram klas dla podanego diagramu przypadków użycia:



# Diagram klas UML

Zaprojektuj diagram klas UML, modelujący:

# Diagram klas UML

Zaprojektuj diagram klas UML, modelujący:

1. przychodnię

# Diagram klas UML

Zaprojektuj diagram klas UML, modelujący:

1. przychodnię
2. sklep internetowy

# Diagram klas UML

Zaprojektuj diagram klas UML, modelujący:

1. przychodnię
2. sklep internetowy
3. bibliotekę