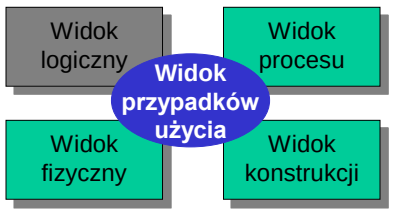
Podstawy projektowania systemów komputerowych

Diagramy klas UML

1

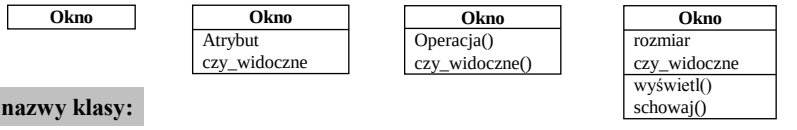
Widok logiczny



- Używany do modelowania części systemu oraz sposobów, w jaki one ze sobą współdziałają.
- Ten widok zazwyczaj tworzą diagramy:
  - Klas,
  - Obiektów,
  - Maszyny stanowej,
  - Interakcji.

2

Podstawowe sposoby przedstawiania klas, różne poziomy szczegółowości

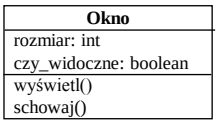


Pole nazwy klasy:

nazwa_ klasy

Pole atrybutów:

dostępność nazwa_atrybutu : typ = wart_początkowa



Pole metod:

dostępność nazwa_metody (lista_arg) : typ_wart_zwracanej

3

Poziomy dostępu

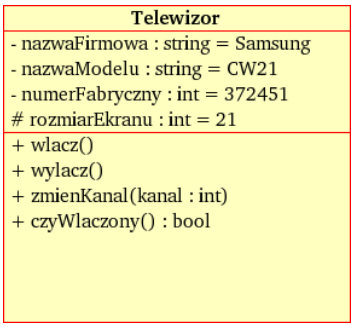
Dostepności:

\+ publiczna

\- prywatna

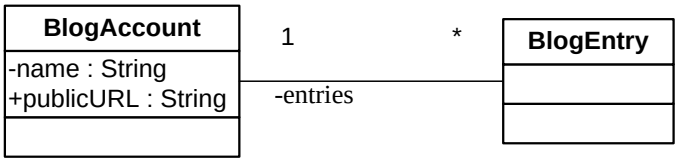
\# chroniona

~ zakres pakietu



4

Atrybuty

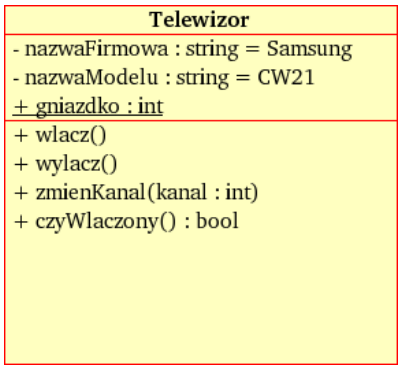


Atrybuty wpisane

Atrybuty zadeklarowane poprzez asocjację

5

Statyczne części klasy



Podkreślenie!

6

Liczebność

Pracownik
Imię[1..2] nazwisko data ur. wiek adres zamieszkania płeć stosunek do służby wojsk. [0..1] lista poprz. miejsc pracy [0..*] adres firmy
policz wiek (imię, nazwisko) policz wiek policz wiek (imię, nazwisko) czy pracował w (nazwa firmy) znajdź najstarszego()

Pozwala określić, że atrybut w rzeczywistości reprezentuje zbiór obiektów.

7

Ogólna deklaracja atrybutu wpisanego

[widoczność] nazwa [liczebność] [: typ] [=wartość początkowa]

- Przykład:

+ wysokosc : double = 10

8

Ogólna deklaracja operacji

[widoczność] nazwa [(lista parametrów)] [: typ wyniku]

- gdzie lista parametrów:

nazwa : typ [=wartość domyślna]

9

Ogólna deklaracja operacji 2

[widoczność] nazwa [(lista parametrów)] [: typ wyniku] [właściwość]

- Nazwa funkcji pisana *kursywą* – metoda abstrakcyjna
- Nazwa podkreślona – metoda wirtualna

10

Związki pomiędzy klasami

- Wymieniane od najsłabszych:
  - Zależność (strzałka przerywana)
  - Asocjacja (pojedyncza linia)
  - Agregacja częściowa (pusta strzałka zakończona rombem)
  - Agregacja całkowita (pełna strzałka zakończona rombem)
  - Dziedziczenie (pusta strzałka)

11

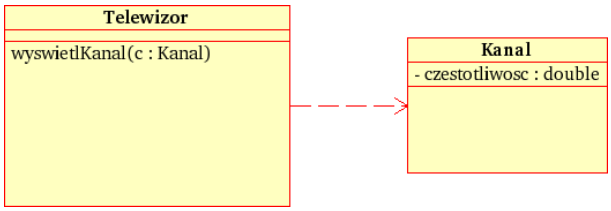
Zależność (1)

- **Zależność** pomiędzy dwiema klasami informuje, że jedna z nich, aby używać obiektów innej, musi mieć o niej informacje.
- Zależność występuje gdy zmiana specyfikacji jednej klasy, może powodować konieczność wprowadzenia zmiany w innej klasie.

12

Zależność (2)

- Najczęściej używa się zależności do pokazania, że jedna klasa używa innej klasy jako parametru jakiejś operacji:



- Obie klasy są zależne od siebie nawzajem w celu zapewnienia poprawnej pracy!

13

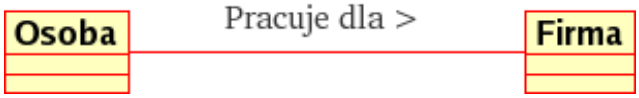
Asocjacja (1)

- Związek asocjacji pozwala jednej klasie na używanie obiektów innej klasy.
- Asocjacja oznacza, że klasa będzie w rzeczywistości:
  - Zawierać w postaci atrybutu odwołanie do obiektu,
  - Zawierać w postaci atrybutu same obiekty,
  - Zawierać inną klasę.

14

Asocjacja (2)

- Domyślnie powiązanie jest dwukierunkowe (jeśli tego nie chcemy, to trzeba do linii dodać strzałkę)



15

Asocjacje (3)

- Na diagramach można umieszczać dodatkowe informacje o powiązaniach
- Rola:

- Krotność:
- Przykładowe krotności: 1, 2, 5, 1..3, *, 3..*, 0,1, 0..6

16

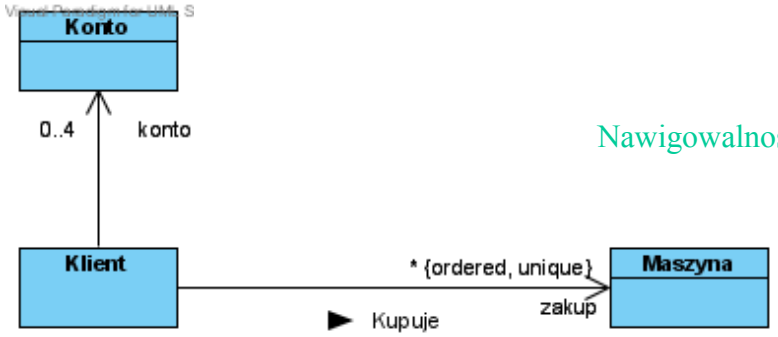
Asocjacje (4)

- Za pomocą dodatkowego atrybutu można określać, czy odwołanie się do powiązania jest dostępne dla innych obiektów nie biorących w powiązaniu udziału (czy jest publiczne):

17

Asocjacje (5)

```
public class Klient {
        public Maszyna[] zakup;
        //….
}
```

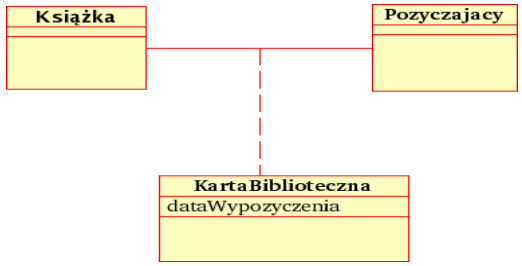


Nawigowalność!

18

Klasy asocjacyjne

- Sama asocjacja może powodować powstawanie nowych klas.
- Powiązanie może mieć atrybuty i operacje, tak samo jak każda inna klasa.



19

Agregacja częściowa

- W rzeczywistości jest silniejszą wersją asocjacji.
- Zaznacza, że jedna klasa w rzeczywistości posiada obiekty innej, ale może je jednocześnie również współdzielić.

20

Agregacja całkowita (kompozycja, złożenie)

- Jeśli obiekt należy tylko do jednej całości, jest tworzony i likwidowany razem z całością, nazywamy to agregacją całkowitą.
- Kompozycja oznacza, że cykl życiowy składowej zawiera się w cyklu życiowym całości, oraz że składowa nie może być współdzielona.

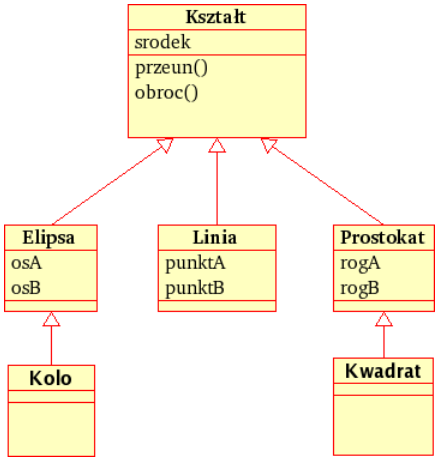
21

Uogólnienie (dziedziczenie)

- Używa się w celu opisania klasy, która jest **rodzajem** innej klasy.
- Uogólnienie/dziedziczenie jest relacją pomiędzy klasą ogólniejszą (nazywaną klasą rodzicem, klasą bazową itp.) a bardziej szczegółową (nazywaną podklasą, albo klasą dzieckiem) – reprezentuje stwierdzenie: “A jest rodzajem B”.

22

Uogólnienie - przykład



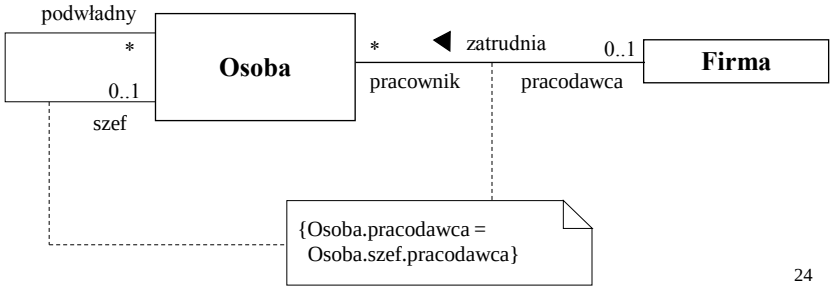
Klasa posiadająca klasy potomne, ale nie posiadająca rodziców jest nazywana klasą korzeniem (ang. *root class*)

Klasa bez potomków nazywana jest klasą liściem (ang. *leaf class*)

23

Przykład ograniczenia

- Określenie niezmienników klasy wymaga użycia języka OCL.



24

Klasy abstrakcyjne

- Nazwa klasy pisana *kursywą*.
- Metody abstrakcyjne pisane *kursywą*.

Bryła
pole podstawy wysokość
policz objętość()

25

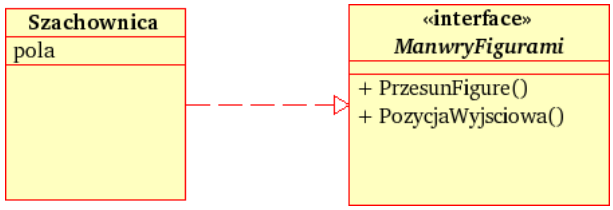
Interfejsy

- Interfejs jest zbiorem operacji, które nie mają odpowiadających im metod implementujących.
- Przypominają klasy abstrakcyjne, w C++ są implementowane w postaci klas abstrakcyjnych.

26

Interfejsy, **realizacja**

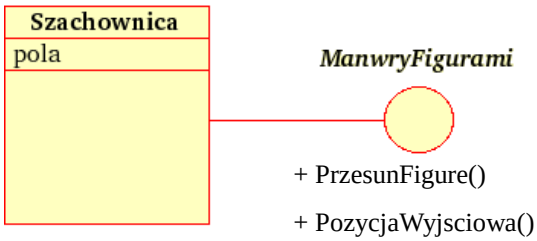
- Interfejs jest przedstawiany podobnie do klasy – nie ma jednak właściwości.



- Związek pomiędzy klasą a interfejsem nazywa się *realizacją*

27

Intrfejsy, notacja „lizakowa”



W tej notacji operacje interfejsu podajemy alternatywnie!

28

Inżynieria wsteczna / w przód

- Inżynieria wprzód – generowanie kodu na podstawie istniejących diagramów UML. Podczas tego procesu traci się część informacji.Choć nie specyfikuje jak takie mapowanie ma być przeprowadzone, UML był tworzony tak aby dało się go przeprowadzić.

- Inżynieria wsteczna – tworzenie diagramów UML na podstawie istniejącego kodu programu.

29