Wydział Elektrotechniki i Informatyki
Politechniki Lubelskiej

Programowanie obiektowe

dr hab. inż. Dariusz Czerwiński, prof. PL

Materiały pomocnicze do wykładu

Program przedmiotu

- **Podstawy programowania obiektowego:**
  - **Pojęcie obiektu i typu obiektowego,**
  - **Interfejs i implementacja obiektu,**
  - **Część prywatna i publiczna obiektu,**
  - **Relacje pomiędzy obiektami**
- **Obiekty w C++:**
  - **Klasy abstrakcyjne jako interfejsy,**
  - **Dziedziczenie,**
  - **Przykłady interfejsów i implementacji**

Program przedmiotu

- **Wprowadzenie do programowania wizualnego.**
  - **Przegląd istniejących środowisk graficznych oraz narzędzi wizualnego programowania.**
  - **Przykłady zastosowań programowania wizualnego.**
  - **Konstruowanie graficznych interfejsów programów użytkowych.**
- **Cechy programowania wizualnego:**
  - **Obiektowa architektura systemów środowisk graficznych,**
  - **Sterowanie komunikacją,**
  - **Struktury ekranu i raportu,**
  - **Technologia szybkiego wytwarzania aplikacji (RAD).**
- **Wprowadzenie do procesu kompilacji.**
  - **Budowa i programowanie kompilatorów**

Program przedmiotu

- **Klasy i zdarzenia**
- **Komunikacja pomiędzy obiektami**
- **Podstawowe biblioteki obiektowe w programowaniu wizualnym:**
  - **Object Window Library (OWL) firmy Inprise,**
  - **Foundation Class Library (MFC) firmy Microsoft.**
- **Rozwój bibliotek komponentów w programowaniu wizualnym:**
  - **Visual Component Library (VCL) firmy Inprise.**
- **Generacja kodu źródłowego w programowaniu wizualnym:**
  - **metody automatycznego generowania kodu,**
  - **analiza kodu, ingerencja programisty.**

Podstawowe aspekty programowania

- **Programowanie:**
  - **Kontrolowanie:**
    - **Maszyna potrafi zrobić tylko to, co chce programista lub na co pozwoli jej napisany program (pod warunkiem pełnej świadomości twórcy).**
  - **Nauczanie:**
    - **Komputer jest w stanie nauczyć się nowych funkcji, jeśli wskażemy mu jak to zrobić.**

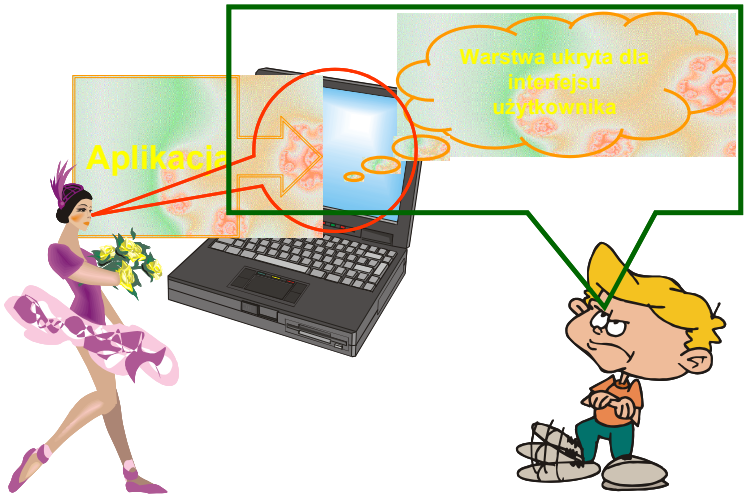
- Proces obliczeniowy zadanego problemu:
  - Zadanie obliczenia najkrótszej trasy przejazdu.
- Kreatywność:
  - Należy dobrać najlepsze rozwiązanie (spośród wielu możliwych) dla zadanego problemu.
- Modelowanie:
  - Przedstawienie w abstrakcyjnej formie, poprzez opis szczególnych właściwości rzeczywistych obiektów.
- Abstrakcja:
  - Wyselekcjonowanie ważnych szczegółów bez opuszczenia szczególnie ważnych właściwości.

Program:

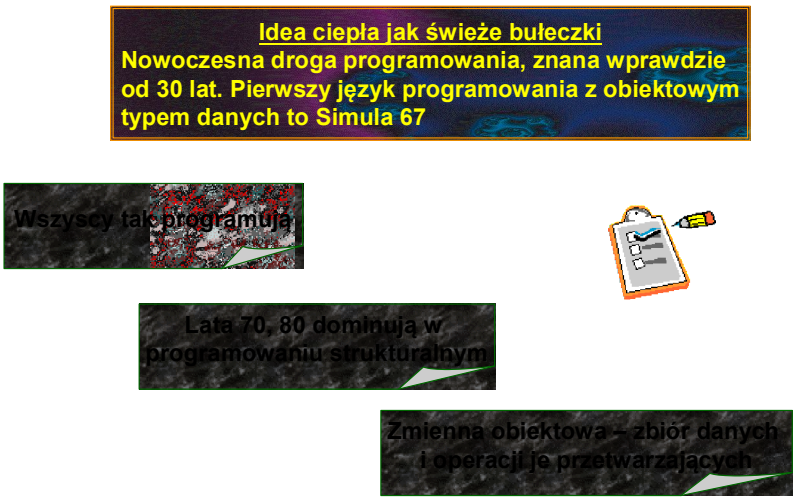
- **Model złożonego systemu:**
  - Model: Zespolony fragment pewnego systemu, reprezentujący szczególnie istotne szczegóły rzeczywistości w abstrakcie opisującym.
  - System: Kolekcja (zbiór) komponentów synchronicznie ze sobą współpracujących.
- **Specyficzne odzwierciedlenie poszczególnych sekwencji w języku programowania:**
  - Składnia: formułowanie ciągu instrukcji według ściśle określonych reguł gramatycznych.
  - Semantyka: rozumienie (interpretacja) zadanych instrukcji.
- **Instrukcje wpisane przez programistę:**
  - Zaimplementowanie w konkretnym języku programowania.
  - Język programowania wymaga precyzyjniejszego opisu niż język komunikacji międzyludzkiej.
  - Programy nie mogą zawierać dwuznaczności.
- **Wykonywany przez komputer za pomocą realizacji indywidualnych instrukcji.**

Warstwy środowiska

Spojrzenie na aplikację

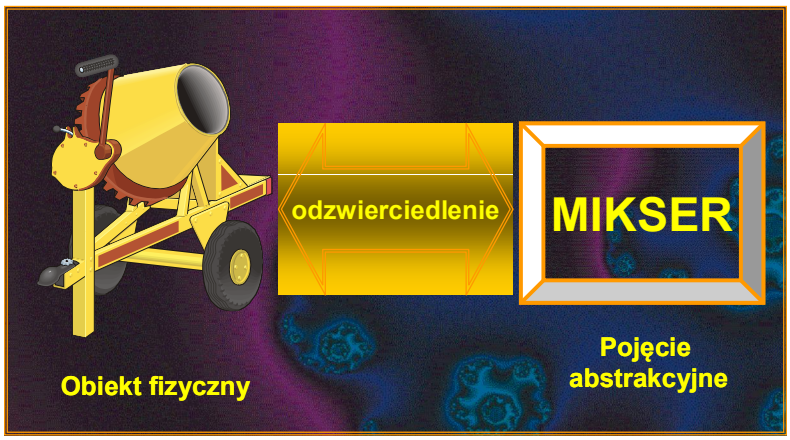


Programowanie zorientowane obiektowo



Definicja obiektu

- **Obiekt: Odzwierciedlenie rzeczywistości w określonym pojęciu abstrakcyjnym:**



Definicja obiektu

Definicja obiektu

```
Mikser = Object
            Liczba_obrotów:Integer;
            Dopuszczalnaładowność:Real;
            RodzajZawieszenia:String;
         End;
```

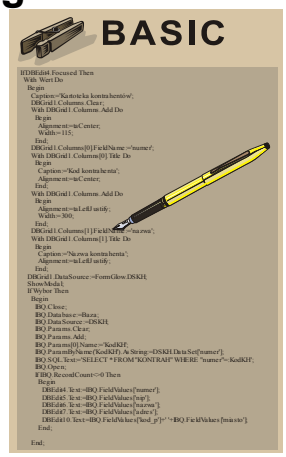
Definicja obiektu

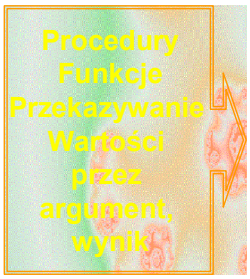
```
Mikser = Object
            Liczba_obrotów:Integer;
            DopuszczalnaładownokśćReal;
            RodzajZawieszenia:String;
            Procedura Zmieszaj;
            Procedura Wylej;
            Procedura Odjeżdżaj;
            Funkcja Samotestowanie:Boolean;
         End;
```

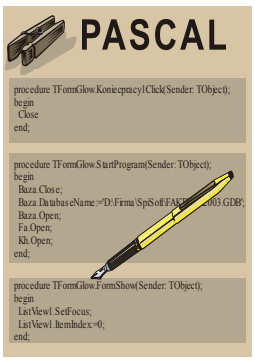
Definicja obiektu

- Obiektowym typem danych nazywamy określony złożony zbiór danych typu prostego połączonych w jedną strukturę oraz procedury i funkcje do przetwarzania tych danych. W definicji obiektu deklaruje się ilość i typ danych oraz predefiniuje się procedury i funkcje do ich przetwarzania.
- Każdą procedurę i funkcję zdefiniowaną w typie obiektowym nazywa się metodą danego typu obiektowego.

Aspekty programowania obiektowego







Programowanie monolityczne

Programowanie modułowe

Aspekty programowania obiektowego

- **Klasa – jest rozszerzeniem obiektowego typu danych, dla którego metody ściśle są ze sobą powiązane (współpracujące) wewnątrz niej samej**
- **Relacje pomiędzy klasami:**
  - **Dziedziczenie (forma wielokrotnego wykorzystania kodu) - nowe klasy tworzy się na podstawie istniejących klas:**
    - **wchłania cechy i zachowania obiektu (klasy potomnej), dołączając je do własnych.**
  - **Zastępowanie metod - przedefiniowanie odziedziczonych metod.**
  - **Podklasa dziedziczy od nadklasy.**
  - **Bezpośrednia nadklasa - podklasa jawnie dziedzicząca**
  - **Pośrednia nadklasa - podklasa dziedziczy od dwóch albo więcej poziomów w górę (hierarchia klas)**
- **Dostęp do modyfikacji:**
  - **Public.**
  - **Private.**
  - **Protected.**

Aspekty programowania obiektowego

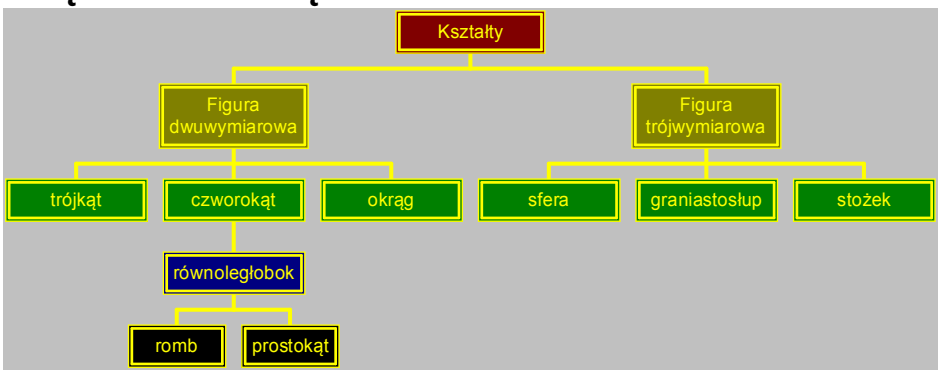
- **Wprowadzenie dostępu chronionego (protected)**
- **Relacje:**
  - **„is" – dziedzictwo:**
    - **Obiekt podklasy "jest" obiektem nadklasy.**
  - **„has" – kompozycja:**
    - **Obiekt „zawiera" obiekt innej klasy jako członka.**
- **Biblioteki klas:**
  - **nowe klasy mogą dziedziczyć od siebie.**
  - **Oprogramowanie może zostać zbudowane od standaryzowanego, nadających się do wielokrotnego użycia komponentów (jak sprzęt komputerowy).**

- Utworzy potężniejsze oprogramowanie.

Klasy podrzędne i nadrzędne

- **Przykład dziedziczenia:**
  - Prostokąt "jest" czworokątem.
  - Prostokąt jest określonym typem czworokąta.
  - Istnieje nadklasa czworokątów – prostokąt jest podklasą.
  - Określenie czworokąt "jest" prostokątem jest niepoprawne.
  - Nazywa może mylić, ponieważ podklasa ma więcej cech niż nadklasa.
  - Podklasa jest określona przez nadklasę.
  - Każda podklasa "jest" przedmiotem nadklasy, ale nie na odwrót.
- **Drzewo dziedzictwa – hierarchia struktur.**

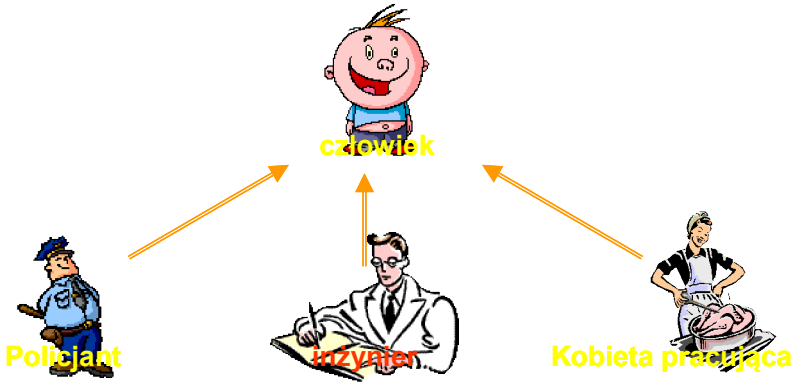
Klasy podrzędne i nadrzędne



Klasy podrzędne i nadrzędne

Użycie dziedzictwa

- pewne zachowania albo dane są wspólne między grupą klas a tworzoną nową klasą
- Część wspólna kilku klas może zostać zdefiniowana przez nadklasę.
- unikalne elementy mogą zostać zdefiniowane przez poszczególne podklasy.



Przykłady

```
Format:
  Policjant =  Class (Człowiek)
                    Definicja klasy podrzędnej (tylko kodu indywidualnego)
               End;

Przykład:
  Smok = Class(Potwór)
          public
            Procedure wyświetl_Inaczej;
            Writeln(„Zieje ogniem");
          End;
```

Relacje pomiędzy klasami

- **Klasa nadrzędna (nadklasa):**
  - **Człon Public: Dostępny, gdziekolwiek program ma odniesienie do nadklasy albo typu podklasy.**
  - **Człon Private: Dostępny tylko w metodach nadklasy.**
  - **Człon Protected: pośrednie zabezpieczenie pomiędzy członem public a private.**
    - **Dostępny przez metody nadklasy, podklasy, albo klas w tej samej hirarchii.**
- **Metody klasy podrzędnej:**
  - **Mogą odwoływać się do metod członów public lub protected przez nazwę.**
- **Obiekt podklasy:**
  - **Może zostać potraktowany jako obiekt nadklasy.**
    - **Nie spełniony jest warunek odwrotny**
  - **Wiele klas może dziedziczyć od jednej nadklasy.**
  - **Wybór odniesień nadklasy**
  - **Przetwarzanie wszystkich obiektów jak obiekty jednej nadklasy**
- **Szczegółowe zaszeregowanie:**
  - **Zmiana odniesienia nadklasy do odniesienie podklasy (downcasting).**
  - **Może zostać wykonane wtedy gdy odniesienia nadklasy właściwie odnoszą się do obiektu podklasy.**

Relacje pomiędzy klasami

- **Metody przełączające**

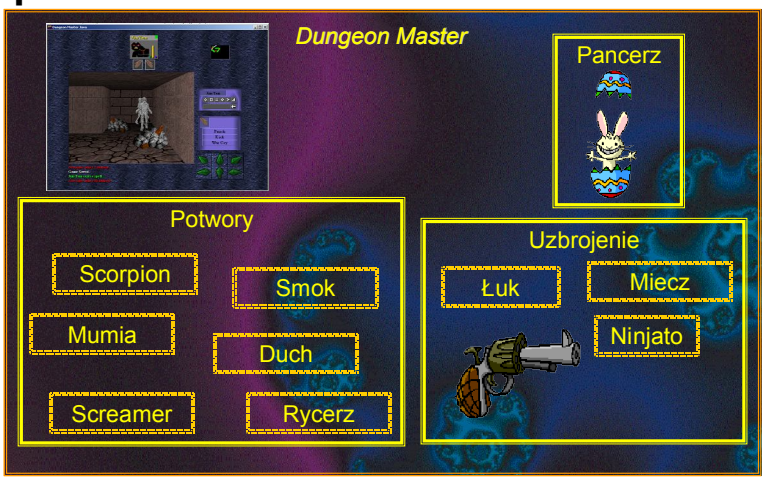
- Podklasa może przedefiniować metody nadklasy
- Odwołanie do konstruktora nadklasy:
  - Constructor(); // jeśli wymagany - podać argumenty
- Jeśli odwołanie jest jawnie, musi być użyte jako pierwsze.

- **Związek("knows-a"):**
  - Relacja może zaistnieć między dwoma klasami:
    - jeśli w metoda jednej klasy lub klas, istnieje jako zmienna lokalna przykład innej klasy
    - jeśli atrybuty jednej klasy są cechą innej klasy.
- **Agregacja ("has-a")**
  - Relacja agregacji istnieje między dwoma klasami jeśli:
    - Pole jednej klasy jest składową innej klasy.
    - Druga klasa jest częścią pierwsza klasy.
- **Dziedzictwo ("is-a")**
  - Relacja dziedzictwa istnieje między dwoma klasami, jeśli jedna klasa jest klasą macierzystą innej klasy.

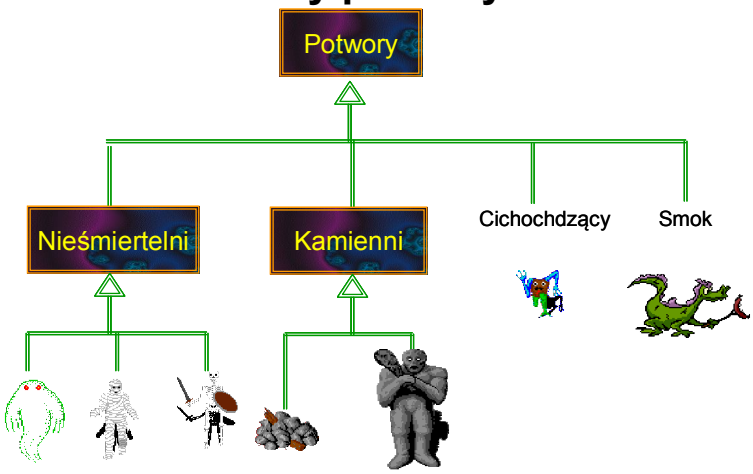
Poziomy dostępu

Poziom	Zakres dostępu		
	Klasa	Podklasa	Nie w podklasie
Public	Tak	Tak	Tak
Protected	Tak	Tak	Brak
Private	Tak	Brak	Brak

Przekleństwo przeszłości



Hierarchia dziedziczenia klasy potwory



Sub-Hierarchia Smoki