Wprowadzenie do systemów informacyjnych

Projektowanie obiektowe

informatyka stosowana

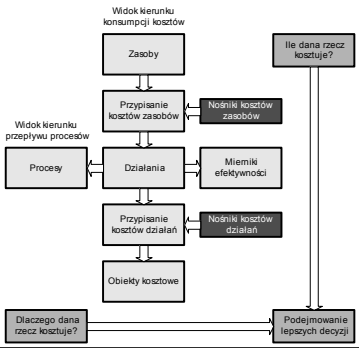
© UEK w Krakowie Ryszard Tadeusiewicz

Uwagi ogólne:

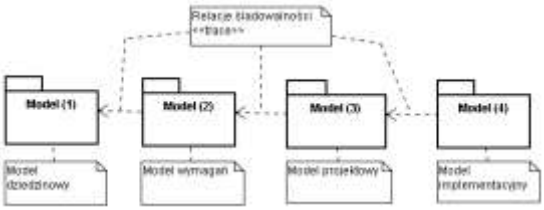
- **Obiektowość jest nową ideologią, która zmienia myślenie realizatorów SI z "zorientowanego na maszynę" na "zorientowane na człowieka".**

- **Obiektowość** jest konsekwencją kryzysu oprogramowania: kosztów związanych z oprogramowaniem, jego zawodnością i trudną do opanowania złożonością.
- Obiektowość przenika wszelkie fazy projektowania, narzędzia i interfejsy.
- Obiektowość dopracowała się własnej kolekcji pojęć i narzędzi.
- Obiektowość jest wciąż jeszcze na początku swojej drogi i musi walczyć z konserwą i spuścizną poprzednich ideologii.

Punktem wyjścia w obiektowym tworzeniu systemu informacyjnego jest zawsze pewien model biznesowy.
Przykład: Diagram przepływu kosztów w metodzie ABC



Potem, stosownie do potrzeb, tworzy się kolejne modele

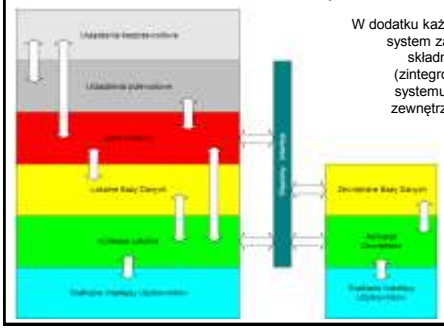


System informacyjny składa się z wielu elementów o zróżnicowanych zadaniach

Na szczęście moduły, z jakich składa system w ujęciu obiektowym mają powtarzalną strukturę

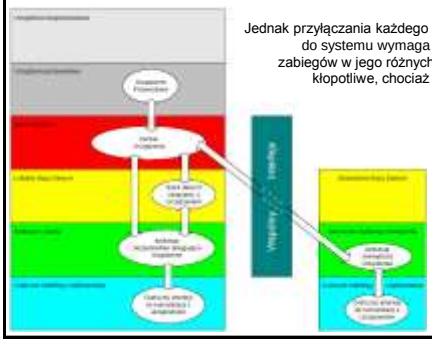
Dane
Aplikacje (klasy)
Graficzny interfejs użytkownika

Orientację w tej złożonej materii ułatwia fakt, że większość systemów informacyjnych ma obecnie strukturę warstwową

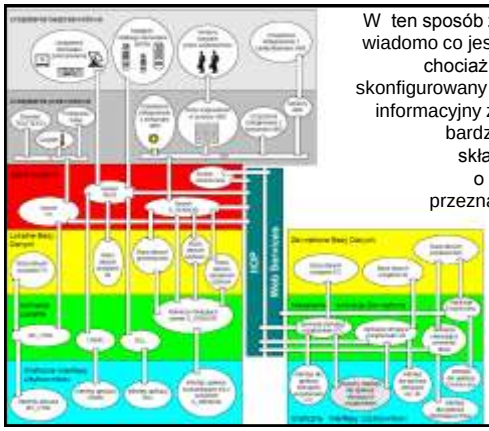


W dodatku każdy nowoczesny system zawiera zarówno składniki wewnętrzne (zintegrowane z jądrem systemu) oraz składniki zewnętrzne (satelitarne)

Dzięki temu współczesne systemy informacyjne odznaczają się dużą elastycznością, co oznacza, że można do nich w każdej chwili dołączać nowe komponenty.

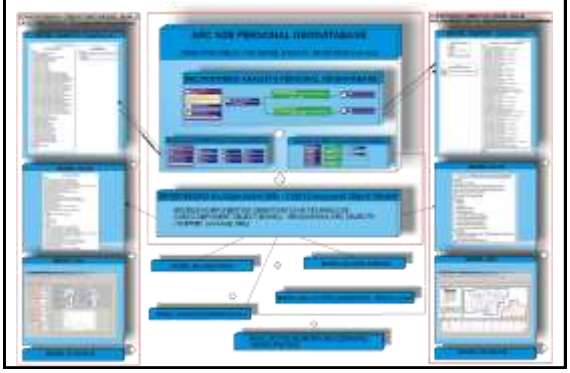


Jednak przyłączania każdego nowego urządzenia do systemu wymaga wykonania szeregu zabiegów w jego różnych warstwach, co jest kłopotliwe, chociaż na ogół odbywa się automatycznie.

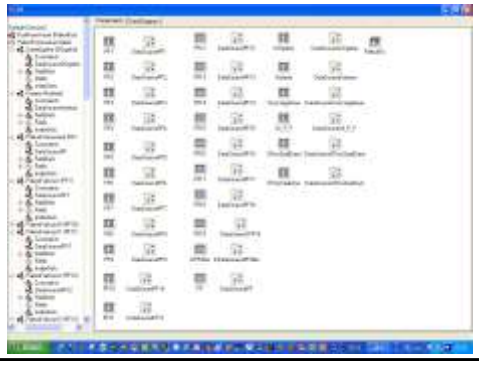


W ten sposób zawsze wiadomo co jest czym, chociaż w pełni skonfigurowany system informacyjny zawiera bardzo wiele składników o różnym przeznaczeniu

Konkrety przykład systemu budowanego obiektowo



Źródła danych i związane z nimi zasoby



Podstawowe koncepcje projektowania obiektowego

- Koncepcja obiektu → stan wewnętrzny; sposób zachowania; tożsamość unikatowa
- Zasada abstrakcji → kompozycyjnej; uogólniającej
- Zasady hierarchizacji → kompozycyjnej; dziedziczenia
- Zasada modularności
- Zasada enkapsulacji

Modelowanie i projektowanie obiektowe - analiza

1. Identyfikacja kluczowych abstrakcji:

a) podejście klasyczne - kategorie a'priori (rzeczy, zdarzenia, role, miejsca, struktury),

b) analiza zachowań:

- analiza odpowiedzialności - najpierw identyfikacja zadań pełnionych przez składowe systemu, następnie grupujemy składowe na bazie podobieństwa zadań,
- analiza funkcji systemu - najpierw identyfikujemy zbiór funkcji systemu, następnie definiujemy obiekty, które mają je realizować,

Modelowanie i projektowanie obiektowe - analiza

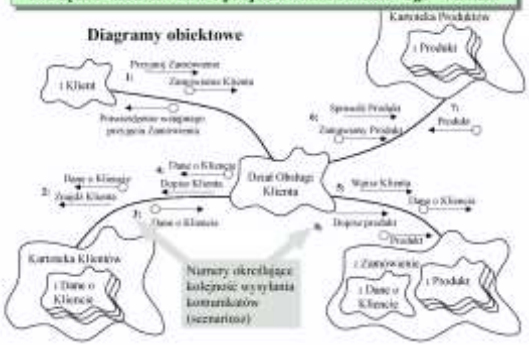
- analiza dziedziny:
konstrukcja ogólnego modelu problemu na bazie wiedzy ekspertów → analiza analogicznych systemów informatycznych - dziedziny rozwiązań → wspólne z ekspertami analizy tych systemów (podobieństwa i różnice) → uszczegółowienie modelu problemu.

2. Konstrukcja ***scenariuszy*** - obiekty otrzymują ***role***.

3. Zdefiniowanie ***odpowiedzialności*** klas - klasy otrzymują zdolności (zbiory akcji) pozwalające ich obiektom odgrywać role.

Narzędzia modelowania i projektowania obiektowego - analiza

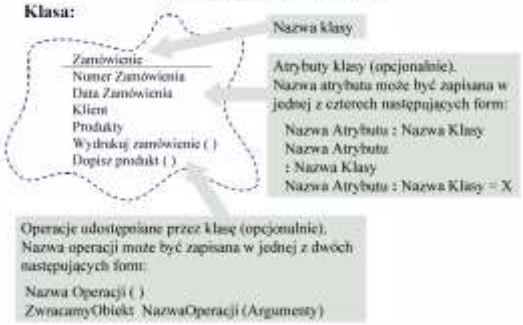
Diagramy obiektowe



Modelowanie i projektowanie obiektowe

Przejście od obiektów do klas

Klasa:



Modelowanie i projektowanie obiektowe - projektowanie

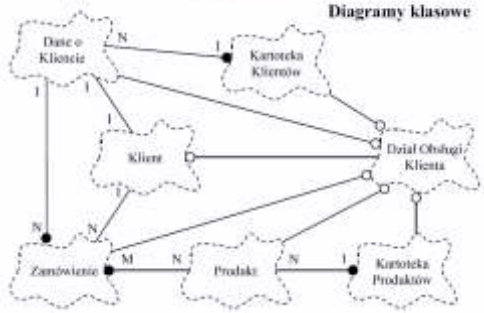
1. Specyfikacja (formalna) warstwy zewnętrznej klas.

2. Konstrukcja modeli ***współpracy*** obiektów (definiują jak obiekty powinny się zachowywać i komunikować, aby uzyskać funkcjonalność opisaną przez scenariusze) oraz ***mechanizmów*** współpracy klas. Powstają struktury: obiektowe i klasowe.

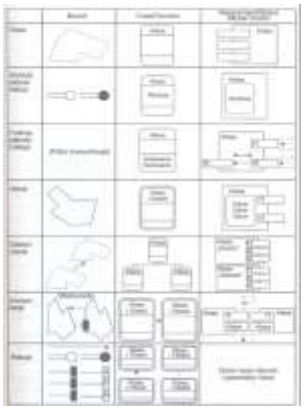
3. Zdefiniowanie architektury systemu (modularyzacja, podsystemy, architektura HW).

Narzędzia modelowania i projektowania obiektowego - projektowanie

Diagramy klasowe

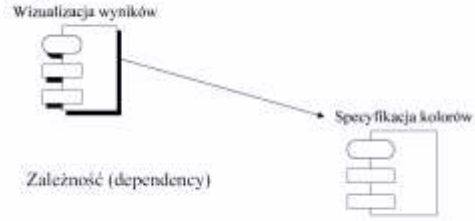


Ogólna „ideologia" obiektowa jest przez różnych autorów różnie realizowana praktycznie (także w sensie reprezentacji graficznej)



Narzędzia modelowania i projektowania obiektowego - projektowanie

Diagramy modulowe



Moduł A jest zależny od modułu B jeśli kompilacja modułu A wymaga wcześniejszej kompilacji modułu B.

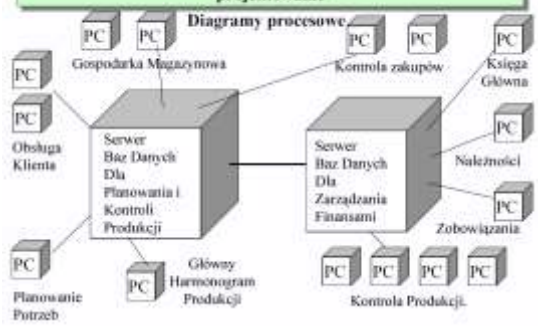
Narzędzia modelowania i projektowania obiektowego - projektowanie

Diagramy modulowe

Służą do grupowania modułów i są wykorzystywane w przypadku rozbudowanych systemów oprogramowania (o dużej liczbie modułów).

Narzędzia modelowania i projektowania obiektowego - projektowanie

Diagramy procesowe



Podstawowe pojęcia metodologii obiektowej:

Obiekt (ang.: *Object*) jest podstawowym pojęciem w podejściu obiektowym. Obiekt reprezentuję sobą konkretny pojedynczy byt.

Obiekt jest charakteryzowany poprzez:
- **identyfikator** (nazwę),
- **stan** (wartości atrybutów obiektu) oraz
- **zachowanie** (operacje obiektu).

Zachowanie może zmieniać stan obiektu, od którego pochodzi i/lub stany innych obiektów.

Klasa (ang.: *Class*) reprezentuje zbiór obiektów, które dzielą strukturę i wspólne zachowanie.

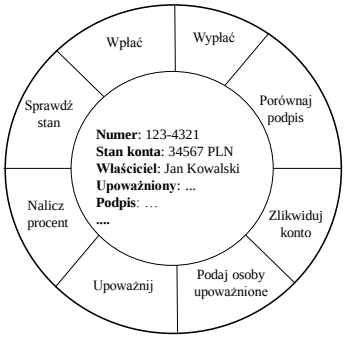
W projektowaniu obiektowym kluczowym pojęciem jest **klasa**

Takie zamknięcie danych w „pancerzu" skojarzonych z nimi metod nazywa się **enkapsulacją**

Obiekty zewnętrzne mają dostęp wyłącznie do funkcji (nazywanych też metodami) za pomocą których mogą **zlecać** wykonanie określonych operacji na danych

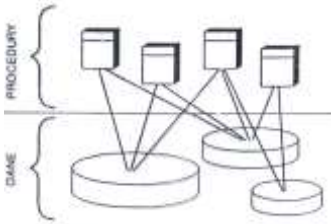
Cechą charakterystyczną koncepcji klasy jest **ukrycie danych** przed obiektami zewnętrznymi.

Przykład obiektu



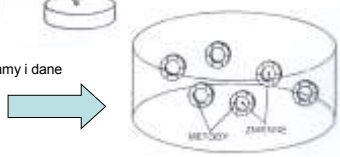
Obiekt
KONTO

Dzięki temu można zmienić tradycyjny układ systemu, w którym wiele programów korzysta z wielu danych

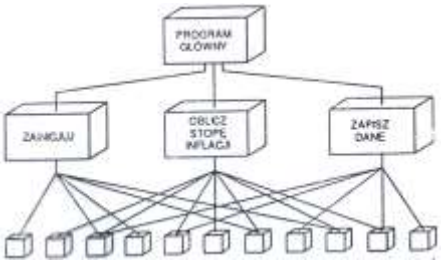


W takim systemie każdy z obiektów może być programowany przez oddzielnego programistę i nie ma potrzeby uzgadniania w zespole projektowym wewnętrznych rozwiązań poszczególnych obiektów

Na nowy układ, w którym programy i dane są zblokowane razem

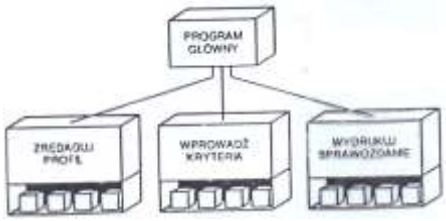


Sposoby dzielenia danych pomiędzy programy w klasycznych (nie obiektowych) metodach projektowania (1)



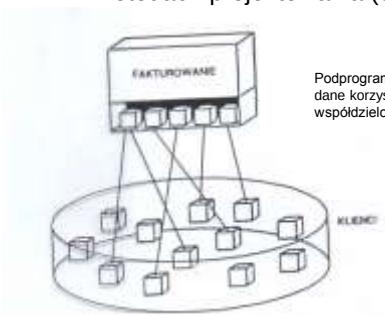
Kilka podprogramów używa tych samych danych

Sposoby dzielenia danych pomiędzy programy w klasycznych (nie obiektowych) metodach projektowania (2)



Dane lokalne w podprogramach

Sposoby dzielenia danych pomiędzy programy w klasycznych (nie obiektowych) metodach projektowania (3)

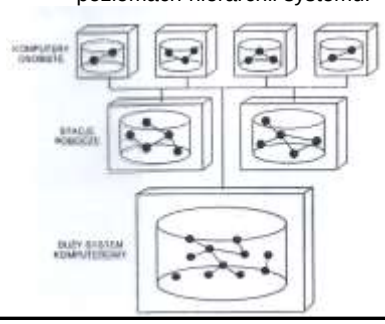


Podprogram mający własne dane korzysta jednak z danych współdzielonych w pliku

Sposoby dzielenia danych pomiędzy programy w klasycznych (nie obiektowych) metodach projektowania (4)

Podprogramy mające własne lokalne dane korzystają ze wspólnej bazy danych

Przy metodzie obiektowej cały system można budować zestawiając powtarzalne obiekty i klasy w stosowych konfiguracjach na poszczególnych poziomach hierarchii systemu.



Klasa a obiekt

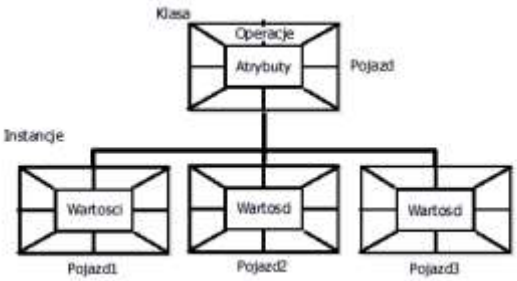
Operacje i **atrybuty** są definiowane jednorazowo (w klasie).

O obiektach, które należą do danej klasy, mówi się, że są **instancjami** tej klasy.

Instancje te zawierają określone własne (czasem nawet określane jako "prywatne") **wartości** atrybutów klasy.

Współdzielą one natomiast **operacje** klasy. Zachowanie tych instancji jest więc jednolite.

Klasa i jej instancje



Enkapsulacja

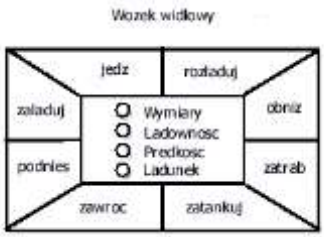
Enkapsulacja jest techniką, w której **dane** są przechowywane w obiektach razem z **operacjami**, jakie można na nich wykonać.

W dodatku dane są zazwyczaj chronione wewnątrz "kapsuły" utworzonej z operacji, co oznacza, że dowolny obiekt zewnętrzny może wywołać działanie określonej operacji, natomiast nie może bezpośrednio zmienić (ani nawet odczytać) żadnej wewnętrznej danej.

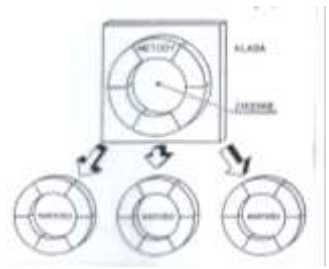
Jedynym sposobem dotarcia do danych ukrytych wewnątrz obiektu jest **użycie operacji** należącej do powłoki kapsuły, która na żądanie wykona stosowną operację na danych.

Eliminuje to **ryzyko** niepoprawnego użycia danych obiektu, ponieważ operują na nich zawsze wyłącznie "autoryzowane" własne operacje tegoż obiektu.

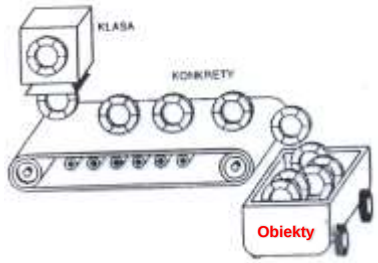
Przykład obiektu



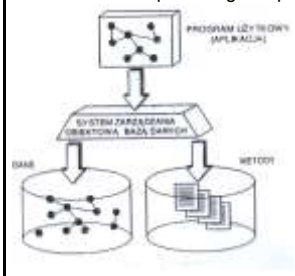
Na podstawie abstrakcyjnej definicji klasy można wygenerować konkretne **obiekty**



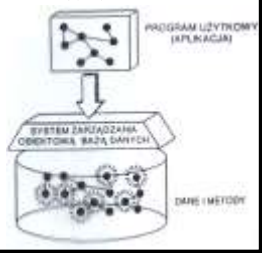
Mając dobrą definicję klasy można wygenerować z niej dowolnie dużo obiektów



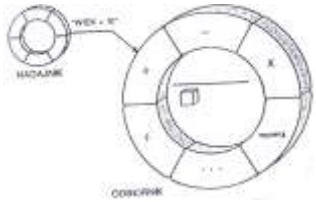
W tradycyjnym podejściu głównym „aktorem" działającym w systemie jest program zarządzający, który aktywizuje poszczególne programy i dane



W podejściu obiektowym wszystkie obiekty mogą być aktywne równocześnie, a każdy z nich wykonuje swoją część zadania



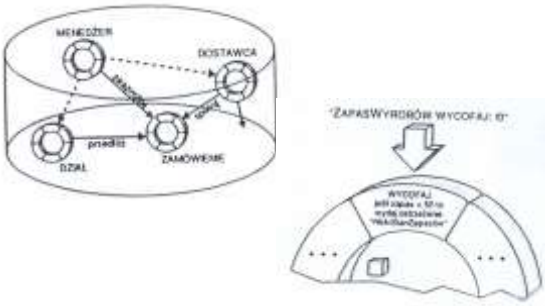
Obiekty są od siebie izolowane, ale porozumiewają się poprzez komunikaty



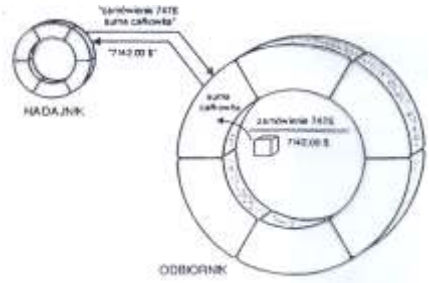
Komunikat może zlecać wykonanie jakiejś czynności na danych zawartych wewnątrz obiektu.

Komunikat odbiera wtedy jedna z metod powłoki i realizuje wymaganą czynność

Komunikaty wiążą kolekcję klas i obiektów w jeden system

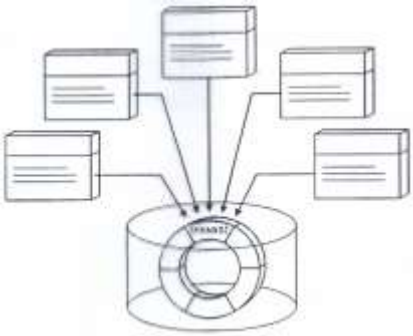


Za pomocą komunikatu można nakazać sprawdzenie wartości jakiejś danej.

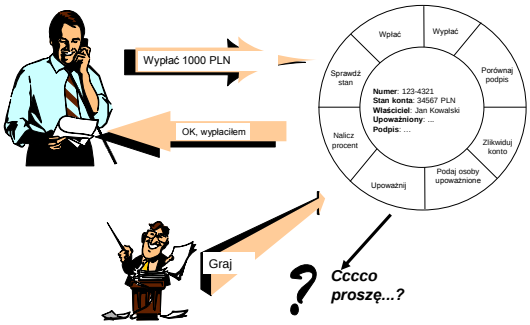


Wynik sprawdzenia jest także odsyłany w formie komunikatu.

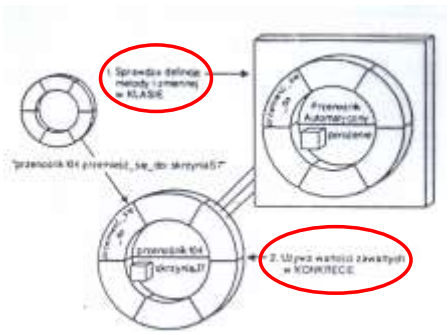
System metod i komunikatów używanych w podejściu obiektowym pozwala uniknąć wielokrotnego oprogramowywania tych samych czynności



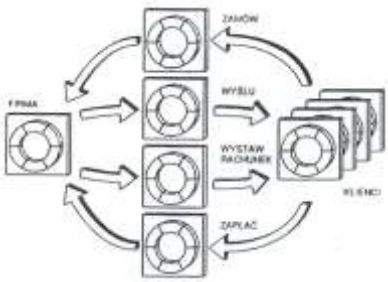
Komunikaty mogą być prawidłowe, lub nie!



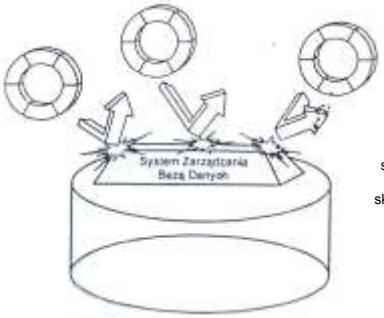
Czynność nakazana komunikatem może być wykonalna lub nie



Przykładowa współpraca między obiektami

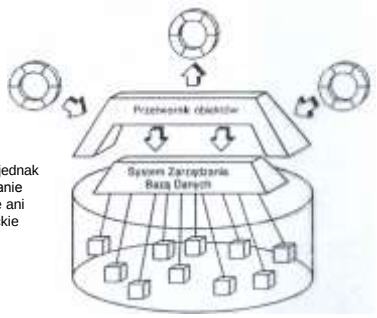


Podejście obiektowe musi być stosowane konsekwentnie



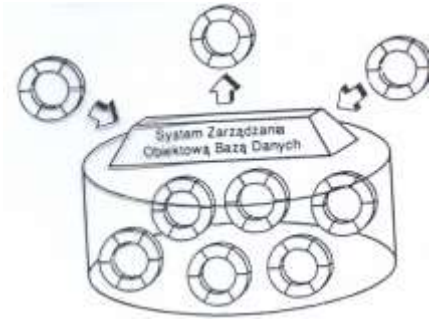
Zbudowane w sposób tradycyjny elementy systemu nie będą dobrze współpracować ze składnikami zbudowanymi w technice obiektowej

Możliwa jest budowa oprogramowania pośredniczącego pomiędzy elementami obiektowymi i tradycyjnymi

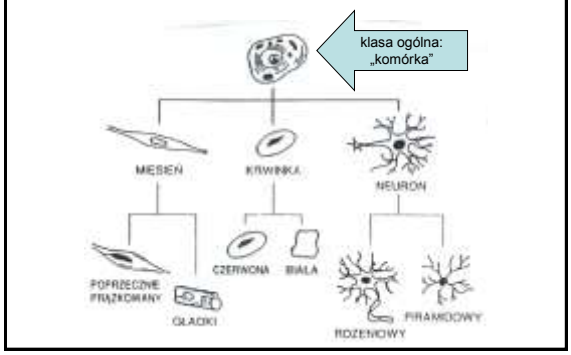


Nie jest to jednak rozwiązanie wygodne ani eleganckie

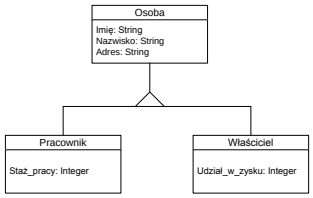
Poprawna droga polega na stosowaniu podejścia obiektowego we wszystkich składnikach projektowanego systemu



Przykład przejścia od obiektu ogólnego do obiektów szczegółowych

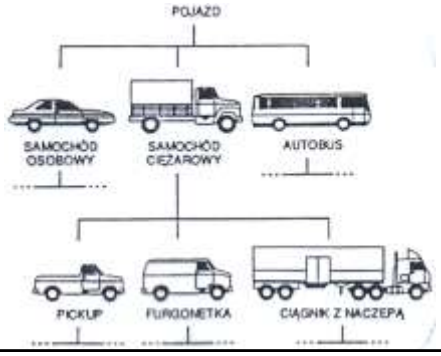


Klasy potomne przejmują atrybuty zdefiniowane w klasie macierzystej



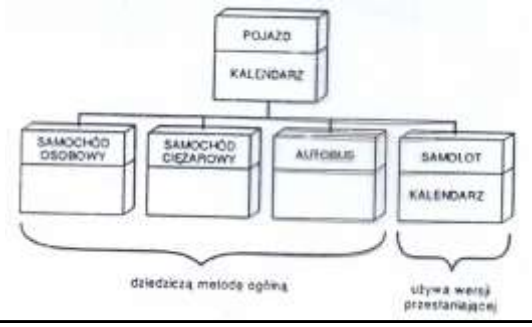
Ten mechanizm nosi nazwę: „dziedziczenie"

W ten sposób powstaje hierarchia klas i podklas, bardzo ułatwiająca operowanie złożonymi obiektami

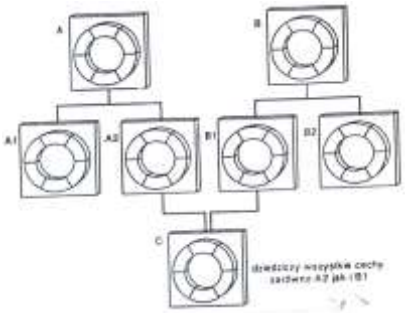


W każdej chwili można także dodać nową podklasę

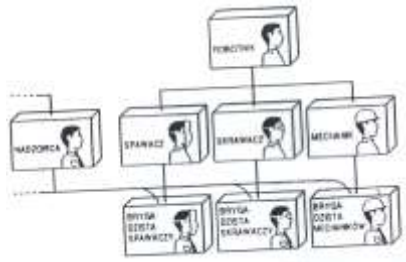
Atrybuty dotyczące obiektów wszystkich podklas wygodnie jest umieszczać w definicji klasy nadrzędnej



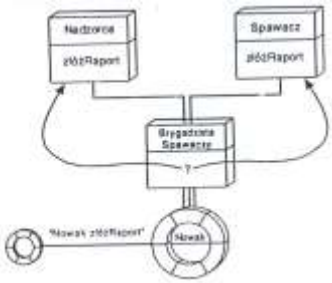
Niekiedy klasa może powstać jako „potomek” kilku klas – i z każdej klasy macierzystej może dziedziczyć pewne elementy



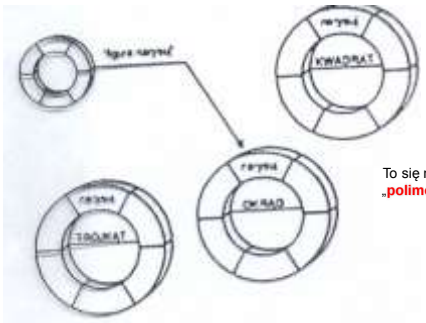
Przykład: „Brygadzista” jako klasa dziedzicząca po określonej podklasie klasy „Robotnik” oraz po klasie „Nadzorca”



Czasem może to powodować problemy związane z niejednoznacznością



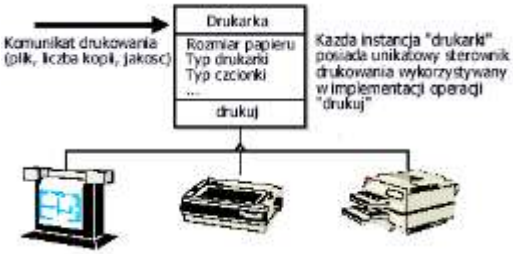
Taki sam komunikat kierowany do różnych obiektów może wywołać różne działania



To się nazywa „**polimorfizm**”

Polimorfizm

Polimorfizm jest techniką, w której ukrywa się szczegóły implementacji we wspólnym interfejsie. Polimorfizm upraszcza komunikację pomiędzy obiektami.



Analiza i projektowanie systemów informacyjnych

Projektowanie obiektowe

© UEK w Krakowie Ryszard Tadeusiewicz