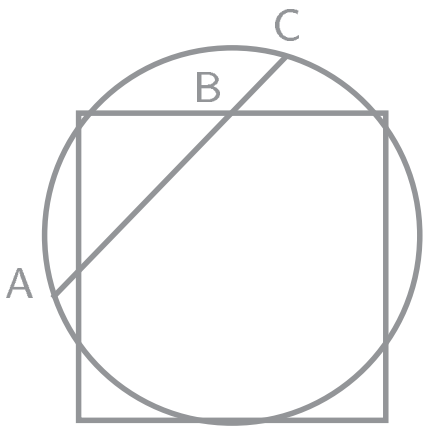
Inżynieria oprogramowania

Radosław Klimek

2015-23



http://home.agh.edu.pl/rklimek

1 Wprowadzenie do podejścia obiektowego

Wprowadzenie do podejścia obiektowego

Pablo PICASSO: *Panny z Avignon*

Inżynieria oprogramowania

O programowanie, i jego procesy wytwórcze, ma w wielu krajach duży wpływ na PKB, wyprzedzając inne dziedziny, takie np. jak: przemysł wytwórczy, sektor bankowo-ubezpieczeniowy, a nawet przemysł energetyczny! – szacuje się że w kosztach opracowania telefonii komórkowej koszty oprogramowania wynoszą ok. 75%.

Złożoność oprogramowania

O wzroście złożoności oprogramowania, wzrastającym zapotrzebowaniu na nie i coraz większych wymaganiach świadczą następujące dane:

- w USA jedynie 2% systemów jest używanych w wersji dostarczonej pierwotnie, 29% nie zostało nigdy dostarczonych do klienta, mimo że zostały zapłacone, a 47% nie jest nigdy wykorzystywane, mimo że zostały dostarczone, z powodu niedostosowania do potrzeb;
- w 1990 r. ważne programy bywały dostarczane z jednorocznym opóźnieniem i tylko 1% dużych projektów programistycznych zostało ukończonych na czas i w granicach projektowanego budżetu, informatyka jest bez wątpienia jedną z ostatnich gałęzi w której można aktualnie obserwować odchylenia większe niż 50% pomiędzy terminami i/lub kosztami planowania na początku projektu i stwierdzonymi po jego zakończeniu;
- udział kosztów obsługi w 1995 r. był szacowany na ponad 70% obrotu dla firm pracujących w dziale informatyki, a 22% tych kosztów jest związane z poprawianiem oprogramowania.

Podejście strukturalne a obiektowe

Podejście strukturalne
Rozpatrywanie zbioru procedur i funkcji działających na pewnych strukturach danych – może to okazać się niewystarczające z punktu widzenia rozpatrywanego i złożonego świata rzeczywistego.
Podejście obiektowe
.....

Podejście strukturalne a obiektowe (cd.)

Podejście strukturalne
.....
Podejście obiektowe
W podejściu tym istotna jest spójność modeli opisujących oprogramowanie, uzyskiwana poprzez zdefiniowanie i zamknięcie w pojedynczej jednostce zarówno danych (struktury) jak i operacji z nimi związanych – tą jedną jednostką jest właśnie **obiekt**, pomyślany jako podstawowy i elementarny komponent.

Podejście strukturalne a obiektowe – podsumowanie

- Rozdzielenie pojęcia struktur danych oraz operacji (funkcji/procedur) powodowało osobny ich rozwój, co mogło grozić utratą spójności.
- Podejście obiektowe burzliwy rozwój odnotowało w drugiej połowie lat osiemdziesiątych oraz w latach dziewięćdziesiątych,
- Zalety metod obiektowych nie przekreślają podejścia strukturalnego, które jednak może być z powodzeniem stosowane w systemach mniejszych i nie magających dużych wymagań.
- Jednak w przypadku systemów złożonych stosowanie technik obiektowych jest niejako koniecznością, jako ułatwiające zarządzanie skomplikowanymi danymi.

Percepcja i klasyfikowanie

Człowiek ma naturalne skłonności do klasyfikowania.

Trzy podstawowe metody percepcji i klasyfikacji mające wpływ na myślenie człowieka:

Percepcja i klasyfikowanie

Człowiek ma naturalne skłonności do klasyfikowania.

Trzy podstawowe metody percepcji i klasyfikacji mające wpływ na myślenie człowieka:

1. różnicowanie pomiędzy poszczególne obiekty i ich atrybuty – przykładowo różnicowanie pomiędzy drzewem i jego rozmiarami w relacji przestrzennej do innych oiektów;

Percepcja i klasyfikowanie

Człowiek ma naturalne skłonności do klasyfikowania.

Trzy podstawowe metody percepcji i klasyfikacji mające wpływ na myślenie człowieka:

1. różnicowanie pomiędzy poszczególne obiekty i ich atrybuty – przykładowo różnicowanie pomiędzy drzewem i jego rozmiarami w relacji przestrzennej do innych oiektów;
2. rozróżnienie pomiędzy całymi obiektami a ich częściami składowymi – przykładowo drzewo składające się z gałęzi czy pnia, które to części składają się na całe drzewo;

Percepcja i klasyfikowanie

Człowiek ma naturalne skłonności do klasyfikowania.

Trzy podstawowe metody percepcji i klasyfikacji mające wpływ na myślenie człowieka:

1. różnicowanie pomiędzy poszczególne obiekty i ich atrybuty – przykładowo różnicowanie pomiędzy drzewem i jego rozmiarami w relacji przestrzennej do innych oiektów;
2. rozróżnienie pomiędzy całymi obiektami a ich częściami składowymi – przykładowo drzewo składające się z gałęzi czy pnia, które to części składają się na całe drzewo;
3. konstruowanie klas obiektów i rozróżnianie ich – przykładowo klasa wszystkich drzew oraz klasa wszystkich kamieni.

Tu już jest używane pojęcie obiektu, ale jest ono używane bardzo ogólnie, w zasadzie można by mówić np. o elementach.

Zarządzanie złożonością systemów informatycznych

Istnieją ważne zasady zarządzania złożonością współczesnych i coraz bardziej skomplikowanych systemów informatycznych:

- **abstrakcja** – ignorowanie od nieistotnych aspektów z punktu widzenia bieżącego celu. Istnieje abstrakcja proceduralna i abstrakcja danych,
- **hermetyzacja**, lub **enkapsulacja** – ukrywanie informacji w pewnym miejscu, z jednoczesnym zamknięciem pojedynczej decyzji projektowej w tym miejscu. Abstrakcja może być uznana jako jedna z form hermetyzacji,
- **dziedzicznie** – pewne podobieństwo wynikające z wykorzystania niektórych cech jednych elementów przez elementy inne, także, inaczej, specjalizacja-generalizacja,
- **skojarzenie**, lub **asocjowanie** – wskazywanie związków pomiędzy pewnymi elementami, np. samochód i właściciel,

Zarządzanie złożonością systemów informatycznych (cd.)

ale także:

- porozumiewanie się poprzez przekazywanie komunikatów – to zostawianie informacji przez pewne byty, w zazwyczaj ustalonych miejscach, do odbioru przez inne byty,
- rozpoznanie i ujednolicenie metod klasyfikacji – już wspominane,
- problem skali – to zasada całość-część, lub **agregacja/kompozycja**, to notacje i strategie prowadzące, w połączeniu ze skalą, poprzez większy model/do większego modelu,
- kategorie zachowań – klasyfikowanie dynamiki systemu: (1) poprzez związki przyczynowe, (2) poprzez podobieństwo ewolucji, (3) ze względu na podobieństwo funkcji.

Podejście obiektowe – pierwsze podsumowanie

Koncepcje podejścia obiektowego mają pozytywny wpływ na proces modelowania systemów. Wynika to faktu, że:

1. obiektowość jest pewną ideą, która ma przybliżyć świat informatyczny do świata rzeczywistego, poprzez:
    1. ogólne podwyższenie poziomu abstrakcji w procesie analizy, projektowania i programowania,
    2. dopasowanie technologii informatycznych inżynierii oprogramowania do oczekiwań analityków i projektantów – celem podejścia obiektowego jest uzyskanie jak najmniejszej luki pomiędzy myśleniem o rzeczywistości, a myśleniem o procesach zachodzących w systemie informatycznym;

Podejście obiektowe – pierwsze podsumowanie (cd.)

2. podejście obiektowe umożliwia przedstawienie systemu z zastosowaniem tego samego modelu, począwszy od fazy początkowej (analiza) aż do końcowej (zazwyczaj implementacja) – model obiektowy przez wszystkie fazy projektu jest co prawda rozwijany i podlega licznym modyfikacjom, ale też jest to ten sam model, tyle że coraz bardziej „dojrzały", co pozwala uniknąć pewnego rozejścia się struktur danych i operacji, jako niezamierzonego efektu ubocznego.

Podejście obiektowe – pierwsze podsumowanie (cd.)

2. podejście obiektowe umożliwia przedstawienie systemu z zastosowaniem tego samego modelu, począwszy od fazy początkowej (analiza) aż do końcowej (zazwyczaj implementacja) – model obiektowy przez wszystkie fazy projektu jest co prawda rozwijany i podlega licznym modyfikacjom, ale też jest to ten sam model, tyle że coraz bardziej „dojrzały", co pozwala uniknąć pewnego rozejścia się struktur danych i operacji, jako niezamierzonego efektu ubocznego.

Można także powiedzieć, że celem obiektowości jest niegubienie semantyki danych i przybliżenie jej do świata rzeczywistego.

Wpływ podejścia obiektowego na różne obszary informatyki

Podejście obiektowe oddziaływuje na wiele obszarów współczesnej informatyki:

- metody analizy i projektowania systemów informatycznych (np. takie pojęcia jak: obiekty, klasy, metody, dziedziczenie, hermetyzacja, polimorfizm);
- języki programowania (np. Smalltalk, C++, Eiffel, Ada95);
- bazy danych i składy obiektów trwałych (standardy: ODMG, Gemstone i inne),
- współdziałanie systemów heterogenicznych (np.: CORBA),
- inne (wizyjne środki programistyczne, biblioteki oprogramowania oraz inne).

Metodyki i metodologie

Metodyki jako zbiory zasad postępowania dla rozwiązywania problemów są ważne także w rozwoju inżynierii oprogramowania.

Definicja

Przez **metodykę** rozumiemy zestaw **pojęć**, **notacji**, **modeli**, **języków**, **technik i sposobów postępowania** służących do analizy dziedziny stanowiącej przedmiot projektowanego systemu oraz do projektowania pojęciowego, logicznego i/lub fizycznego. Z każdą metodyką jest związana **notacja**. ⌟

Wadą niektórych metodyk może być arbitralność, brak głębszego uzasadnienia dla systemów pojęć, zasad, technik i scenariuszy postępowania, oraz idealizacja sytuacji projektowych, która dość często (najczęściej) nie pasuje do realiów i uwarunkowań konkretnego problemu.

Metodyki i metodologie (cd.)

Metodyka ustala:

- fazy projektu,
- modele tworzone w każdej z faz,
- scenariusze postępowania w każdej z faz,
- reguły przechodzenia od fazy danej do następnej,
- notacje które należy używać,
- dokumentację powstającą w każdej z faz.

Metodyka dyscyplinuje przebieg procesu analizy i projektowania, pozwalając tym samym na w miarę obiektywne rozliczenie (także czasowe i finansowe) jego uczestników.

Metodyki obiektowe

Definicja

Metodyka obiektowa (ang. *object-oriented methodology*) to metodyka wykorzystująca pojęcia obiektowości dla celów modelowania pojęciowego oraz analizy i projektowania systemów informatycznych. Podstawowym składnikiem tych metodyk jest diagram obiektów, będący zwykle wariantem notacyjnym i pewnym rozszerzeniem diagramów encja-związek, a także szereg innych diagramów. ⌟

Metodyki obiektowe (cd.)

- Obserwowana była eksplozja metodyk i notacji obiektowych, których obecnie jest kilkadziesiąt, np.: BON, Catalysis, DOOS (Wirfs/Brock), EROOS, Express, Fusion, Goldberg/Rubin, MainstreamObjects, Martin/Odell, MOSES, Objectory (Jacobson), OMT (Rumbaugh), OOA/OOD (Coad/Yourdon), OODA (Booch), OSA, Sintropy, OOSA (Shlaer/Mellor), UML oraz inne.
- Zauważa się już jednak koncentrację i zredukowanie metodyk (UML, OPEN , BON i inne), a na czoło wybija się ostatnio notacja UML autorstwa znanych metodologów Jacobsona, Rumbaugha i Boocha.

Trochę historii podejścia obiektowego – Simula 67

Podejście obiektowe pojawiło się w późnych latach sześćdziesiątych przy okazji języka **Simula 67**, który jest uważany za prekursora (przodka) obiektowości w programowaniu. Sam język powstał w Norweskim Ośrodku Obliczeniowym (Norwegian Computing Center) w Oslo w 1967 r. Jego twórcami byli: Ole-Johan Dahl, Bjørn Myhrhauga i Kristen Nygaard.

H. Oktaba, W. Ratajczak: *Simula 67*. Wydawnictwa Naukowo-Techniczne 1980.

Język Simula był językiem uniwersalnym wysokiego poziomu z wbudowanymi mechanizmami do symulacji. Simula miała udogodnienia dla obliczeń numerycznych, działań na tekstach, działań na zbiorach i strukturach listowych, a także operacje we/wy.

Z punktu widzenia obiektowości, Simula zawierała klasy, podklasy, wirtualne funkcje i aktywne obiekty.

Trochę historii podejścia obiektowego – Smalltalk

Innym znanym przykładem języka obiektowego jest **Smalltalk**, opracowany w latach 1976-83 w Xerox Palo Alto Research Center w Kalifornii. Język wprowadza klasy, podklasy, wirtualne funkcje, przesyłanie komunikatów, meta-klasy. Wszystko jest tu obiektem, w szczególności liczby i klasy.

- Smalltalk był pierwszym konsekwentnie obiektowo zorientowanym językiem i drugim językiem z elementami obiektowymi po języku Simula,
- Smalltalk pracuje pod kontrolą maszyny wirtualnej – tzn. pod kontrolą abstrakcyjnego komputera,
- Maszyna wirtualna posiada tak zwany Garbage Collector, który samodzielnie i efektywnie rozpoznaje, które obiekty nie są już używane i odzyskuje zajmowaną przez nie pamięć.

Simula – schemat deklaracji klasy

Schemat deklaracji klasy w języku Simula:

gdzie:

A	–	nazwa klasy;
PA	–	lista parametrów formalnych;
WA	–	zbiór wartości;
SA	–	zbiór specyfikacji parametrów;
DA	–	zbiór deklaracji lokalnych klasy;
I	–	lista instrukcji.

Obiektowe języki programowania

Obiektowe języki programowania (ang. *object-oriented programming language*), to języki programowania wprowadzające pojęcia takie jak: obiekt, klasa, metoda, dziedziczenie, hermetyzacja i polimorfizm.

Przykładowo są nimi np.:

Smalltalk	C++
Eiffel	Java
Sather	CLOS
Ada95	OO-Cobol
Beta	Cecil
Dylan	Python
Self	Theta

i inne.

(Wyprzedzamy tu pewne pojęcia, np. obiekt, dziedziczenie i inne, ale będą one wprowadzone już wkrótce.)

W kierunku języków obiektowych

Pierwsza generacja – **poddprogramy** – cała struktura danych widoczna dla wszystkich podprogramów, np. FORTRAN, COBOL;

Druga generacja – **proceduralność** – podział na procedury, przekazywanie parametrów, zagnieżdżanie, zakresy działania zmiennych, strukturalne metody projektowania, np. Pascal, Algol;

Trzecia generacja – **modułowość** – grupowanie danych i kodu mających związek logiczny, możliwość pracy w większych zespołach programistycznych;

Czwarta generacja – **obiektowość** – enkapsulacja, hermetyzacja, i inne pojęcia i koncepcje, języki np. C++, SmallTalk, Eiffel i inne.

(Por. rozdz. 1.1 i rys. w: M. Kliszewski, *Inżynieria oprogramowania obiektowego*.)

Zalety podejścia obiektowego

Podejście obiektowe jest powszechnie uznawane i stosowane, m.in. w wyniku następujących zalet:

- **spójność modeli systemu** – powiązanie fazy analizy i projektowania systemu;
- **łatwiejsza abstrakcja elementów dziedziny** – mechanizmy abstrakcji są niejako wbudowane w podejście obiektowe;
- **stabilność względem wprowadzanych zmian** – wprowadzenie zmian jest łatwiejsze i bezpieczniejsze;
- **możliwość wielokrotnego użycia komponentów** – poprzez dziedziczenie i uściślanie możliwość łatwego rozszerzania i redefiniowania komponentów;

Zalety podejścia obiektowego (cd.)

- **skalowalność** – względnie luźne powiązanie elementów w połączeniu z mechanizmami abstrakcji i enkapsulacji ułatwia skalowalność systemu;
- **niezawodność i bezpieczeństwo** – mechanizmy podjeścia obiektowego pozwalają na udostępnienie tylko tych interfejsów które są do tego przeznaczone;
- **wspieranie współbieżności** – w sposób naturalny wsparcie współbieżności poprzez definiowanie niezależnych komponentów.

Definicja obiektu

Obiekt jest jednym z najbardziej fundamentalnych pojęć we współczesnej informatyce (inżynierii oprogramowania). Pojęcie to i zagadnienia z nim związane są już dość dobrze rozpoznane, zarówno na gruncie teorii jak i praktyki.

Definicja

Obiektem (ang. *object*) nazywamy abstrakcyjne pojęcie lub część otaczającego świata, wyróżniające się ograniczonym zakresem oraz interfejsem, za pośrednictwem którego obiekt komunikuje się z jego otoczeniem. ⌟

Definicja obiektu (cd.)

Każdy obiekt jest scharakteryzowany poprzez:

Definicja obiektu (cd.)

Każdy obiekt jest scharakteryzowany poprzez:

- **atrybuty** – odnoszą się do danych zamkniętych we wnętrzu obiektu, opisują jakby własności obiektu,

Definicja obiektu (cd.)

Każdy obiekt jest scharakteryzowany poprzez:

- **atrybuty** – odnoszą się do danych zamkniętych we wnętrzu obiektu, opisują jakby własności obiektu,
- **sposób zachowania** (ang. *behaviour*) – to akcje (działania) podejmowane przez obiekt. Zachowanie dzieli się na:

Definicja obiektu (cd.)

Każdy obiekt jest scharakteryzowany poprzez:

- **atrybuty** – odnoszą się do danych zamkniętych we wnętrzu obiektu, opisują jakby własności obiektu,
- **sposób zachowania** (ang. *behaviour*) – to akcje (działania) podejmowane przez obiekt. Zachowanie dzieli się na:

  proste – dotyczy realizacji obsługi i uzależnione jest od wartości pewnej funkcji, a nie jest uzależnione od historii poprzednich żądań;

  dyskretne – to zachowanie uzależnione jest od pojęcia stanu, a mianowicie obiekt reaguje na pewne zdarzenie w zależności od stanu w którym się znajduje, oraz zachowanie

  ciągłe – oznacza to nieograniczony zbiór możliwych warunków, tak więc zachowanie takiego obiektu, w przeciwieństwie do zachowania dyskretnego, nie może być opisane przez automat, zachowanie obiektów jest uzależnione zarówno do historii jak i strumienia danych wejściowych.

Definicja obiektu (cd.cd.)

- **stan wewnętrzny** (ang. *state*) – wartości zmiennych składające się na strukturę obiektu, określone (przypisywane) w dyskretnych momentach czasowych.

Definicja obiektu (cd.cd.)

- **stan wewnętrzny** (ang. *state*) – wartości zmiennych składające się na strukturę obiektu, określone (przypisywane) w dyskretnych momentach czasowych.
- **zakres stosowania** (ang. *responsibility*) – uzależniony jest od kontekstu użycia obiektu – zachowanie powinno opisywać wszystkie możliwe zastosowania.

Definicja obiektu (cd.cd.)

- **stan wewnętrzny** (ang. *state*) – wartości zmiennych składające się na strukturę obiektu, określone (przypisywane) w dyskretnych momentach czasowych.
- **zakres stosowania** (ang. *responsibility*) – uzależniony jest od kontekstu użycia obiektu – zachowanie powinno opisywać wszystkie możliwe zastosowania.

Uwagi odnośnie stanów:

- stan nie jest wprost tożsamy z wartościami atrybutów,
- stany możemy ustalać badając możliwe wartości atrybutów i sprawdzając czy zachowanie obiektu jest różne dla tych wartości.

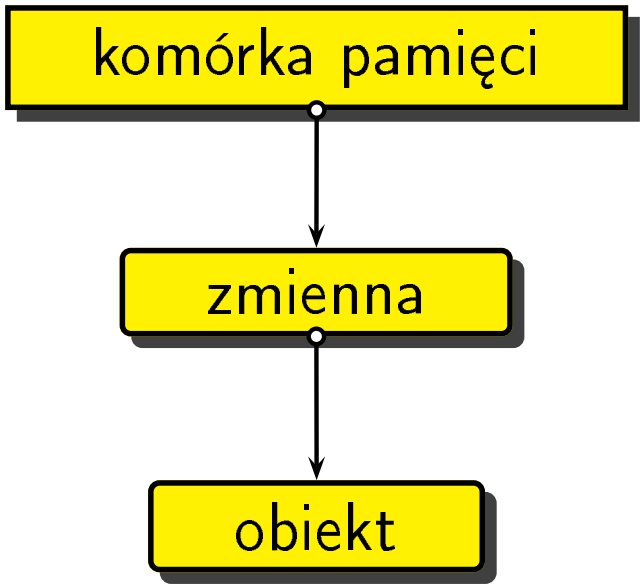
Zdarzenie to godna uwagi zmiana stanu.

Inna definicja obiektu

Definicja

Obiekt (ang. *object*) to konkretny lub abstrakcyjny byt (wyróżnialny w modelowanej rzeczywistości) posiadający nazwę, jednoznaczną identyfikację, określone granice, atrybuty i inne własności. Obiekt może być – i zazwyczaj jest – skojarzony z metodami lub operacjami, które na nim działają; z reguły, są one definiowane/przechowywane w ramach jego klasy oraz jej nadklas. ⌟

Rozwój pojęcia obiektu



komórka pamięci $\longrightarrow$ adres + zawartość

zmienna $\longrightarrow$ identyfikator + zawartość

obiekt $\longrightarrow$ identyfikator + zawartość + operacje

ale to nie koniec rozwoju...

Rozwój pojęcia obiektu (cd.)

- Komórka pamięci ma charakter atomiczny i przechowuje pewne wartości (liczby).
- Zmienna ma charakter pomocniczy w tym sensie, że służy do przechowywania obliczanych wartości jako zawartości zmiennych (komórek pamięci). Do komórej pamięci odwołujemy się poprzez identyfikator syntaktyczny.
- Obiekt jest niepodzielnym bytem, któremu zasadniczo nie odpowiada pojęcie wartości, jedynie (przez referencję) możemy odwoływać się do pewnych atrybutów obiektu.

Rozwój pojęcia obiektu (cd.cd.)

Obiekt w zasadniczej swojej intencji jest dokładnie tym samym co zmienna dla języków programowania. Występujące różnice dotyczą rozwoju historycznego, np. w C++ występują zarówno zmienne (przejęte z C) jak i obiekty. Druga różnica polega na tym, że w większości klasycznych języków programowania nie istnieją mechanizmy pozwalające na związanie metod ze zmiennymi (czyli mechanizmy klas lub abstrakcyjnych typów danych).

Analiza, projektowanie i programowanie obiektowe (OOA)

Analiza – studiowanie dziedziny problemu, prowadzące do wyspecyfikowania zachowania zewnętrznie obserwowalnego. Określenie co jest potrzebne aby osiągnąć pożądaną funkcjonalność, łącznie z charakterystkami operacyjnymi (np. niezawodność, wydajność, itd.). **Analiza obiektowa** OOA (ang. *Object-Oriented Analysis*) wymaga stosowania do powyższej fazy paradygmatów podejścia obiektowego.

.....

.....

Analiza jest kluczowa i wszelkie błędy popełnione w tej fazie będą rzutować na dalszą cześć przedsięwzięcia.

Analiza, projektowanie i programowanie obiektowe (OOD)

.....

Projektowanie – przejęcie specyfikacji zewnętrznie obserwowalnej i określenie co należy zrobić, tj. jakie szczegóły są niezbędne, aby w sposób rzeczywisty osiągnąć wyspecyfikowaną funkcjonalność. **Projektowanie obiektowe** OOD (ang. *Object-Oriented Design*) związane jest ze stosowaniem do powyższej fazy paradygmatów podejścia obiektowego.

.....

Projektowanie obejmuje szereg ważnych decyzji:
określenie komponentów które są współbieżne, wybór mechanizmów synchronizacji, wybór sposobów komunikacji, wybór koncepcji zarządzania danymi, określenie priorytetów, określenie szczególnych warunków działania (inicjalizacja, zakończenie, przypadki awaryjne, sytuacje błędne), itd.

Analiza, projektowanie i programowanie obiektowe (OOP)

.....

.....

Programowanie – model stworzony w fazie projektowania podlega tłumaczeniu na odpowiedni język programowania z uwzględnieniem szeregu aspektów (np. środowisko wykonawcze, architektura sprzętowa i inne). **Programowanie obiektowe** OOP (ang. *Object-Oriented Programming*) związane jest ze stosowaniem do powyższej fazy paradygmatów podejścia obiektowego.

Programowanie: ostatnia faza uściślania modelu (systemu), w której model obiektowy jest tłumaczony na konkretny jeęzyk programowania. Jeżeli jest duża zgodność dostępnych konstrukcji językowych z konstrukacjami wyprowadzonymi dla systemu, to takie tłumacznie jest względnie łatwe.

Obiekt, klasa, instancja

Definicja

Obiekt (ang. *object*) to abstrakcja czegoś w dziedzinie problemu (identyfikacja, stan, zachowanie), ukazuje zdolności systemu do przechowywania o tym informacji oraz wykonywania na tym operacji. ⌟

Obiekt, klasa, instancja

Definicja

Obiekt (ang. *object*) to abstrakcja czegoś w dziedzinie problemu (identyfikacja, stan, zachowanie), ukazuje zdolności systemu do przechowywania o tym informacji oraz wykonywania na tym operacji. ⌟

Definicja

Klasa (ang. *class*) to abstrakcja grupy obiektów o podobnej charakterystyce (wspólne własności, wspólne zachowania), jednolity zbiór atrybutów oraz jednolity zbiór usług łącznie ze sposobem tworzenia (inicjowania) obiektu. Klasa grupuje elementy względem pewnej relacji równoważnościowej. Tak więc jeszcze inaczej można powiedzieć, że **klasa** jest zbiorem obiektów tego samego typu, które mogą się różnić wartością atrybutu lub zachowaniem, lecz pełnią tę samą rolę w systemie. ⌟

Obiekt, klasa, instancja (cd.)

Definicja

Odwoływanie się do każdego z elementów grupy obiektów jest odwoływaniem się do **instancji** (ang. *instance*) odpowiedniego obiektu, tj. do jego konkretnego wystąpienia (egzemplarza). Elementy klasy nazywamy instancjami. ⌟

Obiekt, klasa, instancja (cd.)

Definicja

Odwoływanie się do każdego z elementów grupy obiektów jest odwoływaniem się do **instancji** (ang. *instance*) odpowiedniego obiektu, tj. do jego konkretnego wystąpienia (egzemplarza). Elementy klasy nazywamy instancjami. ⌟

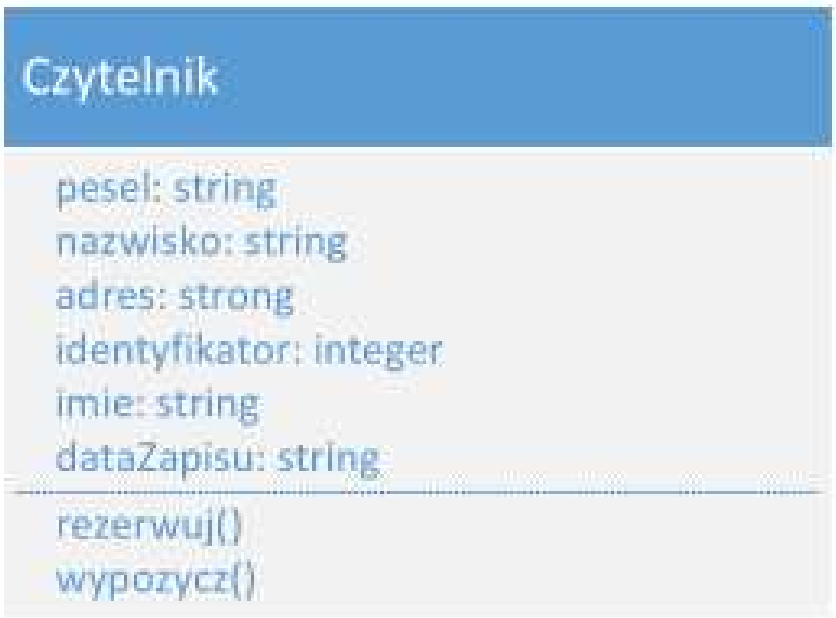
Istotnym pojęciem w analizie obiektowej jest pojęcie klasy. Jest ono analogiczne jak w przypadku teorii zbiorów.

Instancjami są obiekty należące do tej samej klasy.

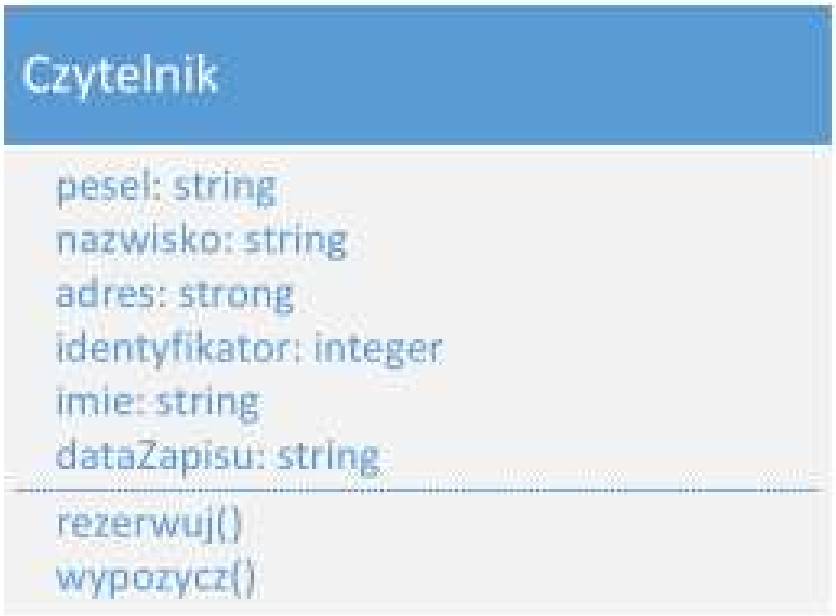
Należy jeszcze zauważyć, że klasa obiektu jest pojęciem węższym niż typ obiektu, który określa obiekty o tej samej strukturze.

W pewnych systemach pojęcie klasy nie jest zawsze wprowadzane, a czasem operuje się wyłącznie pojęciem typu i podtypu, np. Ada95 lub SDL.

Pierwsze diagramy klas



Pierwsze diagramy klas



Podstawowe operacje na obiektach

Wyróżnia się pięć podstawowych operacji działających na obiektach, składających się na jego charakterystykę zachowania:

1. **konstruktor** (ang. *constructor*) – utworzenie obiektu wraz z zainicjowaniem jego zmiennych (zainicjowanie stanu początkowego);
2. **modyfikator** (ang. *modifier*) – zmiana stanu obiektu;
3. **selektor** (ang. *selector*) – udostępnienie informacji o stanie obiektu, bez dokonywania zmiany w obiekcie;
4. **iterator** (ang. *iterator*) – udostępnienie informacji o obiekcie poprzez dostęp do całej jego struktury, np. w wyniku iteracyjnego przeglądania struktury – przeglądanie informacji obiektu w dobrze (i ściśle) określonym porządku;
5. **destruktor** (ang. *destructor*) – niszczenie (usunięcie) obiektu.

Typy obiektów a operacje

Ze względu na rodzaje operacji podejmowanych przez obiekty i związki operacyjne pomiędzy nimi, wyróżnia się trzy rodzaje obiektów:

1. **aktorzy** (ang. *actors*) – obiekty które dokonują operacji na innych obiektach, lecz same nigdy nie podlegają operacjom ze strony innych obiektów;
2. **serwery** (ang. *serwers*) – obiekty podlegające operacjom ze strony innych obiektów i nie wykonujące operacji na innych obiektach;
3. **agenci** (ang. *agents*) – obiekty zarówno operujące na innych obiektach, jak i same podlegające działaniu ze strony obiektów innych.

W innej nomenklaturze aktorzy to inaczej klienci.