Programowanie obiektowe - 1

Mariusz.Masewicz@cs.put.poznan.pl

Programowanie obiektowe

- Programowanie obiektowe (ang. object-oriented programming) to metodologia tworzenia programów komputerowych, która definiuje programy za pomocą "obiektów" - elementów łączących stan (czyli dane) i zachowanie (czyli procedury, tu: metody).
- Obiektowy program komputerowy wyrażony jest jako zbiór takich obiektów, komunikujących się pomiędzy sobą w celu wykonywania zadań. Podejście to różni się od tradycyjnego programowania proceduralnego, gdzie dane i procedury nie są ze sobą bezpośrednio związane. Programowanie obiektowe ma ułatwić pisanie, konserwację i wielokrotne użycie programów lub ich fragmentów.

Podejście obiektowe

- Wszystko jest obiektem
- Program jest zbiorem obiektów które wysyłają sobie komunikaty
- Każdy obiekt posiada pamięć na którą składają się inne obiekty
- Każdy obiekt posiada swój typ
- Wszystkie obiekty tego samego typu mogą otrzymywać te same komunikaty

Cechy obiektu

- stan – dane wewnętrzne
- zachowanie – zestaw metod do wykonania
- identyfikacja – każdy obiekt można w sposób jednoznaczny odróżnić od innych obiektów

Obiekty i klasy

- Klasa to złożony typ danych, składający się z pól, przechowujących dane, oraz posiadający metody, wykonujące zaprogramowane czynności.
- Obiekt może reprezentować cokolwiek. Każdy obiekt należy do pewnej klasy. Definicja klasy zawiera pola, z których składa się ów obiekt, oraz metody, którymi dysponuje.
- **Klasa** jest więc **wzorcem** na podstawie którego powołujemy do życia **obiekty**

Pola i metody

- obiekty zawierają pola, czyli zmienne. Ich rolą jest przechowywanie pewnych informacji o obiekcie - jego charakterystyki
- obiekt może wykonywać na sobie pewne działania, a więc uruchamiać zaprogramowane funkcje; nazywamy je metodami albo funkcjami składowymi. Czynią one obiekt tworem aktywnym - nie jest on jedynie pojemnikiem na dane, lecz może samodzielnie nimi manipulować

Metoda

- Metoda - w programowaniu obiektowym jest to funkcja składowa klasy, której zadaniem jest działanie na rzecz określonych elementów danej klasy lub klas z nią spokrewnionych (zob. też dziedziczenie).
- Metody wiąże się z klasami głównie po to, aby nie zaśmiecać kodu źródłowego i samego programu nadmierną ilością funkcji globalnych, które i tak nie zostaną użyte w celu innym, niż na rzecz konkretnej klasy.
- Inną ich zaletą jest to, że metoda wewnętrzna danej klasy ma dostęp do wszystkich składników tej klasy (także prywatnych i chronionych), bez konieczności deklarowania zaprzyjaźnienia.

Konstruktory i destruktory

- są to specjalne metody wywoływane podczas tworzenia obiektu oraz jego usuwania (destruktory nie występują w Javie).
- konstruktor nie może posiadać definicji zwracanego typu
- konstruktor bez żadnych argumentów jest konstruktorem domyślnym
- nazwa destruktora musi być poprzedzona znakiem tyldy (C++)
- destruktor nie posiada definicji typu zwracanego oraz nie przyjmuje żadnych argumentów (C++)
- nazwy konstruktora i destruktora muszą być zgodne z nazwą klasy
- konstruktor i destruktor muszą być zadeklarowane w części publicznej klasy
- domyślny konstruktor może być przeciążany

Paradygmaty programowania obiektowego

- abstrakcja
- enkapsulacja (hermetyzacja)
- dziedziczenie
- polimorfizm

Abstrakcja

- Umiejętność wyodrębnienia cech istotnych dla danego problemu
- Każdy obiekt w systemie służy jako model abstrakcyjnego "wykonawcy", który może wykonywać pracę, opisywać i zmieniać swój stan, oraz komunikować się z innymi obiektami w systemie, bez ujawniania, w jaki sposób zaimplementowano dane cechy. Procesy, funkcje lub metody mogą być również abstrahowane, a kiedy tak się dzieje, konieczne są rozmaite techniki rozszerzania abstrakcji.
  - identyfikacja obiektów i operacji na nich
  - klasyfikacja podobnych obiektów za pomocą klas
  - zdefiniowanie atrybutów i dopuszczalnych operacji dla poszczególnych klas

Klasa abstrakcyjna

- nie posiada obiektów
- używana do definicji interfejsu dla klas z niej wywiedzionych

Metoda abstrakcyjna

- nie została jeszcze zaimplementowana (tylko zadeklarowana aby zaimplementować ją w klasach potomnych)
- deklarowana jedynie wewnątrz klasy abstrakcyjnej (klasa posiadająca przynajmniej jedną metodę abstrakcyjną jest klasą abstrakcyjną)
- podczas dziedziczenia po klasie abstrakcyjnej:
  - metoda abstrakcyjna musi zostać zaimplementowana
  - klasa potomna jest abstrakcyjna jeżeli nie zaimplementowano w niej metod abstakcyjnych

Enkapsulacja

- Czyli ukrywanie implementacji, hermetyzacja. Zapewnia, że obiekt nie może zmieniać stanu wewnętrznego innych obiektów w nieoczekiwany sposób. Tylko wewnętrzne metody obiektu są uprawnione do zmiany jego stanu. Każdy typ obiektu prezentuje innym obiektom swój "interfejs", który określa dopuszczalne metody współpracy. Pewne języki osłabiają to założenie, dopuszczając pewien poziom bezpośredniego (kontrolowanego) dostępu do "wnętrzności" obiektu. Ograniczają w ten sposób poziom abstrakcji.
  - ukrycie wewnętrznej implementacji klas
  - dostęp do obiektów poprzez wyspecyfikowany interfejs
  - interakcja z obiektami przez wysyłanie komunikatów (wywołania publicznych metod)
  - zalety: ochrona i elastyczność kodu

Poziomy kontroli dostępu

- **public** – dostępne dla każdego
- **private** – tylko twórca klasy
- **protected** – dostęp mają tylko dana klasa i klasy dziedziczące

Dziedziczenie

- Porządkuje i wspomaga polimorfizm i enkapsulację dzięki umożliwieniu definiowania i tworzenia specjalizowanych obiektów na podstawie bardziej ogólnych. Dla obiektów specjalizowanych nie trzeba redefiniować całej funkcjonalności, lecz tylko tę, której nie ma obiekt ogólniejszy. W typowym przypadku powstają grupy obiektów zwane klasami, oraz grupy klas zwane drzewami. Odzwierciedlają one wspólne cechy obiektów.
  - technika wykorzystania istniejących fragmentów kodu polega na tworzeniu nowych klas na bazie już istniejących
  - cechy wspólne dla wszystkich podklas definiowane są w nadklasie
  - podklasa może
    - korzystać z cech nadklasy
    - nadpisywać zachowanie nadklasy
    - dodawać nowe atrybuty i zachowania

Polimorfizm

- Referencje i kolekcje obiektów mogą dotyczyć obiektów różnego typu, a wywołanie metody dla referencji spowoduje zachowanie odpowiednie dla pełnego typu obiektu wywoływanego. Niektóre języki udostępniają bardziej statyczne (w trakcie kompilacji) rozwiązania polimorfizmu - na przykład szablony i przeciążanie operatorów w C++
  - pozwala w jednolity sposób traktować obiekty klas z hierarchii dziedziczenia przy zachowaniu ich charakterystycznego zachowania
  - od strony technicznej sprowadza się do tzw. późnego wiązania metod przy ich wywołaniu (wybór metody na podstawie typu obiektu)
  - w Javie wszystkie metody zachowują się jak metody wirtualne w C++

Wczesne i późne wiązania

- Wczesne wiązanie: kompilator generuje wywołanie funkcji, program łączący zamienia nazwę na konkretny adres.
- Późne wiązanie: kompilator upewnia się ze metoda istnieje, jednak kod jest dopiero ustalony w czasie wykonania.

Różne podejścia do realizacji paradygmatu

- Niektóre języki wprowadzają modyfikacje do założeń obiektowości, na przykład:
  - w niektórych językach każda klasa musi mieć nadklasę
  - w niektórych językach nadklas może być więcej niż jedna (dziedziczenie wielobazowe)

Elementarna charakterystyka cech popularnych języków p.o.

- Powyższa charakterystyka dopuszcza bardzo dużą różnorodność - i w rzeczywistości, o ile systemy programowania strukturalnego czy funkcyjnego były do siebie stosunkowo podobne, o tyle **systemy obiektowe różnią** się tak bardzo że nie wiadomo co tak naprawdę znaczy nazwa obiektowe.

Dziedziczenie wielokrotne

- jest: C++, Python, Incr Tcl
- tylko interfejsy: Object Pascal, Java, C#
- dziedziczenie wielokrotne tylko metod: Ruby

Klasa jest obiektem

- tak: Ruby, Python
- nie: C++, C#, Ocaml

Wszystkie obiekty wywodzą się z jednego korzenia i muszą mieć nadklasę

- tak: Java, C#, Ruby, Object Pascal, Incr Tcl
- nie: C++, Ocaml, Python

Obiekt można pytać do której podklasy należy:

- tak: Ruby, C#, C++ (z RTTI), Java (RTTI lub mechanizm refleksji), Python, Object Pascal, Incr Tcl
- nie: Ocaml

Mechanizm szablonów do automatycznego generowania klas

- tak: Ruby, C++, Ocaml, Java (od wersji 1,5)
- nie: Java (do wersji 1.5)

Literatura

- http://www.google.pl/
- http://java.sun.com/
- Podręczniki do Javy i C++ - http://www.javaic.com/eckel.html
- C++ bez cholesterolu http://www.intercon.pl/~sektor/cbx/
- Megatutorial "Od zera do gier kodera" http://avocado.risp.pl/files/texts/od0dogk/