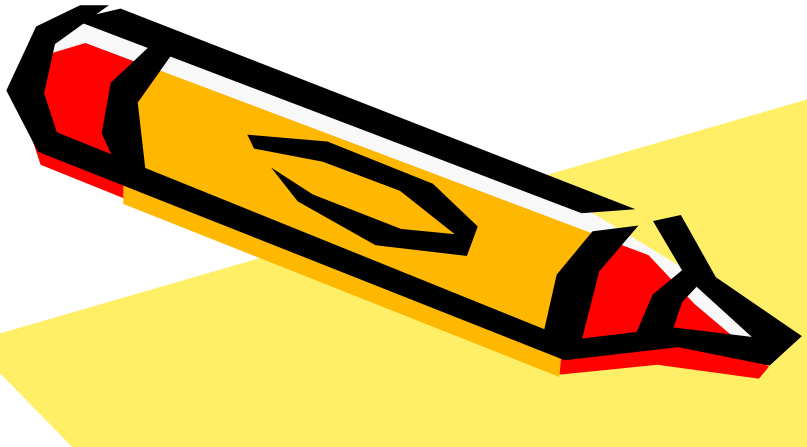


Język programowania

Andrzej Bobyk
http://www.alfabeta.lublin.pl/JP/

Literatura

- K. Reisdorph: *Delphi 6 dla każdego*. Helion, Gliwice 2001
- A. Grażyński, Z. Zarzycki: *Delphi 7 dla każdego*. Helion, Gliwice 2003
- S. Teixeira, X. Pacheco: *Delphi 6. Vademecum profesjonalisty (t. 1+2)*. Helion, Gliwice 2002
- A. Daniluk: *ABC Delphi 7*. Helion, Gliwice 2003

Literatura (c.d.)

- A. Marciniak: *Delphi 5 Professional. Object Pascal.* Nakom, Poznań 2000
- A. Boduch: *Delphi 8 .NET. Kompendium programisty.* Helion, Gliwice 2004
- M. Cantu: *Delphi 6. Praktyka programowania (t. 1+2).* Mikom, Warszawa 2002
- M. Cantu: *Delphi 7.* Mikom, Warszawa 2004

Paradygmaty programowania

- programowanie proceduralne
- programowanie strukturalne
- programowanie obiektowe
- programowanie zdarzeniowe
- programowanie wizualne
- programowanie komponentowe

Tradycyjne paradygmaty programowania

- Programowanie proceduralne – dzielenie kodu na procedury, czyli fragmenty wykonujące ściśle określone operacje
- Programowanie strukturalne – hierarchiczne dzielenie kodu na moduły, które komunikują się jedynie poprzez dobrze określone interfejsy (rozszerzenie koncepcji programowania proceduralnego)

Programowanie obiektowe (OOP)

- metodologia tworzenia programów komputerowych za pomocą obiektów – elementów łączących stan (dane, tzw. pola) i zachowanie (procedury, tzw. metody)
- program komputerowy – zbiór obiektów, komunikujących się pomiędzy sobą w celu wykonywania zadań
- dane i procedury są ze sobą bezpośrednio związane – ułatwia to pisanie, konserwację i wielokrotne użycie programów lub ich fragmentów

Założenia paradygmatu obiektowego

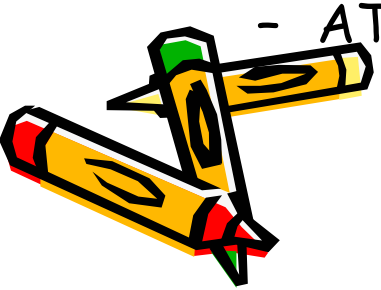
- Abstrakcja – pewne klasy są wzorem dla innych klas, które się z niej wywodzą, ale same nie są wykorzystywane do tworzenia obiektów.
- Enkapsulacja (hermetyzacja) – grupowanie danych i metod w klasy w połączeniu z ukrywaniem szczegółów implementacji
- Polimorfizm – sprawdzanie rzeczywistego typu obiektu w trakcie działania programu
- Dziedziczenie – tworzenie nowych klas na bazie klas już istniejących

Programowanie zdarzeniowe

- program jest cały czas bombardowany zdarzeniami (ang. *events*), na które musi odpowiedzieć
- przepływ kontroli w programie jest całkowicie niemożliwy do przewidzenia z góry
- dominujący typ programowania GUI – zdarzenia to naciśnięcia myszy, klawiszy, żądania odświeżenia przez system okienkowy, różne zdarzenia sieciowe itp.

Programowanie wizualne

- metoda tworzenia programu za pomocą narzędzi umożliwiających wybór standardowych elementów z menu i automatyczną generację kodu źródłowego
- systemy programowania wizualnego:
  - Borland Delphi
  - Borland C++ Builder
  - Microsoft Visual Studio (.NET)
- biblioteki klas:
  - OWL (Borland)
  - MFC (Microsoft)
  - ATL (Microsoft)



Programowanie komponentowe

- stosowanie prefabrykatów programowych, nadających się do wielokrotnego i różnorodnego użytku, zwanych komponentami
- komponenty tworzy się za pomocą metod obiektowych – służą one do budowy wielkich systemów oprogramowania
- technologie: JavaBeans, DCOM, ActiveX, CORBA, IIOP

Klasy

- Definicje obiektów składających się z pól i metod (funkcji i procedur) współdziałających w celu wykonania określonego zadania lub funkcji
- Elementy klas:
  - Kategorie widoczności elementów klasy
  - Konstruktory
  - Destruktory
  - Pola
  - Metody (procedury i funkcje)
  - Ukryty, specjalny wskaźnik o nazwie *Self*

Poziomy dostępu do składników klasy

- Klasy posiadają 4 poziomy dostępu do swoich składników:
  - *Private*
  - *Public*
  - *Protected*
  - *Published*

Konstruktory i destruktory

- Konstruktor jest specjalną metodą używaną przy tworzeniu obiektu danej klasy
- Destruktor jest specjalną metodą wywoływaną automatycznie tuż przed zakończeniem życia obiektu

Pola i metody

- Pola to zmienne umieszczone w klasie przy jej deklarowaniu (podobnie jak w typach rekordowych)
- Zachowują się jak zmienne lokalne dla obiektu
- Można się do nich odwoływać z poziomu wszystkich metod danej klasy
- Ich widzialność na zewnątrz klasy zależy od tego, czy umieszczono je w sekcji *private*, *public* czy *protected*

Metody klasy

- Metody to funkcje i procedury będące składnikami klasy
- Są lokalne w danej klasie i nie istnieją poza nią
- Mogą być wywoływane z wewnątrz klasy lub poprzez obiekt danej klasy
- Mają dostęp do wszystkich pól: prywatnych, publicznych i chronionych
- Mogą być także prywatne, publiczne i chronione

Reguły hermetyzacji (1)

- Część prywatna (*private*) klasy zawiera szczegóły implementacyjne znane tylko twórcy klasy – pola i metody prywatne nie są widziane na zewnątrz klasy (nie można się do nich odwoływać z funkcji i procedur nie będących składnikami klasy)
- Część publiczna (*public*) klasy zawiera interfejs, za pomocą którego świat zewnętrzny komunikuje się z klasą – do pól i metod publicznych można odwoływać się swobodnie z każdego miejsca programu
- Pola i metody chronione (*protected*) dostępne są dla klas będących potomkami klasy podstawowej

Reguły hermetyzacji (2)

- Metody publiczne tworzą interfejs użytkownika – poprzez nie następuje interakcja klasy ze światem zewnętrznym
- Metody prywatne wykonują „czarną robotę" – bezpośrednie ich wywoływanie przez użytkownika nie jest konieczne, a czasem wręcz niepożądane
- Metody chronione nie mogą być wywoływane z zewnątrz, a tylko z klas potomnych danej klasy

Wskaźnik *Self*

- Wszystkie klasy (a tym samym – wszystkie obiekty) posiadają ukryte pole o nazwie *Self* – jest to wskaźnik do tego właśnie obiektu (obiekt zawiera wskaźnik na samego siebie)
- Pozwala metodom uzyskać informację, na którym obiekcie (egzemplarzu klasy) mają działać

Dziedziczenie

- Budowanie nowych klas na bazie już istniejących
- Klasa, z której dziedziczą nowe klasy, nazywa się klasą bazową, klasą macierzystą albo przodkiem, natomiast klasa, która dziedziczy cechy klasy bazowej, nazywa się klasą pochodną lub klasą potomną
- Klasa potomna dziedziczy od swojego przodka wszystkie jego pola i metody