Programowanie obiektowe

dr inż. Marek Wojciechowski
Marek.Wojciechowski@cs.put.poznan.pl
http://www.cs.put.poznan.pl/mwojciechowski/

Plan wykładu

- Wprowadzenie
  - Podstawowe pojęcia
  - Historia programowania obiektowego
  - UML – język do modelowania obiektowego
- Język C++
- Język Java
- Język C#

Literatura

- Bjarne Stroustrup, „Język C++”, Wydawnictwa Naukowo-Techniczne, 1994, 2000 i 2002
- Bruce Eckel, „Thinking in C++”, Free Electronic Book
- Bruce Eckel, „Thinking in Java”, Free Electronic Book
- Stephen C. Perry, „C# i .NET”, Helion, 2006
- G. Booch, J. Rumbaugh, I. Jacobson, „UML przewodnik użytkownika”, Wydawnictwa Naukowo-Techniczne, 2002
- Janusz Jabłonowski, Jacek Sroka (Uniwersytet Warszawski), „Programowanie obiektowe”, Materiały dydaktyczne do kształcenia na odległość
- Wikipedia, the free encyclopedia, http://www.wikipedia.org/
- Wiele innych książek i źródeł internetowych na temat języków C++, Java i C# oraz innych języków obiektowych

Programowanie obiektowe

- Object-oriented programming
- Najpopularniejszy obecnie styl (paradygmat) programowania
- Rozwinięcie koncepcji programowania strukturalnego
  - Podejście tradycyjne: program jako kolekcja funkcji (a wcześniej lista instrukcji)
  - Podejście obiektowe: program jako kolekcja współpracujących ze sobą obiektów
- Obiekty łączą dane i operacje na nich
- Zalety programowania obiektowego:
  - Ułatwia współpracę i podział zadań między programistów
  - Ułatwia pielęgnację i rozbudowę aplikacji
  - Ułatwia ponowne wykorzystywanie wcześniej napisanego kodu
  - Często umożliwia naturalne modelowanie rzeczywistości
  - Odpowiednie dla dużych projektów, popularne w inżynierii oprogramowania

Programowanie strukturalne a programowanie obiektowe

Podejście strukturalne (C++)

```
struct Punkt
{
  int x, y;
};

void narysuj (struct Punkt P)
{
  // ciało funkcji
}
```

Podejście obiektowe (C++)

```
class Punkt
{
  int x, y;
public:
  void narysuj ()
  {
    // ciało funkcji
  }
};
```

- Uwaga: C++ jest przykładem języka wspierającego zarówno styl proceduralny jak i obiektowy

Podstawowe pojęcia (1/5)

- **Klasa** – definiuje charakterystykę „czegoś"; wyznacza modularną strukturę programu zorientowanego obiektowo
  - Charakterystyka obejmuje atrybuty (właściwości) i zachowanie (metody)
  - Atrybuty i metody klasy są określane jako **składowe klasy**
  - Przykład: klasa Pies opisująca cechy i zachowanie wspólne dla wszystkich psów np. kolor sierści i umiejętność szczekania
- **Obiekt** – instancja (wystąpienie) klasy
  - Np. konkretny pies – Reksio o białym kolorze sierści
  - Zbiór wartości atrybutów obiektu w danej chwili jest określany mianem **stanu obiektu**

Podstawowe pojęcia (2/5)

- **Metoda** – operacja, która może być wykonana na obiekcie, reprezentująca jego funkcjonalność
  - Np. metoda szczekaj() klasy Pies, która może być wywołana na rzecz konkretnego obiektu klasy Pies, np. obiektu Reksio
- **Przekazywanie komunikatów** – proces polegający na przekazaniu danych z obiektu do obiektu lub zleceniu wywołania metody na rzecz obiektu

Podstawowe pojęcia (3/5)

- **Kapsułkowanie** – ukrycie szczegółów implementacji klasy przed kodem korzystającym z klasy
  - Składowe klasy dostępne z zewnątrz stanowią interfejs klasy
  - Składowe niedostępne z zewnątrz mogą być zmieniane bez wpływu na pozostały kod aplikacji
  - Realizowane poprzez kwalifikatory dostępu do składowych klasy
  - Podstawowe kwalifikatory dostępu do składowych:
    - **public** – dostęp publiczny
    - **protected** – dostępne w klasie definiowanej i klasach pochodnych
    - **private** – dostępne tylko w definiowanej klasie

Podstawowe pojęcia (4/5)

- **Dziedziczenie** – definiowanie **podklasy** (**klasy pochodnej**) jako **specjalizacji** klasy istniejącej (**klasy bazowej**, **nadklasy**)
  - Np. klasa Jamnik jako specjalizacja klasy Pies
  - Podklasa dziedziczy z klasy bazowej atrybuty i metody
  - Podklasa może posiadać dodatkowe atrybuty i metody oraz redefiniować metody odziedziczone
- **Dziedziczenie wielobazowe** – polega na definiowaniu klasy jako dziedziczącej bezpośrednio z więcej niż jednej klasy
  - Dostępne nie we wszystkich językach
  - Prowadzi do skomplikowanych programów

Podstawowe pojęcia (5/5)

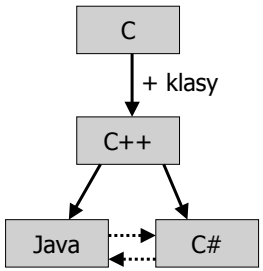
- **Abstrakcja** – praca z obiektami na poziomie ogólności (względem hierarchii dziedziczenia) odpowiednim dla rozwiązywanego problemu
  - Np. traktowanie w danym kontekście instancji klasy Jamnik jako instancji klasy Pies, jeśli nie są w nim wykorzystywane atrybuty i metody niewystępujące w klasie Pies
  - Umożliwia np. traktowanie kolekcji wystąpień konkretnych podklas klasy Pies ogólnie – jako psów
- **Polimorfizm** – różne zachowanie w odpowiedzi na takie samo wywołanie metody w zależności od konkretnej klasy obiektu
  - W połączeniu z dziedziczeniem i abstrakcją
  - **Późne wiązanie** – decyzja o tym, z której klasy metodę wywołać podejmowana w trakcie działania programu, a nie na etapie kompilacji

Historia programowania obiektowego (1/2)

- **Simula** (Simula I, Simula 67)
  - Pierwszy zorientowany obiektowo język programowania
  - Opracowany w Norwegian Computer Center na bazie języka Algol 60
  - Język opracowany z myślą o symulacjach (symulacje statków)
  - Wprowadził pojęcia klas, obiektów, podklas
- **Smalltalk** (Smalltalk 80)
  - Opracowany w Xerox Palo Alto Research Center
  - Graficzny interfejs użytkownika
  - Język w pełni obiektowy („wszystko jest obiektem”)
  - Mechanizm reflekcji (programowy odczyt struktury klasy)
  - Wykorzystywany głównie do celów dydaktycznych

Historia programowania obiektowego (2/2)

- **C++** (początkowo: C z klasami)
  - Opracowany przez Bjarne Stroustrupa w 1983
  - Koncepcje Simuli dodane do języka C
  - Pierwszy język zorientowany obiektowo, który odniósł komercyjny sukces stając się jednym z najpopularniejszych języków programowania lat dziewięćdziesiątych
- **Java**
  - Opracowany w Sun Microsystems w latach 90-tych
  - Składnia podobna do C++, ale wiele uproszczeń
  - Kompilowany do kodu pośredniego i interpretowany
- **C#** (C sharp)
  - Odpowiedź Microsoft na język Java
  - Pojawił się około roku 2000 wraz z platformą .NET
  - Podobny do Javy i C++
  - Kompilowany do kodu pośredniego i interpretowany



Czy możliwe jest programowanie obiektowe bez klas?

- **Klasa** jest kluczowym pojęciem w programowaniu obiektowym i najważniejszych językach programowania
  - C++, Java, C# - języki kompilowane, silnie typowane
- Alternatywą dla programowania obiektowego opartego o klasy jest programowanie obiektowe oparte o **prototypowanie**
  - Koncepcja pojawiła się wraz z językiem Self (Ungar & Smith, 1990)
  - Ułatwia dodawanie funkcjonalności obiektom pochodnym (co „normalnie" wymagałoby rozbudowy hierarchii klas)
  - Obecnie najpopularniejszym prototypowanym językiem zorientowanym obiektowo jest skryptowy język JavaScript
  - Problemy bezpieczeństwa, wydajności i odporności na błędy

Unified Modeling Language (UML)

- Język do modelowania obiektowego
- Język ogólnego przeznaczenia, niezależny od języków programowania obiektowego
- Opracowany przez Object Management Group (OMG)
- Wykorzystywany w inżynierii oprogramowania
- Standaryzuje graficzną notację do tworzenia abstrakcyjnego modelu systemu
- Modele UML mogą być automatycznie transformowane do kodu w konkretnym języku programowania np. Java

Składniki modelu systemu w UML

- **Model funkcjonalny**
  - Reprezentuje funkcjonalność systemu z punktu widzenia użytkownika
  - Zawiera m.in. diagramy przypadków użycia
- **Model obiektowy**
  - Reprezentuje strukturę systemu, obejmującą obiekty, atrybuty, operacje i powiązania
  - Zawiera m.in. diagramy klas
- **Model dynamiczny**
  - Reprezentuje wewnętrzne działanie systemu
  - Zawiera m.in. diagramy aktywności (przepływu sterowania)

Diagramy UML

- Diagram stanowi częściową, graficzną reprezentację modelu
  - Najczęściej uzupełnia go dokumentacja tekstowa
- Przykładowy diagram klas