

Obiektowe bazy danych – Obiektowy model danych

Andrzej Witas

Projekt zaliczeniowy z Podstaw Programowania C#

Streszczenie: Podstawowa Baza Danych z użyciem obiektowego modelu danych.

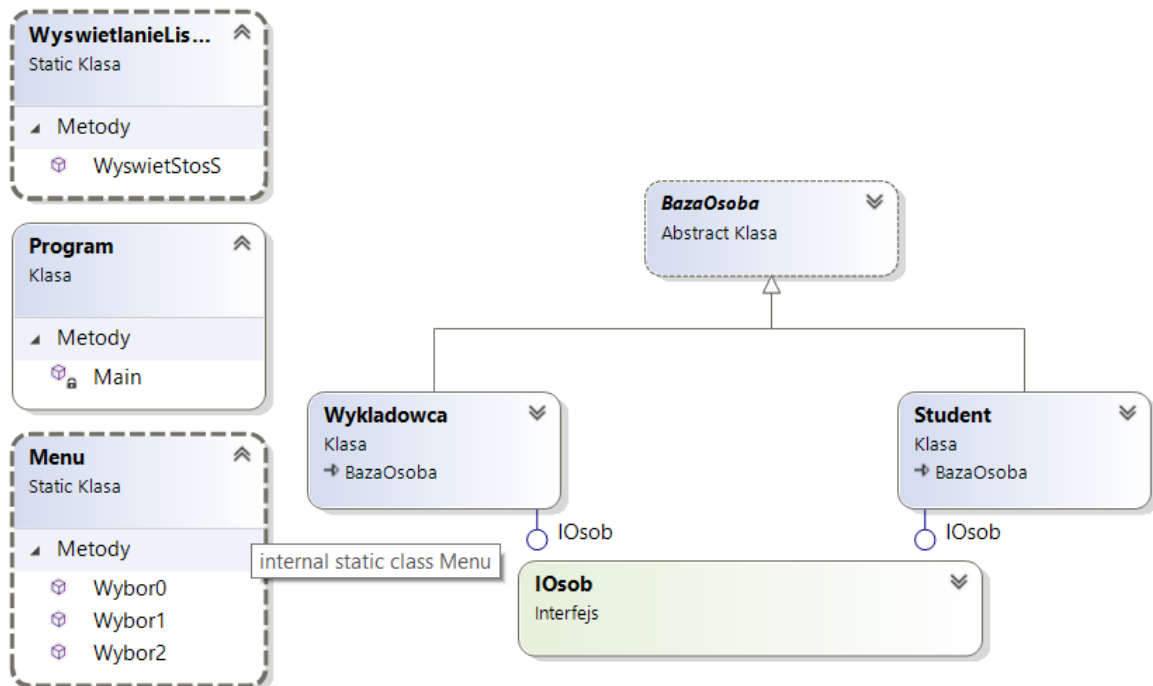
1. Wstęp

Obiektowa baza danych to zbiór obiektów, opisanych przez programistę których stan i związki są ściśle związane z obiektowym modelem danych. Zastosowany tutaj język C# wydaje się być idealnym do tego typu zastosowania. Obiektowy system zarządzania bazą danych jest systemem wspomagającym definiowanie, zarządzanie, utrzymywanie, zabezpieczanie i udostępnianie obiektowej bazy danych. Pierwsze bazy danych (relacyjne) powstawały w latach 80, głównie na użytek bankowości, ubezpieczenia, finanse, gospodarka magazynowa, itp. Niestety problemem była technologia - integracja aplikacji baz danych zapisanych za pomocą języków obiektowych z relacyjnymi bazami danych była trudna i nienaturalna ponieważ system typów bazy danych i system typów aplikacji są całkowicie odmienne. Pierwsze prace nad standaryzacją obiektowych baz danych zaczęły się w roku 1991. Stworzona została grupa do prac nad standardem, została ona nazwana Object Database Management Group ODMG. Grupa zaczęła działać formalnie jako organizacja. Pierwszy rys standardu został opublikowany w roku 1991. W roku 1999 rozpoczął swoją pracę JSR243, który w roku 2002 opublikował standard Java Data Objects 1.0

2. Model Obiektowy

Reprezentacja przedstawionego fragmentu rzeczywistości za pomocą relacyjnego modelu danych nie jest ani prosta, ani naturalna. Z pomocą przychodzą języki obiektowe. Obiekt jest podstawowym pojęciem dla obiektowości, jest definiowany jako abstrakcyjny byt, reprezentujący lub opisujący pewną rzecz lub pojęcie obserwowane w świecie rzeczywistym. Obiekt jest odróżnialny od innych obiektów, ma nazwę i dobrze określone granice. Klasa to wszystkie obiekty mogące zostać zgrupowane w jednej klasie mające ten sam zbiór atrybutów i metod. Obiekt utworzony na podstawie klasy (wzoru) to instancja. Wewnętrzna struktura obiektu oraz implementacja metod są ukryte przed użytkownikami obiektu. Mówi się, że są to prywatne własności obiektów, niedostępne z zewnątrz. Dostęp do obiektów umożliwia ich publiczny interfejs, czyli wywołania metod obiektu. Klasy mogą być definiowane jako specjalizacje innych klas. Klasa wyspecjalizowana jest nazywana podklasą i dziedziczy ona cechy i metody swojej nadklasy. Dziedziczenie

umożliwia współdzielenie implementacji klas. Dodatkowo klasa wyspecjalizowana jest podtypem swojej nadklasy.



Rysunek 2. Diagram aktywności (UML)

3. Projekt i implementacja

Poniżej przedstawione listingi kodu realizującą podstawową bazę danych uczelni przedstawioną na diagramie. Punktem wyjściowym jest klasa bazowa:

Listing 1.1. Fragment kodu źródłowego.

```

abstract public class BazaOsoba // abstrakcyjna
{
    protected string imie;
    protected string nazwisko;
    protected int wiek;
    protected Int64 pesel;
    static Int16 licznik;

    public BazaOsoba (string a, string b, int c, int d) // Konstruktor
    {
        this.imie = a;
        this.nazwisko = b;
        this.wiek = c;
        this.pesel = d;
        licznik++;
    }
}

```

Klasa jest abstrakcyjna (nie pozwala utworzyć instancji). Dostęp do chronionych pól realizowany jest przez właściwości i akcesory: get set, generowane automatycznie.

Klasę dziedziczącą po klasie BazaOsoba stanowi Klasa Student.

Listing 1.2. Fragment kodu źródłowego.

```
public class Student : BazaOsoba , IOsob
{
    private int numerLegitymacji; // zakres od 10000 do 3000
    private int rokStudiow;
    private int ocenaZProgramowania;
}
```

wszystkie pola są chronione i tak jak w klasie BazaOsoba dostęp umożliwiają get i set. Konstruktor utworzony został na podstawie konstruktora klasy bazowej:

Listing 1.2. Fragment kodu źródłowego.

```
public Student (string a, string b, int c, int d, int x, int y, int z)
    :base (a, b, c, d)
{
    this.numerLegitymacji = x;
    this.rokStudiow = y;
    this.ocenaZProgramowania = z;
}
```

Kolejną klasę dziedziczącą po klasie BazaOsoba stanowi Klasa Wykladowca

Listing 1.2. Fragment kodu źródłowego.

```
public class Wykladowca : BazaOsoba, IOsob
{
    private int numerLegitymacji;
    private string tytulNaukowy;

    public Wykladowca(string a, string b, int c, int d, int x, string y)
        : base(a, b, c, d)
    {
        this.NumerLegitymacji = x;
        this.TytulNaukowy = y;
    }
}
```

Całość jak wyżej.

Aby umożliwić dostęp do zapisywanych danych został utworzony interface o nazwie IOsob.

Listing 1.2. Fragment kodu źródłowego.

```
public interface IOsob
{
    string ToString();
}
```

Ponieważ zarówno klasa Student jak i Wykładowca dziedziczy po powyższym interfejsie wymusza to na programiście napisanie osobnej funkcji ToString() pozwalające w zmodyfikowany sposób wyświetlić dane.

Instancje klasy Student i Wykładowca są przechowywane w liście ArrayList. Dzięki temu nie musimy się przejmować zawczasu ile obiektów zostanie utworzonych (tablica dynamiczna).

Menu bazy danych zostało zaimplementowane za pomocą nieskończonej pętli while.

Wpisywane dane zostały poddane sprawdzeniu za pomocą funkcji try...catch.

Listing 1.2. Fragment kodu źródłowego.

```
while (true)
{
    Console.WriteLine("Witam tutaj baza danych, co chcesz zrobic ");
    Console.WriteLine("1.Dodac studenta: ");
    Console.WriteLine("2.Dodac nauczyciela: ");
    Console.WriteLine("3.Wyswietlisc zapisanych studentow: ");
    Console.WriteLine("4.Wyswietlisc nauczycieli: ");
    //Console.WriteLine("4.Zapisac do pliku txt ");
    Console.WriteLine("0.Zakonczyć ");

    int wybor = new int();

    while (true)
    {
        try
        {
            wybor = Convert.ToInt32(Console.ReadLine());
            break;
        }
        catch
        {
            Console.WriteLine("Bład");
        }
    }

    {
        if (wybor == 0)
        {
            Menu.Wybor0();
        }
        if (wybor == 1) // Student(string a, string b, int c, int d, int x, int y, int z)
        {
            Student temp = Menu.Wybor1();
            string a = temp.ToString();
            tablicaStudent.Add(temp);
        }
    }
}
```

```

        if (wybor == 2)
        {
            Wykladowca temp = Menu.Wybor2();
            string a = temp.ToString();
            tablicaWykladowca.Add(temp);
        }
        if (wybor == 3)
        {
            WyszwietlanieListyStudent.WyswietStosS(tablicaStudent);
        }
        if (wybor == 4)
        {
            WyszwietlanieListyStudent.WyswietStosS(tablicaWykladowca);
        }
    }
}

```

Żeby zwiększyć czytelność kodu funkcja wyświetlająca listę dodanych studentów lub wykładowców została zaimplementowana w osobnej statycznej klasie. Publiczna i statyczna funkcja dostaje jako argument tablicę ArrayList.

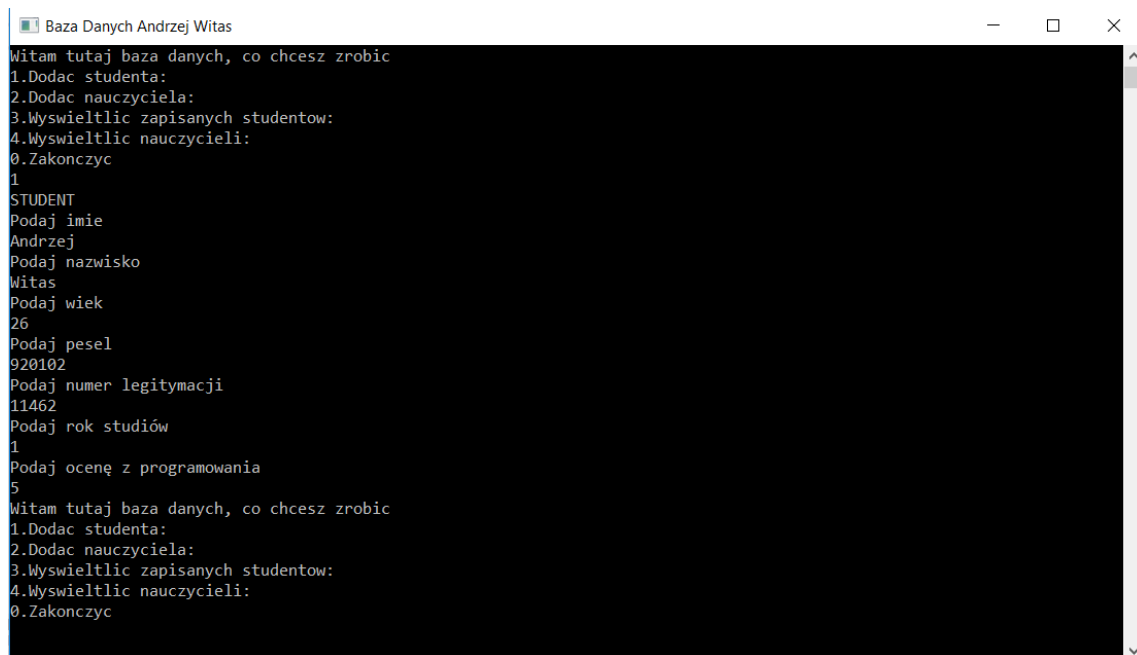
Listing 1.2. Fragment kodu źródłowego.

```

public static class WyszwietlanieListyStudent
{
    public static void WyswietStosS (ArrayList a)
    {
        //Console.WriteLine(a.Count());
        if (a.Count == 0)
        {
            Console.WriteLine("Pusta lista ");
        }

        for (int i = 0; i < a.Count; i++)
        {
            Console.WriteLine(a[i]);
        }
    }
}

```



```
Witam tutaj baza danych, co chcesz zrobic
1.Dodac studenta:
2.Dodac nauczyciela:
3.Wyszukiwanie zapisanych studentow:
4.Wyszukiwanie nauczycieli:
0.Zakonczy
1
STUDENT
Podaj imie
Andrzej
Podaj nazwisko
Witas
Podaj wiek
26
Podaj pesel
920102
Podaj numer legitymacji
11462
Podaj rok studiow
1
Podaj ocene z programowania
5
Witam tutaj baza danych, co chcesz zrobic
1.Dodac studenta:
2.Dodac nauczyciela:
3.Wyszukiwanie zapisanych studentow:
4.Wyszukiwanie nauczycieli:
0.Zakonczy
```

Rysunek 2. Uruchomiona aplikacja

4. Podsumowanie.

Na podstawie powyższego programu można stwierdzić, że programowanie obiektowe świetnie nadaje się do tworzenia baz danych. Dodawanie kolejnych rozbudowanych elementów nie stanowi trudności a polimorfizm pozwala oszczędzić wiele linii kodu i pracy.

Literatura

1. J.D. Ullman, J. Widom, Podstawowy wykład z systemów baz danych, WNT, W-wa, 2000 (seria: Klasyka Informatyki)
2. http://e.wsei.edu.pl/pluginfile.php/22688/mod_resource/content/1/UML.pdf
3. <http://wazniak.mimuw.edu.pl/images/4/4a/ZSBD-2st-1.2-w4.tresc->
4. <http://zasoby.open.agh.edu.pl/~09sbfraczek/diagram-klas%2C1%2C11.html>
5. C#. Programowanie Autor: Jesse Liberty
6. http://c-sharp.ue.katowice.pl/ksiazka/c_sharp_wer2_0.pdf
7. https://pl.wikipedia.org/wiki/Obiektowa_baza_danych