AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Programowanie obiektowe w C#

Wykład 1 – wprowadzenia do języka C#

dr inż. Tymoteusz Turlej

Wydział Inżynierii Mechanicznej i Robotyki
Kraków, 7.10.2021

Organizacja zajęć

Wykład: 14 godzin – 7 spotkań, co 2 tygodnie

Zajęcia laboratoryjne: 14 godzin – 7 spotkań, co dwa tygodnie

Ocena końcowa:

- **aktywność na zajęciach,**
- **wykonanie instrukcji ćwiczeń laboratoryjnych,**
- **realizacja projektu zaliczeniowego.**

Środowisko programistyczne:

- **Visual Studio 2019,**
- **Visual Studio Code**

Godziny konsultacji: do ustalenia ze Starostą/Starostami Grup

Środowisko programistyczne VS2019

- Logowanie na: panel.agh.edu.pl
- Azure For Education
- Uwierzytelnienie mailem w domenie @student.agh.edu.pl
- Oprogramowanie: Visual Studio Enterprise 2019 lub
  Visual Studio Community
- Instalator Visual Studio wymagane pakiety:
  - Programowanie aplikacji klasycznych dla platformy .NET,
  - Programowanie aplikacji klasycznych w języki C++,
  - Opracowanie zawartości dla platformy uniwersalnej,
  - Opracowanie zawartości dla platformy ASP.NET i sieci (opcjonalnie),
  - Magazynowanie i przetwarzanie danych,
  - Aplikacje analiz i przetwarzania danych (opcjonalnie),
  - Programowanie dla wielu platform w środowisku .NET (opcjonalnie),
  - Opracowanie rozwiązań dla oprogramowania Office,
  - Programowanie rozszerzeń programu Visual Studio

Pozycje literaturowe

- Wprowadzenie do WPF : tworzenie aplikacji w WPF przy użyciu XAML i C# / Anna Kempa
- C# 7.0 w pigułce / Joseph Albahari, Ben Albahari
- Język C# w 7 dni : solidne podstawy programowania obiektowego / Gaurav Aroraa
- Programowanie wieloplatformowe z C++ i wxWidgets 3
- Windows 8 : programowanie aplikacji z wykorzystaniem C# i XAML
- C# 6.0 : księga przepisów / Jay Hilyard, Stephen Teilhet
- Nowoczesny C++ : zbiór praktycznych zadań dla przyszłych ekspertów
- Zrozumieć programowanie / Gynvael Coldwind
- WPF 4.5. Księga eksperta-Helion (2015)/ Adam Nathan
- Pro WPF 4.5 in C#, 4th Edition_ Windows Presentation Foundation in .NET 4.5-Apress (2012)/ Matthew MacDonald
- Mastering C# (C Sharp Programming)_ A Step by Step Guide for the Beginner, Intermediate and Advanced User, Including Projects and Exercises (2019)/ Michael B. White

Zakres materiału

- Platforma .NET i historia C#
- Podstawy języka C#:
  - składnia, podstawy typów, typy liczbowe, typ logiczny, operatory,
  - łańcuchy znaków, pojedyncze znaki,
  - tablice,
  - zmienne i parametry,
  - wyrażenia i operatory,
  - instrukcje,
  - przestrzenie nazw

Zakres materiału

- Tworzenie typów:
  - klasy,
  - dziedziczenie,
  - typ object,
  - struktury,
  - modyfikatory dostępu,
  - typy zagnieżdżone i generyczne,
- Zaawansowane elementy języka C#:
  - delegaty,
  - zdarzenia,
  - wyrażenia lambda,

Platforma .NET Framework

Platforma programistyczna:

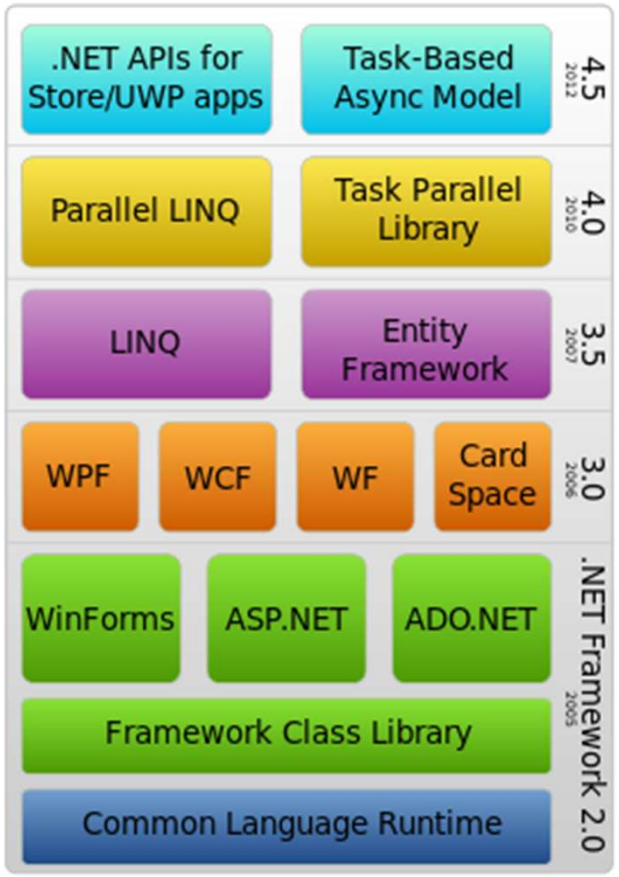
- Środowisko uruchomieniowe (CLR),
- Biblioteki klas.

Bloki składowe platformy .NET:

- CLR (Common Language Runtime),
- CTS (Common Type System),
- CLS (Common Language Specification).

Języki programowania:

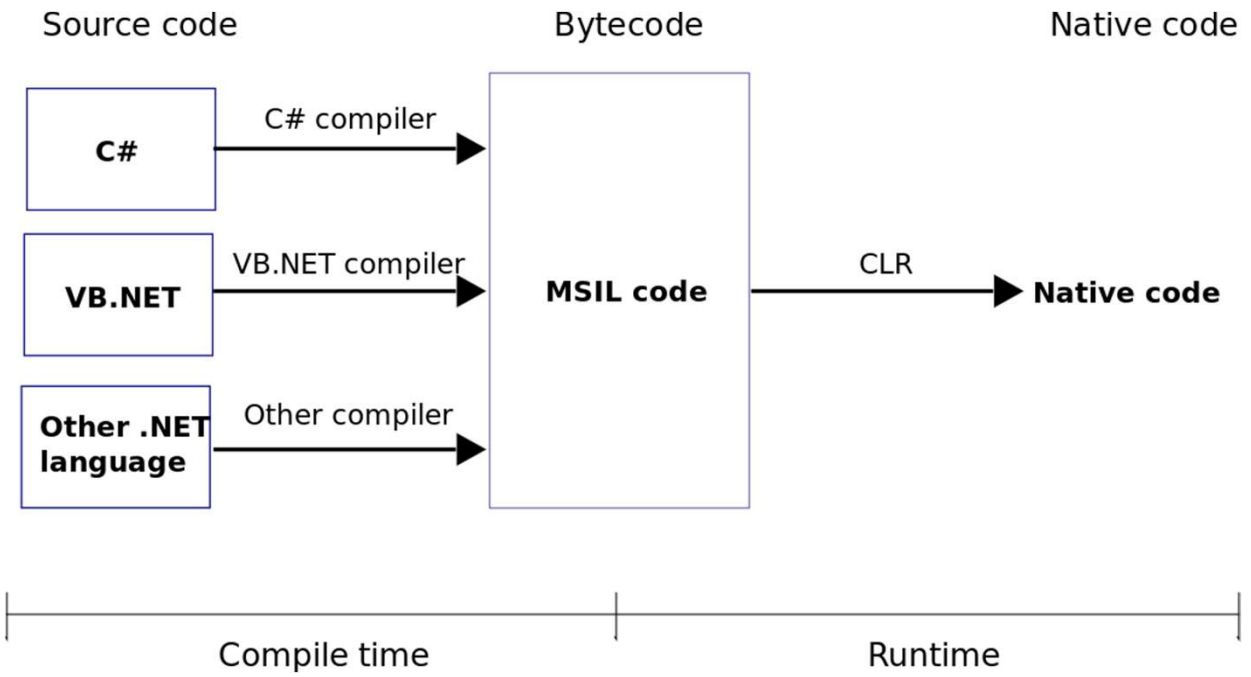
- C#, C++/CLI, VB .NET, J#,
- Python, Smalltalk, itd.



Ź: Wikipedia

CLR – Środowisko Uruchomieniowe

- Wykonanie kodu wyrażonego w CIL (Common Intermediate Language),
- Kompilacja i wykonanie kod aplikacji (w języku CIL),



CTS – System Typu Wspólnego

Funkcjonalność CTS:

- Integracja pomiędzy językami, bezpieczeństwo typów,
- Model zorientowany obiektowo (implementacja wielu języków),
- Definicja reguł dla różnych języków,
- Biblioteka danych z typami pierwotnymi.

Typy w .NET:

- typy wartościowe,
- typy referencyjne.

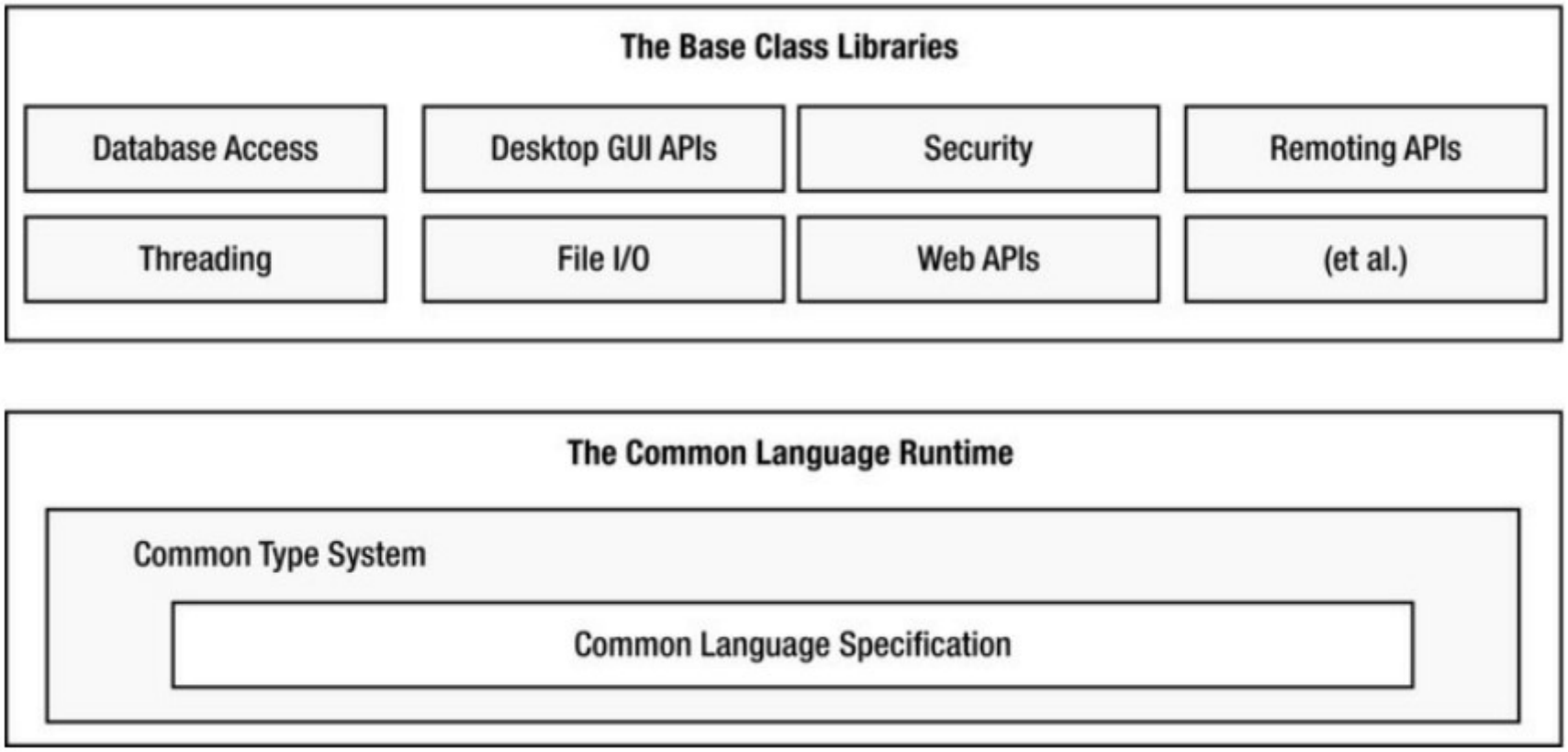
Kategorie typów:

- Klasy,
- Struktury,
- Wyliczenia,
- Interfejsy,
- Delegaty.

Definicje typów obejmują m.in..:

- Atrybuty,
- Dostępność typu,
- Nazwa typu,
- Typ podstawowy typu,
- Zaimplementowane interfejsy,
- Definicje dla elementów typu

Zależności: CLR, CTS, CLS, biblioteki klas bazowych



Język C# 6.0 i platforma .NET 4.5 - Troelsen

Historia C#

- Wersja 1.0:
  - Klasy
  - Struktury
  - Interfejsy
  - Zdarzenia
  - Właściwości
  - Delegaci
  - Operatory i wyrażenia
  - Instrukcje
  - Atrybuty
- Wersja 2.0:
  - Typy ogólne
  - Typy częściowe
  - Metody anonimowe
  - Typy wartości dopuszczające wartość null
  - Iteratory
  - Kowariancja i kontrawariancja

Historia C#

- Wersja 3.0:
  - Właściwości zaimplementowane automatycznie
  - Typy anonimowe
  - Wyrażenia zapytań
  - Wyrażenia lambda
  - Drzewa wyrażeń
  - Metody rozszerzenia
  - Niejawnie wpisanych zmiennych lokalnych
  - Metody częściowe
  - Inicjatory obiektów i kolekcji
- Wersja 4.0:
  - Wiązanie dynamiczne
  - Argumenty nazwane/opcjonalne
  - Kowariantny ogólny i kontrawariant
  - Osadzone typy międzyoptykowe
- Wersja 5.0:
  - Asynchroniczne składowe
  - Atrybuty informacji o wywołującym

Historia C#

- Wersja 6.0:
  - Importy statyczne
  - Filtry wyjątków
  - Inicjatory automatycznych właściwości
  - Wyrażenia elementów członkowskich w słojach wsadowych
  - Propagator wartości null
  - Interpolacja ciągów
  - nameof, operator
- Wersja 7.0:
  - Zmienne wychodzące
  - Krotki i dekonstrukcja
  - Dopasowanie do wzorca
  - Funkcje lokalne
  - Rozwinięte elementy członkowskie wyrażenia w ułomnie
  - Ref locals and returns
  - Inne dostępne funkcje:
  - Odrzucenia
  - Literały binarne i separatory cyfr
  - Wyrażenia throw

Historia C#

- Wersja 8.0:
  - Elementy członkowskie tylko do odczytu
  - Domyślne metody interfejsu
  - Ulepszenia dopasowania wzorca:
  - Przełączanie wyrażeń
  - Wzorce właściwości
  - Wzorce krotki
  - Wzorce pozyacyjne
  - Używanie deklaracji
  - Statyczne funkcje lokalne
  - Jednorazowe struktury ref
  - Typy referencyjne dopuszczające wartość null
  - Strumienie asynchroniczne
  - Indeksy i zakresy
  - Przypisanie z łańcuszką wartości null
  - Typy niezamanagedowane skonstruowane
  - Stackalloc w wyrażeniach zagnieżdżonych
  - Ulepszenie ciągów dosłownych interpolowanych

Historia C#

- Wersja 9.0:
  - Rekordy
  - Metody ustawiające tylko do inicjowania
  - Instrukcje najwyższego poziomu
  - Ulepszenia dopasowania wzorców
  - Wydajność i międzyopłat
  - Liczby całkowite o wielkości natywnej
  - Wskaźniki funkcji
  - Pomijanie emitowania flagi localsinit
  - Funkcje dopasowania i zakończenia
  - Wyrażenia z typem new docelowym
  - static funkcje anonimowe
  - Wyrażenia warunkowe z typem docelowym
  - Kowariantne typy zwracane
  - Obsługa GetEnumerator rozszerzeń dla foreach pętli
  - Parametry odrzucania wyrażenia lambda
  - Atrybuty funkcji lokalnych
  - Obsługa generatorów kodu
  - Inicjatory modułów
  - Nowe funkcje dla metod częściowych

Pierwszy program - składnia

```
using System;                    //import przestrzeni nazw

class Test                 // deklaracja klasy
{
  static void Main()       // deklaracja metody
  {
    int x = 12 * 100;      // instrukcja 1
    Console.WriteLine (x);  // instrukcja 2
  }                                  // koniec metody
}                                    // koniec klasy
```

```
using System;
class Test
{
static void Main()
{
Console.WriteLine (CmToMeters (10));
Console.WriteLine (CmToMeters (50));
}
static int CmToMeters (int cm)
{
int meters = cm * 12;
return meters;
}
}
```

Składnia – identyfikatory, słowa kluczowe, literały

```
using System;
class Test
{
  static void Main()
  {
    int x = 12 * 100;
    Console.WriteLine (x);
  }
}
```

abstract
as
base
bool
break
byte
case
catch
char
checked
class
const
continue
decimal
default
delegate
do
double
else
enum
event
explicit
extern
false
finally
fixed
Float
for
foreach
goto
if
implicit
in
int
interface
internal
is
lock
longnamespace
new
null
object
operator
out
override
params
private
protected
public
readonly
ref
return
sbyte
sealed
short
sizeof
stackalloc
static
string
struct
switch
this
throw
true
try
typeof
uint
ulong
unchecked
unsafe
ushort
using
virtual
void
volatile
while

Podstawy typów w C#

Składowe typu:

- dane,
- funkcje.

```
using System;
public class UnitConverter
{
int ratio;                                                  // pole
public UnitConverter (int unitRatio) {ratio = unitRatio; }  // konstruktor
public int Convert (int unit) {return unit * ratio; }       // metoda
}
class Test
{
static void Main()
{
UnitConverter feetToInchesConverter = new UnitConverter (15);
UnitConverter milesToFeetConverter = new UnitConverter (520);
Console.WriteLine (feetToInchesConverter.Convert(50));
Console.WriteLine (feetToInchesConverter.Convert(80));
Console.WriteLine (feetToInchesConverter.Convert(milesToFeetConverter.Convert(1)));
}
}
```

Składowe egzemplarza i składowe statyczne

```
public class Lion
{
public string Name;                 // pole egzemplarza
public static int Population;        // pole statyczne
public Lion (string n)               // konstruktor
{
Name = n;                           // przypisanie wartości do pola egzemplarza
Population = Population + 1;        // zwiększenie wartości statycznego pola Population
}
}
```

```
using System;
class Test
{
static void Main()
{
Lion kicius1 = new Lion („Simba");
Lion kicius2 = new Lion („Skaza");
Console.WriteLine (p1.Name);
Console.WriteLine (p2.Name);
Console.WriteLine (Lion.Population);}
}
```

Konwersje, typy wartościowe, typy referencyjne

Konwersje:

- niejawna,
- jawna.

```
int x = 12345;          // int to 32-bitowa liczba całkowita
long y = x;             // niejawna konwersja na 64-bitowy typ
                           całkowitoliczbowy
short z = (short)x;     // jawna konwersja na 16-bitowy typ
                           całkowitoliczbowy
```

Kategorie typów w języku C#:

- typy wartościowe,
- typy referencyjne,
- parametry typów generycznych,
- typy wskaźnikowe.

Systematyka typów predefiniowanych

Typy wartościowe:

- Liczbowe,
- liczby całkowite ze znakiem (sbyte, short, int, long),
- liczby całkowite bez znaku (byte, ushort, uint, ulong),
- liczby rzeczywiste (float, double, decimal),
- logiczny (bool),
- znakowy (char),

Typy referencyjne:

- łańcuchowy (string),
- obiektowy (object).

Typy liczbowe

Typ C#	Typ systemu	Przyrostek	Rozmiar	Przedział wartości
Całkowitoliczbowe ze znakiem				
sbyte	SByte		8 bitów	-2^7 do 2^7-1
short	Int16		16 bitów	-2^{15} do $2^{15}-1$
int	Int32		32 bity	-2^{31} do $2^{31}-1$
long	Int64	L	64 bity	-2^{63} do $2^{63}-1$
Całkowitoliczbowe bez znaku				
byte	Byte		8 bitów	0 do 2^8-1
ushort	UInt16		16 bitów	0 do $2^{16}-1$
uint	UInt32	U	32 bity	0 do $2^{32}-1$
ulong	UInt64	UL	64 bity	0 do $2^{64}-1$
Liczby rzeczywiste				
float	Single	F	32 bity	$\pm$ (~10^{-45} do 10^{38})
double	Double	D	64 bity	$\pm$ (~10^{-324} do 10^{308})
decimal	Decimal	M	128 bitów	$\pm$ (~10^{-28} do 10^{28})

Ź: J. Albahari

Operatory

- Operatory arytmetyczne (+, -, *, /, %);
- Operatory inkrementacji i dekrementacji (x++, ++x);
- Operatory sprawdzenia przepełnienia całkowitoliczbowego (checked, unchecked);

```
int a = int.MinValue;
a--;
Console.WriteLine (a == int.MaxValue);          // jaki wynik?

int a = 10000000;
int b = 10000000;
int c = checked (a * b);                // sprawdzenie wyłącznie wyra żenia
checked                                 // sprawdzenie wszystkich wyrażeń wyrażenia
{                                          w bloku instrukcji
...
c = a * b;
...
}
int x = int.MaxValue;
int y = unchecked (x + 1);
unchecked { int z = x + 1; }
```

Następne zajęcia

- 18.19.10.2021 – ćwiczenia laboratoryjne.
- 21.10.2021 – wykład.

24.10.2021 (niedziela) – termin przesyłania własnych tematów projektów!

Do zobaczenia!