Основи платформи .*NET Framework*

2006-2010

Зміст

- Платформа .*NET.*
- Складові частини *CLI*.
- Основи поняття .*NET*-додатків та середовища виконання
- .*NET* -компіляція. Загальна проміжна мова *CIL.*
- Поняття керованого коду.
- Загальна система типів *CTS.*
- Метадані. Міжмовна інтеграція у .*NET.*
- Збірки .*NET.*
- *Just In Time* (*JIT*) компіляція.
- Управління пам'яттю. Збирання сміття.
- Порівняння компонентних підходів *COM* та .*NET.*

Від *COM* до *.NET*

За іронією долі платформа *.NET Framework* з'явилася у результаті зусиль, спрямованих на те, щоб ***спростити розробку COM-додатків***. Тобто сталося так, що не зовсім вдала платформа ***COM*** була не просто підправлена чи виправлена, а ***фактично замінена новою*** – *.NET Framework* вважається ***новою платформою компонентного програмування від Microsoft***.

Платформа .*NET Framework.* Трохи історії

Розробка платформи почалася у 1998 році. Перша робоча назва – *Project* 42, потім була назва *COM Object Runtime* чи, скорочено, *COR*, пізніше *Lightning*, *COM*+ 2.0, *Next Generation Web Services* і, нарешті, *Framework*.

Ще у 1998 році була представлена архітектура ***середовища виконання*** .*NET* – "***віртуальної машини .NET"*** :

"Більш ніяких GUID, ніяких HRESULT, ніяких IUnknown!".

Початкова назва ***середовища виконання*** – ***віртуальна система виконання VES***(***Virtual Execution System***), пізніше почав використовуватись термін ***загальномовне середовище виконання*** (***Common Language Runtime, CLR***).

13 лютого 2002 року – представлено пакет ***.NET Framework.***

Загалом, .*NET Framework* – це пакет засобів для так званих програм нового покоління. Його ядром виступає ***CLR*** – "***віртуальна машина .NET***", інструкції якої визначаються об'єктно-орієнтованою мовою ***CIL*** (***Common Intermediate Language***, ***загальна проміжна мова***).

Не менш важливим компонентом платформи .*NET* є ***Framework Class Library*** (***FCL***) – єдина бібліотека класів для всіх мов платформи .*NET*.

Framework

Платформа *.NET Framework*. Основні складові частини

.NET Framework – це пакет засобів для так званих програм нового покоління

- Ядром *.NET Framework* виступає ***CLR (Common Language Runtime)*** – "***віртуальна машина .NET***", інструкції якої визначаються об'єктно орієнтованою мовою ***CIL*** (***Common Intermediate Language, загальна проміжна мова***).

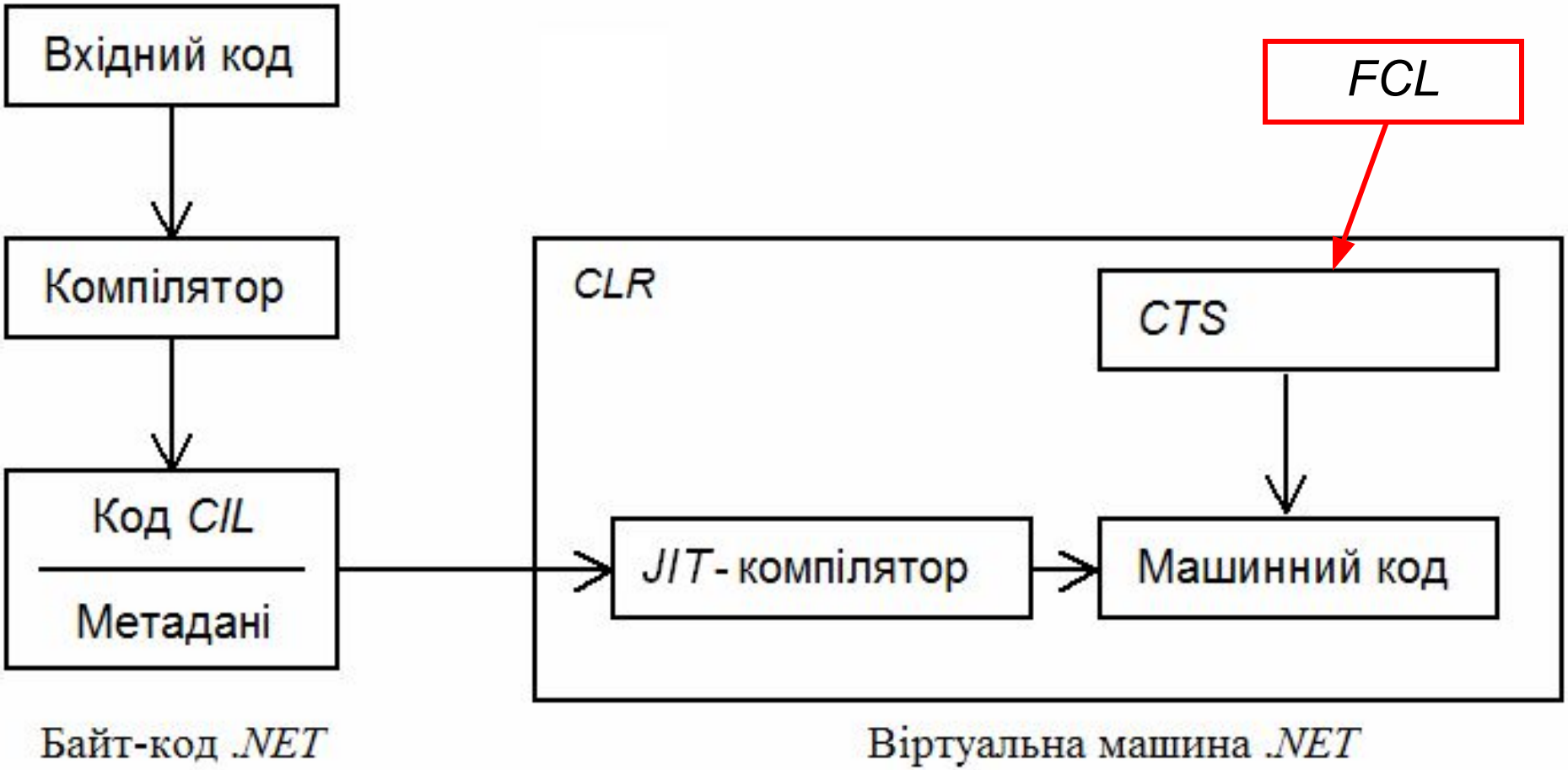
об'єктна парадигма

- Другим важливим компонентом платформи *.NET Framework* є ***Framework Class Library*** (***FCL***) – єдина бібліотека класів для всіх мов платформи *.NET*.

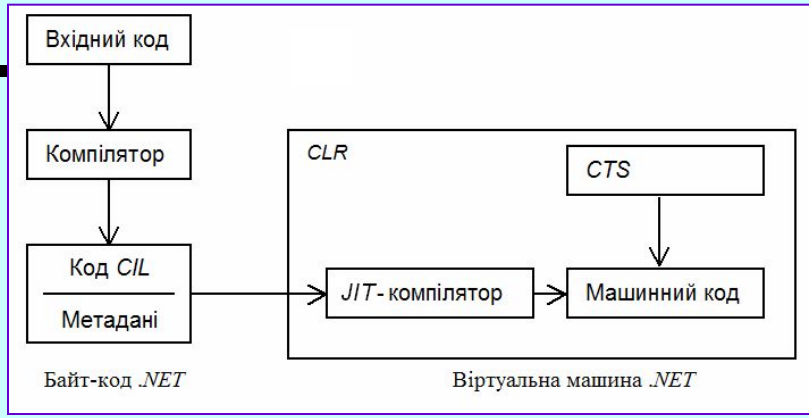
Основні поняття *.NET*- додатків та середовища виконання

Ключові специфікації:

- *CIL* (*Common Intermediate Language*) — загальна проміжна мова;
- *CTS* (*Common Type System*) — система загальних типів;
- *CLR (Common Language Runtime)* — "віртуальна машина *.NET*".



Специфікація *Microsoft* загальної інфраструктури мов (*Common Language Infrastructure, CLI*)



Складові частини *CLI*:

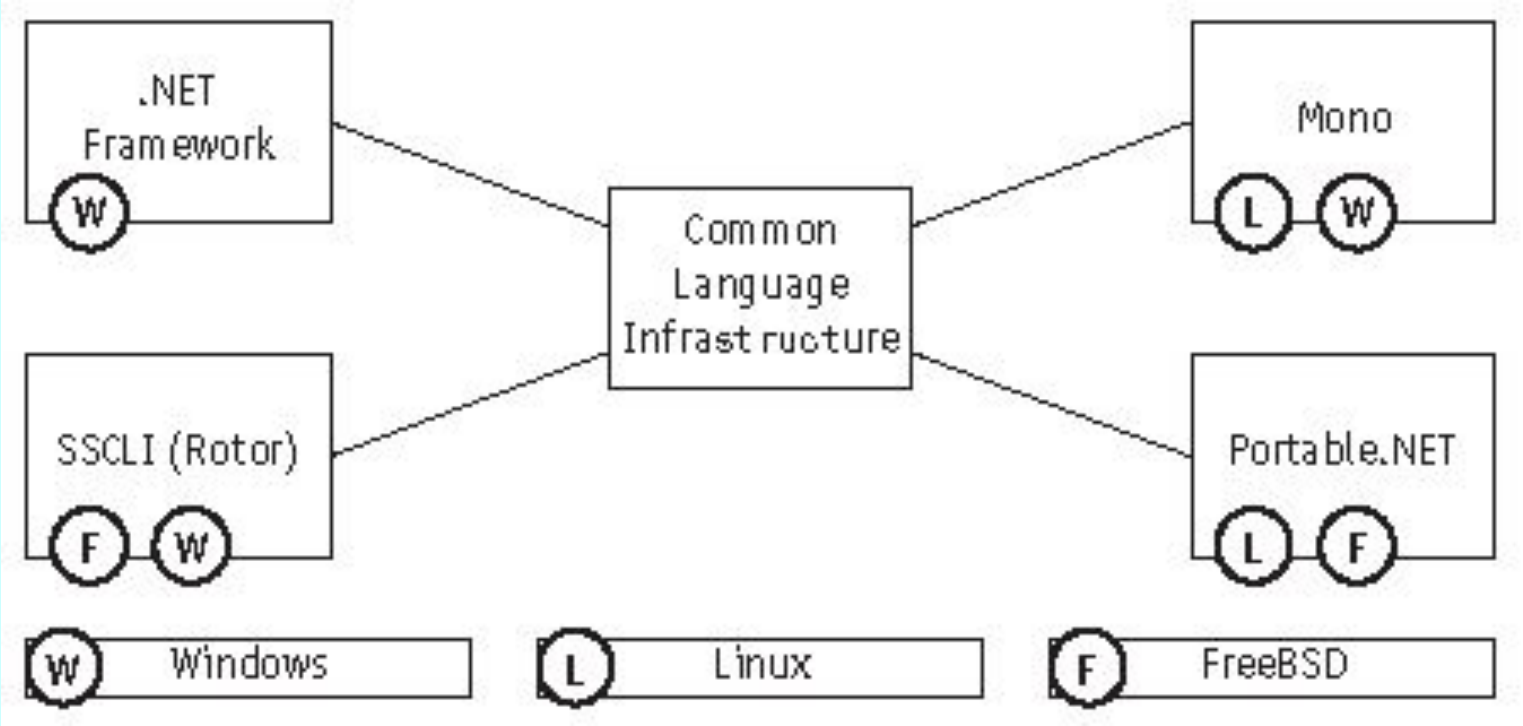
- ***Система загальних типів CTS*** (*Common Type System*) – містить основні засади використання типів, які зустрічаються у розповсюджених мовах програмування.
- ***Загальна проміжна мова*** *CIL* (*Common Intermediate Language*). (Вона є цільовою мовою компіляторів, що орієнтуються на *CLI*).
- ***Віртуальна система виконання*** *VES* (або *CLR*) – відповідає за завантаження і виконання *CIL*-програм (віртуальна машина).
- ***Система метаданих*** (*Metadata System*) – описує збірки, типи у збірках. Зберігається в незалежному від конкретної мови програмування вигляді. Використовується для підтримки компонентної (у тому числі міжмовної) інтеграції.
- ***Загальна специфікація мов*** *CLS* (*Common Language Specification*) – аспекти сумісності мов, що є важливим для реалізації компіляторів (надмовні поняття: ***збірки***, ***метадані*** ; засади міжмовної інтеграції: створення типів, використання "зовнішніх" типів).

Специфікація *Microsoft* загальної інфраструктури мов (*CLI*). Реалізації специфікації *CLI*

- ***.NET Framework – одна з можливих реалізацій*** загальної інфраструктури мов *CLI*.
- Окремі частини *CLI* стандартизовані асоціацією *ECMA* (*European Computer Manufactures Association*).

Окрім *.NET Framework* відомі й інші спроби реалізації *CLI*: платформа ***Mono*** компанії *Ximian*, проект ***Portable.NET***. Крім того, *Microsoft* пропонує вихідні коди ще однієї своєї реалізації *CLI* під *Windows* та *FreeBSD* – це *Shared Source CLI*, для якої використовується також назва ***Rotor***.

Реалізації *CLI* та платформи:



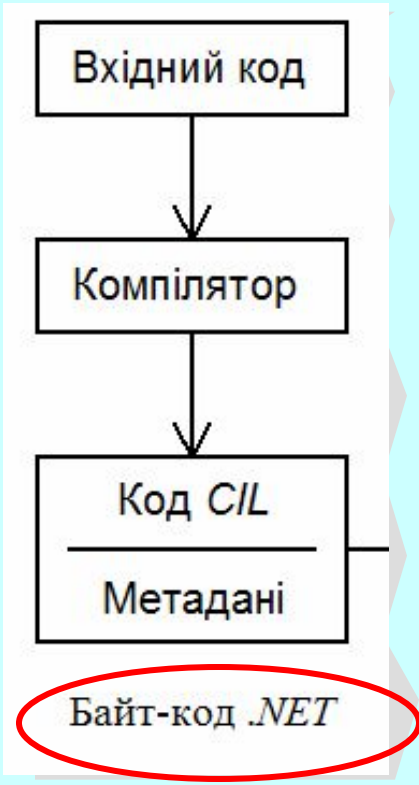
.*NET*-компіляція

.NET-компілятор за вхідним кодом генерує ***байт-код*** (***керований код***), що складається з ***програмного коду*** (набору інструкцій) у проміжній мові *CIL* та ***метаданих***.

У *.NET Framework* результатом компіляції виступають *PE*-файли (*Portable Executable*) – *.NET Framework* розширює специфікацію *PE*-файлів, за рахунок чого програмісти *.NET* найчастіше мають справу зі знайомими ***розширеннями файлів EXE та DLL.***

Але у даному випадку це знайомі "***незнайомці***" – модулі ***керованого коду*** (байт-коди *.NET*).

(Модулі із розширенням *NETMODULE* називають "сирими". Вони не можуть, наприклад, окремо завантажуватись, а тим більше виконуватись).



.NET Compilers

(http://blogs.ugidotnet.org/nettools/articles/8060.aspx)

Ada
- A# - port of Ada to .NET (Dr. Martin C. Carlisle)

APL
- Dyalog APL (Dyalog Ltd)

AsmL
- Abstract State Machine Language (MS Research)

BrainFuck
- BrainFuck.Net

C Standard
- lcc.NET

C#
- C# (MS)
- mcs (Mono/Ximian)
- cscc (DotGNU Portable.NET)

Caml
- F# (ML and Caml), Abstract IL, ILX (MS Research)

C++
- Managed Extensions for C++ (MS)
- Managed and Unmanaged C++ (GotDotNet)

Cobol
- NetCOBOL - COBOL for .NET (Fujitsu)
- Net Express (Micro Focus)

Delphi
- Borland Delphi and C++Builder Support for .NET (Borland)
- Delphi.NET - interoperability tools (Marcus Schmidt)

Eiffel
- Eiffel for .NET (Interactive Software Engineering)

Forth
- Delta Forth .NET (Valer BOCAN)

Fortran
- Lahey/Fujitsu Fortran for .NET (Lahey Computer Systems, Inc.)
- FTN95 - Fortran for Microsoft NET (Salford Software Ltd.)

Java
- Visual J# .NET (MS)
- IKVM.NET - Java VM for .NET

JavaScript
- JScript .NET (GotDotNet)
- JANET - JavaScript-compatible language

Haskell
- Hugs 98 for .NET

LOGO
- MonoLOGO (Richard Hestilow)

Lua
- Lua.NET: Integrating Lua with Rotor (PUC-RIO)

Mercury
- Mercury on .NET

Mondrian
- Mondrian and Haskell for .NET (Nigel Perry)

Oberon
- Active Oberon for .net (ETH Zuerich)

Perl
- Perl for .NET, PerlNET (ActiveState SRL.)

Pascal
- Component Pascal (QUT)
- Chrome

PHP
- PHP Sharp
- Phalanger

PL/1
- PL/IL

Prolog
- P#

Python
- KOBRA
- Open Source Python for .NET (Mark Hammond)
- IronPython

Ruby
- NetRuby
- RubyDotNet

RPG
- ASNA Visual RPG for NET

Scala
- Scala

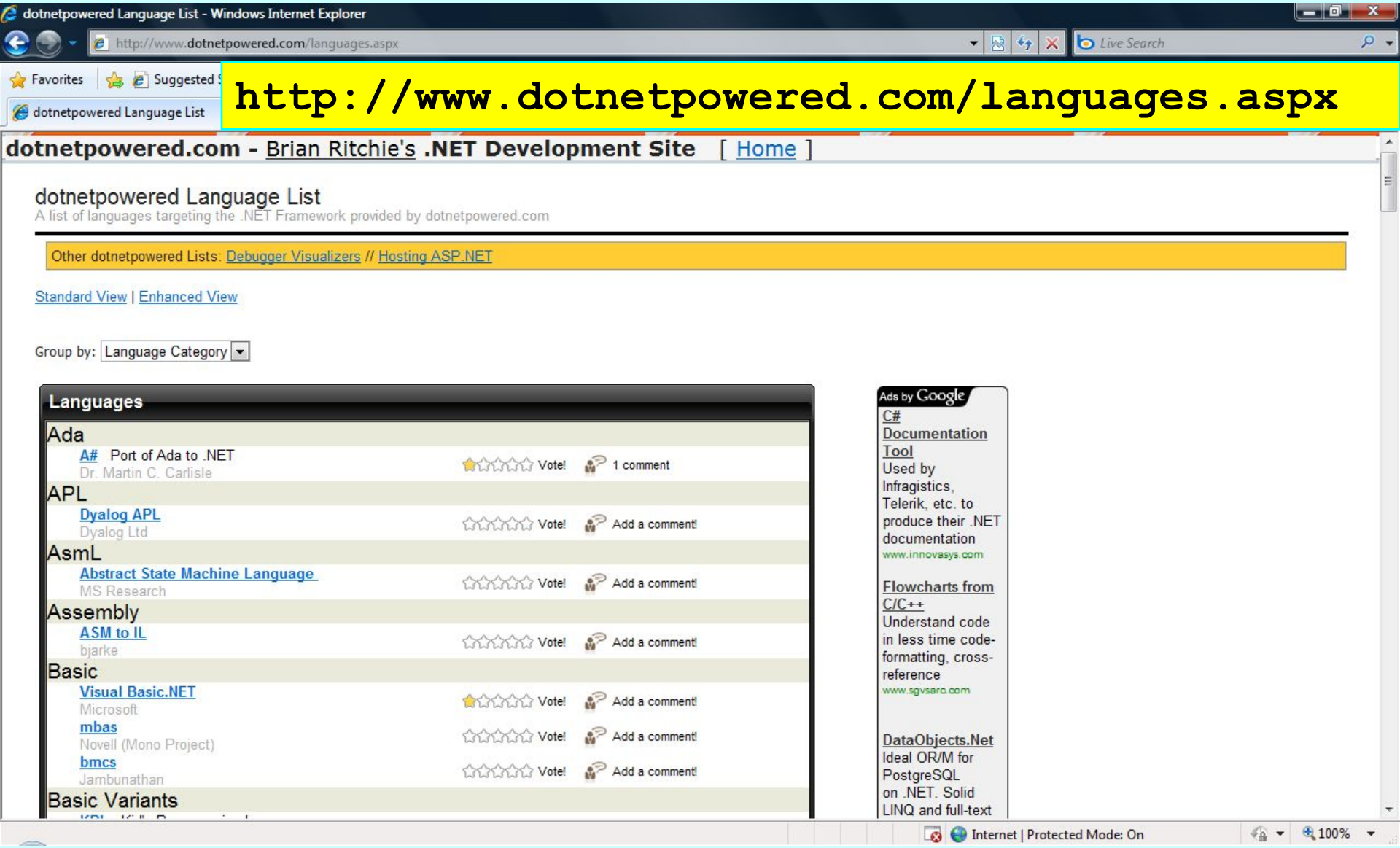
Scheme
- Scheme (Northwestern University)

Small Talk
- S# (SmallScript LLC)
- #Smalltalk

SML (Standard Meta Language)
- SML.NET (MS Research, University of Cambridge)

Visual Basic
- VB.NET (MS)
- mbas (Mono/Ximian)

.*NET*-компілятори



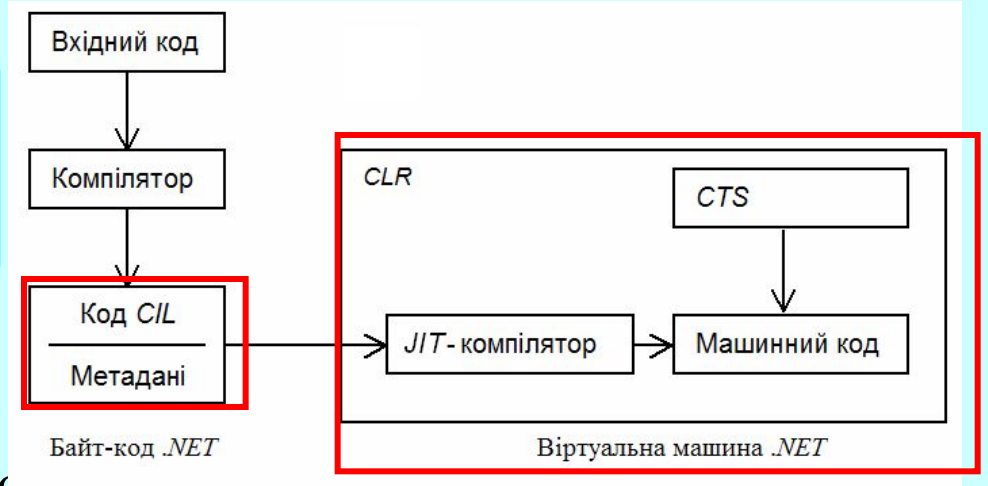
Керований код *.NET*. Як запускається віртуальна машина *.NET*?

CLR ***повністю контролює виконання*** байт-коду *.NET* (саме тому виконуваний *.NET*-код називається ***керованим*** – ***managed***). Зокрема, *CLR* забезпечує різноманітні служби і серед них ***автоматичний збирач сміття***.

Windows, опрацьовуючи виконуваний *.NET*-файл, сканує ***таблицю імпорту*** та завантажує згадувані у ній бібліотеки, серед яких завжди є ***mscoree.dll*** (часто вона буває взагалі єдиною у таблиці імпорту, *mscoree* – абревіатура від *Microsoft Component Object Runtime Execution Engine*). Бібліотека ***mscoree.dll*** – це фасад *CLR* (***фасад віртуальної машини .NET***). А ***точка входу*** керованого коду (виконуваного *.NET*-файлу *EXE* чи *DLL*) є заглушкою з командою (інструкцією) ***передачі управління віртуальній машині*** *.NET*. (Одно- чи багатопроцесорний варіант *CLR* - ***mscorwks.dll***, ***mscorksr.dll***; *JIT*-компілятор ***mscojit.dll*** *etc*).

Основним принципом загальномовного середовища виконання *CLR* є ***концепція управління кодом***.

Заглушка відрізняється від функції тим, що не повертає результат.



Основи

Загальна проміжна мова *CIL*

Код у мові *CIL* (*Common Intermediate Language*) – це незалежний від процесора набір інструкцій, які, з одного боку, можна ефективно ***перетворити в машинний код***, а з іншого боку – забезпечують ***об'єктну підтримку***, зокрема, виклики методів об'єктів. (У *CIL* забезпечується також можливість використання традиційних машинних інструкцій, повязаних із виконанням арифметичних, логічних операцій передач управління, обробкою виключень тощо).

об'єктна парадигма

- *MS* дизасемблер ***ildasm.exe*** забезпечує можливість:
  - дизасемблювання байт-коду в "асемблерний" код;
  - перегляду "асемблерного" коду *CIL* та метаданих.
- *MS* асемблер ***ilasm exe*** :
  - забезпечує підтримку програмування у *CIL* -"асемблерній" мові.

.*NET*-компіляція. Метадані

- ***Метадані*** є засобом ***самоопису*** коду, зокрема, метадані містять інформацію про всі ***класи***, визначен у вхідному коді (модулі), що дозволяє ефективно розв'язувати наступні важливі задачі компонентного проектування програм:
  - міжмовна інтеграція;
  - віддалена взаємодія між об'єктами (. *NET Remoting*);
  - механізми рефлексії (.*NET Reflection*);
  - серіалізація даних (*Serialization*).
- Метадані є абсолютно ***прозорими*** для програмістів.

Збірки .*NET*

Збірки — фундаментальні одиниці, ***"будівельні" блоки*** .*NET*. Саме збірки при потребі завантажуються та виконуються віртуальною машиною. Саме зі збірками пов'язані механізм керування версіями та механізм захисту. "Повне ім'я" будь-якого типу в *CIL* прив'язане до збірки.

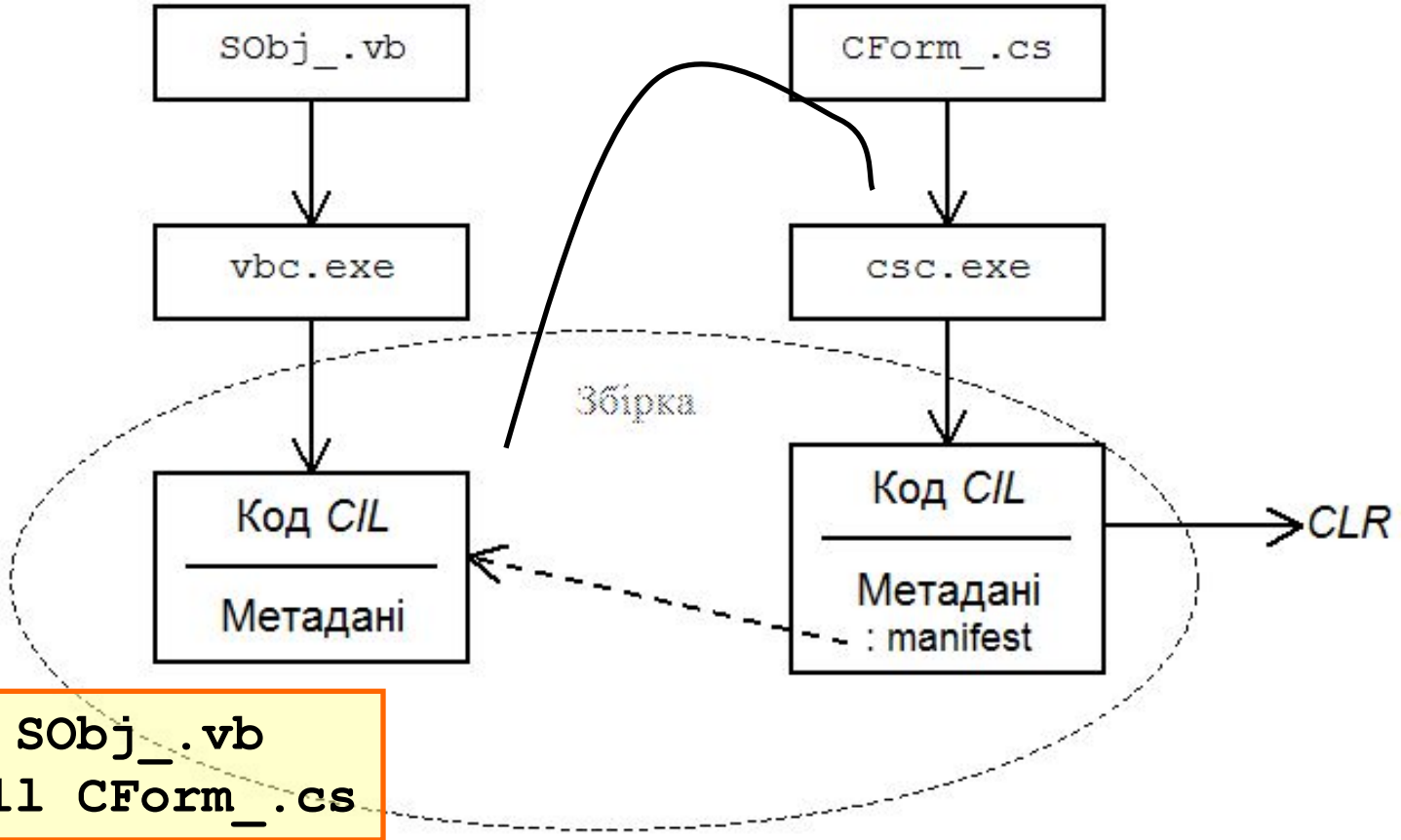
Збірка (*assembly*) є ***логічним об'єднанням*** програмних модулів, ресурсних файлів, а також інших збірок . Про "склад" збірки можна дізнатись з її ***маніфесту***, який є частиною ***метаданих***.

Збірка може розгортатись в каталозі проекту. Така збірка називається ***закритою***, оскільки вона не є доступною для проектів в інших каталогах. Щоб забезпечити використання збірки різними додатками, її потрібно занести у *кеш глобальних збірок* ***GAC*** (***Global Assembly Cache***).

(Збірки можуть бути також ***динамічними***. Такі збірки створюються у пам'яті під час виконання проекту.)

.*NET*-компіляція. Метадані. Збірки .*NET*

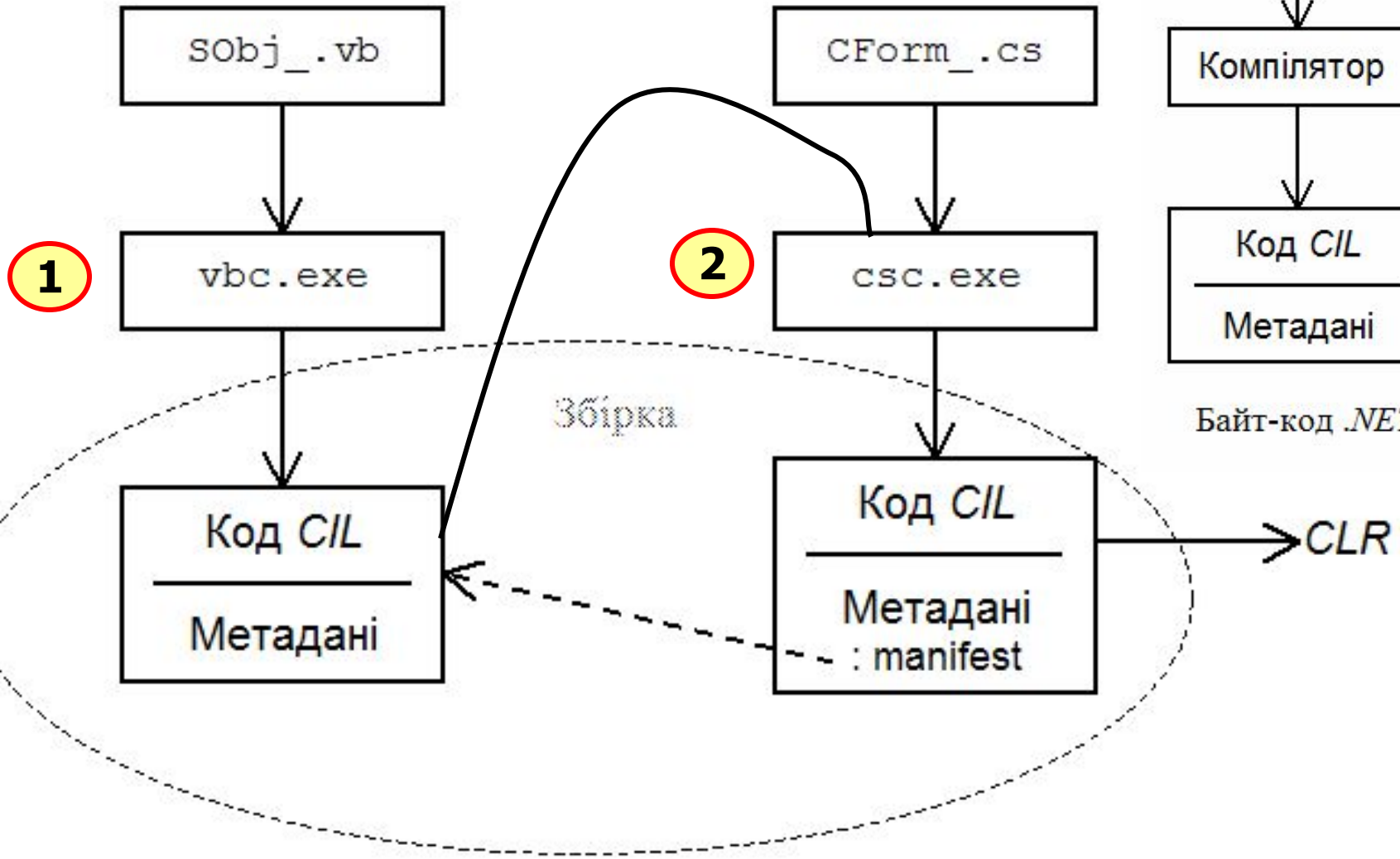
- Метадані містять інформацію не тільки про ***типи - інтерфейси, класи*** (ім'я, вид доступу, ієрархія успадкування, описи полів, методів, властивостей, вкладених типів тощо), але й про ***збірки*** (ім'я, версія, експортовані типи, залежності збірок, атрибути захисту тощо).
- Про "склад" збірки можна дізнатись з її ***маніфесту***, який є частиною ***метаданих***.




```
vbc /t:library SObj_.vb
csc /r:SObj_.dll CForm_.cs
```

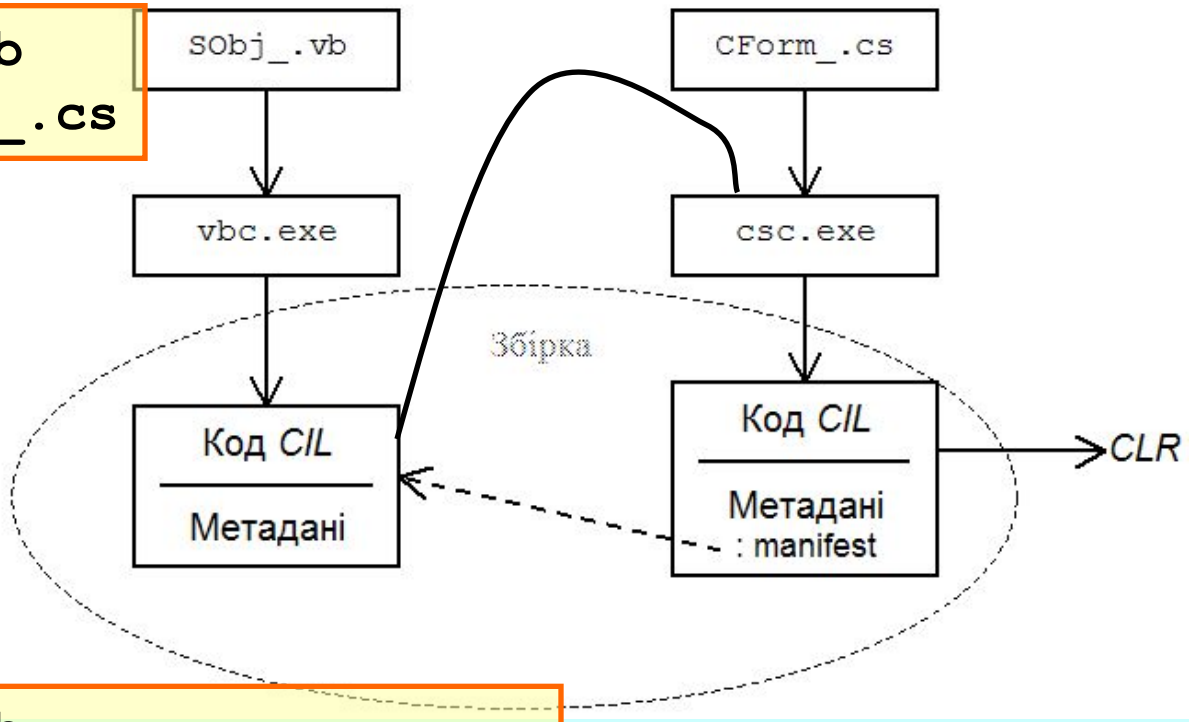
Міжмовна інтеграція у .*NET*.
Мовна безшовність .*NET*

```
vbc /target:library SObj_.vb
csc /reference:SObj_.dll CForm_.cs
```



Міжмовна інтеграція у .*NET*. Модулі .*NET*

```
vbc /t:library SObj_.vb
csc /r:SObj_.dll CForm_.cs
```




```
vbc /t:library SObj_.vb
csc /r:SObj_.dll /t:winexe CForm_win.cs
```

```
vbc  /t:module SObj_.vb
csc  /addmodule:SObj_.netmodule CForm_.cs
```

```
vbc  /t:module SObj_.vb
csc  /addmodule:SObj_.netmodule /t:winexe CForm_.cs
```

Міжмовна інтеграція у *.NET*. Приклад. (“Сервер у клієнтському процесі”, “внутрішній сервер”)

```
vbc /t:library SObj_.vb
csc /r:SObj_.dll CForm_.cs
```

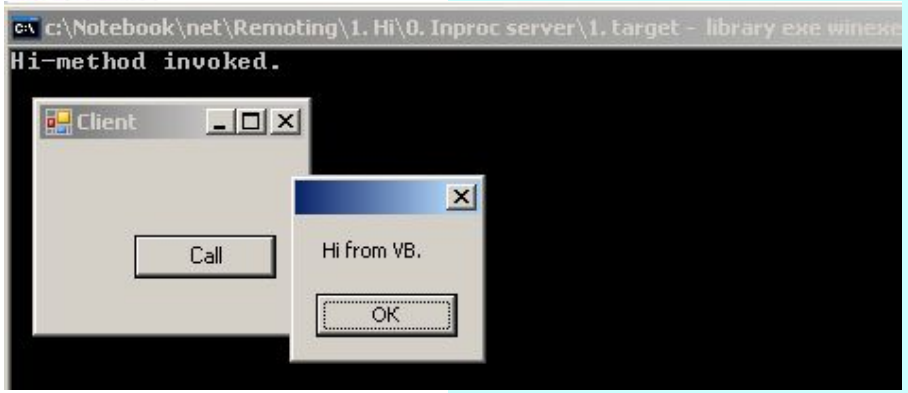
```
public class ClientForm : System.Windows.Forms.Form
{
    private ServerObject m_server;

    public ClientForm()
    {
        // Required for Windows Form Designer support
        InitializeComponent();
        m_server = new ServerObject();
    }

    static void Main()
    {
        Application.Run(new ClientForm());
    }

    private void cmdCall_Click(object sender, System.EventArgs e)
    {
        MessageBox.Show(m_server.Hi());
    }
}
```

CForm_.cs

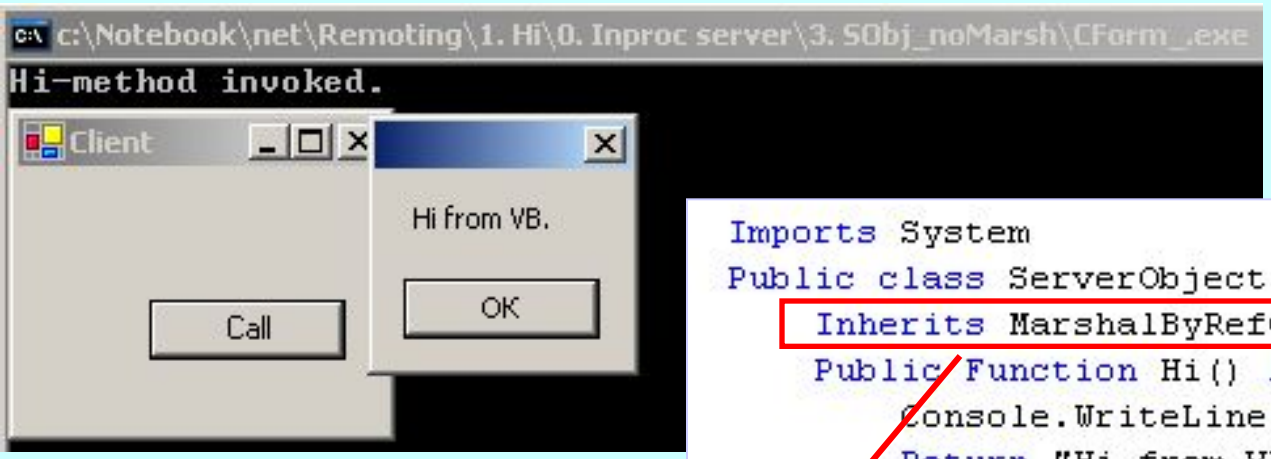



```
Imports System
Public class ServerObject
    Inherits MarshalByRefObject
    Public Function Hi() As String
        Console.WriteLine("Hi-method invoked.")
        Return "Hi from VB. "
    End Function  'Hi
End Class   'ServerObject
```

SObj_.vb

Загалом, *ServerObject* готовий і до віддаленого використання (“серверний” клас)

Міжмовна інтеграція *.NET*. Приклад. (Звичайний, "не серверний" клас у *DLL*).




```vb
Imports System
Public class ServerObject
    Inherits MarshalByRefObject
    Public Function Hi() As String
        Console.WriteLine("Hi-method invoked.")
        Return "Hi from VB. "
    End Function  'Hi
End Class   'ServerObject
```

SObj_.vb

SObj_noMarsh_.vb*

```vb
' for compile:  vbc /t:library SObj_noMarsh_.vb

Imports System
Public class ServerObject
    Inherits Object
    Public Function Hi() As String
        Console.WriteLine("Hi-method invoked.")
        Return "Hi from VB. "
    End Function  'Hi
End Class   'ServerObject
```

vbc /t:library SObj_noMarsh_.vb
csc /r:SObj_noMarsh_.dll CForm_.cs

Система загальних типів *CTS* – фундамент міжмовної взаємодії (1/2)

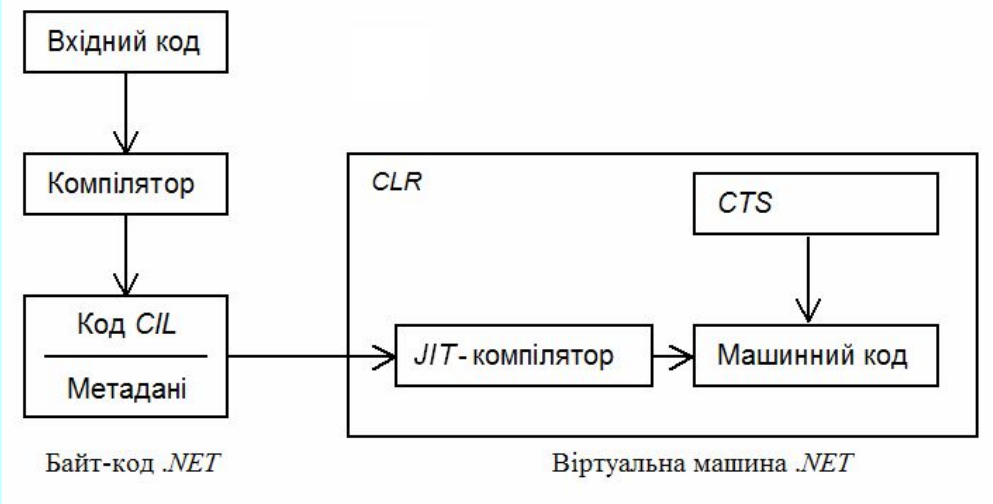
- Усі типи успадковуються від ***System.Object***:
  - "усі типи є класами";
  - не має дивувати конструкція

```
7.ToString();
```

  - дуже важливий для рефлексії метод ***GetType*** – тип (клас) ***Type*** надає широкий спектр методів рефлексії).

об'єктна парадигма

- ***Атрибути*** – об'єкти, що прикріплюються до типів чи їх членів. (***Декларативний стиль програмування***).
- Регламентуються ***члени класів*** (поля, методи, властивості, події, вкладені методи), регламентуються ***успадкування*** типів (лінійне для класів та структур і множинне для інтерфейсів).



Система загальних типів *CTS* – фундамент міжмовної взаємодії (2/2)

- ***Value*** та ***reference*** типи:
  - ***value types*** – біти у стековій пам'яті, заборона на конструктори без параметрів (конструктори за замовчуванням):
    - вбудовані типи;
    - типи, що визначаються користувачем:
      - *enum, struct;*
      - *System.DateTime, System.Decimal, System.Guid* – "наперед" визначені типи *FCL.*
  - ***reference types*** – посилання: вказівник + біти у динамічній пам'яті (купі), при цьому вказівник інкапсулює одночасно адресу та інформацію про тип.
    - Порівняння: ідентичність (посилання на один і той самий об'єкт) та рівність (однакові біти у пам'яті) (*.NET* каже "ні" безтиповому підходу, адресній арифметиці. Є лише дві загальнодоступні операції над посиланнями: читання та занесення даного).
    - Політика безпечності типів (контроль доступу з боку *CLR*).
    - Автоматичне збирання сміття.
  - ***boxing*** (пакування, обгортання), ***unboxing***.

CTS (http://msdn.microsoft.com) (1/3)

Руководство разработчика по .NET Framework

Система общих типов CTS

Обновлен: Ноябрь 2007

Система общих типов определяет способ объявления, использования и управления типами во время выполнения, а также является важным элементом поддержки межъязыковой интеграции в среде выполнения. Система общих типов выполняет следующие функции.

- Устанавливает инфраструктуру, помогающую обеспечивать межъязыковую интеграцию, строгую типизацию и высокопроизводительное выполнение кода.
- Предоставляет объектно-ориентированную модель, поддерживающую полную реализацию многих языков программирования.
- Определяет правила, которых необходимо придерживаться в языке. Эти правила помогают обеспечить взаимодействие объектов, написанных на разных языках.

CTS (http://msdn.microsoft.com) (2/3)

Определения типов
Описание пользовательских типов.

Члены типов
Описание событий, полей, вложенных типов, методов, свойств и понятий, таких как перегрузка членов, переопределение и наследование.

Типы значений в системе общих типов CTS
Описание встроенных и пользовательских типов значений.

Классы в системе общих типов CTS
Описание характеристик классов среды CLR.

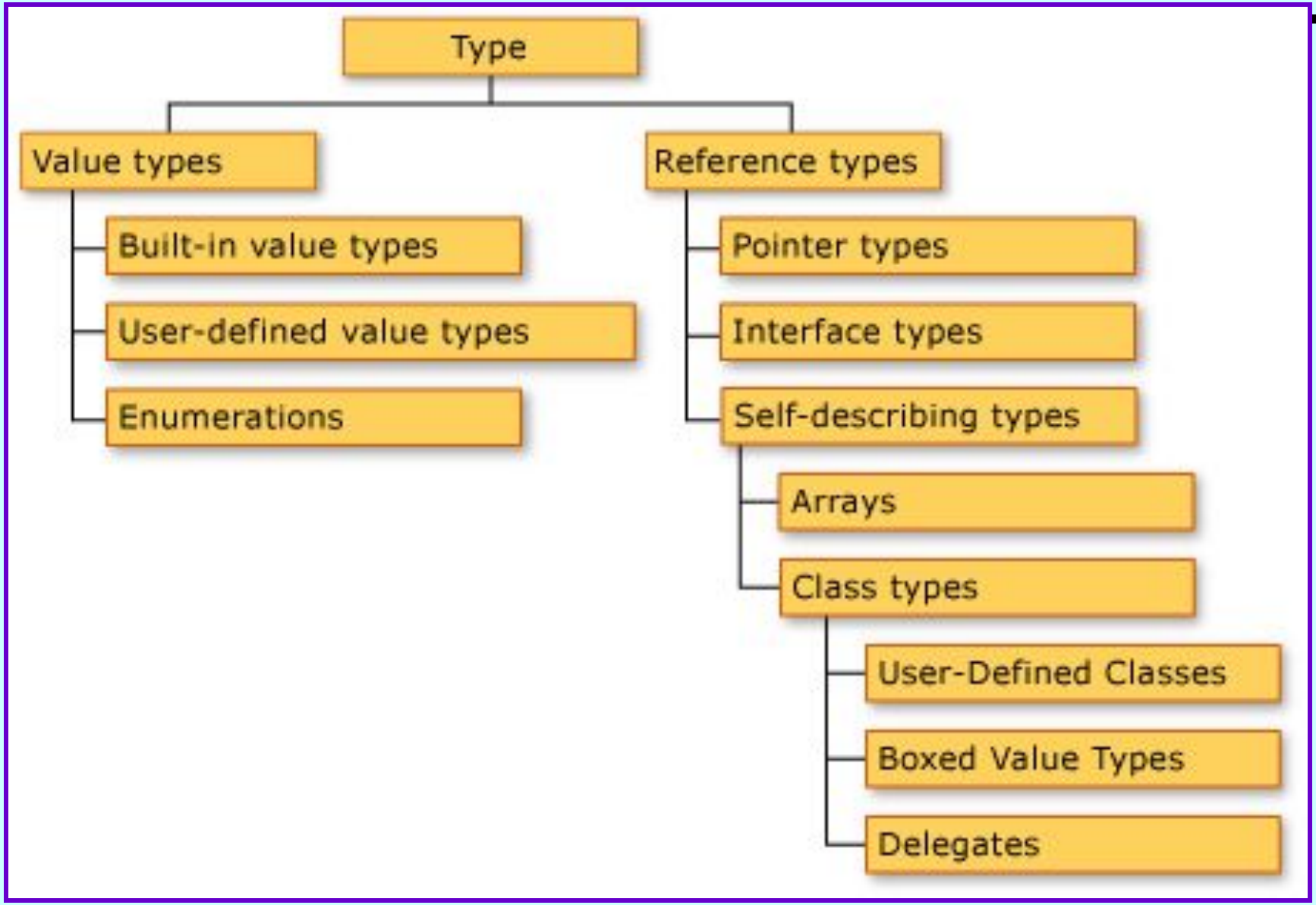
Делегаты в системе общих типов CTS
Описание объекта делегата, который является управляемой альтернативой для неуправляемых указателей на функции.

Массивы в системе общих типов CTS
Описание типов массивов среды CLR.

Интерфейсы в системе общих типов CTS
Описание характеристик интерфейсов, а также ограничений на интерфейсы, заданных средой CLR.

Указатели в системе общих типов CTS
Описание управляемых и неуправляемых указателей, а также неуправляемых указателей функций.

CTS (http://msdn.microsoft.com) (3/3)



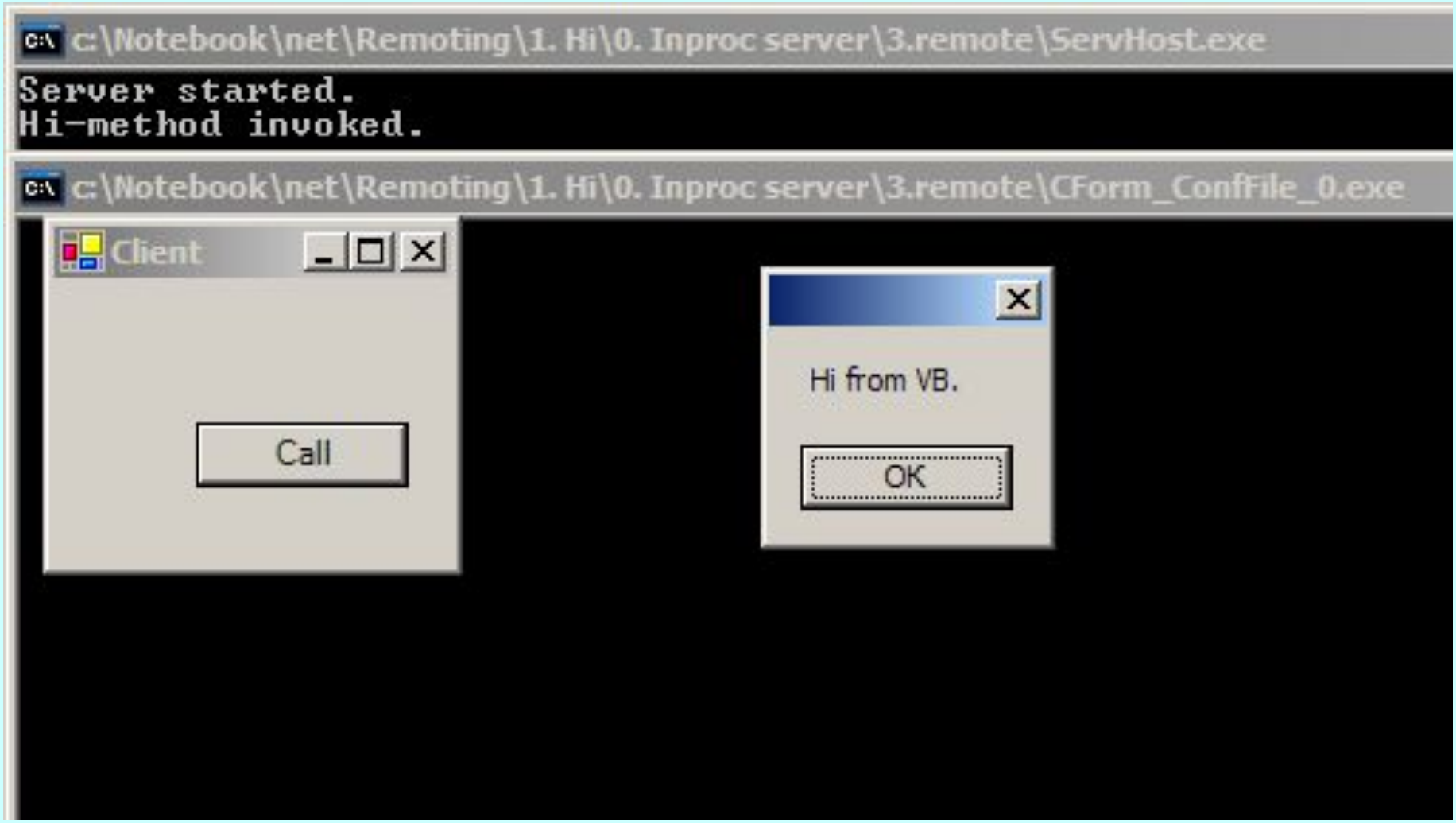
Типи .*NET Framework – FCL (Framework Class Library) та CLS*-сумісність

Категория	Имя класса	Описание	Тип данных в Visual Basic	Тип данных в C#	Тип данных в управляемых расширениях для C++	Тип данных в JScript
Целое число	Byte	8-разрядное целое число без знака.	**Byte**	**byte**	**char**	**Byte**
	SByte	8-разрядное целое число со знаком. Не является CLS-совместимым.	**SByte** Не является встроенным типом.	**sbyte**	**signed char**	**SByte**
	Int16	16-разрядное целое число со знаком.	**Short**			
	Int16	32-разрядное целое число со знаком.	**Integer**	**int**	**int** -или- **long**	**int**
	Int64	64-разрядное целое число со знаком.	**Long**	**long**	**__int64**	**long**
	UInt16	16-разрядное целое число без знака. Не является CLS-совместимым.	**UInt16** Не является встроенным типом.	**ushort**	**unsigned short**	**UInt16**

Не має відповідного ключового слова, треба вказувати повне ім'я класу - **`[mscorlib]System.SByte`**

Framework

Приклад віддаленої взаємодії (*remoting*)



Just In Time (*JIT*) компіляція у машинний код (компіляція за потребою, компіляція "на льоту")

Особливості *JIT*-компіляції:

- Саме компіляція (у машинний код), а не інтерпретація.
- Не вся програма (байт-код) компілюється. (Компіляція ***за потребою*** – враховується, що при виконанні програми не весь її код може бути використаним, а отже не весь варто компілювати).
  - Збірки, потреба у яких не виникає у даний момент навіть ***не завантажуються***, не кажучи вже про їх компіляцію. (Нагадаємо, що збірка є одиницею розгортання.)
  - Спочатку (під час завантаження) для кожного з методів створюється спеціальна "заглушка". При першому виклику метода заглушка забезпечує передачу управління *JIT*-компілятору, метод компілюється і заглушка змінюється так, щоб наступні виклики метода призводили до передачі управління отриманому машинному коду, що реалізує даний метод.

JIT-компіляція та строга типізація коду

У процесі компіляції у машинний код *CLR* забезпечує перевірку, чи є код ***строго типізованим***, тобто чи виконуються наступні вимоги:

- усі ***посилання*** є, по-перше, ***типізованими*** і, по-друге – сумісними з тими даними, які адресуються відповідними вказівниками;
- для об'єкта викликаються тільки ***правильно визначені операції***;
- посвідчення типів (“повні” імена типів з “префіксами” – іменами збірок) є коректними (“справжніми”).

Строга типізація дозволяє ізолювати об'єкти один від одного і унеможливити їх ненавмисне чи навмисне ушкодження. Це дуже важливо для ***безпеки*** коду.

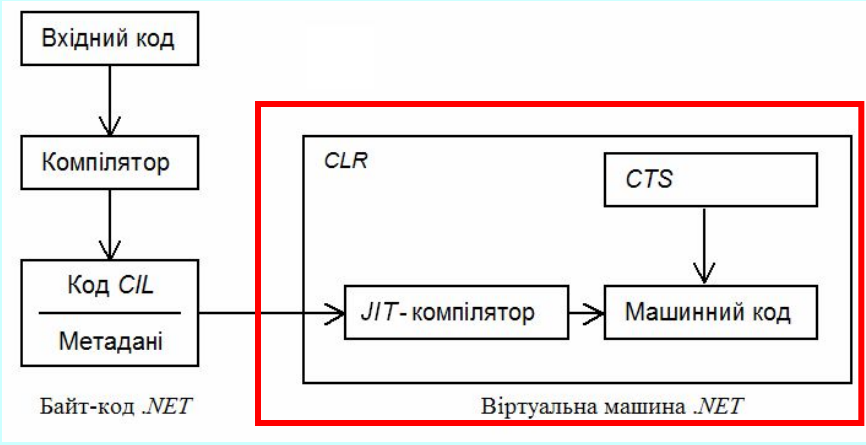
Саме строга типізація є підґрунтям застосування ***доменів***, а також підходу до реалізації автоматичного ***збирача сміття***.

Кероване виконання машинного коду. Використання служб

Основним принципом загальномовного середовища виконання *CLR* є ***концепція управління кодом***.

Загальномовне середовище виконання (віртуальна машина *.NET*) надає інфраструктуру, яка забезпечує ***кероване*** (***managed***) ***виконання*** машинного коду: під час виконання машинного (керованого) коду віртуальна машина *.NET* забезпечує наступні служби:

- ***збирання сміття***;
- ***безпеки***;
- ***взаємодії з некерованим кодом***;
- ***віддаленої взаємодії***;
- ***управління потоками***;
- ***підтримки відстеження версій***.



Управління пам'яттю. Збирання сміття

Автоматичне управління пам'яттю це одна із служб, яку "віртуальна машина" (*CLR*) надає у процесі ***керованого виконання***.

Зокрема, вирішуються ключові проблеми управління пам'яттю:

- "***витікання пам'яті***";
- "***висячі посилання***".

Підхід .*NET* наполягає на тому, щоб відмовитись від явного управління пам'яттю, натомість пропонується використовувати ***екземпляри reference типів*** (найчастіше ***класів***).

Збирання сміття (*garbage collection, GC*). Основні поняття

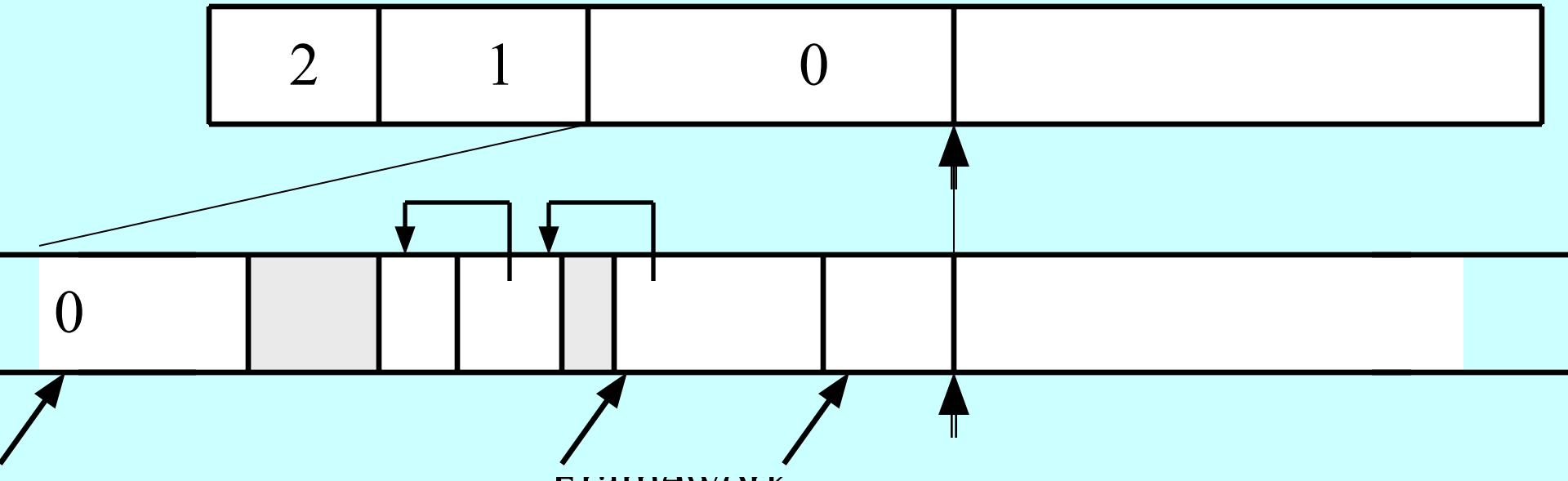
"Керована" купа – ***неперервна*** (без дефрагментацій!) область адресного простору (забезпечується висока швидкість виділення пам'яті).

"Корені". Кожен корінь або посилається на об'єкт (типізований!), який міститься у керованій купі, або має порожнє значення. (Корені пов'язані з глобальними та статичними об'єктами, локальними змінними, параметрами у стеку).

Покоління. (0-, 1-, 2-покоління).

Ущільнення пам'яті.

Метод ***Finalize***, черга фіналізації.

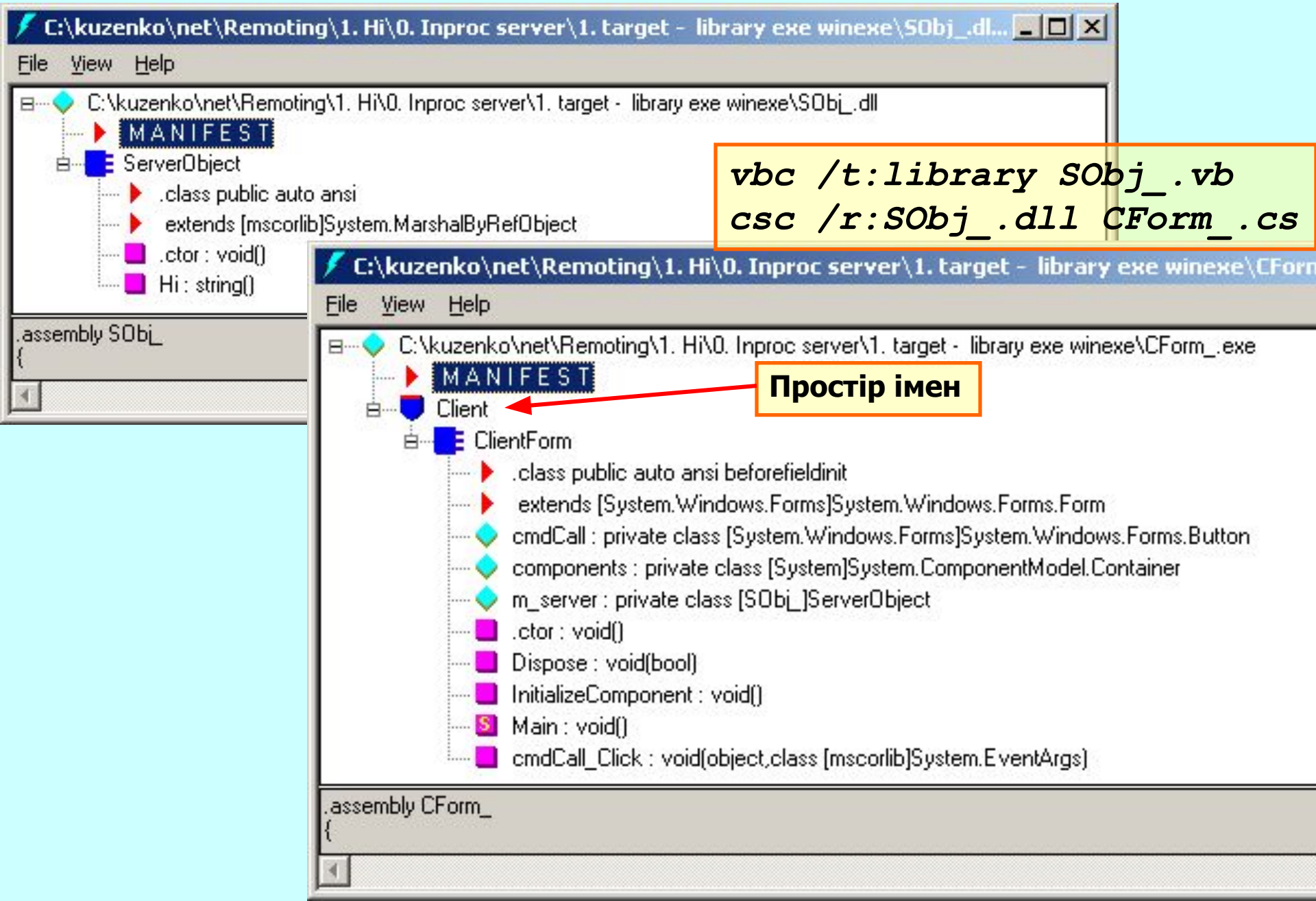


Порівняння компонентних підходів *COM* та *.NET*. Ключова розбіжність – віртуалізація (*virtualization*) контрактів *.NET*

- ***Контракти*** компонентів *COM* є "***фізичними***" ("***двійковими***"), звідки жорсткі умови міжкомпонентних викликів: точні зміщення у таблиці віртуальних методів *vtable*, точна дисципліна стека (*_stdcall*), точний формат вказівників на інтерфейс тощо. Практична необхідність використання метазасобів для ***опису контрактів***. (Два метазасоби: мова ***IDL*** та бібліотеки типів ***TLB***).
- ***Контракти*** компонентів *.NET* є ***віртуалізованими*** – не використовується "фізичний" (“двійковий”) рівень, ніяких угод про представлення у пам'яті. Контракти представляються у форматі ***метаданих*** (*metadata*). (***Прозорість метаданих***).
  - ~ Інструментальні засоби читання та генерування метаданих (генерування прозоре, автоматичне).
  - ~ Для компонентів обов'язкове (автоматичне!) “супроводження” метаданими (на відміну від *COM*).
  - ~ *CIL* абстрагований від “машинного рівня”, у ньому виклики компонентних методів посилаються на метадані з використанням звичайних імен, а не вказівників чи зміщень.

Додаток

Дизасемблер *ildasm.exe*. Перегляд *SObj_.dll, CForm_. dll*



Дизасемблер *ildasm.exe*.
Перегляд маніфесту *CForm_.exe*

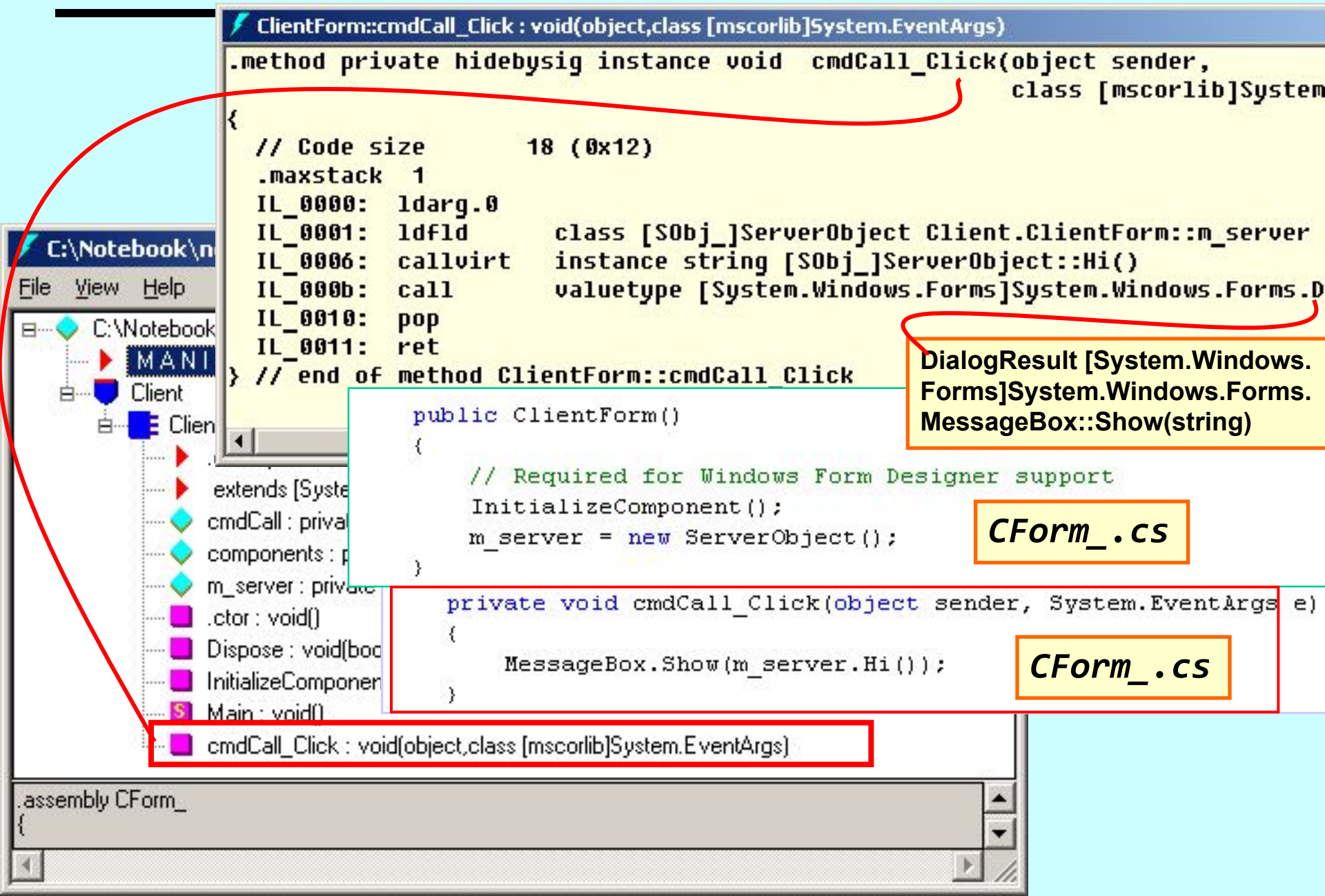
```
vbc /t:library SObj_.vb
csc /r:SObj_.dll CForm_.cs
```

MANIFEST

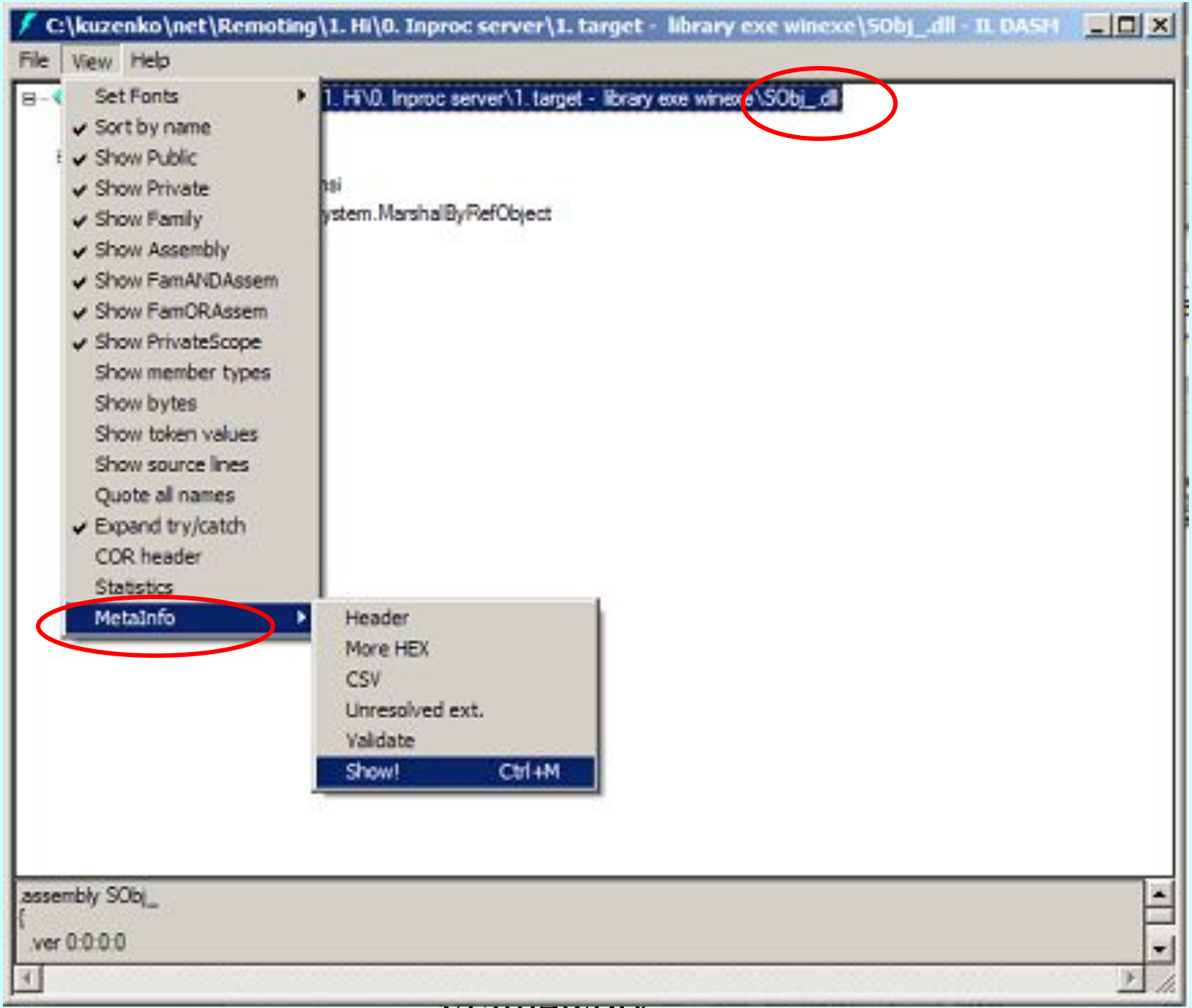
```
.assembly extern System.Windows.Forms
{
  .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )                    // .z\V.4..
  .ver 1:0:3300:0
}
.assembly extern SObj_
{
  .ver 0:0:0:0
}
.assembly extern mscorlib
{
  .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )                    // .z\V.4..
  .ver 1:0:3300:0
}
.assembly extern System
{
  .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )                    // .z\V.4..
  .ver 1:0:3300:0
}
.assembly extern System.Drawing
{
  .publickeytoken = (B0 3F 5F 7F 11 D5 0A 3A )                    // .?_....:
```

mscorlib - Multilanguage Standard Common Object Runtime Library

Дизасемблер *ildasm.exe*. Перегляд метода *cmdCall_Click* (файл *CForm_.exe*)



Дизасемблер *ildasm.exe*. Метадані (*metainfo*) *SObj_.dll* (1/4)



Дизасемблер *ildasm.exe*. Метадані (*metainfo*) *SObj_.dll* (2/4)

MetaInfo

```
Coff symbol name overhead:  0
ScopeName : SObj_.dll
MVID      : {5D639CD6-E1C9-4AF7-B32E-
=====================================
Global functions
-------------------------------------

Global fields
-------------------------------------

Global MemberRefs
-------------------------------------

TypeDef #1
-------------------------------------------------------
        TypDefName: ServerObject (02000002)
        Flags     : [Public] [AutoLayout] [Class] [AnsiClass]  (00000001)
        Extends   : 01000001 [TypeRef] System.MarshalByRefObject
        Method #1
        ---------------------------------------------------------
                MethodName: .ctor (06000001)
                Flags     : [Public] [ReuseSlot] [SpecialName] [RTSpecialName] [.ctor]  (00001806)
                RVA       : 0x00002050
                ImplFlags : [IL] [Managed]  (00000000)
                CallCnvntn: [DEFAULT]
                hasThis
                ReturnType: Void
                No arguments.

        Method #2
        ---------------------------------------------------------
                MethodName: Hi (06000002)
                Flags     : [Public] [ReuseSlot]  (00000006)
                RVA       : 0x00002058
                ImplFlags : [IL] [Managed]  (00000000)
                CallCnvntn: [DEFAULT]
                hasThis
                ReturnType: String
                No arguments.


TypeRef #1 (01000001)
```

SObj_.vb

```
Imports System
Public class ServerObject
    Inherits MarshalByRefObject
    Public Function Hi() As String
        Console.WriteLine("Hi-method invoked.")
        Return "Hi from VB. "
    End Function  'Hi
End Class   'ServerObject
```

TypeDefName

ServerObject

ServerObject *ctor*

ServerObject *Hi*

Дизасемблер *ildasm.exe*. Метадані (*metainfo*) *SObj_.dll* (3/4)

MetaInfo

```
TypeRef #1 (01000001)
----------------------------------------------------
Token:             0x01000001
ResolutionScope:   0x23000001
TypeRefName:       System.MarshalByRefObject
        MemberRef #1
        -------------------------------------------------------
                Member: (0a000001) .ctor:
                CallCnvntn: [DEFAULT]
                hasThis
                ReturnType: Void
                No arguments.

TypeRef #2 (01000002)
----------------------------------------------------
Token:             0x01000002
ResolutionScope:   0x23000001
TypeRefName:       System.Console
        MemberRef #1
        -------------------------------------------------------
                Member: (0a000002) WriteLine:
                CallCnvntn: [DEFAULT]
                ReturnType: Void
                1 Arguments
                        Argument #1:  String

Signature #1 (0x11000001)
----------------------------------------------------
                CallCnvntn: [LOCALSIG]
                1 Arguments
                        Argument #1:  String

Assembly
----------------------------------------------------
        Token: 0x20000001
        Name : SObj_
        Public Key    :
        Hash Algorithm : 0x00008004
        Major Version: 0x00000000
        Minor Version: 0x00000000
        Build Number: 0x00000000
        Revision Number: 0x00000000
        Locale: <null>
```

TypeRef

MarshalByRefObject

System.Console

System.Console

WriteLine

Assembly

*SObj_*

Дизасемблер *ildasm.exe*. Метадані (*metainfo*) *SObj_.dll* (4/4)

MetaInfo

```
Assembly
-------------------------------------------------------
        Token: 0x20000001
        Name : SObj_
        Public Key    :
        Hash Algorithm : 0x00008004
        Major Version: 0x00000000
        Minor Version: 0x00000000
        Build Number: 0x00000000
        Revision Number: 0x00000000
        Locale: <null>
        Flags : [SideBySideCompatible]  (00000000)

AssemblyRef #1
-------------------------------------------------------
        Token: 0x23000001
        Public Key or Token: b7 7a 5c 56 19 34 e0 89
        Name: mscorlib
        Major Version: 0x00000001
        Minor Version: 0x00000000
        Build Number: 0x00000ce4
        Revision Number: 0x00000000
        Locale: <null>
        HashValue Blob:
        Flags: [none] (00000000)

AssemblyRef #2
-------------------------------------------------------
        Token: 0x23000002
        Public Key or Token: b0 3f 5f 7f 11 d5 0a 3a
        Name: Microsoft.VisualBasic
        Major Version: 0x00000007
        Minor Version: 0x00000000
        Build Number: 0x00000ce4
        Revision Number: 0x00000000
        Locale: <null>
        HashValue Blob:
        Flags: [none] (00000000)


User Strings
-------------------------------------------------------
70000001 : (18) L"Hi-method invoked."
70000027 : (12) L"Hi from VB. "
```

*SObj_*

mscorlib

mscorlib - Multilanguage Standard Common Object Runtime Library

Самоопис керованого коду .*NET*. Механізм рефлексії

Керований код .*NET* містить ***метадані*** (***metadata***), які представляють собою ***опис*** самого ***коду***, зокрема опис усіх використаних у коді ***типів***.

.*NET Framework*, спираючись на метадані, забезпечує підтримку ***механізму рефлексії*** (***reflection***). У просторі імен ***System.Reflection.Emit*** надаються функції, які дозволяють отримувати дані про типи, що використані у збірці. Далі, за отриманими даними про типи, .*NET Framework* дозволяє ***динамічно*** (під час виконання програми) ***створювати об'єкти*** – екземпляри таких типів.