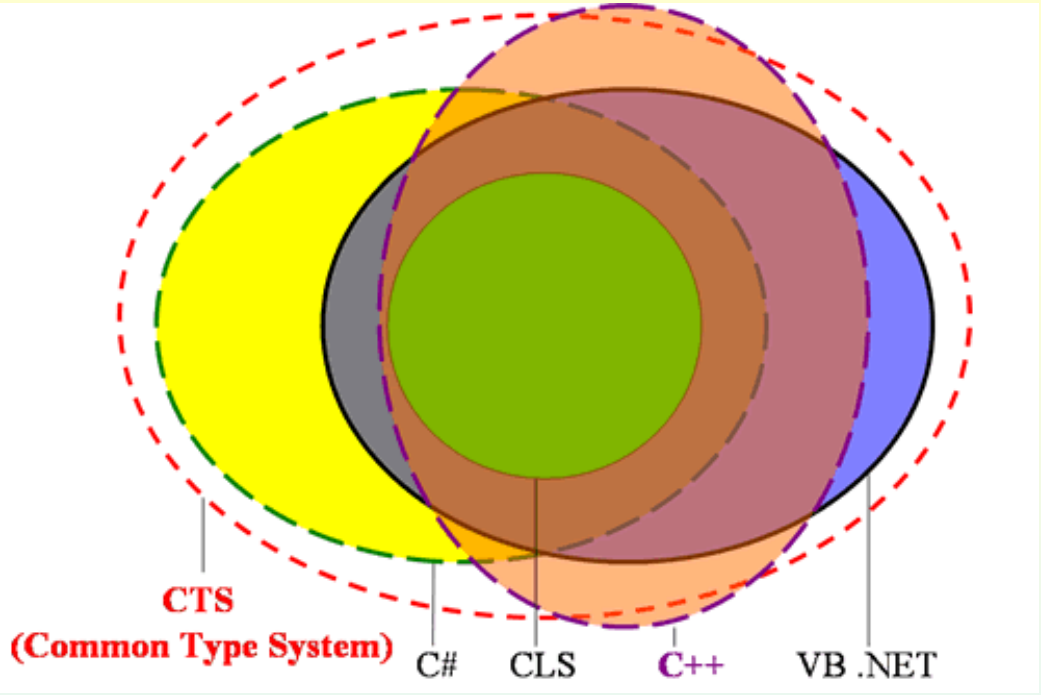
Лекція 2.
CTS, вбудовані типи C#, приведення типів

Клакович Л.М.

1. Common Type System і C#
2. Типи-значення та типи-посилання
3. Вбудовані типи C#
4. Перетворення типів. Неявні та явні перетворення
5. Простори назв

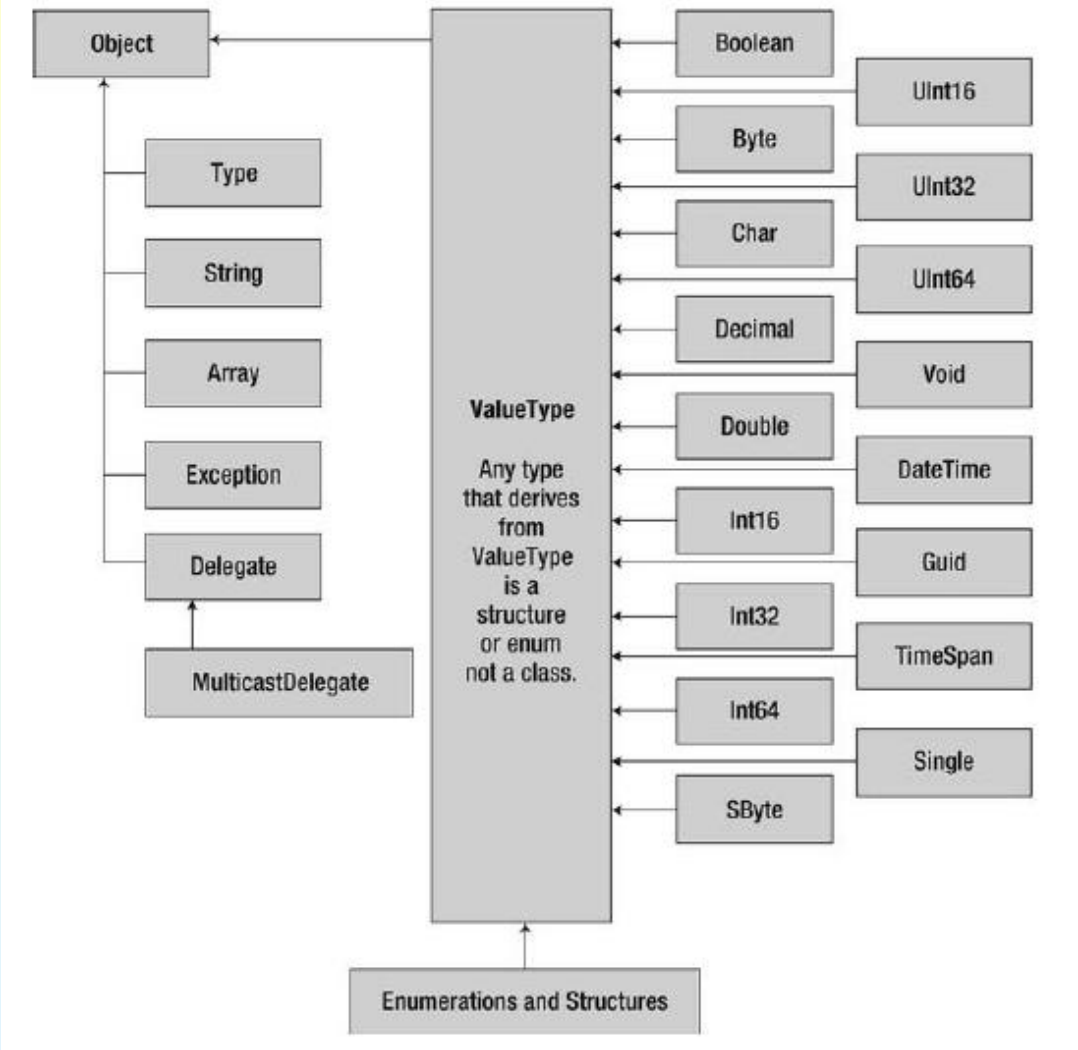
Common Type System
Загальна система типів



• **CTS** визначає спосіб оголошення, використання і управління типами в середовищі CLR, і є важливою складовою частиною підтримки міжмовної інтеграції
• **CLS** - Common Language Specification (загальномовна специфікація) – підмножина CTS, яку підтримують всі .NET мови.

- CST виконує такі функції:
  - Забезпечує міжмовну інтеграцію, безпеку типів і високопродуктивне виконання коду.
  - Надає об'єктно-орієнтовану модель.
  - Визначає правила, яких необхідно дотримуватися в мові. Ці правила допомагають забезпечити взаємодію об'єктів, написаних на різних мовах.
  - Надає бібліотеку, яка містить типи-примітиви (наприклад, Boolean, Byte, Char, Int32 і UInt64).

Клас object. Ієрархія типів



Common Type System і C#

- ***System.Object*** – базовий клас
- Дві категорії типів стосовно виділення пам'яті та функціонування :
  - **типи-значення**: *примітиви, структури, переліки* – об'єкти безпосередньо містять значення. Наслідуються від ValueType

```
int i = 32; // змінна типу System.Int32 у стеку
```

  - **типи-посилання:** *класи, інтерфейси, делегати, масиви* – опосередкований доступ до об'єктів у купі (*heap*) або ***null***

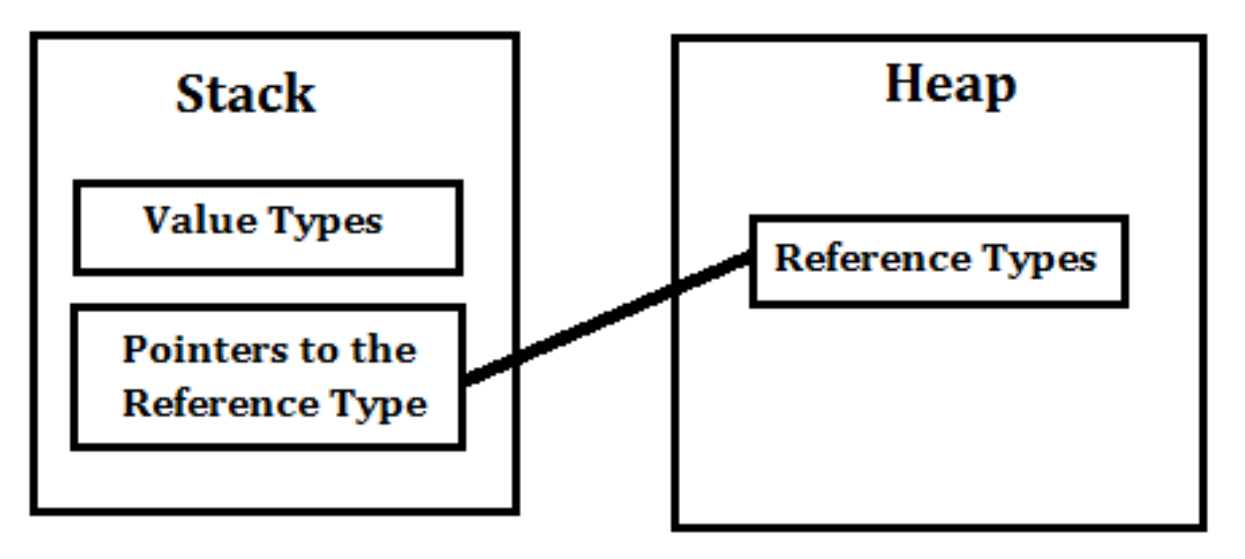
```
Student stud = new Student();

// посилання на розміщений в купі об'єкт, проініціалізований рядком символів
string s = "Hello, World";
```

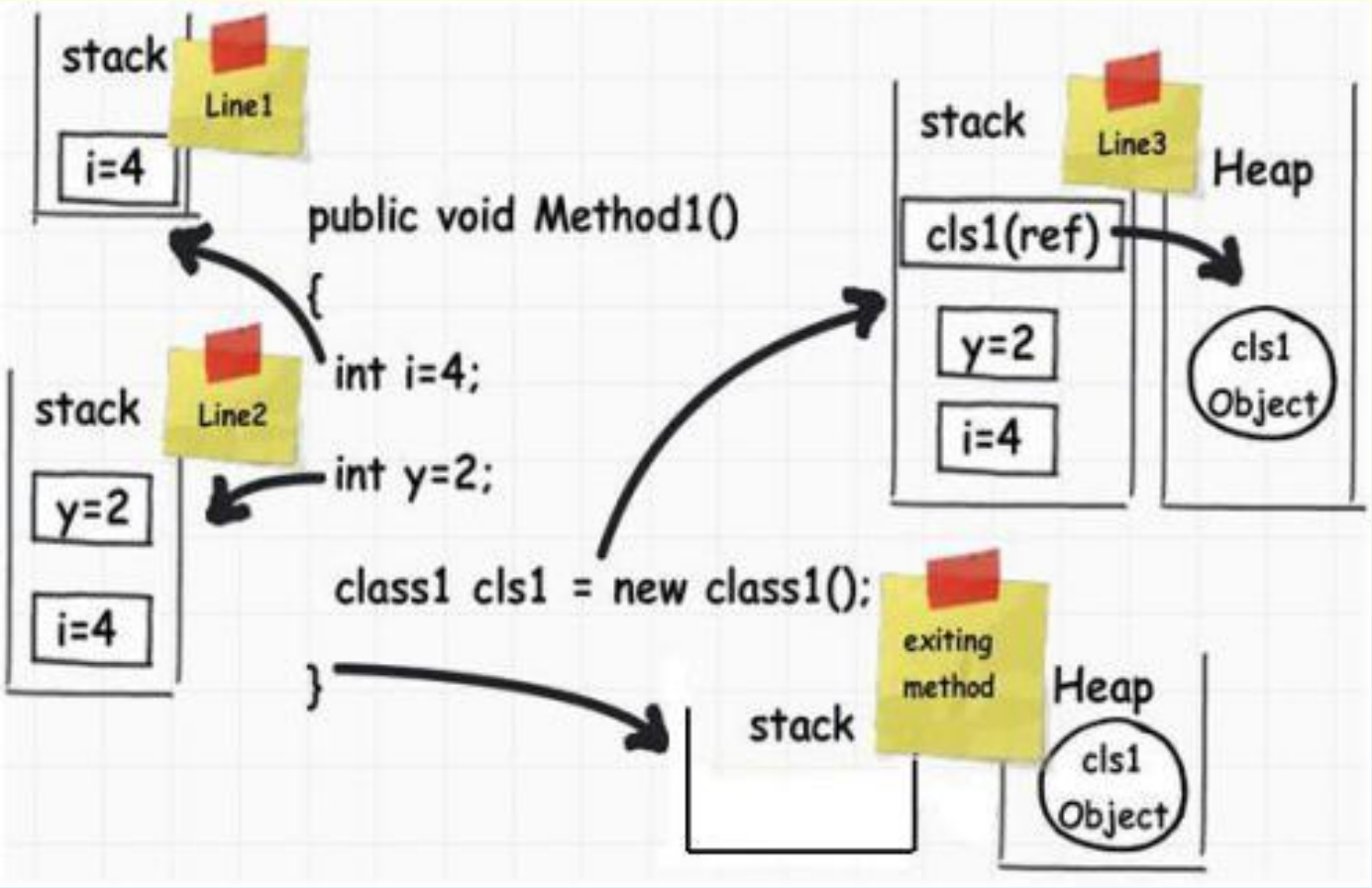
- *boxing-unboxing, Пакування-розпакування типів*

Типи-значення та типи-посилання

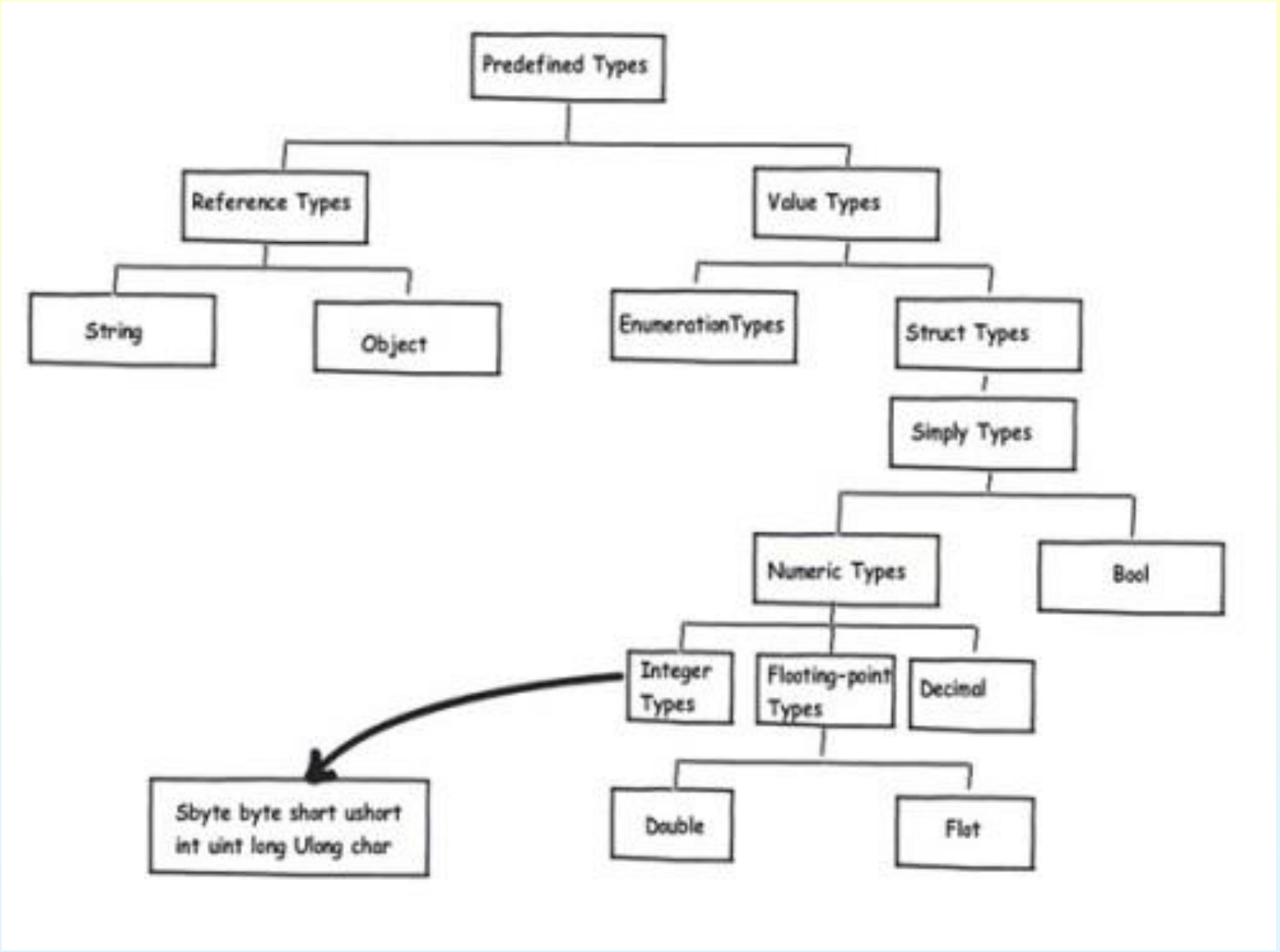
- **Типи-значення (Value type)** містять дані в стеку, змінна ототожнюється з даними. Локальні змінні, параметри-методів, struct, enum, вбудовані
- **Типи-посилання (Reference type):** змінна розміщена в стеку, і містить адресу пам'яті в купі, де знаходяться дані (аналог вказівника C++). Великі об'єкти, масиви, Типи створені class



Типи-значення та типи-посилання



Вбудовані типи C#

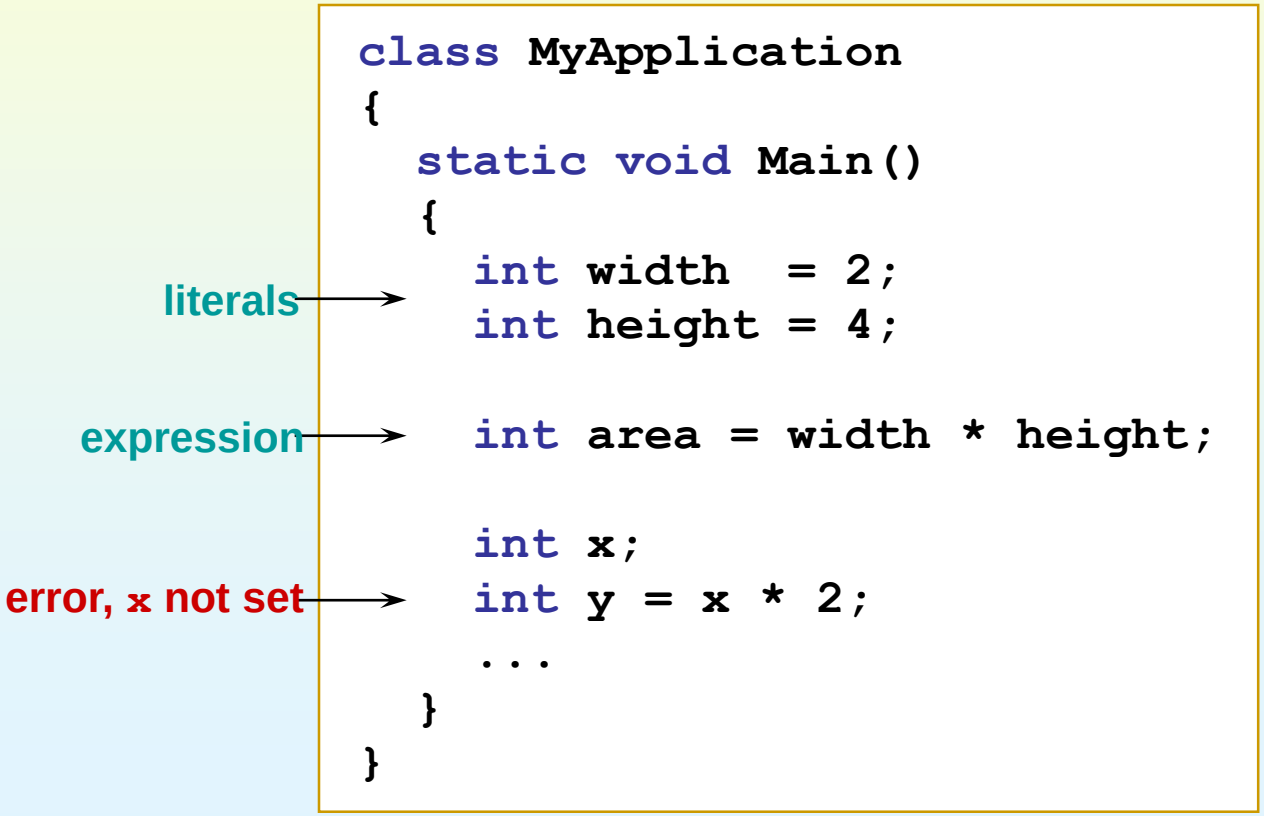


Вбудовані типи C#

CTS Type Name	C# Alias	Description	Range
System.Object	*object*	Base class for all CTS	
System.SByte	*sbyte*	Signed 8-bit byte	[-128, 127]
System.Byte	*byte*	Unsigned 8-bit byte	[0, 255]
System.Int16	*short*	Signed 16-bit value	[-32768, 32767]
System.UInt16	*ushort*	Unsigned 16-bit value	[0, 65535]
System.Int32	*int*	Signed 32-bit value	[-2^{31}, $2^{31}-1$]
System.UInt32	*uint*	Unsigned 32-bit value	[0, 4 294 967 295]
System.Int64	*long*	Signed 64-bit value	[-2^{63}, $2^{63}-1$]
System.UInt64	*ulong*	Unsigned 64-bit value	[0, 18 446 744 073 709 551 615]
System.Single	*float*	IEEE 32-bit float	[$1.4\text{x}10^{-45}$, $3.4\text{x}10^{38}$] 6-7
System.Double	*double*	IEEE 64-bit float	[$5.0\text{x}10^{-324}$, $1.7\text{x}10^{308}$] 15-16
System.Decimal	*decimal*	128-bit data	[$1.0\text{x}10^{-28}$, $7.9\text{x}10^{28}$] 28-29
System.Boolean	*bool*	Boolean value	*true* / *false*
System.Char	*char*	16-bit Unicode char	[u+0000, u+ffff] ‘\u0058’
System.String	*string*	String	

Визначення та ініціалізація локальних змінних

Помилка компіляції при використанні локальної змінної без попередньої ініціалізації



Визначення та ініціалізація змінних вбудованих типів

• Можна **утворювати змінні вбудованих типів використовуючи new –** дефолтний конструктор автоматично проініціалізує змінну відповідними значеннями:

```
int width  = new int();         //width=0
double radius = new double(); //radius=0.0
```

- **bool** - false.
- **Числові типи** - 0 (або 0.0).
- **char** - один порожній символ.
- **BigInteger** - 0. (System.Numerics.dll)
- **DateTime** - 1/1/0001 12:00:00 AM.
- **Object references** (включаючи **strings**) - null.

Функціонування вбудованих типів

```
char c1 = '1';
char ca = 'a';
Console.WriteLine(c1.Equals('1')); //True
Console.WriteLine(c1.GetType());     //System.Char
Console.WriteLine(char.IsDigit(c1)); //True
Console.WriteLine(char.IsLetter('6')); //False
Console.WriteLine(char.IsPunctuation(','));//True
Console.WriteLine('a'.Equals('A'));  //False
```

Неявне визначення локальних змінних

• **Можна оголошувати змінні типу var з одночасною ініціалізацією.** Тип цієї змінної буде виведено з типу ініціалізатора.

```
static void DeclareExplicitVars()
{
  // Explicitly typed local variables
  // are declared as follows:
  // dataType variableName = initialValue;
  int myInt = 0;
  bool myBool = true;
  string myString = "Time, marches on...";
}
```

```
static void DeclareImplicitVars()
{
  // Implicitly typed local variables
  // are declared as follows:
  // var variableName = initialValue;
  var myInt = 0;
  var myBool = true;
  var myString = "Time, marches on...";
}
```

```
  // Print out the underlying type.
  Console.WriteLine("myInt is a: {0}", myInt.GetType().Name);
  Console.WriteLine("myBool is a: {0}", myBool.GetType().Name);
  Console.WriteLine("myString is a: {0}", myString.GetType().Name);
}
```

• **Не можна використовувати var** для визначення полів класу, параметрів методів чи об’єктів з ініціалізацією null:

```
class ThisWillNeverCompile
{
  // Error! var cannot be used as field data!
  private var myInt = 10;

  // Error! var cannot be used as a return value
  // or parameter type!
  public var MyMethod(var x, var y){}
}
```

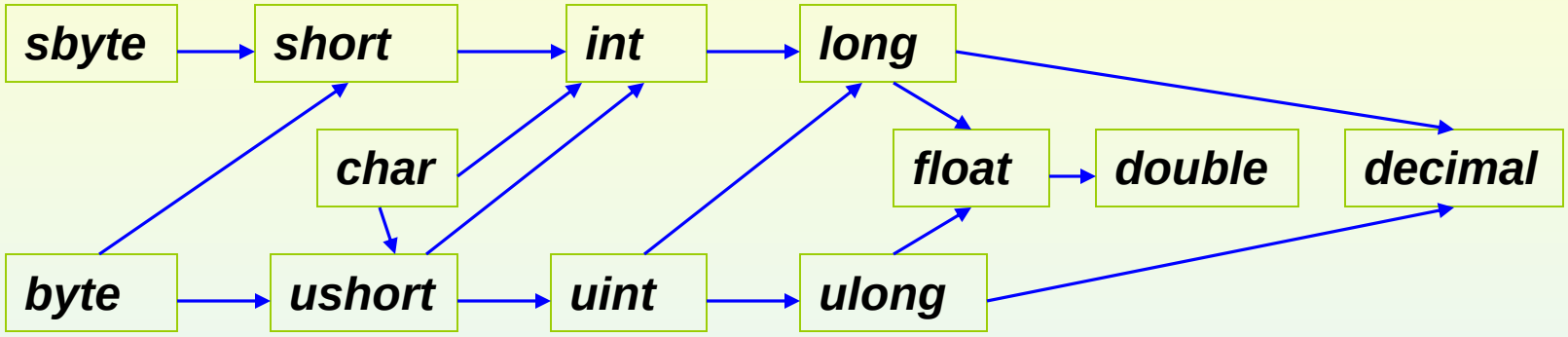
```
// Error! Can't assign null as initial value!
var myObj = null;

// OK, if SportsCar is a reference type!
var myCar = new SportsCar();
myCar = null;
```

Перетворення типів

Неявні перетворення (допустиме використання замість очікуваного типу):

а) просування вбудованих типів




```
uint a = 10U;
ulong b = a;

// void f(ulong x){…}
f(a);
```

Неявні перетворення

b) пакування (boxing) вбудованих типів (приведення типу-значення до типу-посилання)

```
int foo = 42;            // value type
object bar = foo;        // foo is boxed to bar
```

c) підстановка замість об’єкта базового класу (upcasting)

```
// class B{…}   class D: B{…}    void f(B x){…}

        B x = new B();      f( x );
        D y = new D();      f( y );
```

Перетворення типів

Явні перетворення (контроль наслідків некоректних перетворень)

a) приведення (casting) вбудованих типів

```
long a= 10L;
int b = (int) a;
```

b) розпакування (unboxing) вбудованих типів

```
int foo = 42;          //   Value type
object bar = foo;      //   foo is boxed to bar
int foo2 = (int)bar;   //   Unboxed back to int
```

Простори назв

- **Визначення простору назв**
  - послідовно в одному модулі
  - вкладені з необхідною глибиною

namespace *ідентифікатор*
{
визначення просторів назв
визначення типів
}

```
using  System;
namespace A
{
  namespace A.B
  {
   class X{public int mx; …}
    }
   class Y{public int my; …}
}
namespace C
{
  class Z{public int mz; …}
}
```

- Назва простору імен повинна включати назву компанії, назву технології і опційно feature, design

```
CompanyName.TechnologyName[.Feature][.Design]

LNU.PetsShop.Design
```

Простори назв

- **Використання простору назв**
  - **using** перед просторами назв стосується всіх класів у модулі компіляції
  - **using** всередині простору назв стосується класів лише цього простору

using *специфікований ід простору назв* **;**

- **Використання аліясів (псевдонімів)**

using *аліяс* = *специфікований ід простору назв* **;**
using *аліяс* = *специфікований ід класу* **;**

```
using  A;
using  A.B;
using  D=A.B.X;
…
{ … X.mx    … Y.my
     A.B.X.mx… D.mx
} …
```