

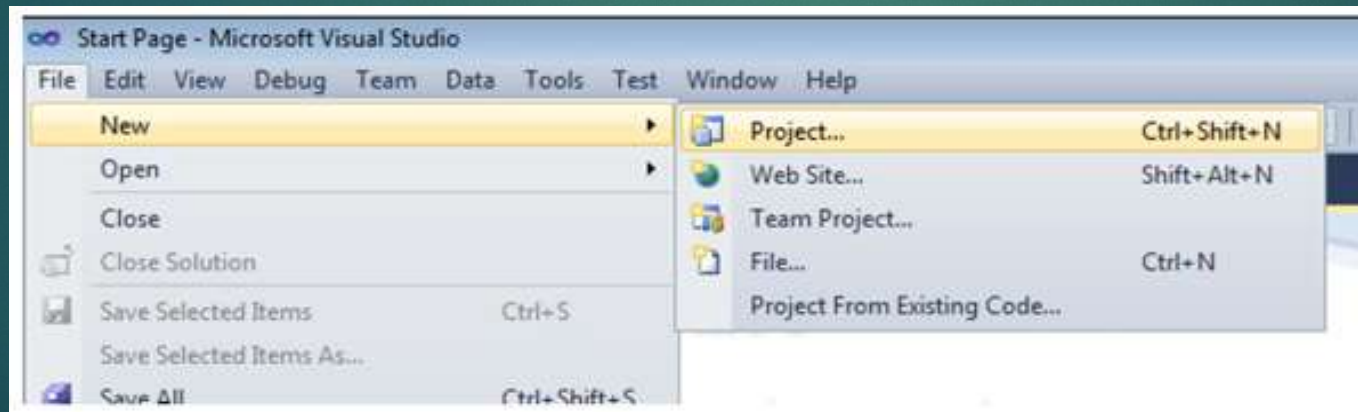


C# Căn bản (Phần 1)

TS. TRẦN ANH TUẤN

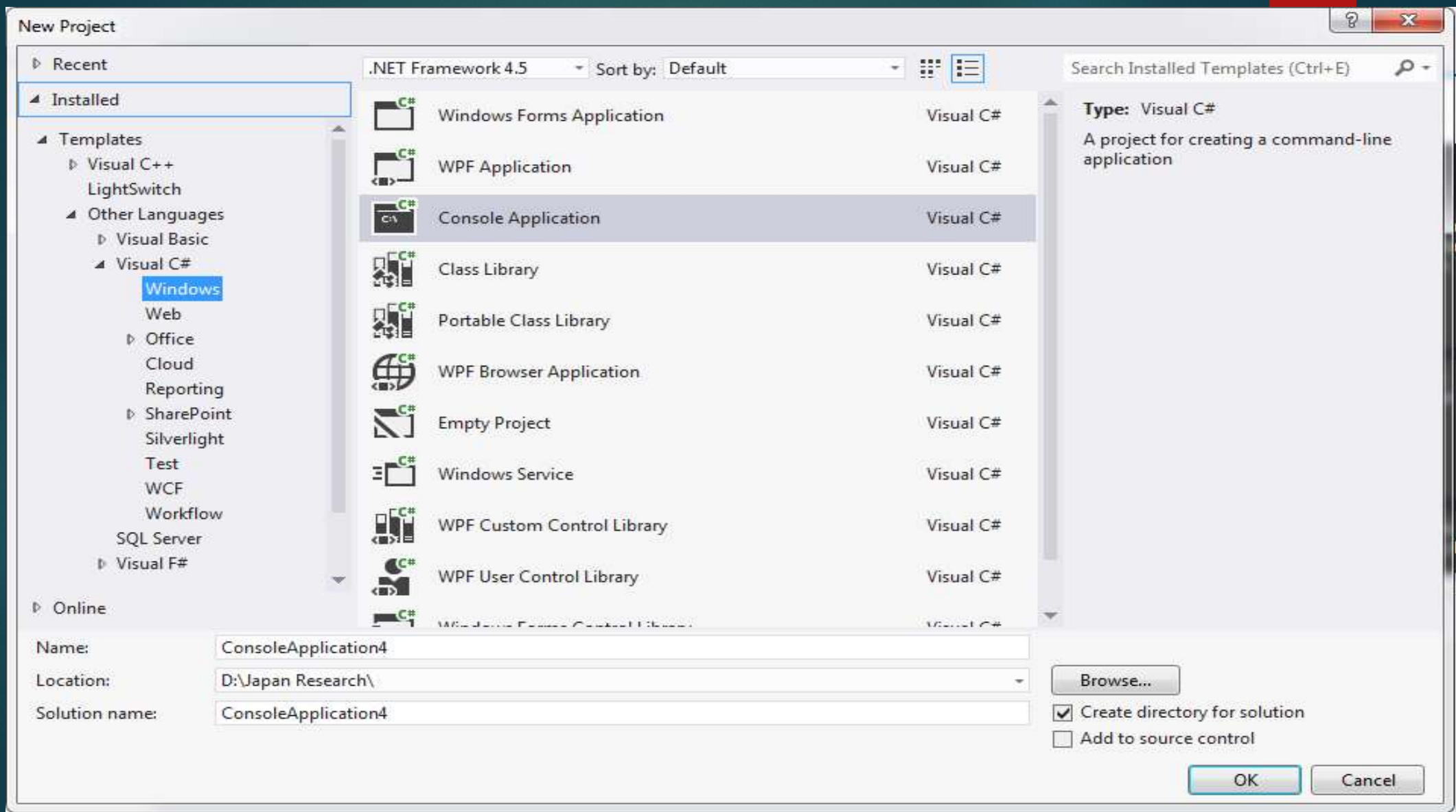
Tạo Project mới

- ▶ Mở Microsoft Visual Studio sau đó chọn File > New > Project
- ▶ Trong cửa sổ New Project, chọn ngôn ngữ lập trình C#
- ▶ Trong ngôn ngữ lập trình C#, chọn ứng dụng Windows



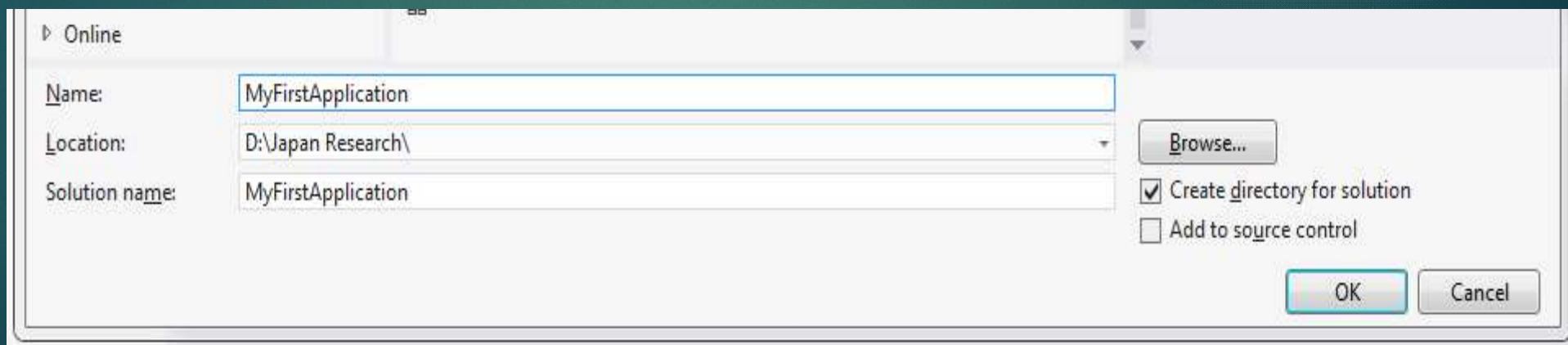
Tạo Project mới

- ▶ Chúng ta sẽ thấy các ứng dụng dạng :
 - ▶ Window Form Application : Tạo các ứng dụng chạy độc lập truyền thống
 - ▶ WPF Application : Tạo các ứng dụng desktop hướng thương mại bắt mắt
 - ▶ Console Application : Tạo các ứng dụng console
 - ▶ Class Library : Tạo lớp hay các thành phần có thể dùng cho project khác
 - ▶ Window Service: Tạo các ứng dụng dịch vụ cho Window



Tạo ứng dụng Console



- ▶ Sau khi chọn Console Application, chúng ta nhập tên project > chọn vị trí > Chọn tên solution
- ▶ Mặc định thì tên Location sẽ trùng với tên project (Nếu solution chỉ có 1 project)
 - ▶ Solution là một nhóm của một hay nhiều project cùng chung để tạo một ứng dụng.
 - ▶ Project được sử dụng trong một solution để quản lý, xây dựng và debug các phần mà tạo nên ứng dụng một cách hợp lý nhất.









The screenshot shows the 'Online' dialog box in Visual Studio. The 'Name' field is set to 'MyFirstApplication'. The 'Location' field is set to 'D:\Japan Research\'. The 'Solution name' field is also set to 'MyFirstApplication'. There is a 'Browse...' button next to the 'Location' field. Below the fields, there are two checkboxes: 'Create directory for solution' (checked) and 'Add to source control' (unchecked). At the bottom right, there are 'OK' and 'Cancel' buttons.

Online	
Name:	MyFirstApplication
Location:	D:\Japan Research\
Solution name:	MyFirstApplication
	<input checked="" type="checkbox"/> Create directory for solution
	<input type="checkbox"/> Add to source control
	OK Cancel

Ứng dụng console

Name	Date modified	Type	Size
 MyFirstApplication	9/18/2016 7:54 AM	File folder	
 MyFirstApplication.sln	9/18/2016 7:54 AM	Microsoft Visual S...	

Name	Date modified	Type	Size
 bin	9/18/2016 7:54 AM	File folder	
 obj	9/18/2016 7:54 AM	File folder	
 Properties	9/18/2016 7:54 AM	File folder	
 App.config	9/18/2016 7:54 AM	XML Configuratio...	
 MyFirstApplication.csproj	9/18/2016 7:54 AM	Visual C# Project f...	
 Program.cs	9/18/2016 7:54 AM	Visual C# Source f...	

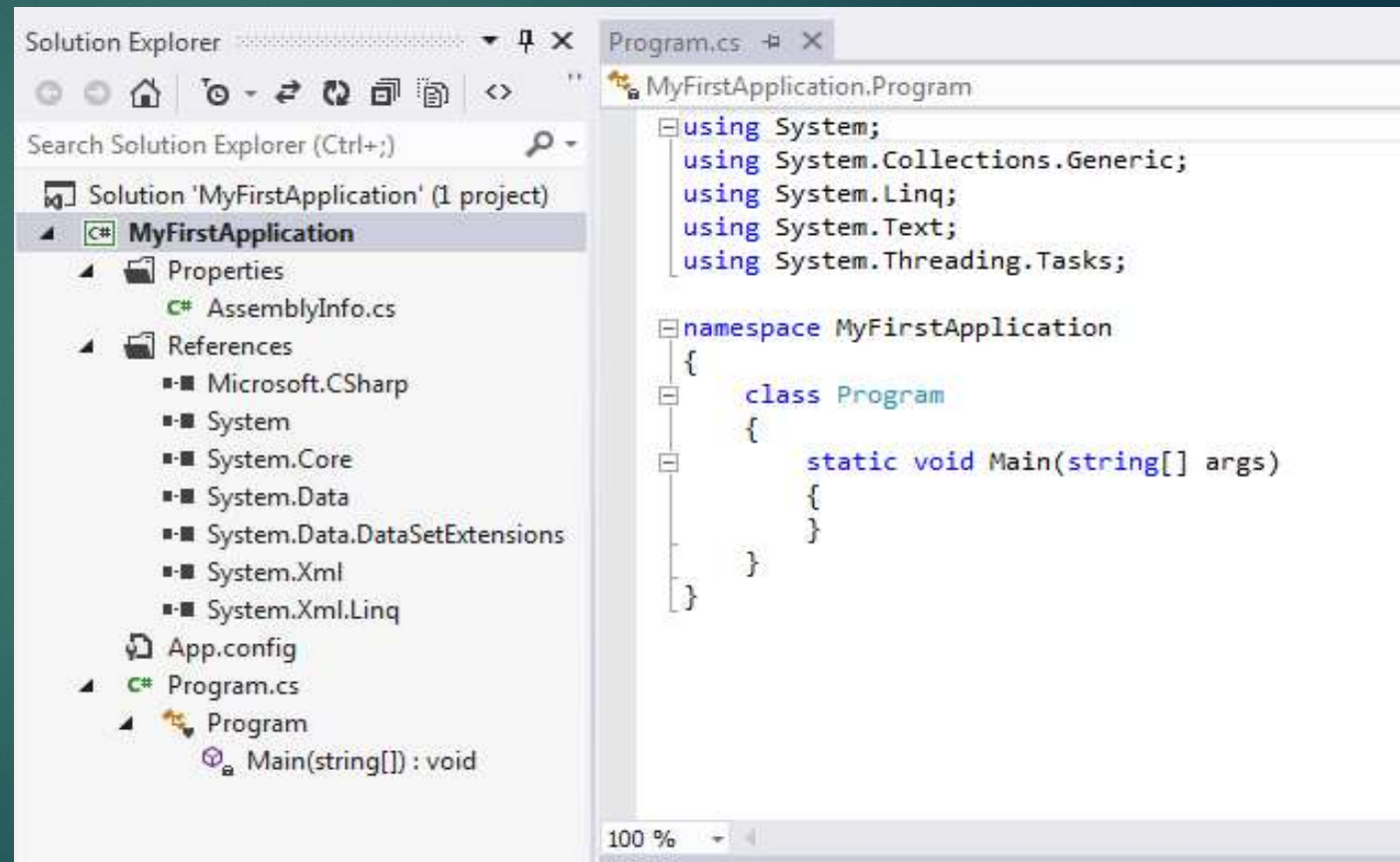
Ứng dụng console

- ▶ Tập tin sln : tập tin Solution cho ứng dụng
- ▶ Tập tin csproj : tập tin project cho ứng dụng
- ▶ Tập tin cs: tập tin chính chứa các dòng code csharp
- ▶ Bin folder : Là thư mục chứa các assemblies (.dll files), các thành phần hay những đoạn code mà được reference đến
- ▶ Obj folder: là thư mục chứa các object file tạm cho quá trình biên dịch
- ▶ Properties: là thư mục chứa các thông tin Assembly của ứng dụng (tên, phiên bản, mô tả v.v...)

Ứng dụng Console

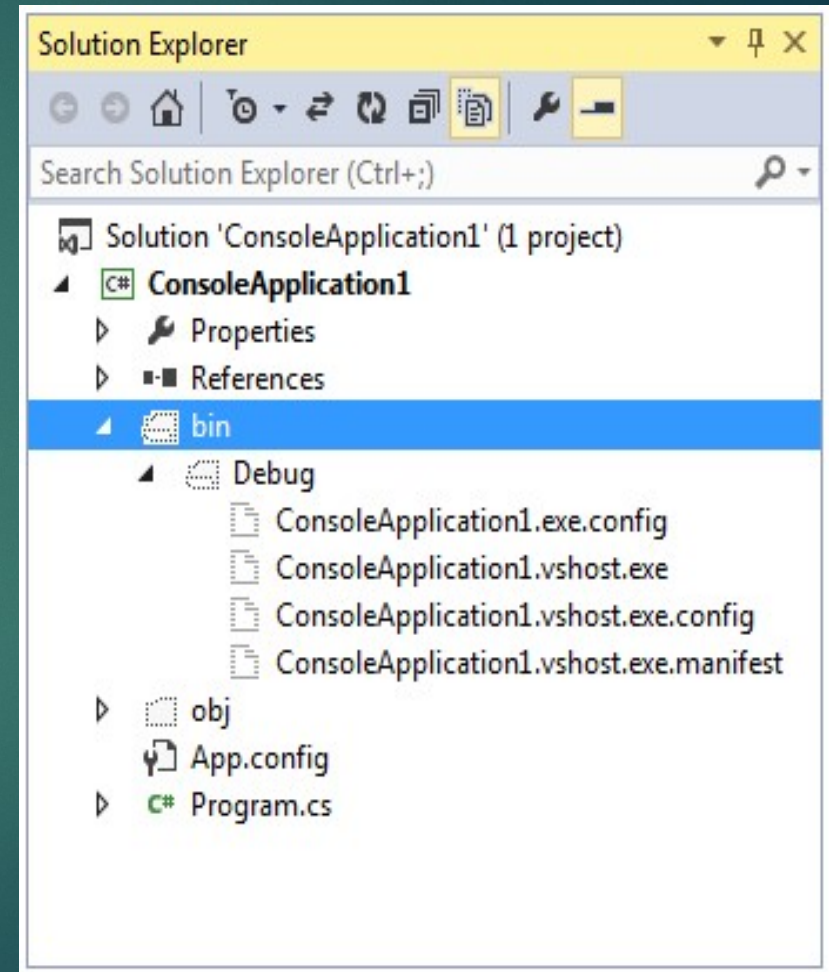
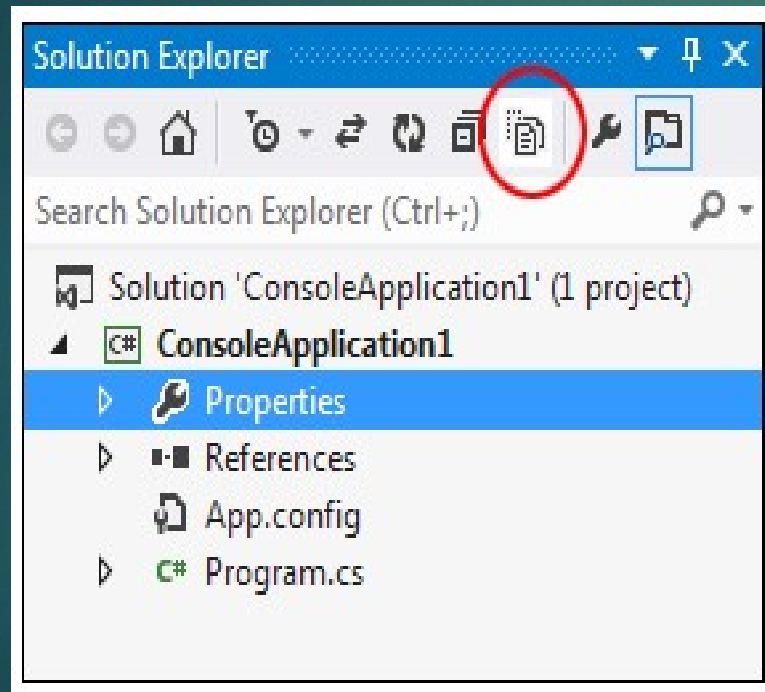
► Solution Explorer:

- Chứa tất cả các file, các reference, assembly của một ứng dụng. Giúp cho việc dễ dàng tìm kiếm và quản lý. Có thể mở tất bằng các chọn View > Solution Explorer)



Ứng dụng Console

- ▶ Nhấn vào nút Show All Files : có thể thấy thư mục bin và obj của ứng dụng



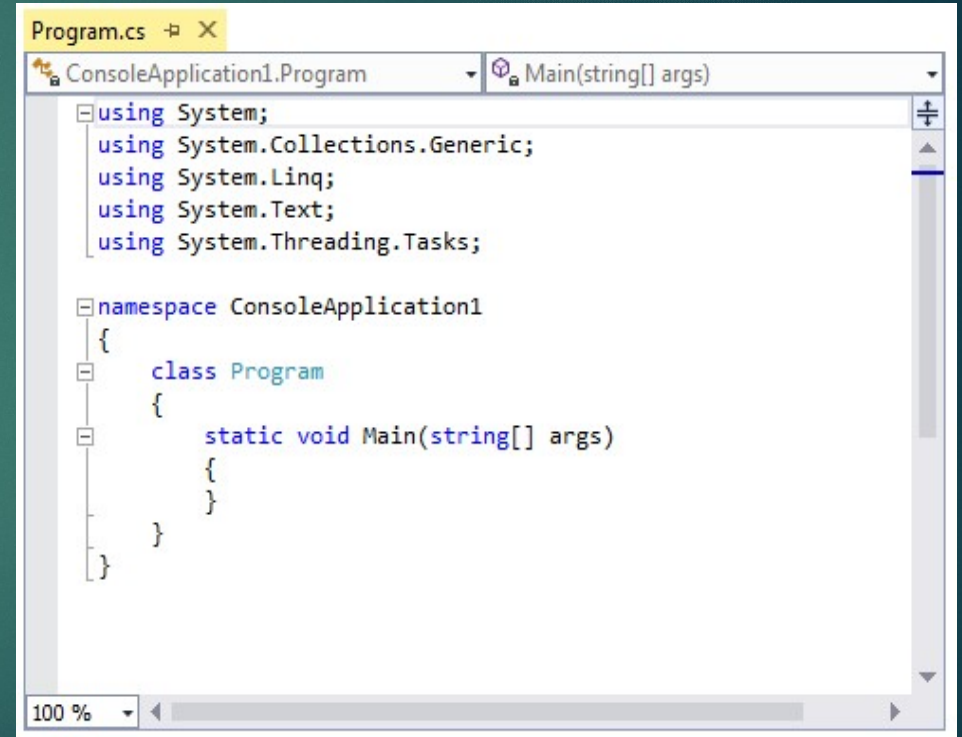
AssemblyInfo.cs

- ▶ Chứa các thông tin về ứng dụng

```
// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("MyFirstApplication")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("Microsoft")]
[assembly: AssemblyProduct("MyFirstApplication")]
[assembly: AssemblyCopyright("Copyright © Microsoft 2016")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]
|
```

Program.cs

- ▶ Chứa các dòng code chính của chương trình
- ▶ Chứa hàm Main tự động thực thi khi chạy chương trình
- ▶ Thứ tự code chương trình :
 - ▶ Using : để sử dụng các namespace chuẩn của .NET
 - ▶ Namespace: tạo ra namespace ứng dụng
 - ▶ Class: Tạo ra lớp của chương trình
 - ▶ Main : Hàm chính thực thi trong ứng dụng Console



```
Program.cs
ConsoleApplication1.Program
Main(string[] args)

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```


Nhập và xuất trong Console

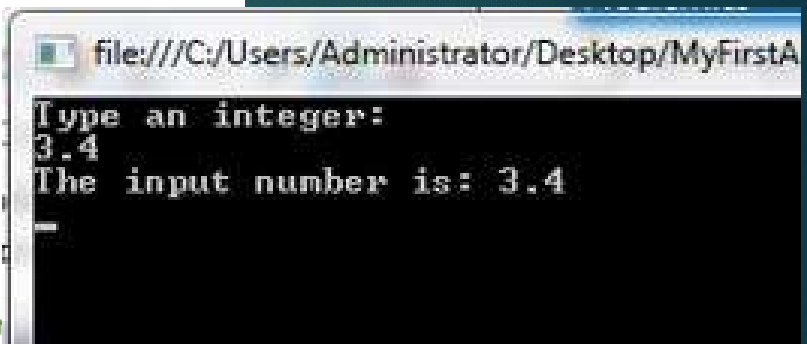
- ▶ Console.WriteLine dùng để xuất một chuỗi ra màn hình Console
- ▶ Console.ReadLine dùng để đọc một chuỗi từ màn hình Console
- ▶ Không có nhập trực tiếp số mà phải nhập chuỗi rồi dùng TryParse để chuyển thành số (Ví dụ : Int.TryParse(chuoi, out so) hay so = int.Parse(chuoi))
- ▶ Tham số để xuất ra màn hình sẽ theo thứ tự {0}, {1}, v.v...

Example

```
string s1 = Console.ReadLine();  
string s2 = Console.ReadLine();  
  
int i = int.Parse(s1);  
int j = int.Parse(s2);  
  
Console.WriteLine("{0} plus {1} equals {2}", i, j, i + j);
```

Nhập và xuất trong Console

```
namespace MyFirstApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Type an integer:");
            string line = Console.ReadLine(); // Read string from console
            float value;
            if (float.TryParse(line, out value)) // Try to parse the string as a float
            {
                Console.WriteLine("The input number is: {0}", value);
            }
            else
            {
                Console.WriteLine("Not an integer!");
            }
            Console.ReadLine();
        }
    }
}
```



```
file:///C:/Users/Administrator/Desktop/MyFirstA
Type an integer:
3.4
The input number is: 3.4
```

100 %

Namespaces

- ▶ Một namespaces là một cách để đóng gói một tập các tên (biến, hàm, lớp) để tách biệt với một namespaces khác.
- ▶ Hai đối tượng (biến, hàm, hay lớp) có thể trùng tên nếu chúng thuộc hai namespaces khác nha

```
namespace namespace_name  
{  
    // code declarations  
}
```


Namespaces

```
using System;
namespace first_space
{
    class namespace_cl
    {
        public void func()
        {
            Console.WriteLine("Inside first_space");
        }
    }
}

namespace second_space
{
    class namespace_cl
    {
        public void func()
        {
            Console.WriteLine("Inside second_space");
        }
    }
}
```

```
class TestClass
{
    static void Main(string[] args)
    {
        first_space.namespace_cl fc = new first_space.namespace_cl();
        second_space.namespace_cl sc = new second_space.namespace_cl();
        fc.func();
        sc.func();
        Console.ReadKey();
    }
}
```

Sử dụng lớp

- ▶ C# là ngôn ngữ lập trình hướng đối tượng
- ▶ Một chương trình sẽ gồm nhiều đối tượng tương tác với nhau bằng các phương thức của lớp
- ▶ Ví dụ lớp Rectangle sẽ có hai thuộc tính là length và width và hai phương thức là tính diện tích và hiển thị thông tin

Sử dụng lớp

```
static void Main(string[] args)
{
    Rectangle r = new Rectangle();
    r.Acceptdetails();
    r.Display();
    Console.ReadLine();
}
```

```
Length: 4.5
Width: 3.5
Area: 15.75
```

```
class Rectangle
{
    // member variables
    double length;
    double width;
    public void Acceptdetails()
    {
        length = 4.5;
        width = 3.5;
    }

    public double GetArea()
    {
        return length * width;
    }

    public void Display()
    {
        Console.WriteLine("Length: {0}", length);
        Console.WriteLine("Width: {0}", width);
        Console.WriteLine("Area: {0}", GetArea());
    }
}
```


Kế thừa trong lớp

- ▶ Kế thừa cho phép một lớp con có thể
- ▶ dùng dữ liệu và thao tác của lớp cha

```
static void Main(string[] args)
{
    Rectangle Rect = new Rectangle();

    Rect.setWidth(5);
    Rect.setHeight(7);

    // Print the area of the object.
    Console.WriteLine("Total area: {0}", Rect.getArea());
    Console.ReadKey();
}
```

```
class Shape
{
    public void setWidth(int w)
    {
        width = w;
    }
    public void setHeight(int h)
    {
        height = h;
    }
    protected int width;
    protected int height;
}
```

```
// Derived class
class Rectangle: Shape
{
    public int getArea()
    {
        return (width * height);
    }
}
```

Kế thừa trong lớp

```
// Derived class
class Rectangle : Shape, PaintCost
{
    public int getArea()
    {
        return (width * height);
    }
    public int getCost(int area)
    {
        return area * 70;
    }
}
```

```
class Shape
{
    public void setWidth(int w)
    {
        width = w;
    }
    public void setHeight(int h)
    {
        height = h;
    }
    protected int width;
    protected int height;
}

// Base class PaintCost
public interface PaintCost
{
    int getCost(int area);
}
```

Kế thừa trong lớp

```
static void Main(string[] args)
{
    Rectangle Rect = new Rectangle();
    int area;
    Rect.setWidth(5);
    Rect.setHeight(7);
    area = Rect.getArea();

    // Print the area of the object.
    Console.WriteLine("Total area: {0}", Rect.getArea());
    Console.WriteLine("Total paint cost: ${0}" , Rect.getCost(area));
    Console.ReadKey();
}
```


Đa hình trong lớp

- ▶ Đa hình cho phép chúng ta khai báo lớp con có cùng phương thức với lớp cha
- ▶ Ở lớp cha khai báo phương thức là virtual, còn ở lớp con khai báo là override
- ▶ Nếu phương thức ở lớp con có thì sẽ được gọi, còn không sẽ gọi phương thức của lớp cha

Đa hình trong lớp

```
class Shape
{
    protected int width, height;
    public Shape( int a=0, int b=0)
    {
        width = a;
        height = b;
    }
    public virtual int area()
    {
        Console.WriteLine("Parent class area :");
        return 0;
    }
}
```

```
class Rectangle: Shape
{
    public Rectangle( int a=0, int b=0): base(a, b)
    {
    }
    public override int area ()
    {
        Console.WriteLine("Rectangle class area :");
        return (width * height);
    }
}
class Triangle: Shape
{
    public Triangle(int a = 0, int b = 0): base(a, b)
    {
    }
    public override int area()
    {
        Console.WriteLine("Triangle class area :");
        return (width * height / 2);
    }
}
```

Đa hình trong lớp

Rectangle class area:

Area: 70

Triangle class area:

Area: 25

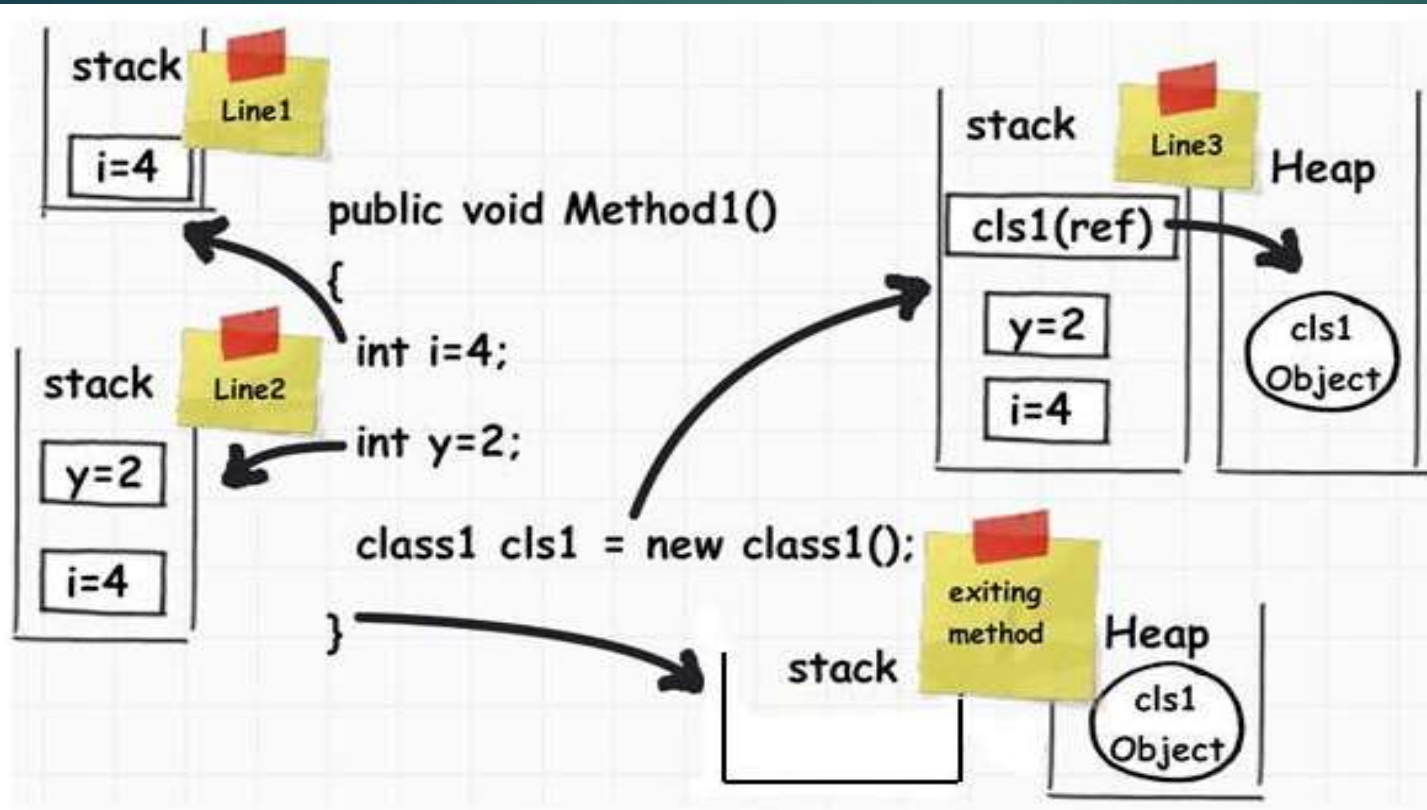
```
class Caller
{
    public void CallArea(Shape sh)
    {
        int a;
        a = sh.area();
        Console.WriteLine("Area: {0}", a);
    }
}

class Tester
{
    static void Main(string[] args)
    {
        Caller c = new Caller();
        Rectangle r = new Rectangle(10, 7);
        Triangle t = new Triangle(10, 5);
        c.CallArea(r);
        c.CallArea(t);
        Console.ReadKey();
    }
}
```

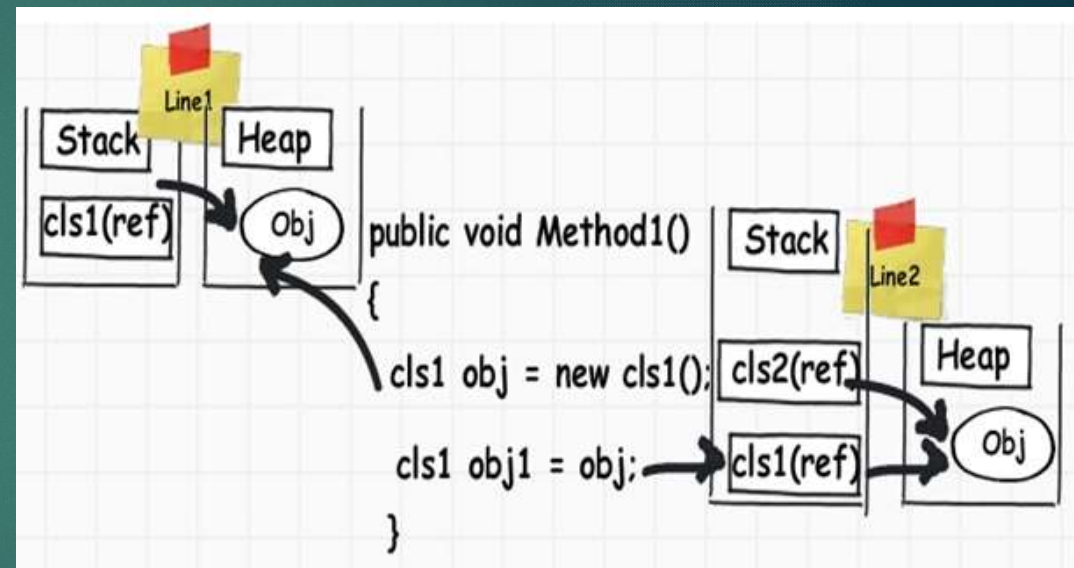
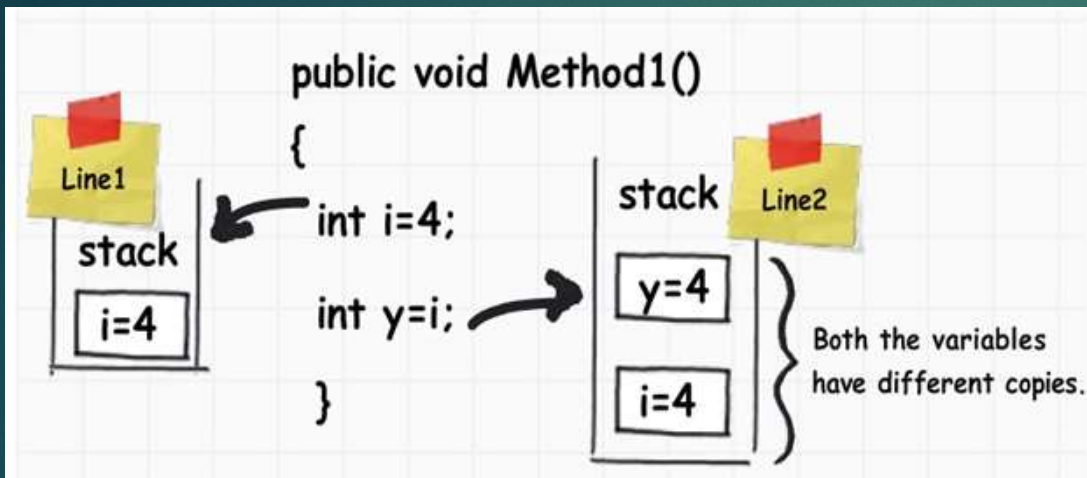

Stack và Heap memory trong C#

- ▶ Stack là nơi để lưu trữ và theo dõi các vùng nhớ đang được sử dụng trong chương trình
- ▶ Heap là nơi chỉ để lưu trữ khi cần truy xuất chứ không dùng để theo dõi vùng nhớ đang được sử dụng hay không
- ▶ Stack là dạng static memory, Heap là dạng dynamic memory

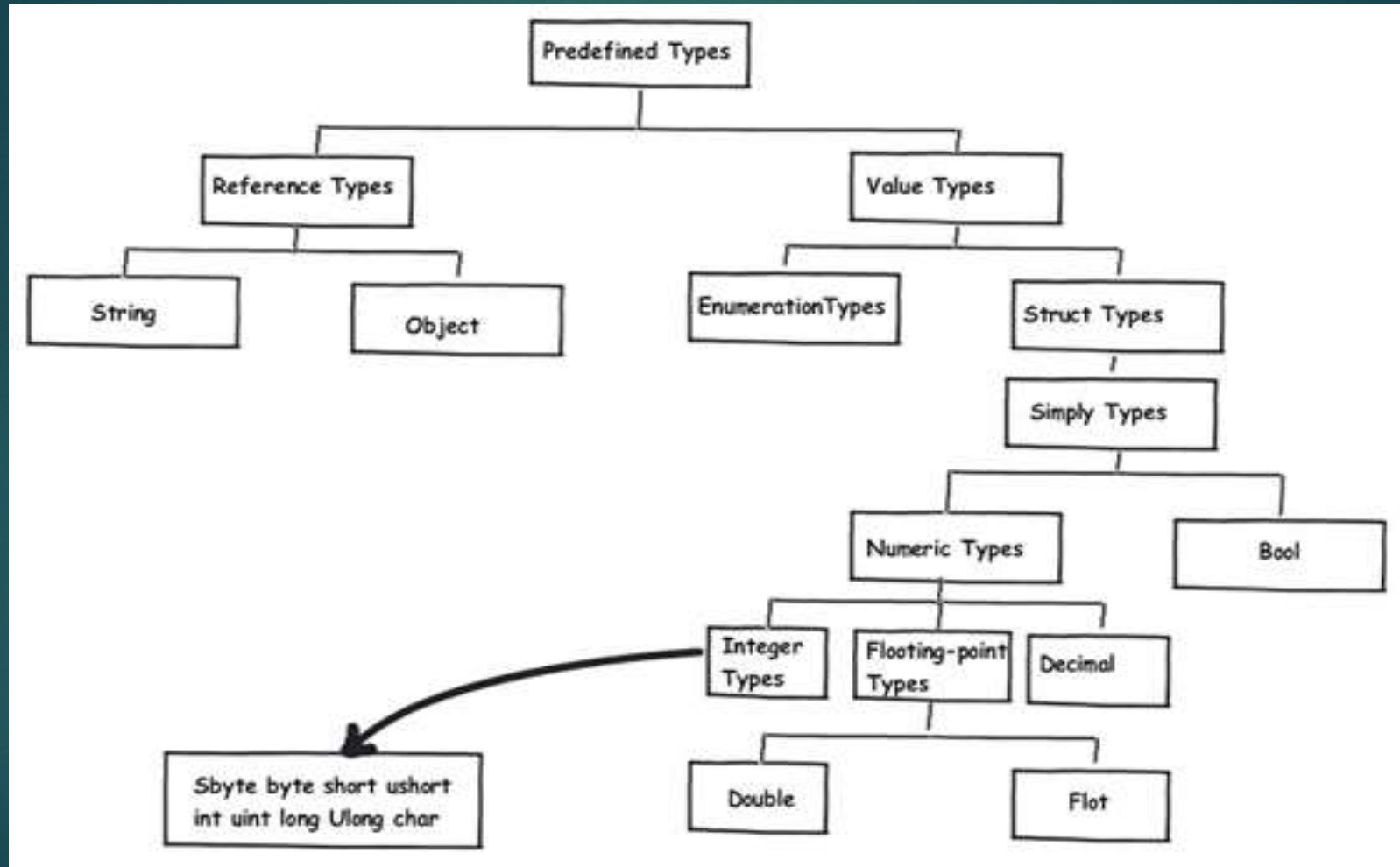
Stack và Heap memory trong C#



Stack và Heap memory trong C#

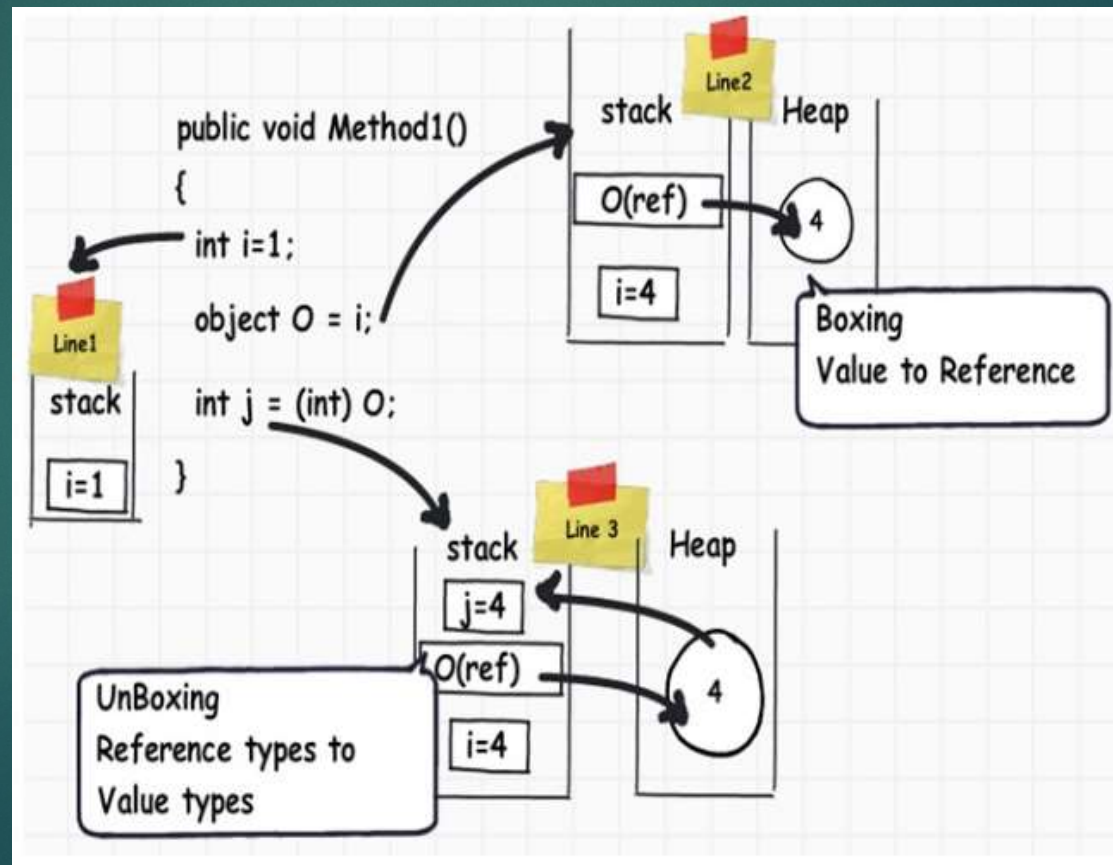


Kiểu dữ liệu Value (Stack) và Reference (Heap)



Boxing và Unboxing dữ liệu

- Là thao tác chuyển từ kiểu value sang kiểu reference (boxing) và ngược lại (unboxing)



Boxing và Unboxing dữ liệu

C#

```
int i = 123;  
// The following line boxes i.  
object o = i;
```

C#

```
o = 123;  
i = (int)o; // unboxing
```

private void BoxUnbox()

```
{  
    int i = 123;  
    object j = i;  
}
```

This function was
looped 10000 times
value , it took
3542 milliseconds
to execute

private void SimpleVariableAssignment()

```
{  
    int i = 123;  
    int j = i;  
}
```

This function was
looped 10000 times
value , it took
2477 milliseconds
to execute

Boxing unboxing function took 3542 Milliseconds while
with out boxing unboxing it took 2477 Milliseconds.

Boxing và Unboxing dữ liệu

```
class TestBoxing
{
    static void Main()
    {
        int i = 123;

        // Boxing copies the value of i into object o.
        object o = i;

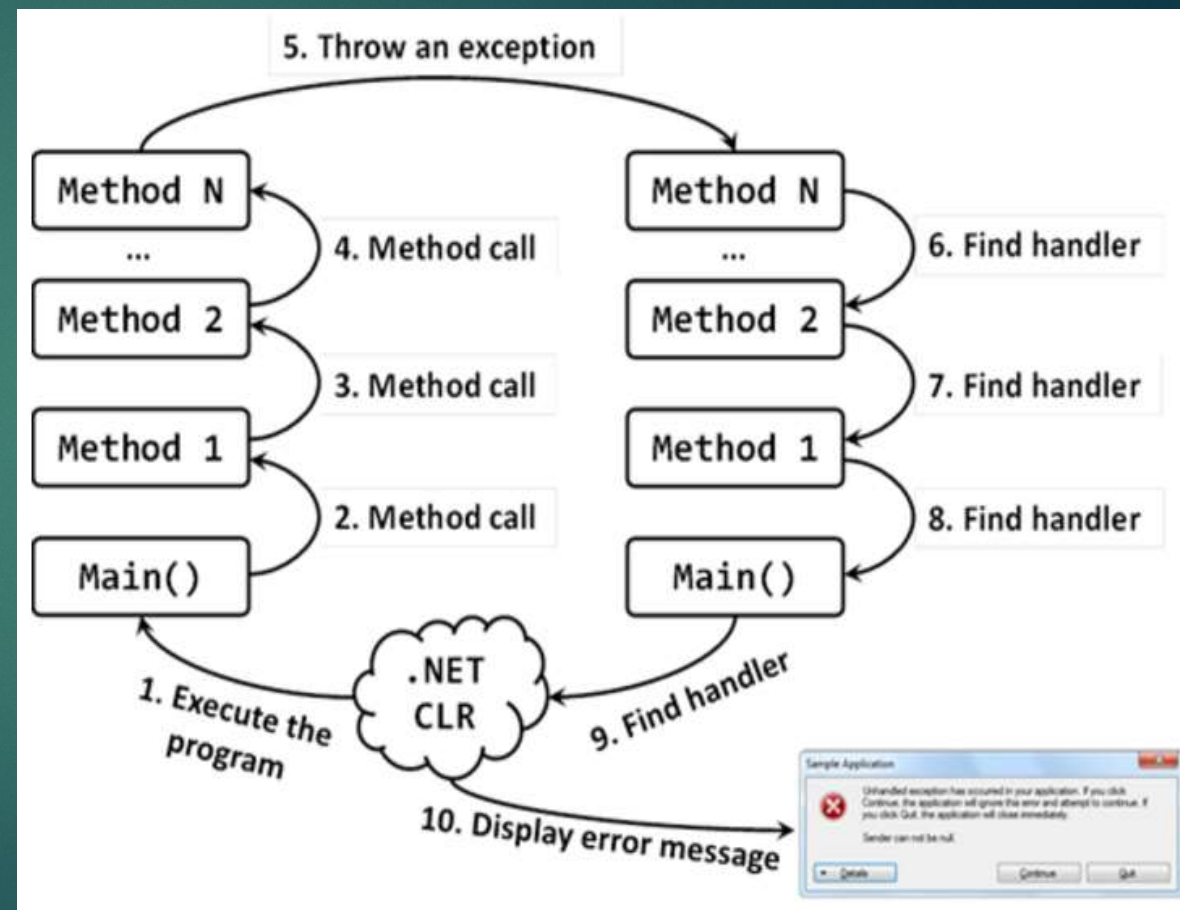
        // Change the value of i.
        i = 456;

        // The change in i does not effect the value stored in o.
        System.Console.WriteLine("The value-type value = {0}", i);
        System.Console.WriteLine("The object-type value = {0}", o);
    }
}

/* Output:
    The value-type value = 456
    The object-type value = 123
*/
```

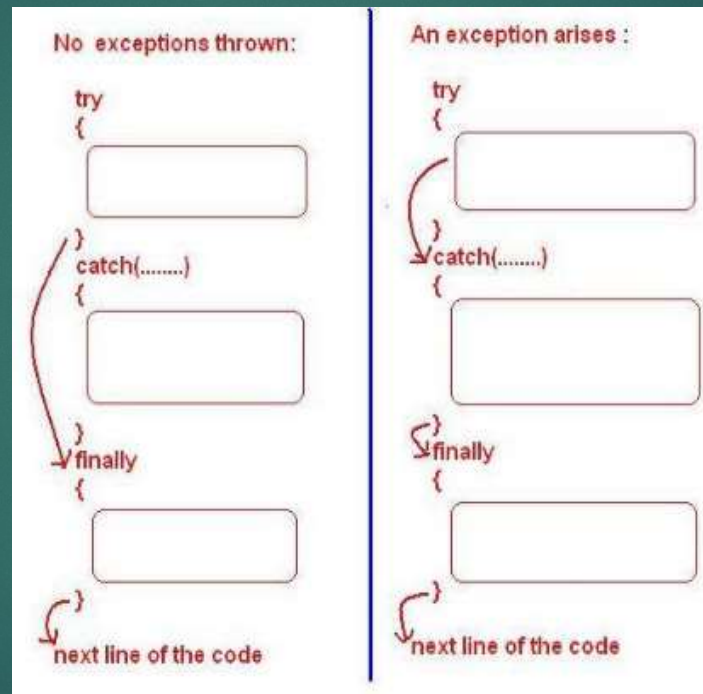
Xử lý Exception

- ▶ Một Exception là một vấn đề xảy ra khi thực thi chương trình
- ▶ Exception handling là hành động điều hướng xử lý của chương trình khi có lỗi xảy ra
- ▶ Các phương thức sẽ được lưu trong stack, khi xảy ra lỗi CLR sẽ cố gắng tìm Exception Handler bằng cách truy ngược lại từng phương thức. Nếu không tìm được Exception Handler thì sẽ tự động trả lỗi ra màn hình.



Xử lý Exception

- ▶ Try : là nơi chứa các đoạn code có khả năng phát sinh lỗi
- ▶ Catch : là nơi chứa các đoạn code xử lý trong trường hợp phát sinh ra lỗi
- ▶ Finally: là nơi chứa các đoạn code cần thực thi dù có lỗi hay không
- ▶ Throw: là câu lệnh chủ động gọi xử lý lỗi khi chúng ta phát hiện ra nó



```
try
{
    // statements causing exception
}
catch( ExceptionName e1 )
{
    // error handling code
}
catch( ExceptionName e2 )
{
    // error handling code
}
catch( ExceptionName eN )
{
    // error handling code
}
finally
{
    // statements to be executed
}
```


Xử lý Exception

- Các exception chúng ta có thể bắt

Exception Class	Description
System.IO.IOException	Handles I/O errors.
System.IndexOutOfRangeException	Handles errors generated when a method refers to an array index out of range.
System.ArrayTypeMismatchException	Handles errors generated when type is mismatched with the array type.
System.NullReferenceException	Handles errors generated from dereferencing a null object.
System.DivideByZeroException	Handles errors generated from dividing a dividend with zero.
System.InvalidCastException	Handles errors generated during typecasting.
System.OutOfMemoryException	Handles errors generated from insufficient free memory.
System.StackOverflowException	Handles errors generated from stack overflow.

Xử lý Exception

```
Exception caught: System.DivideByZeroException: Attempted to divide by zero.  
at ...  
Result: 0
```

```
using System;  
namespace ErrorHandlingApplication  
{  
    class DivNumbers  
    {  
        int result;  
        DivNumbers()  
        {  
            result = 0;  
        }  
        public void division(int num1, int num2)  
        {  
            try  
            {  
                result = num1 / num2;  
            }  
            catch (DivideByZeroException e)  
            {  
                Console.WriteLine("Exception caught: {0}", e);  
            }  
            finally  
            {  
                .WriteLine("Result: {0}", result);  
            }  
        }  
    }  
    static void Main(string[] args)  
    {  
        DivNumbers d = new DivNumbers();  
        d.division(25, 0);  
        Console.ReadKey();  
    }  
}
```



Tự định nghĩa Exception

```
using System;
namespace UserDefinedException
{
    class TestTemperature
    {
        static void Main(string[] args)
        {
            Temperature temp = new Temperature();
            try
            {
                temp.showTemp();
            }
            catch(TempIsZeroException e)
            {
                Console.WriteLine("TempIsZeroException: {0}", e.Message);
            }
            Console.ReadKey();
        }
    }
}

public class TempIsZeroException: Exception
{
    public TempIsZeroException(string message): base(message)
    {
    }
}
```



Tự định nghĩa Exception

```
public class Temperature
{
    int temperature = 0;
    public void showTemp()
    {
        if(temperature == 0)
        {
            throw (new TempIsZeroException("Zero Temperature found"));
        }
        else
        {
            Console.WriteLine("Temperature: {0}", temperature);
        }
    }
}
```

TempIsZeroException: Zero Temperature found

Checked và Unchecked

- ▶ Checked: phép toán tràn số sẽ cho ra exception
- ▶ Unchecked: phép toán tràn số sẽ không cho ra exception mà tự động cắt bớt (truncated)

```
short x = 32767;    // Max short value
short y = 32767;
try {
    z = checked((short)(x + y));
}
catch (System.OverflowException e) {
    System.Console.WriteLine(e.ToString());
}
```

```
unchecked {
    int z = x + y;
    return z;    // Returns -2
}
```

Câu lệnh điều kiện if và switch

```
if (boolean_expression)
    statement_or_statement_block;
else if (boolean_expression)
    statement_or_statement_block;
else
    statement_or_statement_block;
```

```
switch ( variable ) {
    case val1: statement-or-block;
        goto case val2;
    case val2: statement-or-block;
        goto default;
    default: statement-or-block;
        break;
}
```


: Câu lệnh lặp do...while , và while

```
int number = 0;
do {
    Console.WriteLine(number++.ToString());
} while (number <= 10);
```

```
while ( condition )
    statement-or-block;
```

```
for (int i = 1, j = 100 * i;    // initialization
     i < 100 && j > 0;         // condition
     i++, j--) {               // iteration
    System.Console.WriteLine((i+j).ToString());
}
```

Câu lệnh foreach

- Khi chúng ta có một mảng các đối tượng, có thể dùng lệnh foreach để xét từng phần tử trong mảng một cách nhanh chóng và hiệu quả

foreach statement

```
foreach ( var_declaration in enumerable_exp )  
    statement-or-block;
```

Example

```
string[] itemsToWrite = {"Alpha", "Bravo",  
                          "Charlie", "Denny"};  
  
foreach (string item in itemsToWrite)  
    System.Console.WriteLine(item);
```

Giá trị Nullables

- ▶ C# cung cấp một kiểu dữ liệu đặc biệt tên là kiểu dữ liệu nullable cho phép gán giá trị null vào kiểu dữ liệu đó.
- ▶ Cách khai báo < data_type> ? <variable_name> = null;

```
int? num1 = null;
int? num2 = 45;
double? num3 = new double?();
double? num4 = 3.14157;

bool? boolval = new bool?();

// display the values

Console.WriteLine("Nullables at Show: {0}, {1}, {2}, {3}", num1, num2, num3, num4);
Console.WriteLine("A Nullable boolean value: {0}", boolval);
Console.ReadLine();
```

```
Nullables at Show: , 45, , 3.14157
A Nullable boolean value:
```


Giá trị Nullables

- ▶ Phép toán kiểm tra null để gán giá trị đúng

```
using System;
namespace CalculatorApplication
{
    class NullablesAtShow
    {
        static void Main(string[] args)
        {
            double? num1 = null;
            double? num2 = 3.14157;
            double num3;
            num3 = num1 ?? 5.34;
            Console.WriteLine(" Value of num3: {0}", num3);
            num3 = num2 ?? 5.34;
            Console.WriteLine(" Value of num3: {0}", num3);
            Console.ReadLine();
        }
    }
}
```



Value of num3: 5.34

Value of num3: 3.14157

Mảng và ma trận

- ▶ Là một tập có số lượng xác định các phần tử cùng kiểu có thứ tự
- ▶ Mảng/Ma trận là kiểu reference nên cần phải dùng new để khởi tạo vùng nhớ (hoặc gán giá trị cụ thể)
- ▶ Mảng/Ma Trận là mảng có số lượng xác định nên cần truyền vào một số lượng cụ thể
- ▶ Có thể dùng foreach để duyệt mảng/Ma trận

```
int [] n = new int[10]; /* n is an array of 10 integers */

/* initialize elements of array n */
for ( int i = 0; i < 10; i++ )
{
    n[i] = i + 100;
}

/* output each array element's value */
foreach (int j in n )
{
    int i = j-100;
    Console.WriteLine("Element[{0}] = {1}", i, j);
}

Console.ReadKey();
```

Mảng và ma trận

```
int[] numbers = { 4, 5, 6, 1, 2, 3, -2, -1, 0 };  
foreach (int i in numbers)  
{  
    System.Console.Write("{0} ", i);  
}  
// Output: 4 5 6 1 2 3 -2 -1 0
```

```
int[,] numbers2D = new int[3, 2] { { 9, 99 }, { 3, 33 }, { 5, 55 } };  
// Or use the short form:  
// int[,] numbers2D = { { 9, 99 }, { 3, 33 }, { 5, 55 } };  
  
foreach (int i in numbers2D)  
{  
    System.Console.Write("{0} ", i);  
}  
// Output: 9 99 3 33 5 55
```


Chuỗi

- ▶ Chuỗi là một dãy các ký tự (S)
- ▶ Dùng
 - ▶ S.Length : Xác định chiều dài của chuỗi
 - ▶ bool Check = String.Compare(S,S1) : để so sánh hai chuỗi
 - ▶ S = String.Concat(S1,S2) : để nối hai chuỗi
 - ▶ S.Contains(S1) : để kiểm tra có chứa chuỗi con hay không
 - ▶ S = String.Join(“ ”, arrayofstring) : Để nối một mảng các chuỗi lại
 - ▶ S.ToLower(), S.ToUpper(), S.Trim() : để viết thường / viết hoa và xóa khoảng trắng đầu cuối chuỗi
 - ▶ S1 = S.Substring(index) : để trích chuỗi con từ chuỗi cha từ vị trí index

Chuỗi

```
string str1 = "This is test";
string str2 = "This is text";

if (String.Compare(str1, str2) == 0)
{
    Console.WriteLine(str1 + " and " + str2 + " are equal.");
}
else
{
    Console.WriteLine(str1 + " and " + str2 + " are not equal.");
}
Console.ReadKey() ;
```

```
//methods returning string
string[] sarray = { "Hello", "From", "Tutorials", "Point" };
string message = String.Join(" ", sarray);
Console.WriteLine("Message: {0}", message);
```

Chuỗi

```
using System;
namespace StringApplication
{
    class StringProg
    {
        static void Main(string[] args)
        {
            string str = "Last night I dreamt of San Pedro";
            Console.WriteLine(str);
            string substr = str.Substring(23);
            Console.WriteLine(substr);
        }
    }
}
```



```
string str = "This is test";
if (str.Contains("test"))
{
    Console.WriteLine("The sequence 'test' was found.");
}
Console.ReadKey() ;
```


Kiểu dữ liệu enum

- ▶ Là một tập gồm các giá trị nguyên được đặt tên

```
using System;
namespace EnumApplication
{
    class EnumProgram
    {
        enum Days { Sun, Mon, tue, Wed, thu, Fri, Sat };

        static void Main(string[] args)
        {
            int WeekdayStart = (int)Days.Mon;
            int WeekdayEnd = (int)Days.Fri;
            Console.WriteLine("Monday: {0}", WeekdayStart);
            Console.WriteLine("Friday: {0}", WeekdayEnd);
            Console.ReadKey();
        }
    }
}
```



Monday: 1

Friday: 5

THANK YOU
HẸN GẶP CÁC BẠN Ở PHẦN 2

