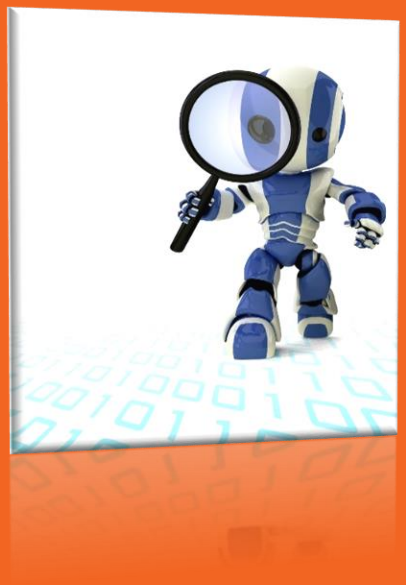
FPT Education

FPT POLYTECHNIC



LẬP TRÌNH C# 2

BÀI 2: IMPLICITLY,DYNAMIC,NULLABLE, ANONYMOUS TYPED

www.poly.edu.vn

Mục tiêu

- Implicitly, Dynamic
- Nullable, Anonymous

NỘI DUNG

- Phần I: Implicitly,Dynamic
  - Implicitly
  - Dynamic
- Phần II: Nullable, Anonymous
  - Nullable
  - Anonymous

KIỂU NGẦM ĐỊNH (IMPLICITLY TYPE)

- Khai báo biến kiểu ngầm định (khai báo không tường minh) là **biến được khai báo mà không cần phải chỉ ra kiểu dữ liệu**
- Kiểu dữ liệu của biến sẽ được xác định bởi trình biên dịch dựa vào biểu thức được gán khi khai báo biến
- Sử dụng từ khóa "var" khi khai báo và cần khởi tạo giá trị

```
var variable_name = value;
```

```
var x = 50; // Implicitly typed
int y = 50; // Explicitly typed
```

KIỂU NGẦM ĐỊNH (IMPLICITLY TYPE)

```
class Program

{
    0 references
    static void Main(string[] args)
    {
        var i = 100;
        Console.WriteLine("i value: {0}, type: {1}", i, i.GetType());
        var j = "Welcome to FPoly";
        Console.WriteLine("j value: {0}, type: {1}", j, j.GetType());
        var k = true;
        Console.WriteLine("k value: {0}, type: {1}", k, k.GetType());
        var l = 20.50;
        Console.WriteLine("l value: {0}, type: {1}", l, l.GetType());
        Console.ReadLine();
    }
}
```

KIỂU NGẦM ĐỊNH (IMPLICITLY TYPE)

- Các hạn chế khi dùng từ khóa "var"
  - Không thể sử dụng từ khóa var bên ngoài phạm vi của một method

```
public class Fpoly
{
    // declaring and initializing implicitly
    // typed local variable at class level.
    // It will give compile time error
    var imp = 15;
    // Main method
    0 references
    static public void Main()
    {
        // trying to print the value of 'imp'
        Console.WriteLine(Fpoly.imp);
    }
}
```

KIỂU NGẦM ĐỊNH (IMPLICITLY TYPE)

- Các hạn chế khi dùng từ khóa "var"
  - Không thể khởi tạo giá trị là null.

```
var value = null;   // invalid
```

  - Biến phải được khởi tạo giá trị khi nó được khai báo

```
var ivalue;   // invalid
```

  - Nếu biến được gán giá trị, thì kiểu dữ liệu phải giống nhau

```
var x = 3;
x = "Fpoly";      // ERROR
```

KIỂU NGẦM ĐỊNH (IMPLICITLY TYPE)

- Các hạn chế khi dùng từ khóa "var"
  - Giá trị khởi tạo phải là một biểu thức. Giá trị khởi tạo không được là một đối tượng hay tập hợp các giá trị. Nhưng nó có thể sử dụng toán tử new bởi một đối tượng hoặc tập hợp các giá trị.

```
// Not allowed
var data = { 23, 24, 10 };
// Allowed
var data = new int[] { 23, 34, 455, 65 };
```

KIỂU NGẦM ĐỊNH (IMPLICITLY TYPE)

- Cho biết kết quả khi chạy từng dòng lệnh bên dưới là valid hay Invalid?

```
var x = 10; //valid
var y; y = 10; // Error: Implicitly-typed variables must be initialized
var z = null; // Error: Cannot assign null to implicitly typed variable
var x = 10, y = 20, z = 30; // Invalid: Compile-time Error
var x = 10; // valid
var y = 20; // valid
var z = 30; // valid
// var variable as function Parameter
void GetDetails(var x)  // Invalid: Compile-time error
{
// your code
}
int x = (x = 20); // valid
var y = (y = 20); // invalid
```

KIỂU ĐỘNG (DYNAMIC TYPE)

- Kiểu động - ngầm định - khai báo với từ khóa dynamic, thì kiểu thực sự của biến đó được xác định bằng đối tượng gán vào ở thời điểm chạy (khác với kiểu ngầm định var kiểu xác định ngay thời điểm biên dịch)

```
dynamic x = 50;
dynamic y = 10.25;
```

Ví dụ khai báo phương thức có sử dụng tham số kiểu dynamic

```
static void TestFunc(dynamic dvar) {
    Console.WriteLine(dvar.abc); // ở thời điểm biên dịch - không biết dvar có thuộc tính abc hay không, nhưng nó vẫn biên dịch
}
```

KIỂU ĐỘNG (DYNAMIC TYPE)

Ví dụ Dynamic Type

```
class Program
{
    0 references
    static void Main(string[] args)
    {
        dynamic i = 100;
        Console.WriteLine("i value:{0}, type: {1}", i, i.GetType());
        dynamic j = "Welcome to FPoly";
        Console.WriteLine("j value:{0}, type: {1}", j, j.GetType());
        dynamic k = true;
        Console.WriteLine("k value:{0}, type: {1}", k, k.GetType());
        dynamic l = 20.50;
        Console.WriteLine("l value:{0}, type: {1}", l, l.GetType());
        Console.ReadLine();
    }
}
```

KIỂU ĐỘNG (DYNAMIC TYPE)

- Ví dụ dùng dynamic Type truyền tham số

```
class Program
{
    static void Main(string[] args)
    {
        GetDetails(100);
        GetDetails("Welcome to FPoly");
        GetDetails(true);
        GetDetails(20.50);
        Console.ReadLine();
    }
    static void GetDetails(dynamic d)
    {
        Console.WriteLine(d);
    }
}
```

KIỂU ĐỘNG (DYNAMIC TYPE)

- Gán đối tượng cho dynamic Type

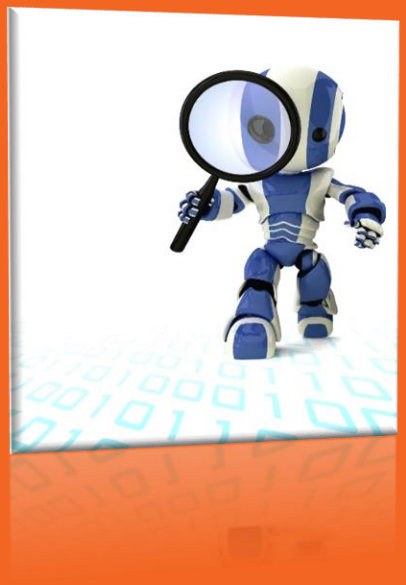
```
class Program
{
    static void Main(string[] args)
    {
        dynamic dobj = new User();
        dobj.GetInfo();
    }
}
class User
{
    public int Id { get; set; }
    public string Name { get; set; }
    public void GetDetails(dynamic d)
    {
        Console.WriteLine(d);
    }
}
```

STATIC CLASS

- Hiện thực các ví dụ

FPT Education

FPT POLYTECHNIC



LẬP TRÌNH C# 2

BÀI 2: IMPLICITLY,DYNAMIC,NULLABLE, ANONYMOUS TYPED(P2)

www.poly.edu.vn

NULLABLE TYPED

- null là một giá trị cố định không có tham chiếu (trỏ) đến đối tượng nào.
- null chỉ có thể gán được cho các biến kiểu tham chiếu (biến có kiểu dữ liệu là các lớp), không thể gán null cho những biến có kiểu dữ liệu dạng tham trị như int, float, bool ...

```
class Program
{
    static void Main(string[] args)
    {
        int x = null;
        bool y = null;
        Console.WriteLine("x = {0}, y = {1}", x, y);
        Console.ReadLine();
    }
}
```

Code	Description
CS0037	Cannot convert null to 'int' because it is a non-nullable value type
CS0037	Cannot convert null to 'bool' because it is a non-nullable value type

Ví dụ sử dụng giá trị null

```
class MyClass {
    public void ShowData() {
        Console.WriteLine("Show Data ... ");
    }
}
```

```
MyClass refvar1, refvar2;
refvar1 = new MyClass();     // refvar1 tham chiếu (gán) bằng một đối tượng
refvar2 = refvar1;           // refvar1, refvar2 cùng tham chiếu một đối tượng

refvar1 = null;              // refvar1 gán bằng null =>  không trỏ đến đối tượng nào
refvar2.ShowData();          // refvar2 có trỏ đến đến tượng, nên có thể truy cập các thành viên của đối tượng
refvar1.ShowData();          // refvar1 không trỏ đến đối tượng nào, nên truy cập thành viên sẽ lỗi

int myvar = 10;              //  int là kiểu tham trị, nó có thể gán giá trị cho biến myvar (10)
int myvar = null;            //  lỗi - kiểu tham  trị  không được gán null hay bằng tham chiếu đến đến tượng
```

NULLABLE TYPED

- Sử dụng Nullable Types cho kiểu dữ liệu tham trị
- Cú pháp

```
Nullable<T> variable_name

or

T? variable_name;
```

- Ví dụ

```
Nullable<int> x = 10;
int? y = 20;
bool? z = null;
Nullable<double> a = null;
int?[] arr = new int?[10];
```

- Cần gán giá trị khi khai báo

```
static void Main(string[] args)
{
    Nullable<int> x;
    Console.WriteLine("x = {0}", x);
    CS0165  Use of unassigned local variable 'x'
}
```

NULLABLE TYPED

- Cần gán giá trị khi khai báo, nếu không sẽ bị error

```
static void Main(string[] args)
{
    Nullable<int> x;
    Console.WriteLine("x = {0}", x);
    CS0165  Use of unassigned local variable 'x'
}
```

- Nên kiểm tra giá trị trước khi dùng bằng HasValue

```
class Program
{
    static void Main(string[] args)
    {
        Nullable<int> x = null;
        Console.WriteLine("x = {0}", x.Value);
        Console.ReadLine();
    }
}
```

Exception Unhandled

System.InvalidOperationException: 'Nullable object must have a value.'

View Details | Copy Details

Exception Settings

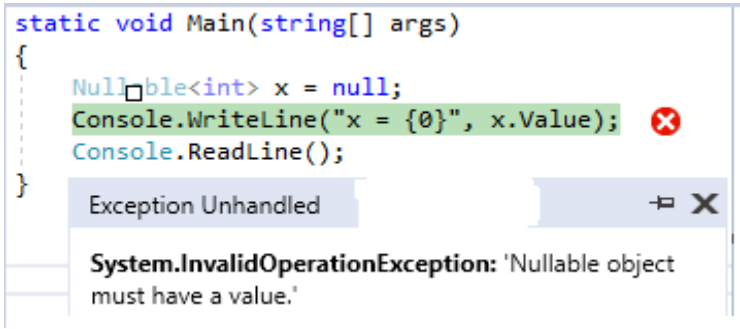
```
static void Main(string[] args)
{
    Nullable<int> x = null;
    if (x.HasValue)
    {
        Console.WriteLine("x = {0}", x.Value);
    }
    else
    {
        Console.WriteLine("Value is Empty");
    }
    Console.ReadLine();
```

NULLABLE TYPED

- Truy cập giá trị theo cú pháp NullableType.Value

```
Nullable<int> x = 10;
Console.WriteLine("x = {0}", x.Value);
```

- Bị exception nếu giá trị là null



- Dùng phương thức GetValueOrDefault()

```
static void Main(string[] args)
{
    Nullable<int> x = null;
    Console.WriteLine("x = {0}", x.GetValueOrDefault());
    Console.ReadLine();
```

NULLABLE TYPED

- Dùng toán tử ?? thực hiện gán Nullable Type cho Non-Nullable Type

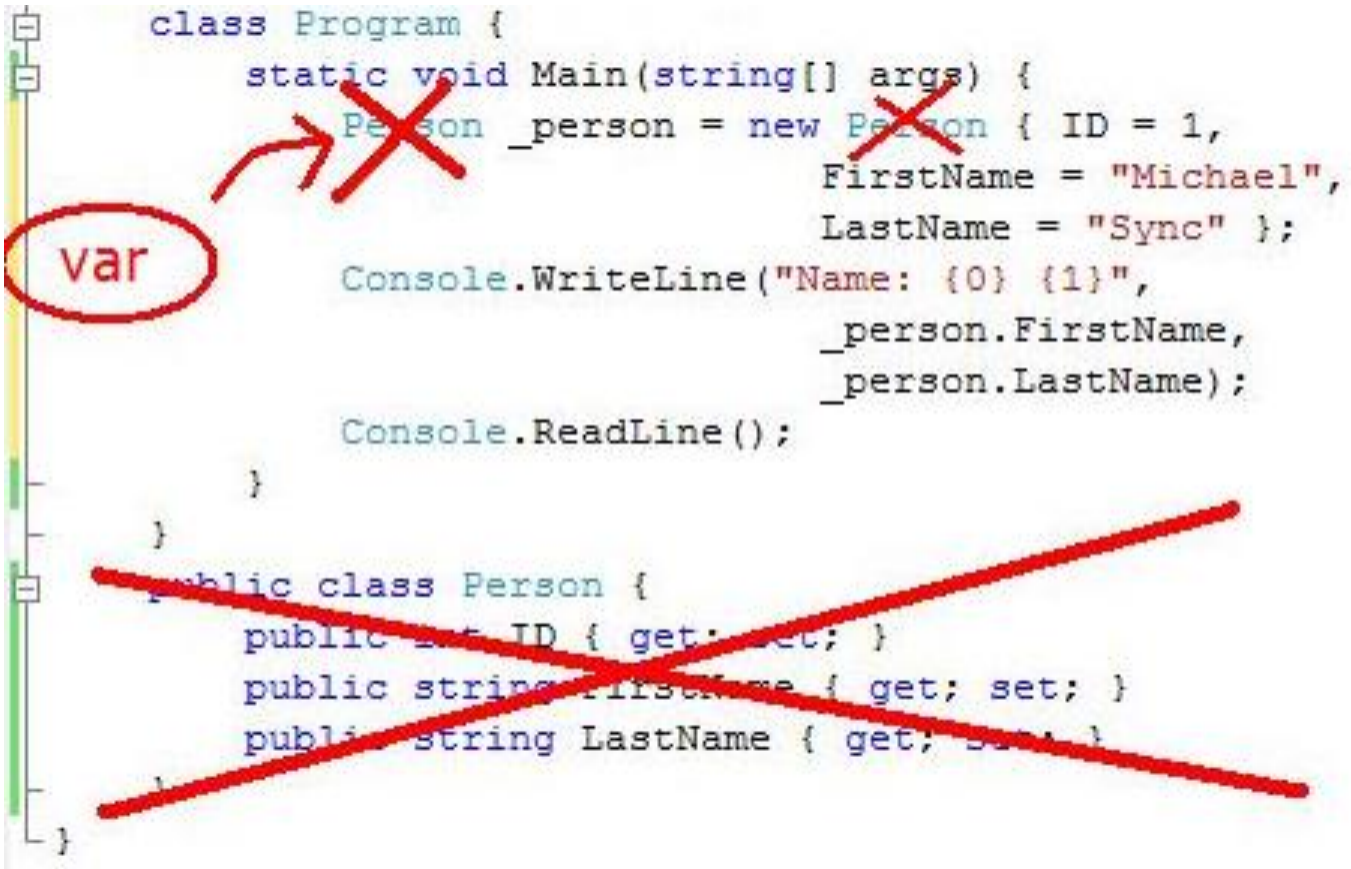
```
class Program
{
    static void Main(string[] args)
    {
        int? x = null;
        // y = x if x is not null, y = 0 if x is null
        int y = x ?? 0;
        Console.WriteLine("y = {0}", y);
        Console.ReadLine();
    }
}
```

ANONYMOUS TYPED

- Kiểu ẩn danh (Anonymous Type) cung cấp một cách thuận tiện để đóng gói (encapsulate) một tập các thuộc tính chỉ đọc (read-only properties) vào một đối tượng mà không cần phải xác định rõ ràng loại (type) của nó ngay lúc viết code
- Cho phép tạo type mới (user-defined) mà không cần xác định tên của nó
- Tạo các type ẩn danh này bằng cách sử dụng toán tử new

```
var userInfo = new { Id = 1, name = "Suresh", isActive = true };
```

- Ví dụ


class Program {
static void Main(string[] args) {
Person _person = new Person { ID = 1,
FirstName = "Michael",
LastName = "Sync" };
var
Console.WriteLine("Name: {0} {1}",
_person.FirstName,
_person.LastName);
Console.ReadLine();
}
}
public class Person {
public int ID { get; set; }
public string FirstName { get; set; }
public string LastName { get; set; }
}

ANONYMOUS TYPED

- Ví dụ dùng anonymous types lồng nhau

```
class Program
{
    static void Main(string[] args)
    {
        // Create Nested anonymous type object
        var user = new
        {
            Id = 1,
            Name = "Suresh Dasari",
            IsActive = true,
            jobInfo = new { Designation = "Lead", Location = "Hyderabad" }
        };
        // Access anonymous type object properties
        Console.WriteLine("Id: " + user.Id);
        Console.WriteLine("Name: " + user.Name);
        Console.WriteLine("IsActive: " + user.IsActive);
        Console.WriteLine("Designation: {0}, Location: {1}", user.jobInfo.Designatio
n, user.jobInfo.Location);
        Console.ReadLine();
    }
}
```

ANONYMOUS METHOD

- *Phương thức vô danh* (anonymous method) là một phương thức:
  - Không cần khai báo tên phương thức khi định nghĩa phương thức
  - Có thể khai báo trực tiếp ở chỗ cần dùng, không cần định nghĩa trước
  - Được dùng như tham số của delegate

```
public delegate void Print(int value);

0 references
static void Main(string[] args)
{
    Print print = delegate (int val) {
        Console.WriteLine("Inside Anonymous method. Value: {0}", val);
    };

    print(100);
}
```

ANONYMOUS METHOD

- Anonymous methods truy cập biến từ bên ngoài

```
public delegate void Print(int value);
0 references
static void Main(string[] args)
{
    int i = 10;

    Print prnt = delegate (int val) {
        val += i;
        Console.WriteLine("Anonymous method: {0}", val);
    };

    prnt(100);
}
```

ANONYMOUS METHOD

- Kết hợp Anonymous methods và delegate truyền tham số cho hàm

```
public delegate void Print(int value);
0 references
class Program
{
    1 reference
    public static void PrintHelperMethod(Print printDel, int val)
    {
        val += 10;
        printDel(val);
    }
    0 references
    static void Main(string[] args)
    {
        PrintHelperMethod(delegate (int val) { Console.WriteLine("Anonymous method: {0}", val); }, 100);
    }
}
```

ANONYMOUS METHOD

- Một số giới hạn Anonymous methods
  - Không khái báo được các lệnh goto, break or continue bên trong phương thức
  - Không truy cập được các tham số ref hoặc out bên ngoài
  - Phải được dùng kết hợp với delegate

STATIC CLASS

- Hiện thực các ví dụ

Tổng kết bài học

- Phần I: Implicitly,Dynamic
  - Implicitly
  - Dynamic
- Phần II: Nullable, Anonymous
  - Nullable
  - Anonymous

KẾT THÚC