

Bài 5.6: Kiểu giá trị có thể null

- ✓ Mục đích và cú pháp
- ✓ Chuyển đổi giữa kiểu có thể null và kiểu gốc
- ✓ Thao tác với các toán tử
- ✓ Xác định kiểu với typeof
- ✓ Ví dụ minh họa

Mục đích sử dụng và cú pháp

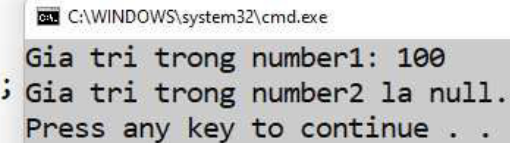
- ✓ Giá trị **null** đại diện cho một giá trị không xác định, không cụ thể, không rõ ràng. Thường được sử dụng để làm giá trị khởi tạo cho các kiểu tham chiếu.
- ✓ Kiểu giá trị có thể null(nullable value type) có dạng **T?** trong đó T là tên kiểu giá trị như int, float, bool.
- ✓ Kiểu giá trị có thể null được sử dụng khi cần thể hiện giá trị không xác định của kiểu cơ bản.
- ✓ Ví dụ một trường dữ liệu chứa giá trị bool trong CSDL có thể là NULL(không xác định), true hoặc false.
- ✓ Ví dụ:

```
bool? isOk = null; // có thể gán giá trị null
double? pi = 3.14; // có thể gán giá trị cụ thể
int?[] arr = new int?[10]; // mảng được khởi tạo giá trị null
int x = 200;
int? y = x; // có thể gán giá trị biến thông thường
```

Đánh giá đối tượng kiểu giá trị có thể null

- ✓ Ta có thể sử dụng các thuộc tính sau để đánh giá và lấy giá trị của một biến kiểu giá trị có thể null:
- ✓ **HasValue**: trả về true nếu giá trị trong biến khác null.
- ✓ **Value**: trả về giá trị của biến nếu HasValue là true. Nếu không sẽ vắng ngoại lệ.

```
int? number1 = 100;
int? number2 = null;
if(number1.HasValue)
{
    Console.WriteLine($"Gia tri trong number1: {number1.Value}");
} else
{
    Console.WriteLine("Gia tri trong number2 la null.");
}
if(number2.HasValue)
{
    Console.WriteLine($"Gia tri trong number2: {number2.Value}");
} else
{
    System.Console.WriteLine("Gia tri trong number2 la null.");
}
```



```
C:\WINDOWS\system32\cmd.exe
Gia tri trong number1: 100
Gia tri trong number2 la null.
Press any key to continue . .
```

Chuyển đổi kiểu

- ✓ Khi ta muốn gán giá trị của biến kiểu giá trị có thể null cho biến kiểu gốc, ta không thể gán trực tiếp.
- ✓ Để thực hiện được việc gán này ta sử dụng toán tử ?? Nhằm xác định giá trị thay thế được dùng khi gặp null.
- ✓ Cú pháp: **type variable = nullableVariable ?? replacementValue;**
- ✓ Trong đó type là kiểu biến giá trị thông thường, nullableVariable là biến kiểu giá trị có thể null. replacementValue là giá trị thay thế gán cho variable khi giá trị trong nullableVariable là null.

```
int? number1 = 100;
int? number2 = null;
int x = number1; // error
int y = number1 ?? -1; // ok, y = 100
int z = number2 ?? -1; // ok, z = -1
```

Thao tác với các toán tử

- ✓ Ta có thể sử dụng các toán tử có sẵn hoặc nạp chồng áp dụng cho các kiểu giá trị có thể null.
- ✓ Với các phép toán số học như $+$, $-$, $*$, $/$, $\%$ nếu ít nhất 1 trong 2 toán hạng là null thì kết quả sẽ là null.
- ✓ Với các toán tử so sánh $>$, $>=$, $<$, $<=$ nếu ít nhất 1 trong hai toán hạng là null, kết quả sẽ là false.
- ✓ Với toán tử so sánh $==$, nếu cả hai toán hạng cùng null thì kết quả là true. Nếu 1 trong 2 null thì kết quả là false. Nếu cả 2 đều khác null, so sánh theo giá trị của 2 toán hạng.
- ✓ Với toán tử so sánh $!=$, nếu cả hai toán hạng là null, kết quả là false. Nếu chỉ 1 trong 2 là null, kết quả là true. Các trường hợp khác so sánh theo giá trị của 2 toán hạng.

Ví dụ

```
int? number1 = 100;
int? number2 = null;
int? number3 = 200;
// phép cộng
number1++; // number1 tăng lên 101
Console.WriteLine("number1 = " + number1);
number1 += number3; // number1 có giá trị là 301
Console.WriteLine($"number1 = {number1}");
number3 += number2; // number3 có giá trị là null
// phép so sánh:
Console.WriteLine($"number1 > number2? {number1 > number2}");
Console.WriteLine($"number1 < number2? {number1 < number2}");
Console.WriteLine($"number1 >= number2? {number1 >= number2}");
Console.WriteLine($"number1 == number2? {number1 == number2}");
Console.WriteLine($"number1 != number2? {number1 != number2}");
Console.WriteLine($"number2 == null? {number2 == null}");
Console.WriteLine($"number2 != null? {number2 != null}");
```

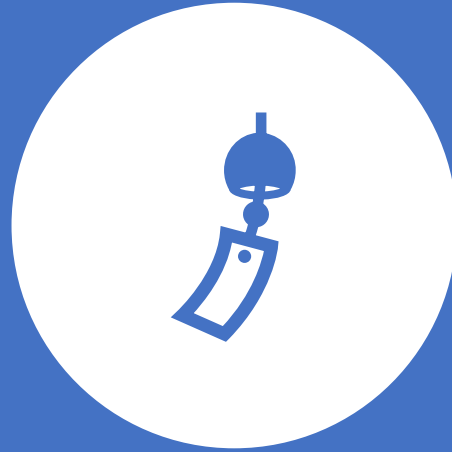
Cách xác định nullable value type

- ✓ Không sử dụng phương thức GetType() hay toán tử is để kiểm tra xem một đối tượng có phải thuộc kiểu giá trị có thể null.
- ✓ Để thực hiện, ta dùng toán tử typeof(kiểu) kết hợp với Nullable.GetUnderlyingType().
- ✓ Nếu phương thức trên trả về khác null, kiểu đang xét là kiểu giá trị có thể null. Ngược lại nó là kiểu giá trị cơ bản không null.

```
// kiểm tra xem một đối tượng kiểu có phải kiểu giá trị có thể null
Console.WriteLine(Nullable.GetUnderlyingType(typeof(int)) != null? "nullable" : "non-nullable");
Console.WriteLine(Nullable.GetUnderlyingType(typeof(int?)) != null? "nullable" : "non-nullable");
Console.WriteLine(Nullable.GetUnderlyingType(typeof(bool)) != null? "nullable" : "non-nullable");
Console.WriteLine(Nullable.GetUnderlyingType(typeof(bool?)) != null? "nullable" : "non-nullable");
Console.WriteLine(Nullable.GetUnderlyingType(typeof(double)) != null? "nullable" : "non-nullable");
Console.WriteLine(Nullable.GetUnderlyingType(typeof(double?)) != null? "nullable" : "non-nullable");
```

C:\WINDOWS\system32\cmd.exe

```
non-nullable
nullable
non-nullable
nullable
non-nullable
nullable
Press any key to continue . . .
```



Nội dung tiếp theo

Biểu thức lambda