Bài 10.1: C# generic

- ✓ Tổng quan về generic
- ✓ Lớp generic
- ✓ Phương thức generic
- ✓ Các ràng buộc với kiểu generic
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

Tổng quan về generic trong C#

- ✓Generic nghĩa là dạng thức chung, không dập khuôn vào một kiểu cụ thể nào.
- ✓Ví dụ ta viết 1 phương thức sắp xếp dùng chung. Sau đó mọi tập hợp muốn sử dụng chức năng sắp xếp đều có thể sử dụng. Khi đó ta có thể sắp xếp danh sách các phần tử kiểu int, float, double, long,… chỉ với 1 phương thức thay vì mỗi kiểu phải viết riêng 1 phương thức.
- ✓C# cho phép định nghĩa các lớp, interface, các trường, thuộc tính, sự kiện, delegate và toán tử ở kiểu generic sử dụng tham số kiểu(type parameter).
- ✓Type parameter là một chỗ chứa cho một kiểu cụ thể khi ta tạo đối tượng cụ thể của kiểu generic.

Lớp generic

✓Cú pháp tổng quát của lớp generic:

```
class ClassName<TypeParameters>
{
    // class members
}
```

✓Trong đó:

- ✓ ClassName: tên lớp generic đặt theo quy tắc đặt tên lớp.
- ✓ Cặp <> là bắt buộc.
- ✓ TypeParameters: danh sách các tham số kiểu. Thường sử dụng các chữ cái K, T, U, V,… để đại diện. Khuyến nghị nên đặt tên càng tường minh càng tốt. Ví dụ TKey, TValue, TSession…
- ✓ Bên trong lớp có thể chứa các thành phần như constructor, method, field, property…

✓Trong lớp generic ta có thể sử dụng các tham số kiểu làm kiểu trả về, tham số, đối số các thành phần khác như một kiểu thông thường.

Lớp generic TranscriptOfStudent

✓Ví dụ sau sử dụng lớp TranscriptOfStudent<S, T> để ghép cặp sinh viên với bảng điểm:

```
class Student
{
    0 references
    public string StudentId { get; set; }
}

0 references
class Transcript
{
    0 references
    public float Gpa { get; set; }
}

1 reference
class TranScriptOfStudent<S, T>
{
    1 reference
    public string SubjectName { get; set; }
    1 reference
    public S Student { get; set; }
    1 reference
    public T Transcript { get; set; }

    0 references
    public TranScriptOfStudent(S student, T transcript, string subjectName)
    {
        Student = student;
        Transcript = transcript;
        SubjectName = subjectName;
    }
    // ...
}
```

Ví dụ lớp Pair

```
class Lesson101
{
    0 references
    static void Main()
    {
        Console.OutputEncoding = Encoding.UTF8;
        Pair<string, string> hello = new Pair<string, string>("Hello", "Xin chào");
        Pair<string, string> lovely = new Pair<string, string>("Lovely", "Đáng yêu");
        Console.WriteLine($"Từ tiếng Anh: {hello.Key} => Nghĩa tiếng Việt: {hello.Value}");
        Console.WriteLine($"Từ tiếng Anh: {lovely.Key} => Nghĩa tiếng Việt: {lovely.Value}");
    }
}

5 references
class Pair<TKey, TValue>
{
    3 references
    public TKey Key { get; set; }
    3 references
    public TValue Value { get; set; }

    2 references
    public Pair(TKey key, TValue value)
    {
        Key = key;
        Value = value;
    }
    // ...
}
```

Một số đặc điểm của lớp generic

- ✓Lớp generic làm tăng tính tái sử dụng code. Càng nhiều tham số kiểu càng dễ tái sử dụng. Tuy nhiên không nên sử dụng quá nhiều generic làm cho chương trình khó hiểu và khó bảo trì.
- ✓Một lớp generic có thể là lớp cha của một lớp generic, non-generic khác.
- ✓Một lớp generic có thể kế thừa từ một lớp/interface generic hoặc non-generic khác.

Phương thức generic

- ✓Phương thức có kiểu trả về hoặc tham số là tham số kiểu thì nó là phương thức generic.
- ✓Các phương thức trong lớp generic là phương thức generic.
- ✓Các phương thức trong các lớp non-generic có thể trở thành phương thức generic khi ta thêm dấu hiệu generic vào cho nó.
- ✓Cú pháp: thêm <TypeParams> vào sau tên phương thức, trước dấu ngoặc (là xong.
- ✓Ví dụ:

```
0 references
class Printer
{
    0 references
    public static void Print<T>(T value)
    {
        Console.WriteLine(value);
    }
}
```

Ví dụ

```
    int[] integers = new int[] { 1, 2, 3, 5, 8, 9, 7, 0, 4 };
    double[] gpas = new double[] { 1.2, 3.5, 4.0, 2.54, 3.65, 3.24, 3.17 };
    Console.WriteLine($"Max integer number: {FindMax(integers)}"); // 9
    Console.WriteLine($"Max gpa: {FindMax(gpas)}"); // 4
}

// phương thức tìm phần tử lớn nhất trong mảng
2 references
public static T FindMax<T>(T[] arr) where T : IComparable<T>
{
    T max = arr[0];
    foreach (var item in arr)
    {
        if (item.CompareTo(max) > 0)
        {
            max = item;
        }
    }
    return max;
}
```

Ràng buộc trong generic

- ✓Ràng buộc chỉ định khả năng và mong muốn cần đáp ứng của một tham số kiểu.
- ✓Khai báo các ràng buộc tức là bạn có thể sử dụng các thao tác và phương thức của các kiểu đã ràng buộc.
- ✓Nếu lớp hoặc phương thức generic của bạn sử dụng các chức năng không được hỗ trợ bởi lớp Object, hãy sử dụng tới ràng buộc.
- ✓Ví dụ: ràng buộc về lớp cha sẽ giúp ta giới hạn việc phương thức, lớp chấp nhận đối số chỉ xảy ra nếu đối tượng truyền vào là đối tượng của lớp cha hoặc hậu duệ của lớp cha đó.
- ✓Cú pháp rang buộc: bổ sung where T : constraint1, constraint2… sau ngoặc > hoặc) và trước { của lớp/phương thức.
- ✓Trong đó T là tên tham số kiểu đang sử dụng. Constraint1, constraint2 là ràng buộc mà tham số T phải thỏa mãn.

Các ràng buộc generic

Ràng buộc	Mô tả
where T : struct	Kiểu của đối số phải là kiểu giá trị không null. Không kết hợp được với ràng buộc new(), unmanaged.
where T : class	Kiểu của đối số phải là kiểu tham chiếu. Áp dụng cho bất kì kiểu class, interface, delegate, array nào. Từ C# 8 về sau, T phải không thể null.
where T : class?	Kiểu của đối số phải là kiểu tham chiếu, có thể null hoặc không. Áp dụng cho bất kì kiểu class, interface, delegate, array nào.
where T : notnull	Kiểu của đối số phải không thể null. Từ C# 8 về sau, đối số có thể là kiểu tham chiếu non-nullable hoặc value type non-nullable.
where T : default	Ràng buộc này xử lý việc lẫn lộn giữa việc xác định một tham số kiểu không ràng buộc trong override phương thức hoặc cung cấp một triển khai interface mặc định. Ràng buộc này ngụ ý rằng phương thức gốc không có ràng buộc struct hoặc class.
where T : unmanaged	Kiểu của đối số phải là kiểu không được quản lý và không thể null. Ngụ ý áp dụng ràng buộc struct. Không thể kết hợp với ràng buộc struct hoặc new().
where T : new()	Tham số kiểu phải có một constructor mặc định không tham số với access modifier public. Khi kết hợp với các ràng buộc khác, ràng buộc new() phải được xác định sau cùng. Ràng buộc này không thể kết hợp với ràng buộc struct, unmanaged.

Các ràng buộc generic

where T : <base class name>	Kiểu của đói số phải là kiểu được chỉ định hoặc là kiểu dẫn xuất của lớp cha được chỉ định. Từ C# 8, T phải là kiểu tham chiếu không thể null dẫn xuất từ kiểu cha được chỉ định.
where T : <base class name>?	Kiểu của đối số phải là hoặc kiểu dẫn xuất từ kiểu được chỉ định. Từ C#8, T có thể null hoặc không, là kiểu được chỉ định hoặc kiểu dẫn xuất từ kiểu được chỉ định.
where T : <interface name>	Kiểu của đối số phải là interface được chỉ định hoặc là nó triển khai interface được chỉ định. Có thể áp dụng ràng buộc với nhiều interface. Interface trong ràng buộc có thể là generic. Từ C# 8 về sau, T phải là kiểu tham chiếu không thể null mà triển khai interface interface được chỉ định.
where T : <interface name>?	Giống ở trên và mở rộng việc chấp nhận T có thể là kiểu tham chiếu hoặc giá trị, nullable hoặc non-nullable.
where T : U	Đối số cung cấp cho T phải là hoặc dẫn xuất từ đối số cung cấp cho U. Trong ngữ cảnh nullable, nếu U là kiểu tham chiếu non-nullable , T cũng phải là kiểu tham chiếu non-nullable. Nếu U là nullable thì T có thể nullable hoặc non-nullable.

Ví dụ

```
    int[] integers = new int[] { 1, 2, 3, 5, 8, 9, 7, 0, 4 };
    Sort(integers); // sắp xếp các phần tử mảng
    ShowElements(integers); // hiển thị mảng sau sắp xếp
    Console.WriteLine($"Max integer number: {FindMax(integers)}"); // 9
}

// phương thức sắp xếp tăng dần các phần tử trong mảng
1 reference
public static void Sort<T>(T[] arr) where T : IComparable<T>
{
    for (int i = 0; i < arr.Length - 1; i++)
    {
        for (int j = arr.Length - 1; j > i; j--)
        {
            if(arr[j].CompareTo(arr[j - 1]) < 0)
            {
                var tmp = arr[j];
                arr[j] = arr[j - 1];
                arr[j - 1] = tmp;
            }
        }
    }
}
```

```
C:\WINDOWS\system32\cmd.exe
0 1 2 3 4 5 7 8 9
Max integer number: 9
```

Tóm tắt

- ✓Sử dụng các kiểu generic để tối đa hóa việc tái sử dụng code, an toàn về kiểu và hiệu năng.
- ✓Generic sử dụng phổ biến nhất trong quản lý dữ liệu dạng collection và các thao tác liên quan collection.
- ✓Thư viện .NET chứa các lớp collection trong namespace System.Collections.Generic. Các generic collection nên được sử dụng bất cứ khi nào có thể để thay cho ArrayList.
- ✓Ta có thể tự định nghĩa các class, interface, method, event, delegate generic riêng.
- ✓Các kiểu generic có thể được ràng buộc để cho phép truy cập tới các phương thức cụ thể của một kiểu nào đó.
- ✓Thông tin về kiểu dữ liệu được sử dụng trong kiểu generic có thể được quản lý tại thời điểm chạy của chương trình bằng cách sử dụng ánh xạ(reflection).

Nội dung tiếp theo

Tìm hiểu về lớp List<T>