

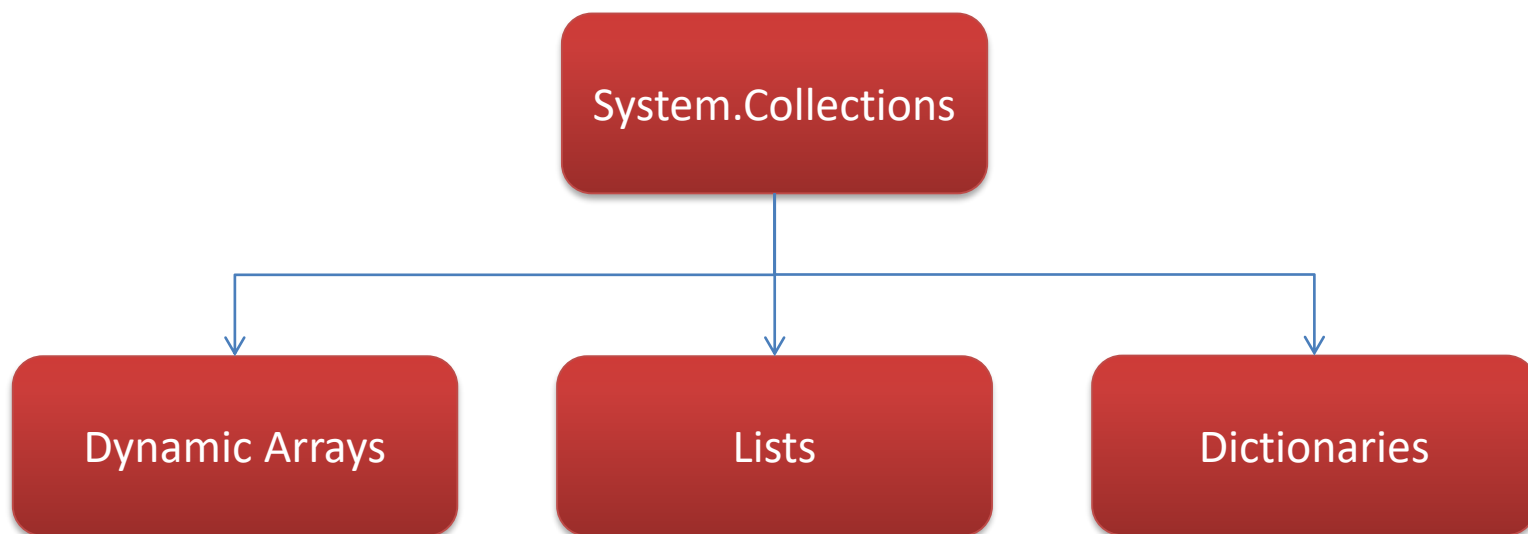


Bài 06

Generics và Collections

- Giới thiệu về tập hợp (Collection)
- Tìm hiểu một số Collection (lớp `ArrayList`, `Hashtable`, `SortedList`)
- Giới thiệu về tập hợp có định kiểu (`Generic Collection`)
- Tìm hiểu một số Generic Collection (`List<>`, `Dictionary<>`, `SortedList<>`)

- Collection là một tập dữ liệu có liên quan nhưng không nhất thiết phải cùng kiểu.
 - Nó có thể thay đổi động tại thời điểm chạy.
 - Truy cập vào collection giống như truy cập vào mảng.



“ArrayList” class

- **ArrayList** là mảng động, cho phép lưu trữ các phần tử có giá trị null và trùng nhau
- Giá trị các phần tử có thể thuộc các kiểu khác nhau
- Các phần tử được truy cập thông qua chỉ số (index) và giá trị (value)

0	1	2	3
Jack	45	Engineer	\$5000.00
Name	Age	Profession	Salary

Thêm phần tử	
Add	Thêm 1 phần tử vào cuối danh sách
AddRange	Thêm 1 tập phần tử vào cuối danh sách
Insert	Chèn 1 phần tử vào vị trí bất kỳ
InsertRange	Chèn 1 tập phần tử vào vị trí bất kỳ

Ví dụ

```
//tạo mảng
ArrayList arr = new ArrayList();
//thêm giá trị chuỗi
arr.Add("Xin chào");
//thêm giá trị số
arr.Add(10);
//chèn giá trị bool
arr.Insert(1, true);
//tạo mảng names
string[] names = { "Long", "Hai", "Dung" };
//thêm mảng names vào arraylist
arr.AddRange(names);
```

Xóa phần tử	
Remove	Xóa 1 phần tử có giá trị xác định
RemoveAt	Xóa 1 phần tử tại vị trí xác định
RemoveRange	Xóa các phần tử tại vị trí xác định
Clear	Xóa tất cả các phần tử

Ví dụ

```
//xóa phần tử có giá trị "Xin chào"  
arr.Remove("Xin chào");  
//xóa phần tử ở vị trí 1  
arr.Remove(1);  
//xóa 3 phần tử từ vị trí 1  
arr.RemoveRange(1, 3);  
//xóa tất cả các phần tử  
arr.Clear();
```

▪ Duyệt mảng

Ví dụ

```
//duyet mang dung for
for (int i = 0; i < arr.Count; i++)
{
    Console.WriteLine(arr[i]);
}
//duyet mang dung foreach
foreach (var item in arr)
{
    Console.WriteLine(item);
}
//lay bo du lieu dang liet ke va duyet dung while
IEnumerator items = arr.GetEnumerator();
while (items.MoveNext())
{
    Console.WriteLine(items.Current);
}
```

Sắp xếp và tìm kiếm

Sort	Sắp xếp các phần tử phân biệt chữ hoa chữ thường
IndexOf	Tìm phần tử đầu tiên, nếu tìm thấy trả về vị trí, nếu không trả về -1
LastIndexOf	Tìm phần tử ở cuối (giống IndexOf)
Contains	Kiểm tra trong tập hợp có chứa phần tử cần tìm không

Ví dụ

```
//Tạo mảng
ArrayList tree = new ArrayList() {"Tung", "Cuc", "Truc", "Mai" };
//sắp xếp phân biệt chữ hoa chữ thường
tree.Sort();
//sắp xếp không phân biệt chữ hoa thường
tree.Sort(new CaseInsensitiveComparer());
//Kiểm tra xem có phần tử "Cuc" không, nếu có thì xóa
if (tree.Contains("Cuc"))
{
    int pos = tree.IndexOf("Cuc");
    tree.RemoveAt(pos);
}
else
    Console.WriteLine("Khong tim thay");
```


- Lớp **HashTable** là tập hợp để lưu trữ các phần tử, mỗi phần tử gồm một cặp thông tin key(khóa) và value (giá trị)
 - Key của các phần tử phải là duy nhất
 - Cho phép tìm kiếm phần tử theo key
- **Tạo đối tượng**

```
Hashtable pb = new Hashtable();
```

Thêm, xóa phần tử

Add	Thêm một phần tử vào cuối danh sách
Remove	Xóa một phần tử theo key xác định
Clear	Xóa tất cả các phần tử

Ví dụ

```
//tạo đối tượng
Hashtable pb=new Hashtable();
//thêm 3 phần tử vào hashtable
pb.Add("HR", "Human Resource");
pb.Add("IT", "Information Technology");
pb["MK"] = "Marketing";
//xóa phần tử
pb.Remove("IT");
//xóa hết
pb.Clear();
```

▪ Duyệt **Hashtable** và truy xuất tới key, value

Ví dụ

```
//tạo đối tượng
Hashtable pb=new Hashtable();
//thêm 3 phần tử vào hashtable
pb.Add("HR", "Human Resource");
pb.Add("IT", "Information Technology");
pb["MK"] = "Marketing";
foreach (var key in pb.Keys)
{
    Console.WriteLine(key + ":" + pb[key]);
}
```

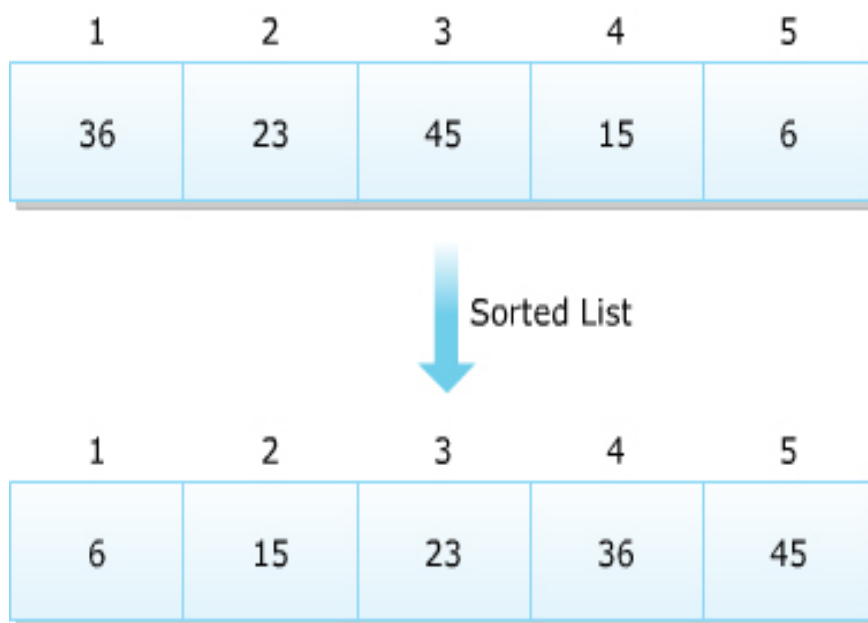
Tìm kiếm phần tử

ContainsKey	Trả về true nếu trong tập hợp có chứa key chỉ ra, ngược lại trả về false
ContainsValue	Trả về true nếu trong tập hợp có chứa value chỉ ra, ngược lại trả về false

Ví dụ

```
//kiểm tra xem trong tập hợp có chứ key là "SC" không, nếu không thì bổ sung vào  
if (!pb.ContainsKey("SC"))  
    pb.Add("SC","Security");
```

- Lớp **SortedList** giống lớp **Hashtable** nhưng key được sắp xếp và cho truy xuất phần tử theo chỉ số hoặc theo key.
- Tuy nhiên về tốc độ thì chương trình sử dụng **SortedList** chậm hơn so với chương trình sử dụng **Hashtable**



- **Thêm phần tử**

Sử dụng phương thức **Add()** để thêm một phần tử với (key-value) vào cuối danh sách

- **Xóa phần tử**

Sử dụng các phương thức **Remove, RemoveAt, Clear**

- **Truy xuất phần tử**

Phương thức **GetKey(chỉ_số)** lấy key của phần tử có chỉ_số

Phương thức **GetByIndex(chỉ_số)** lấy value của phần tử có chỉ_số

Thao tác với SortedList 2-2

Ví dụ

```
//tao doi thuong
SortedList staff = new SortedList();
//them gia tri vao staff
staff.Add("Long", "Administrator");
staff.Add("Hung", "Human Resources");
staff.Add("Thuy", "Finance");
staff.Add("Dung", "Marketing");
staff.Add("Thang", "Manager");
Console.WriteLine("Danh sach nhan vien sap xep theo ten:");
//duyet danh sach truy cap theo chi so
for (int i = 0; i < staff.Count; i++)
{
    Console.WriteLine("Key:" + staff.GetKey(i) + "/" + "Values:" + staff.GetByIndex(i));
}
//tim kiem
if (!staff.ContainsKey("Hoang"))
    staff.Add("Hoang", "Support");
Console.WriteLine("Danh sach nhan vien sap xep theo ten:");
//duyet danh sach truy cap theo key
foreach (var item in staff.Keys)
{
    Console.WriteLine("Key:" + item + "/" + "Values:" + staff[item]);
}
```

- Generic là một phần trong hệ thống kiểu của .NET Framework, nó phép định kiểu mà không quan tâm nhiều đến các chi tiết bên trong
- .NET Framework cung cấp nhiều lớp generic trong namespace **System.Collections.Generic**.
 - Các lớp này hoạt động tương tự các lớp thông thường
 - Chúng tăng hiệu năng thực hiện và truy xuất an toàn về kiểu (safe-type)
- Ngoài những lớp Generic do .NET cung cấp, người dùng cũng có thể tự tạo ra các lớp Generic tùy biến.

- Bảng sau liệt kê danh sách các lớp Collection và Generic Collections tương ứng

Collections	Generic Collections
ArrayList	List<>
Hashtable	Dictionary<>
SortedList	SortedList<>
DictionaryEntry	KeyValuePair<>

- Lớp `List<>`: tương tự `ArrayList` nhưng các phần tử phải chỉ ra kiểu dữ liệu trước

Ví dụ

```
//Tạo tập hợp chứa kiểu dữ liệu số nguyên
List<int> numbers = new List<int>();
numbers.Add(10);
numbers.Add(15);
numbers.Add(25);
numbers.Add("CShape");//Dòng này biên dịch sẽ báo lỗi
foreach (int n in numbers)
{
    Console.WriteLine(n);
}
```

- Lớp **Dictionary<>**: tương tự lớp **Hashtable** nhưng key và value phải được định kiểu trước

Ví dụ

```
Dictionary<int, string> week = new Dictionary<int, string>();  
week.Add(2, "Thu 2");  
week.Add(3, "Thu 3");  
week.Add(4, "Thu 4");  
week.Add(5, "Thu 5");  
week.Add("6", "Thu 6"); //dòng này sẽ báo lỗi nhé  
foreach (int key in week.Keys)  
{  
    Console.WriteLine(key + ":" + week[key]);  
}
```

- Lớp `SortedList<>`: tương tự lớp `Dictionary<>` nhưng các phần tử được sắp xếp theo key.

Ví dụ

```
SortedList<string, int> number = new SortedList<string, int>();  
number.Add("Three", 3);  
number["One"] = 1;  
number.Add("Two", 2);  
//in ra danh sách sắp xếp theo key  
foreach (string key in number.Keys)  
{  
    Console.WriteLine(key + ":" + number[key]);  
}
```

- Từ C# 3.0 trở lên, .NET cung cấp cách khởi tạo nhanh một tập hợp mà không cần sử dụng phương thức **Add/AddRange** như phiên bản trước.

Khởi tạo nhanh với List<>

```
//cách khởi tạo cũ
List<string> animal = new List<string>();
animal.Add("Long");
animal.Add("Ly");
animal.Add("Quy");
animal.Add("Phuong");
//cách khởi tạo mới
List<string> animal1 = new List<string>(){ "Long", "Ly", "Quy", "Phuong" };
```

Khởi tạo nhanh với Dictionary<>

```
//Khởi tạo nhanh Dictionary
Dictionary<string, int> flowers = new Dictionary<string, int>()
{
    {"Hoa Lan",20000}, {"Hoa Hong",24000}, {"Hoa Ly",23000}
};
```

Khởi tạo nhanh với dữ liệu dạng class

```
//Khởi tạo nhanh với dữ liệu dạng class
List<Employee> list = new List<Employee>()
{
    new Employee{Id=1,Name="Thuy",Address="Ha Noi"},
    new Employee{Id=2,Name="Dung",Address="Ha Noi"},
    new Employee{Id=3,Name="Long",Address="Ha Noi"}
};
```

Hỏi Đáp

