CHƯƠNG 2

LỚP VÀ ĐỐI TƯỢNG

GVHD: TS. GVC LÊ THỊ THÚY NGA

BỘ MÔN: ĐIỀU KHIỂN HỌC

NỘI DUNG

2.1 Lớp và đối tượng

2.2 Tạo và sử dụng đối tượng

2.3 Các kỹ thuật trong xây dựng lớp

2.4 Kết tập và kế thừa

2.5 Các kiểu dữ liệu tập hợp trong C#

2.6 Ngoại lệ và xử lý ngoại lệ

2.5 CÁC KIỂU DỮ LIỆU TẬP HỢP (COLLECTIONS) TRONG C#

- Trong nội dung của các mục trước, chúng ta đã sử dụng mảng để lưu trữ danh sách các đối tượng thuộc lớp. Đối với dữ liệu kiểu mảng chỉ có thể truy cập vào phần tử của mảng thông qua chỉ số mà không thể thêm bớt các phần tử của mảng trong quá trình run – time.
- Để khắc phục nhược điểm trên, Collections cung cấp một cách linh hoạt hơn để làm việc với danh sách. Ta có thể tăng giảm số lượng phần tử một cách tự động. Một số Collections còn hỗ trợ lưu trữ danh sách dưới dạng **Key – Value** giúp truy xuất, tìm kiếm một cách nhanh chóng.

2.5 CÁC KIỂU DỮ LIỆU TẬP HỢP TRONG C#

- Collection là một tập hợp các dữ liệu không cùng kiểu. Collection cung cấp rất nhiều các phương thức giúp người dùng thao tác với dữ liệu một cách đơn giản và dễ dàng.
- Một trong những ưu điểm lớn nhất của Collection là khả năng tương tác và thay đổi dữ liệu bên trong nó ngay tại thời điểm chạy (run-time).
- Microsoft cung cấp một tập hợp các lớp Collections trong thư viện System.Collections. Khai báo thư viện:

```
using System.Collections.
```

❖Một số Collections thông dụng:

LỚP	MÔ TẢ
ArrayList	Lớp cho phép lưu trữ và quản lý các phần tử giống mảng. Tuy nhiên, không giống như trong mảng, ta có thể thêm hoặc xoá phần tử một cách linh hoạt và có thể tự điều chỉnh kích cỡ một cách tự động.
HashTable	Lớp lưu trữ dữ liệu dưới dạng cặp **Key – Value**. Khi đó ta sẽ truy xuất các phần tử trong danh sách này thông qua **Key** (thay vì thông qua chỉ số phần tử như mảng bình thường).
SortedList	Là sự kết hợp của ArrayList và HashTable. Tức là dữ liệu sẽ lưu dưới dạng **Key – Value**. Ta có thể truy xuất các phần tử trong danh sách thông qua **Key** hoặc thông qua chỉ số phần tử. Đặc biệt là các phần tử trong danh sách này luôn được sắp xếp theo giá trị của **Key**.
Stack	Lớp cho phép lưu trữ và thao tác dữ liệu theo cấu trúc LIFO (Last In First Out).
Queue	Lớp cho phép lưu trữ và thao tác dữ liệu theo cấu trúc FIFO (First In First Out).
BitArray	Lớp cho phép lưu trữ và quản lý một danh sách các bit. Giống mảng các phần tử kiểu bool với true biểu thị cho bit 1 và false biểu thị cho bit 0. Ngoài ra BitArray còn hỗ trợ một số phương thức cho việc tính toán trên bit.

ARRAYLIST

❖**ArrayList** trong C#:

- Là một Collections giúp lưu trữ và quản lý một danh sách các đối tượng theo kiểu mảng (truy cập các phần tử bên trong thông qua chỉ số index).
- Rất giống mảng các object nhưng có thể thêm hoặc xoá các phần tử một cách linh hoạt và có thể tự điều chỉnh kích cỡ một cách tự động.

❖Cú pháp khai báo: **List<ClassName> ObjectName = new List<ClassName>();**

❖Ví dụ: List<sinhvien> a = new List<sinhvien>(); *// a là đối tượng thuộc lớp sinhvien có kiểu dữ*

sinhvien [] a=new sinhvien[100] *//liệu là ArrayList.*

❖Một số thuộc tính thông dụng trong ArrayList:

Tên thuộc tính	Ý nghĩa
Count	Trả về 1 số nguyên là **số phần tử hiện có** trong ArrayList.
Capacity	Trả về 1 số nguyên cho biết số phần tử mà ArrayList **có thể chứa** (sức chứa). Nếu số phần tử được thêm vào chạm sức chứa này thì hệ thống sẽ tự động tăng lên. Ngoài ra ta có thể gán 1 sức chứa bất kỳ cho ArrayList.

Ví dụ: **b=a.Count**; *// b là số phần tử của mảng danh sách a*

❖Một số phương thức thông dụng trong ArrayList:

TÊN PHƯƠNG THỨC	Ý NGHĨA
Add(object Value)	Thêm đối tượng **Value** vào cuối ArrayList.
AddRange(ICollection ListObject)	Thêm danh sách phần tử **ListObject** vào cuối ArrayList.
BinarySearch(object Value)	Tìm kiếm đối tượng **Value** trong ArrayList theo thuật toán tìm kiếm nhị phân. Nếu tìm thấy sẽ trả về vị trí của phần tử ngược lại trả về giá trị âm. **Lưu ý** *là ArrayList phải được sắp xếp trước khi sử dụng hàm.*
Clear()	Xoá tất cả các phần tử trong ArrayList.
Clone()	Tạo 1 bản sao từ ArrayList hiện tại.
Contains(object Value)	Kiểm tra đối tượng **Value** có tồn tại trong ArrayList hay không.

⇒

Ví dụ:

a.Add(b); *// thêm đối tượng b*

//vào vị trí cuối của danh sách a

❖Một số phương thức thông dụng trong ArrayList:

TÊN PHƯƠNG THỨC	Ý NGHĨA
GetRange(int StartIndex, int EndIndex)	Trả về 1 ArrayList bao gồm các phần tử từ vị trí **StartIndex** đến **EndIndex** trong ArrayList ban đầu.
IndexOf(object Value)	Trả về vị trí đầu tiên xuất hiện đối tượng **Value** trong ArrayList. Nếu không tìm thấy sẽ trả về -1.
⇒ Insert(int Index, object Value)	Chèn đối tượng **Value** vào vị trí **Index** trong ArrayList.
InsertRange(int Index, ICollection ListObject)	Chèn danh sách phần tử **ListObject** vào vị trí **Index** trong ArrayList.
LastIndexOf(object Value)	Trả về vị trí xuất hiện cuối cùng của đối tượng **Value** trong ArrayList. Nếu không tìm thấy sẽ trả về -1.

Ví dụ:

a.Insert(1,b); *// chèn đối tượng b*

//vào chỉ số 1 (vị trí 2)

//của danh sách a

❖Một số phương thức thông dụng trong ArrayList:

TÊN PHƯƠNG THỨC	Ý NGHĨA
⇒ Remove(object Value)	Xoá đối tượng **Value** xuất hiện đầu tiên trong ArrayList.
Reverse()	Đảo ngược tất cả phần tử trong ArrayList.
Sort()	Sắp xếp các phần tử trong ArrayList theo thứ tự tăng dần.
ToArray()	Trả về 1 mảng các object chứa các phần tử được sao chép từ ArrayList.
⇒ RemoveAt(index)	Xóa đối tượng có chỉ số index tương ứng

Ví dụ:

a.Remove(b);

// xóa đối tượng b

//đầu tiên của danh

//sách a

Ví dụ:

a.RemoveAt(2);

// xóa đối tượng có

//chỉ số 2 (tương

//đương vị trí 3) của

//danh sách a

BÀI TẬP 2.5

Ứng dụng **Console** để thực hiện các công việc sau:

- Xây dựng lớp sinh viên:

  + thuộc tính: họ tên, năm sinh, quê quán.

  + phương thức: khởi tạo, nhập thông tin.

- Xây dựng lớp sinh viên đại học kế thừa lớp sinh viên:

  Bổ sung:

+ thuộc tính: điểm năm 1, điểm năm 2, điểm năm 3, điểm năm 4, điểm năm 5.

+ phương thức: tính điểm trung bình (điểm trung bình= (điểm năm 1+ điểm năm 2+ điểm năm 3+ điểm năm 4+ điểm năm 5)/5), in thông tin gồm cả điểm trung bình.

BÀI TẬP 2.5

Yêu cầu:

a. Thực hiện nhập thông tin, in thông tin (tất cả các thông tin) của m (m>3) đối tượng thuộc lớp sinh viên đại học.

b. Xóa sinh viên đại học ở vị trí thứ 1 của danh sách a) (danh sách m đối tượng thuộc lớp sinh viên đại học), sau đó in thông tin của danh sách mới.

c. Chèn vào danh sách b, một sinh viên mới vào vị trí thứ 2, in thông tin của tất cả các sinh viên trong danh sách mới.

d. Hiển thị tất cả thông tin của sinh viên cuối cùng trong danh sách c)