LẬP TRÌNH HƯỚNG SỰ KIỆN

Giảng viên: ThS. Phan Thanh Toàn

BÀI 3
GIAO DIỆN VÀ TẬP HỢP

Giảng viên: ThS. Phan Thanh Toàn

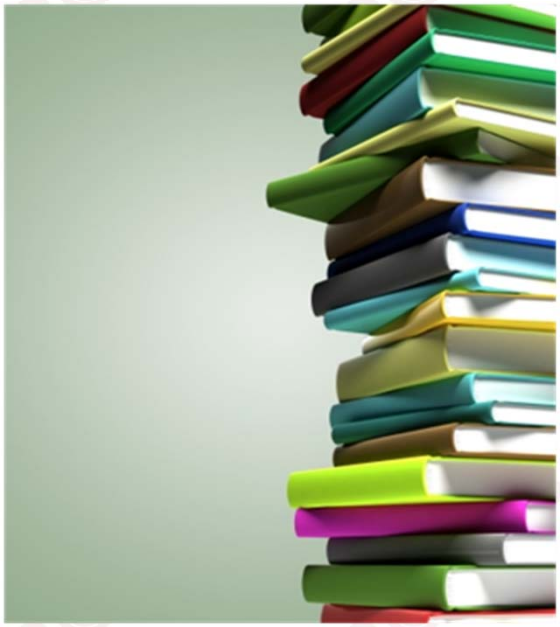
MỤC TIÊU BÀI HỌC

- Phân biệt được namespace và interface
- Liệt kê được các tính chất cơ bản của collection.
- Vận dụng ngôn ngữ C# vào triển khai namespace và interface.
- Sử dụng được các lớp đối tượng cơ bản trong thư viện collection để xây dựng một số ứng dụng đơn giản.

CÁC KIẾN THỨC CẦN CÓ

Để học được môn học này, sinh viên phải học xong các môn học:

- Cơ sở lập trình;
- Lập trình hướng đối tượng;
- Cơ sở dữ liệu;
- Hệ quản trị cơ sở dữ liệu SQL Server.



HƯỚNG DẪN HỌC

- Đọc tài liệu tham khảo.
- Thảo luận với giáo viên và các sinh viên khác về những vấn đề chưa hiểu rõ.
- Trả lời các câu hỏi của bài học.

CẤU TRÚC NỘI DUNG

3.1 Giao diện (Interface)

3.2 Collections

3.3 Namespace

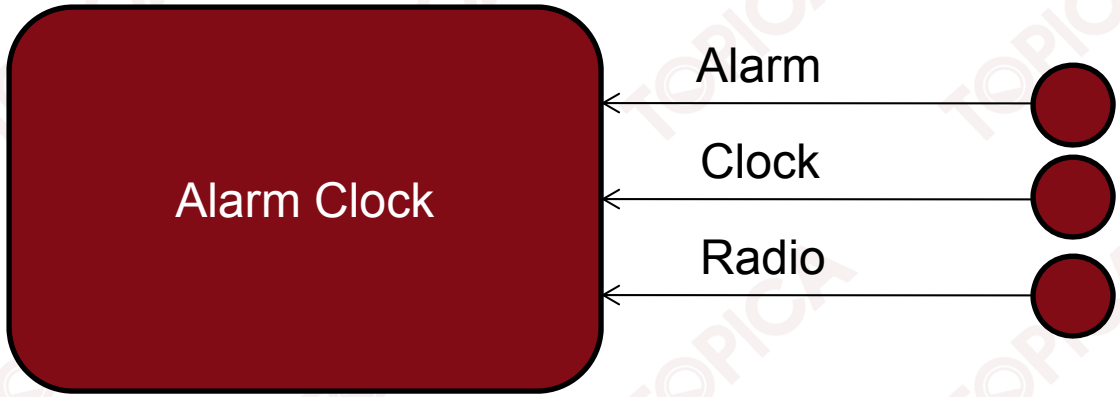
3.1. GIAO DIỆN (INTERFACE)

3.1.1. Khái niệm

3.1.2. Khai báo và khởi tạo lớp interface

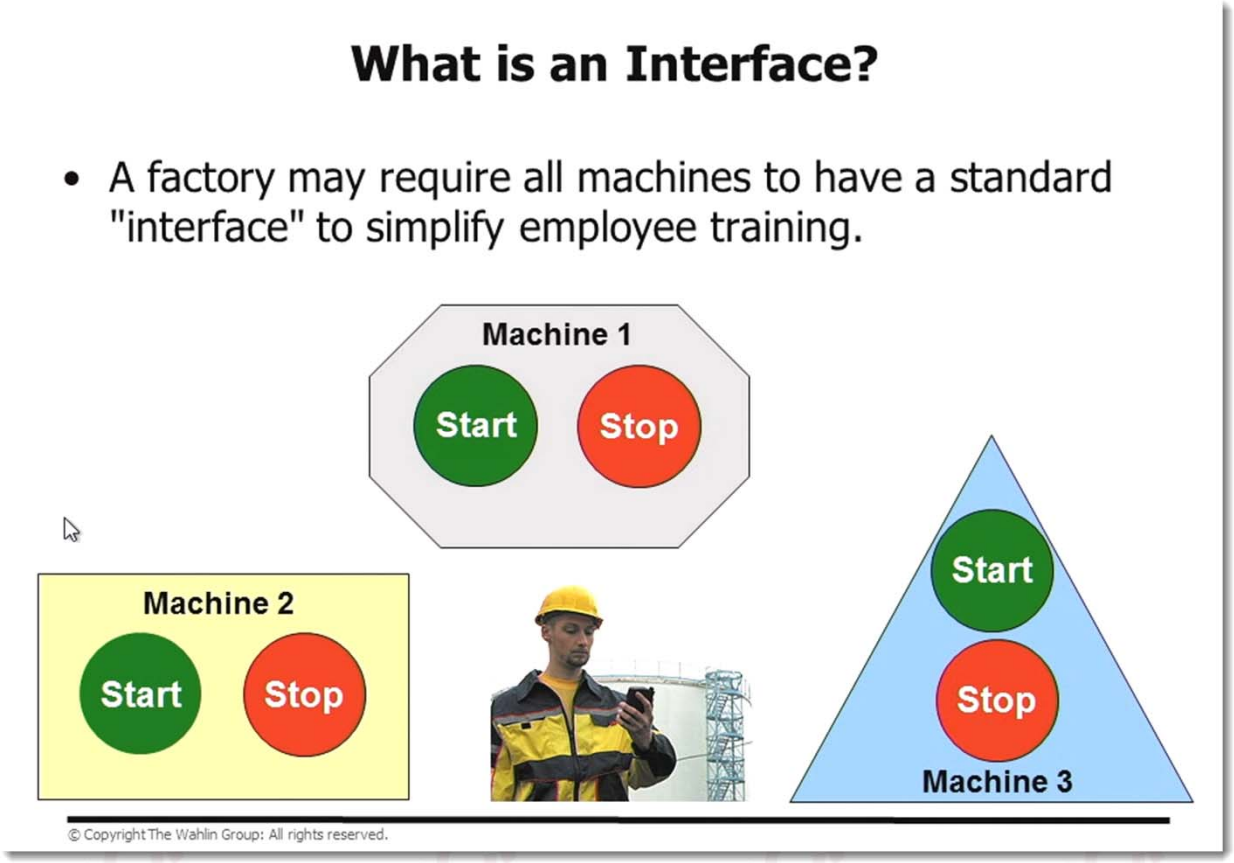
3.1.1. KHÁI NIỆM

- Giao diện (interface) cung cấp hình thức trừu tượng hóa cho việc phát triển các ứng dụng theo phương pháp xây dựng các thành phần cơ sở.
- Giao diện cung cấp các thỏa thuận chung cho phép các thành phần làm việc với nhau.
- Ví dụ: Một đồng hồ báo thức nhưng sẽ cung cấp 3 thành phần giao diện khác nhau là Alarm, Clock, Radio.



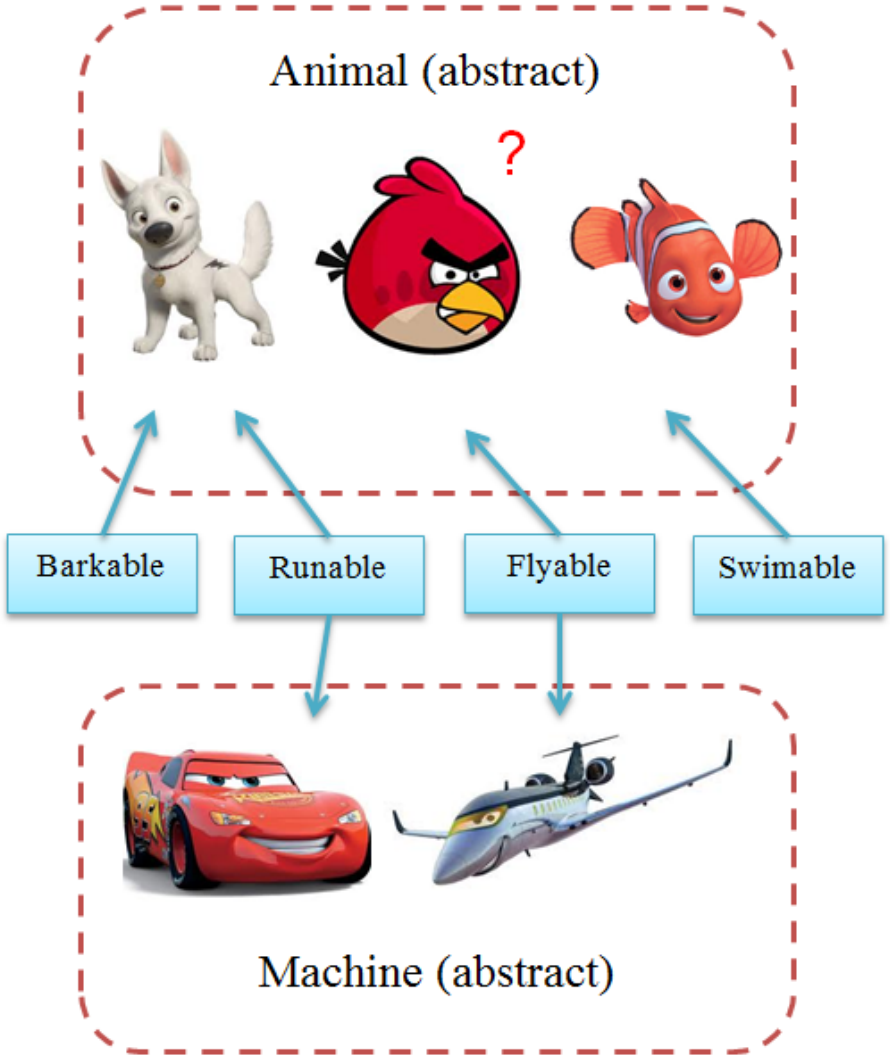
3.1.1. KHÁI NIỆM (tiếp theo)

Giao diện không phải là một lớp, do vậy không có bất cứ một triển khai cụ thể nào trong giao diện. Giao diện chỉ tạo ra các ràng buộc mà mọi lớp thực thi từ giao diện phải tuân thủ theo.



3.1.1. KHÁI NIỆM (tiếp theo)

- Giao diện sẽ khai báo một tập các thành phần. Các lớp thực thi giao diện sẽ cung cấp các định nghĩa về các thành phần đã được khai báo trong giao diện.
- Giao diện là một kiểu tham chiếu.
- Khi một lớp thực thi từ một interface thì phải cài đặt tất cả các thành phần đã khai báo trong interface.



3.1.2. KHAI BÁO VÀ KHỞI TẠO LỚP INTERFACE

- Khai báo interface không chứa các thành phần dữ liệu.
- Interface chỉ chứa các phương thức.
- Các phương thức trong interface chỉ chứa phần khai báo, không có code thực thi của phương thức.
- Các thành phần của interface có phạm vi ngầm định là public, khi khai báo không được đặt các từ khóa về phạm vi truy cập trước các phương thức.
- Cú pháp khai báo interface:

```
<Phạm vi> interface <Tên interface>
{
    //Khai báo các thành phần của interface;
}
```

- Ví dụ:

  Xây dựng giao diện Animal với các phương thức: getName(), setName(string name), makeSound(), eat().

3.1.2. KHAI BÁO VÀ KHỞI TẠO LỚP INTERFACE (tiếp theo)

- Giao diện Animal:

```
interface Animal
{
  string getName();
  void setName(string name);
  string makeSound();
  void eat();
}
```

3.1.2. KHAI BÁO VÀ KHỞI TẠO LỚP INTERFACE (tiếp theo)

- Thực thi interface

```
<Phạm vi> class <Tên lớp>: <Tên giao diện>
{
    //Khai báo các thành phần của lớp
    //Cài đặt các mô tả trong interface;
}
```

- Ví dụ: Triển khai 2 lớp Lion và Dog từ interface Animal

```
public class Lion: Animal
 {
     private string animalName;
          public Lion (string name)
          {
       animalName = name;
    }
```

3.1.2. KHAI BÁO VÀ KHỞI TẠO LỚP INTERFACE (tiếp theo)

```
public string getName()
    {
      return animalName;
    }
public void setName(string name)
    {
      animalName = name;
    }
    public string makeSound()
    {
      return "Roar";
    }
    public void eat()
    {
      Console.WriteLine("Lions Devour");
    }
  }
```

3.1.2. KHAI BÁO VÀ KHỞI TẠO LỚP INTERFACE (tiếp theo)

- Triển khai lớp Dog từ interface Animal

```
class Dog:Animal
{
  private string animalName;
  public Dog(string name)
  {
   animalName = name;
  }
  public string getName()
  {
   return animalName;
  }
```

3.1.2. KHAI BÁO VÀ KHỞI TẠO LỚP INTERFACE (tiếp theo)

```
 public void setName(string name)
 {
  animalName = name;
 }
 public string makeSound()
 {
  return "woof";
 }
 public void eat()
 {
  Console.WriteLine("Dog chews a bone");
 }
}
```

3.1.2. KHAI BÁO VÀ KHỞI TẠO LỚP INTERFACE (tiếp theo)

- Tạo lập và sử dụng đối tượng

```
static void Main(string[] args)
{
  Animal[] a = {new Lion("Dobby"), new Dog("Miky") };
  for (int i = 0; i < a.Length; i++)
   a[i].eat();
}
```

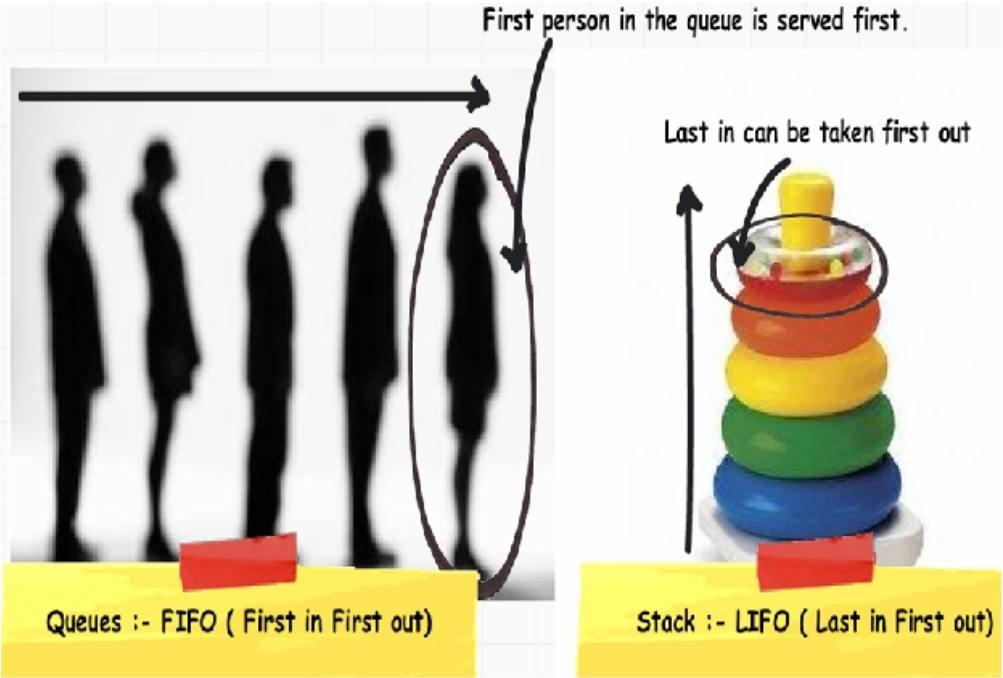
3.2. COLLECTIONS

3.2.1. Khái niệm về collections

3.2.2. Thao tác với các đối tượng trong collections

3.2.1. KHÁI NIỆM VỀ COLLECTIONS

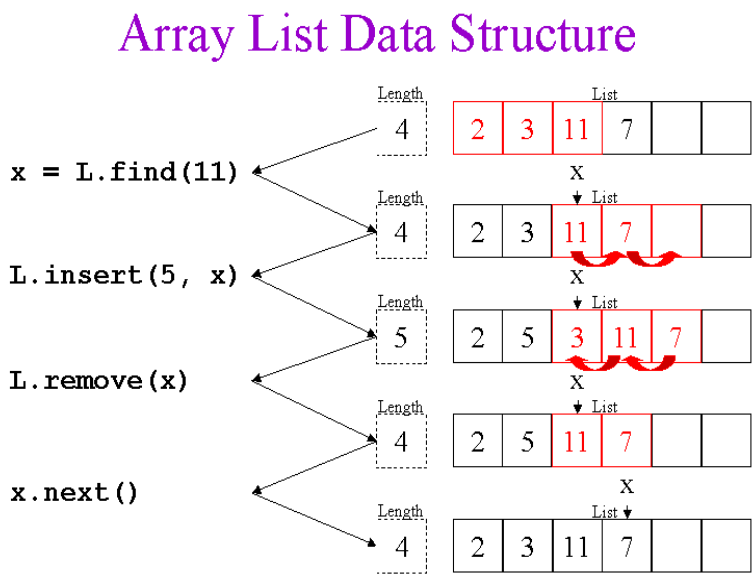
- Collections là tập hợp các lớp hỗ trợ thu thập và quản lý các đối tượng một cách tuần tự, hỗ trợ tìm kiếm, lưu trữ và duyệt các đối tượng trong tập hợp.
- Trong C# các lớp đối tượng về tập hợp được đặt trong thư viện System. Collections.
- Một số lớp đối tượng cơ bản của Collections gồm:
  - ➢ ArrayList;
  - ➢ Queue;
  - ➢ Stack;
  - ➢ Hashable;
  - ➢ Dictionary.



3.2.2. THAO TÁC VỚI CÁC LỚP ĐỐI TƯỢNG TRONG COLLECTIONS

a. Lớp ArrayList

- Là lớp được tổ chức theo cấu trúc mảng.
- Sử dụng để lưu trữ các phần tử.
- Số lượng các phần tử trong ArrayList không bị cố định như mảng (Array), số phần tử tự tăng/giảm khi thêm/xóa phần trong ArrayList.
- ArrayList cung cấp nhiều phương thức cho phép xử lí các phần tử một cách dễ dàng.
- ArrayList có thể chứa các đối tượng thuộc các class khác nhau.



3.2.2. THAO TÁC VỚI CÁC LỚP ĐỐI TƯỢNG TRONG COLLECTIONS

- Cú pháp khai báo và tạo lập ArrayList:

```
ArrayList <Tên biến mảng> = new ArrayList();
```

Hoặc:

```
ArrayList <Tên biến mảng> = new ArrayList(num); //num số nguyên, xác định số phần tử của ArrayList
```

- Một số phương thức của lớp ArrayList:
  - Phương thức Add: Thêm một phần tử vào cuối danh sách trong ArrayList

```
<Tên biến ArrayList>.Add(object value);
```

Ví dụ: Giả sử đã tạo 2 lớp Person và Staff, tạo ArrayList là staffs để chứa danh sách các nhân viên

```
ArrayList staffs = new ArrayList(20);
Person p = new Person("Hung", 20);
staffs.Add(p);
Staff s = new Staff("Nam", 22, 3, 1.33F);
staffs.Add(s);
```

  - Phương thức: Insert chèn thêm một phần tử vào vị trí bất kì trong ArrayList

```
<Tên biến ArrayList>. Insert(int index, object value);
```

3.2.2. THAO TÁC VỚI CÁC LỚP ĐỐI TƯỢNG TRONG COLLECTIONS (tiếp theo)

Ví dụ: Chèn thêm một nhân viên vào vị trí số 1

```
Staff s1 = new Staff("Bao", 25, 4, 2.33F);
staffs.Insert(1, s1);
```

Truy cập các phần tử trong ArrayList

```
<Tên biến ArrayList>[Chỉ số];
```

Ví dụ: Chương trình khởi tạo mảng các nhân viên và duyệt để in thông tin về các nhân viên

```
ArrayList staffs = new ArrayList();
Staff s = null;
s = new Staff("Hung", 32, 8, 3.33F);
staffs.Add(s);
s = new Staff("Nam", 25, 3, 1.86F);
staffs.Add(s);
for (int i = 0; i < staffs.Count; i++)
{
  s =(Staff) staffs[i];
  s.Show();
}
```

Thuộc tính Count cho biết số phần tử của ArrayList.

3.2.2. THAO TÁC VỚI CÁC LỚP ĐỐI TƯỢNG TRONG COLLECTIONS (tiếp theo)

- Phương thức Remove: Loại bỏ một phần tử khỏi ArrayList

```
<Tên biến ArrayList>.Remove(object value);
```

- Phương thức RemoveAt: Loại bỏ một phần tử ở vị trí index ra khỏi ArrayList

```
<Tên biến ArrayList>.RemoveAt(int index);
```

- Phương thức Clear: Loại bỏ tất cả các phần tử ra khỏi ArrayList

```
<Tên biến ArrayList>.Clear()
```

- Phương thức IndexOf: Trả về vị trí của phần tử value trong ArrayList

```
<Tên biến ArrayList>.IndexOf(object value);
```

- Phương thức Contains: Kiểm tra xem có tồn tại phần tử value trong ArrayList hay không?

```
<Tên biến ArrayList>. Contains(object value);
```

3.2.2. THAO TÁC VỚI CÁC LỚP ĐỐI TƯỢNG TRONG COLLECTIONS (tiếp theo)

b. Lớp Queue

- Là cấu trúc danh sách tuần tự cho phép truy xuất các phần tử theo phương thức First – in, First – out (thêm phần tử vào cuối danh sách, lấy phần tử ở đầu danh sách).
- Cách tạo Queue:

```
Queue <Tên biến> = new Queue();
```

- Thuộc tính: Count cho biết số phần tử trong Queue.
- Phương thức:
  - EnQueue: Thêm một phần tử vào cuối danh sách;
  - DeQueue: Loại bỏ một phần tư khỏi danh sách;
  - Peek: Lấy một phần tử ở đầu danh sách.
- Ví dụ: Sử dụng Queue lưu một danh sách các nhân viên sau đó duyệt Queue để đọc thông tin về từng nhân viên trong danh sách.

3.2.2. THAO TÁC VỚI CÁC LỚP ĐỐI TƯỢNG TRONG COLLECTIONS (tiếp theo)

```
Queue staffs= new Queue();
Staff s1 = new Staff("Hung", 32, 8, 3.33F);
Staff s2 = new Staff("Nam", 25, 3, 1.86F);
Staff s3 = new Staff("Bao", 28, 5, 1.86F);
staffs.Enqueue(s1);
staffs.Enqueue(s2);
staffs.Enqueue(s3);
foreach (Staff s in staffs)
{
  s.Show();
}
```

3.2.2. THAO TÁC VỚI CÁC LỚP ĐỐI TƯỢNG TRONG COLLECTIONS (tiếp theo)

c. Lớp Stack

- Là cấu trúc danh sách tuần tự cho phép truy xuất các phần tử theo phương thức Last – in, First – out (thêm và lấy phần tử đều thực hiện ở đỉnh Stack).
- Cách tạo Stack:

```
Stack<Tên biến> = new Stack();
```

- Thuộc tính: Count cho biết số phần tử trong Stack.
- Phương thức:
  - Push: Thêm một phần tử vào Stack;
  - Pop: Lấy một phần tử ra khỏi Stack (và loại bỏ phần tử khỏi Stack);
  - Peek: Lấy một phần ra khỏi Stack (không loại bỏ phần tử khỏi Stack).

3.2.2. THAO TÁC VỚI CÁC LỚP ĐỐI TƯỢNG TRONG COLLECTIONS (tiếp theo)

d. Lớp Hashtable

- Là một cấu trúc bảng băm và sử dụng để ánh xạ một khóa (key) vào một giá trị (value).
- Mỗi phần tử trong danh sách có một key duy nhất.
- Cách khai báo và tạo đối tượng Hashtable:

```
Hashtable <Tên biến> = new Hashtable();
```

- Thuộc tính: Count cho biết số phần tử của Hashtable.
- Phương thức:
  - Add: Thêm một phần tử với khóa key và giá trị value vào Hashtable

    ```
    <Tên biến Hashtable>.Add(object key, object value);
    ```

  - Remove: Loại bỏ một phần tử với khóa key khỏi Hashtable

    ```
    <Tên biến Hashtable>.Remove(object key);
    ```

  - Clear: Loại bỏ tất cả các phần tử khỏi Hashtable

    ```
    <Tên biến Hashtable>.Clear();
    ```

  - Contains: Kiểm tra xem có phần tử với khóa key trong Hashtable hay không

    ```
    <Tên biến Hashtable>. Contains(object key);
    ```

  - ContainsValue: Kiểm tra xem có tồn tại phần tử với giá trị value trong Hashtable hay không?

    ```
    <Tên biến Hashtable>. ContainsValue(object value);
    ```

  - GetEnumerator: Trả về giá trị kiểu IDictionaryEnumerator, là tập tất cả các phần tử trong Hashtable.

3.2.2. THAO TÁC VỚI CÁC LỚP ĐỐI TƯỢNG TRONG COLLECTIONS (tiếp theo)

- Ví dụ: Chương trình lưu danh sách các nhân viên vào bảng băm Hashtable và duyệt bảng băm để in ra thông tin các nhân viên

```
Hashtable staffs = new Hashtable();
Staff s1 = new Staff("Hung", 32, 8, 3.33F);
Staff s2 = new Staff("Nam", 25, 3, 1.86F);
Staff s3 = new Staff("Bao", 28, 5, 1.86F);
staffs.Add(1, s1);
staffs.Add(2, s2);
staffs.Add(3, s3);
IDictionaryEnumerator enumerator = staffs.GetEnumerator();
while (enumerator.MoveNext())
{
 int key =(int) enumerator.Key;
  Staff s = (Staff)enumerator.Value;
  Console.WriteLine("ID:" + key);
  s.Show();
}
```

3.3. NAMESPACE

- Namespace giúp quản lí các class, interface, structure.
- Namespace tránh việc trùng lặp giữa các class, interface
- Trong các namespace có thể có các class hay interface trùng tên nhau
- Các thành phần trong namespace: interface, structure, class, enumeration.
- Đặc trưng và lợi ích của namespace:
  - Cung cấp một cấu trúc thứ bậc, tạo điều kiện thuận tiện trong quản lý và sử dụng các lớp;
  - Cho phép có nhiều lớp trùng tên trong các namespace khác nhau;
  - Tạo nên hệ thống các bộ phận tháo rời.
- Một số namespace của hệ thống:
  - System.Data: Chứa các lớp tạo lên kiểu cấu trúc ADO.NET, cho phép truy xuất đến các nguồn dữ liệu khác nhau.
  - System.IO: Bao gồm các class cho phép đọc hoặc viết từ các luồng dữ liệu lên tệp tin.
  - System.NET: Bao gồm các class cho phép tạo các ứng dụng Web cơ sở.

3.3. NAMESPACE

- Cú pháp khai báo namespace:

```
namespace <Tên namespace>
{
   //khai báo các interface
   //Khai báo các class
}
```

- Ví dụ: Tạo 2 namespace là BankTypes và ClubTypes, mỗi namespace đều chứa các lớp id.

3.3. NAMESPACE (tiếp theo)

```
namespace BankTypes
{
public class Id
{
  string _identifier;
  public Id(string Identifier)
  {
   _identifier = Identifier;
  }
  string GetId()
  {
   return _identifier;
  }
 }
}
```

3.3. NAMESPACE (tiếp theo)

```
namespace ClubTypes
{
  public class Id
  {
    string _identifier;
    public Id(string Identifier)
    {
      _identifier = Identifier;
    }
    string GetId()
    {
      return _identifier;
    }
  }
}
```

- Truy xuất đến các thành phần trong namespace

  `<Tên namespace>.<Tên thành phần>;`

  Ví dụ:

  `BankTypes.Id bank = new BankTypes.Id("09-887-667");`

TÓM LƯỢC CUỐI BÀI

Trong bài này, chúng ta đã nghiên cứu các nội dung chính sau:

- Interface và cách triển khai interface trong C#;
- Namespace và cách triển khai namespace trong C#;
- Collections và một số lớp đối tượng cơ bản trong collections.