

# LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG NÂNG CAO VỚI C#

## A. MỤC TIÊU:

- ✓ Hướng dẫn SV nâng cao kỹ năng xây dựng lớp đối tượng trong C#
- ✓ Xây dựng giao diện, kế thừa và thực thi giao diện.
- ✓ Nâng cao: SV tự nghiên cứu kỹ thuật dùng phương thức Sort được cài sẵn trong lớp ArrayList, thực thi giao diện chuẩn IComparable của .NET..., SV làm quen với việc sử dụng giao diện IComparer, hỗ trợ sắp xếp theo nhiều tiêu chuẩn khác nhau.
- ✓ Hướng dẫn SV sử dụng File Stream I/O để lưu trữ thông tin của đối tượng, nhập xuất file bao gồm FileStream, StreamWriter và StreamReader.
- ✓ Sử dụng các tham số có kiểu enumeration của FileStream như: FileMode, FileAccess.
- ✓ Dùng phương thức tiện ích Split của lớp string để chia chuỗi lớn thành các chuỗi con theo ký tự phân cách được cung cấp.

## B. NỘI DUNG:

**Bài tập 1:** Xây dựng một ứng dụng Console cơ bản quản lý danh sách các cuốn sách, mỗi cuốn sách này chứa các thông tin như sau: tên sách, tên tác giả, nhà xuất bản, năm xuất bản, số hiệu ISBN (International Standard Book Number) và danh mục các chương sách (chỉ chứa tên chương).


*Thực hiện theo các yêu cầu sau:*

- ✓ Xây dựng một interface có tên là **IBook**, mô tả property và method cần thiết cho các lớp dạng Book thực thi.
- ✓ Xây dựng lớp Book kế thừa từ IBook, thực hiện các mô tả trong IBook và các chi tiết riêng của Book.
- ✓ Xây dựng lớp BookList quản lý danh sách các đối tượng Book, lớp này chứa các thao tác trên danh sách các đối tượng Book.
- ✓ Thực thi giao diện IComparable, định nghĩa quan hệ thứ tự trong phương thức CompareTo...
- ✓ Sử dụng giao diện IComparer, hỗ trợ sắp xếp theo nhiều tiêu chuẩn khác nhau...
- ✓ Viết hàm Main thực thi yêu cầu sau:
  - Cho nhập vào một mảng chứa những cuốn sách.
  - Xuất danh sách thông tin những cuốn sách.
  - Lần lượt xuất danh sách ra theo thứ tự được sắp theo tên tác giả, tên sách, năm xuất bản.

*Hướng dẫn:*

- ✓ **Bước 1:** Tạo ứng dụng Console có tên BookManaging
- ✓ **Bước 2:** Tạo giao diện IBook được minh họa như hình 1, gồm các mô tả
  - Một chỉ mục
  - Property Title

- Property Author
- Property Publisher
- Property Year
- Property ISBN
- Một phương thức void Show() không tham số



```
/// <summary>
/// Mô tả giao diện chung
/// cho các loại sách
/// </summary>
interface IBook
{
    /// <summary>
    /// Mô tả indexer chương sách
    /// </summary>
    string this[int index]
    {
        get;
        set;
    }
    /// <summary>
    /// Property Title đại diện cho
    /// tên của sách
    /// </summary>
    string Title
    {
        get;
        set;
    }
    /// <summary>
    /// Property Author đại diện cho
    /// tên của tác giả cuốn sách
    /// </summary>
    string Author
    {
        get;
        set;
    }
}

/// <summary>
/// Publisher đại diện cho tên của NXB
/// </summary>
string Publisher
{
    get;
    set;
}
/// <summary>
///
/// </summary>
string ISBN
{
    get;
    set;
}
/// <summary>
/// Property Year: năm xuất bản
/// </summary>
int Year
{
    get;
    set;
}
/// <summary>
/// Mô tả phương thức hiển thị
/// thông tin cuốn sách
/// </summary>
void Show();
} // end of IBook
```

Hình 1: Mô tả giao diện IBook

- ✓ **Bước 3:** Định nghĩa lớp Book, lớp này có thực thi giao diện IBook
- ```
class Book : IBook
{
    // nội dung của lớp Book
}
```

Khai báo các field cho lớp Book

```
class Book : IBook
{
    // dữ liệu thành viên
    #region Định nghĩa dữ liệu

    private string isbn;
    private string title;
    private string author;
    private string publisher;
    private int year;
    // lưu trữ mảng chứa tên chương
    private ArrayList chapter = new ArrayList();

    #endregion
}
```

Hình 2: Phần khai báo data member

Thực thi lần lượt các property mô tả trong IBook, hình 3 minh họa một cách cài đặt bộ chỉ mục cho các chương sách

```
#region Thực thi giao diện IBook

public string this[int index] // indexer chương sách
{
    get
    {
        if (index >= 0 && index < chapter.Count)
            return (string)chapter[index];
        else // phát sinh ngoại lệ
            throw new IndexOutOfRangeException();
    }
    set
    {
        if (index >= 0 && index < chapter.Count)
            chapter[index] = value; // cập nhật lại chương
        else if (index == chapter.Count)
            chapter.Add(value); // thêm chương mới
        else // phát sinh ngoại lệ
            throw new IndexOutOfRangeException();
    }
}

public string Title...
public string Author...
public string Publisher...
public string ISBN...
public int Year...
```

Hình 3: Mô tả phần thực thi giao diện IBook

Định nghĩa phương thức Show (phương thức này có mô tả trong IBook)

```
public void Show()
{
    // xuất thông tin sách ra màn hình console
    Console.WriteLine("-----");
    Console.WriteLine("Title: " + title);
    Console.WriteLine("Author: " + author);
    Console.WriteLine("Publisher: " + publisher);
    Console.WriteLine("Year: " + year);
    Console.WriteLine("ISBN: " + isbn);
    Console.WriteLine("Chapter: ");
    for (int i = 0; i < chapter.Count; i++)
        Console.WriteLine("\t{0}: {1}", i + 1, chapter[i]);
    Console.WriteLine("-----");
}
```

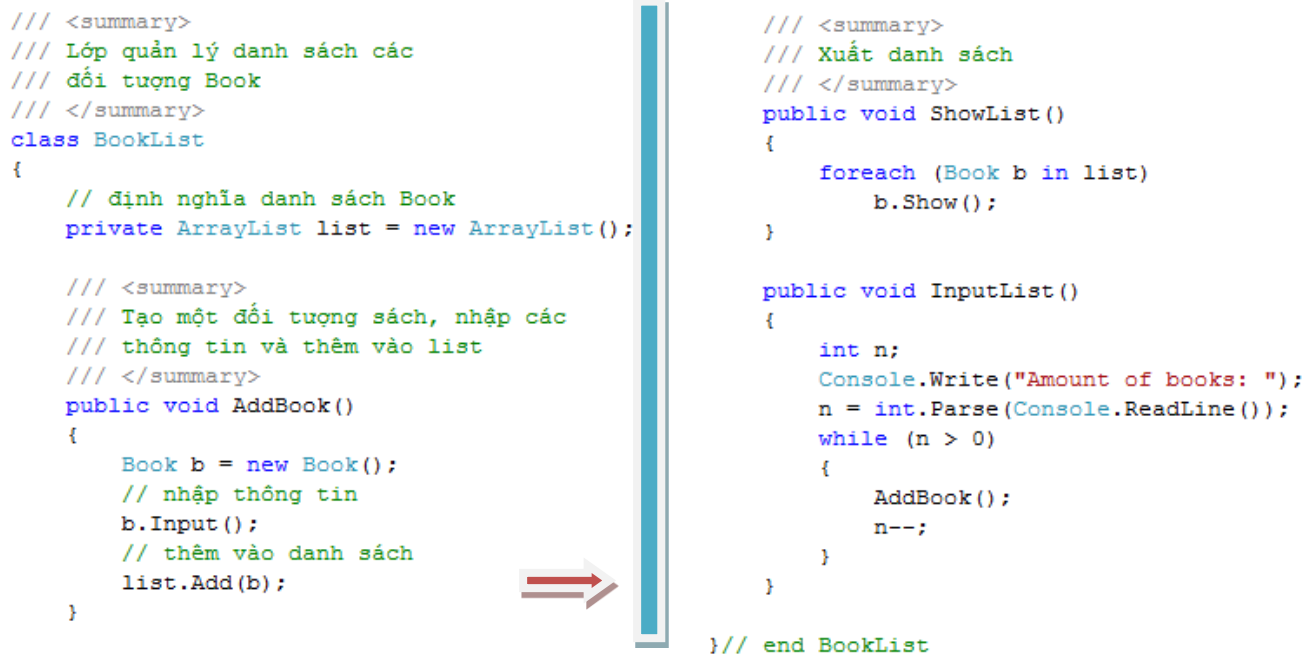
Hình 4: Phần định nghĩa phương thức Show

Định nghĩa phương thức Input()

```
public void Input()
{
    Console.Write("Title: ");
    title = Console.ReadLine();
    Console.Write("Author: ");
    author = Console.ReadLine();
    Console.Write("Publisher: ");
    publisher = Console.ReadLine();
    Console.Write("ISBN: ");
    isbn = Console.ReadLine();
    Console.Write("Year: ");
    year = int.Parse(Console.ReadLine());
    Console.WriteLine("Input chapter (finished with empty string)");
    string str;
    do // nhập lần lượt các chương sách
    {
        str = Console.ReadLine();
        if (str.Length > 0)
            chapter.Add(str);
    } while (str.Length > 0); // kết thúc khi chuỗi rỗng
}
```

Hình 5: Phần định nghĩa phương thức Input

- ✓ **Bước 4:** Tạo lớp BookList để quản lý danh sách các đối tượng Book, đây là dạng container class



```

/// <summary>
/// Lớp quản lý danh sách các
/// đối tượng Book
/// </summary>
class BookList
{
    // định nghĩa danh sách Book
    private ArrayList list = new ArrayList();

    /// <summary>
    /// Tạo một đối tượng sách, nhập các
    /// thông tin và thêm vào list
    /// </summary>
    public void AddBook()
    {
        Book b = new Book();
        // nhập thông tin
        b.Input();
        // thêm vào danh sách
        list.Add(b);
    }

    /// <summary>
    /// Xuất danh sách
    /// </summary>
    public void ShowList()
    {
        foreach (Book b in list)
            b.Show();
    }

    public void InputList()
    {
        int n;
        Console.Write("Amount of books: ");
        n = int.Parse(Console.ReadLine());
        while (n > 0)
        {
            AddBook();
            n--;
        }
    }
}
    
```

Hình 6: Định nghĩa lớp container BookList

✓ **Bước 5:** Tạo đoạn code demo như sau

```

class Program
{
    static void Main(string[] args)
    {
        // tạo một đối tượng BookList
        BookList bl = new BookList();

        bl.InputList();

        bl.ShowList();

        Console.ReadLine();
    }
}
    
```

Hình 7: Phần code demo trong hàm Main()

## Bài tập 2:

Bổ sung chức năng hỗ trợ để sắp xếp danh sách book theo một thứ tự nào đó, ví dụ sắp danh sách theo thứ tự alphabet của title, thứ tự theo author, thứ tự theo publisher, thứ tự theo năm...

Có 2 cách thực hiện:

- ✓ Thực thi giao diện IComparable
- ✓ Sử dụng giao diện Comparer, tạo các lớp hỗ trợ sắp xếp theo các tiêu chuẩn khác nhau

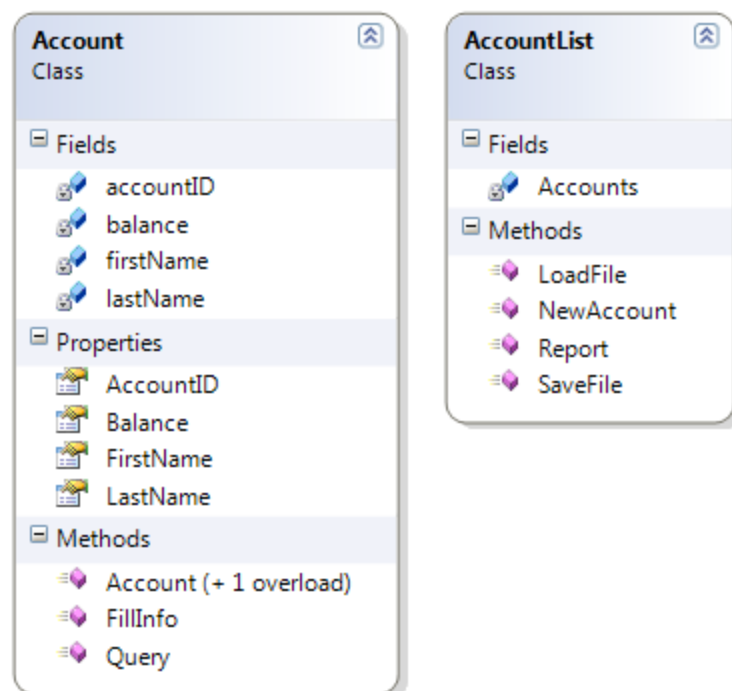
## Bài tập 3:

- Tạo một lớp **Account** chứa các thông tin tài khoản ngân hàng như sau:
  - Account ID: mã số tài khoản
  - First Name
  - Last Name
  - Balance: số dư tài khoản

- Viết các phương thức constructor, phương thức hiển thị thông tin tài khoản, phương thức nhập thông tin tài khoản (từ bàn phím).
- Tạo lớp **AccountList** chứa danh sách các Account, sử dụng ArrayList để lưu trữ danh sách này. Viết các phương thức sau
  - **NewAccount**: thêm một account mới vào danh sách
  - **SaveFile**: lưu danh sách account vào file
  - **LoadFile**: lấy danh sách account từ file vào danh sách
  - **Report**: xuất ra màn hình tất cả danh sách các account

**Hướng dẫn:**

- ✓ **Bước 1:** Tạo ứng dụng Visual C# - Windows - Console Application.
- ✓ **Bước 2:** Tạo lớp **Account** có các data member như mô tả bên trên, tạo các property cho các field đó, tạo các phương thức constructor, có hai phương thức thành viên tiện ích là:
  - **FillInfo**: cho phép nhập thông tin của account từ bàn phím: bao gồm Account ID, First Name, Last Name, Balance.
  - **Query**: hiển thị thông tin của tài khoản {account ID, First Name, Last Name, Balance}
- ✓ **Bước 3:** Tạo lớp AccountList chứa danh sách các đối tượng Account, sử dụng ArrayList cho danh sách đối tượng này. Bổ sung thêm các phương thức tiện ích như sau
  - **NewAccount**: tạo mới account và đưa vào danh sách
  - **SaveFile**: cho phép user nhập vào tên file, và thực hiện việc lưu danh sách account này vào file trên.
  - **LoadFile**: từ một tên file do user nhập vào, đọc lần lượt các account và đưa vào danh sách các account.



Hình 1: Sơ đồ lớp Account và AccountList

Phần chức năng **SaveFile** của lớp **AccountList** có thể được làm như sau:

```

public void SaveFile()
{
    // nhập tên file
    Console.WriteLine("Input file name to save: ");
    string filename = Console.ReadLine();
    // đọc file
    try
    {
        // tạo luồng truy cập file
        FileStream output = new FileStream(filename,
            FileMode.CreateNew, FileAccess.Write);

        // thiết lập writer
        StreamWriter writer = new StreamWriter(output);

        // duyệt qua từng đối tượng trong Accounts
        foreach (Account acc in Accounts)
        {
            // lưu các thông tin của một account trên dòng, phân
            // cách nhau bằng dấu ,
            writer.WriteLine("{0},{1},{2},{3}", acc.AccountID,
                acc.FirstName, acc.LastName, acc.Balance);
        }
        // đóng kết nối
        writer.Close();
        output.Close();
    }
    catch (IOException e) // phát sinh ngoại lệ nếu có
    {
        Console.WriteLine(e.Message);
    }
}
}

```

Chức năng **LoadFile** có thể được thực hiện như sau:

```

public void LoadFile()
{
    // đọc tên file chứa dữ liệu
    Console.WriteLine("Input file name to load: ");
    string filename = Console.ReadLine();
    // xóa danh sách
    Accounts.Clear();
    try
    {
        // tạo luồng đọc file
        FileStream input = new FileStream(filename,
            FileMode.Open, FileAccess.Read);
        StreamReader reader = new StreamReader(input);

        string str;
        // đọc qua từng dòng, kết thúc khi chuỗi null
        while ((str = reader.ReadLine()) != null)
        {
            // tách chuỗi đọc ra thành các chuỗi con
            // có phân cách nhau dấu ','
            string[] list = str.Split(',');

            // tạo đối tượng Account mới
            Account acc = new Account(int.Parse(list[0]),
                list[1], list[2], decimal.Parse(list[3]));
            // đưa đối tượng mới đọc vào danh sách account
            Accounts.Add(acc);
        }
    }
}

```

```
// đóng luồng
input.Close();
reader.Close();
} // try
catch (IOException e)
{
    Console.WriteLine(e.Message);
}
} //end method
```

✓ **Bước 4:** Tạo phương thức Main thực hiện theo mô tả như sau:

- Tạo một vòng lặp chờ user nhập lệnh,
- Bao gồm các lệnh sau: Add, Save, Load, Report, Exit.
- Mỗi lệnh sẽ thực hiện chức năng tương ứng,
  - Nhập “Add” thì chương trình gọi chức năng “thêm một account vào danh sách”.
  - Nhập “Save” thì sẽ gọi chức năng lưu danh sách account vào file...
  - Nhập “Load” gọi chức năng đọc file dữ liệu account trên đĩa
- Lệnh Exit sẽ kết thúc chương trình.

**Bài tập 4:**

- ✓ Bổ sung thêm chức năng Remove xóa một account ra khỏi danh sách. Sử dụng BinarySearch của ArrayList để xác định chỉ mục của đối tượng có khóa nào đó, theo tiêu chí so sánh trong các lớp IComparer được xây dựng hỗ trợ cho Account.
- ✓ Sắp xếp danh sách theo thứ tự tăng dần của Account ID, First Name, Balance.
- ✓ Sinh viên tìm hiểu Serialization và sử dụng để lưu trữ các đối tượng account thay thế cho File I/O cơ bản bên trên.

**-----Hết Lab 02----**