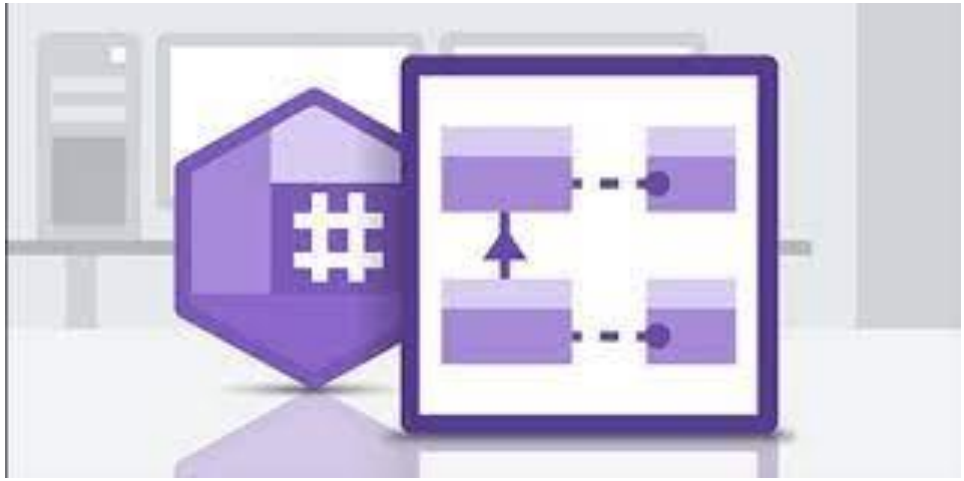
C# 8 – Lớp trừu tượng và giao diện

Giảng viên: **ThS. Lê Thiện Nhật Quang**

Email: quangltn.dotnet.vn@gmail.com

Website: http://dotnet.edu.vn

Điện thoại: **0868.917.786**



MỤC TIÊU

- Giải thích lớp trừu tượng - abstract
- Giải thích giao diện - interface
- So sánh lớp trừu tượng và giao diện

1.1. LỚP TRỪU TƯỢNG - ABSTRACT

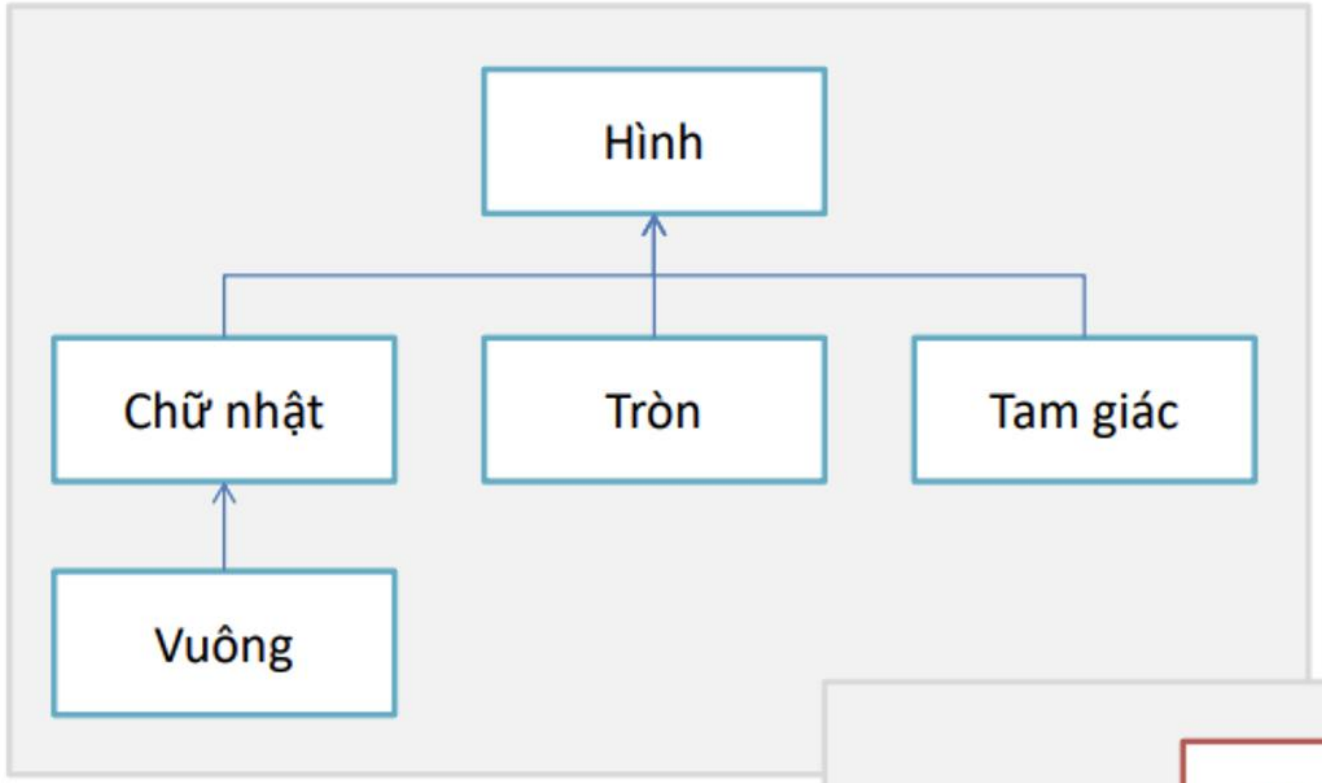
❑Lớp trừu tượng là lớp có các hành vi chưa được xác định rõ

❖Ví dụ 1: Đã là hình thì chắc chắn là có diện tích và chu vi nhưng chưa xác định được cách tính mà phải là một hình cụ thể như chữ nhật, tròn, tam giác… mới có thể xác định cách tính

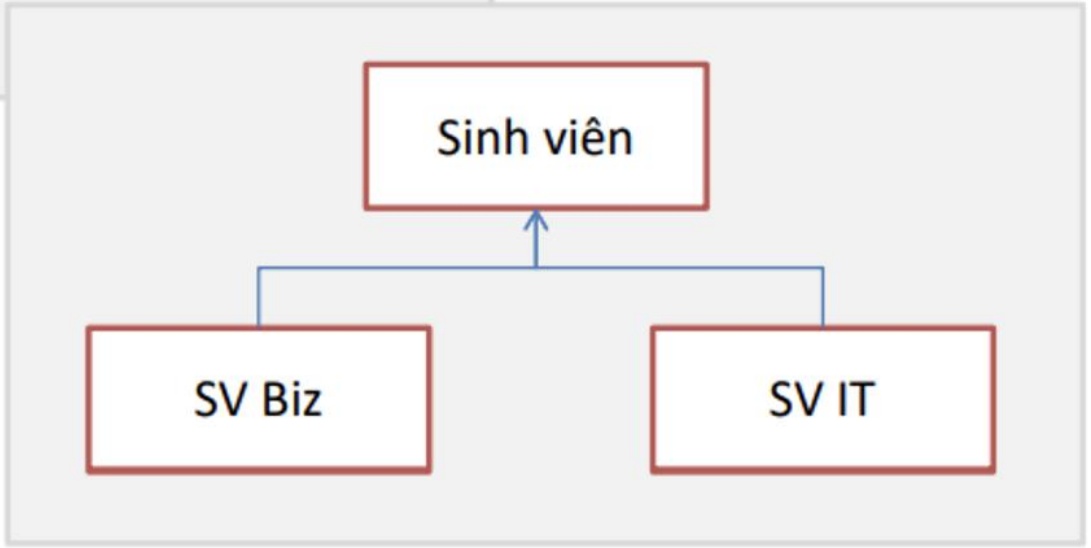
❖Ví dụ 2: Sinh viên thì chắc chắn có điểm trung bình nhưng chưa xác định được cách tính như thế nào mà phải là sinh viên của ngành nào mới biết được môn học và công thức tính điểm cụ thể.

❑Vậy lớp hình và lớp sinh viên là các lớp trừu tượng vì phương thức tính chu vi, diện tích và tính điểm chưa thực hiện được.

1.1. LỚP TRỪU TƯỢNG – ABSTRACT (2)



Hình và Sinh viên là các lớp trừu tượng



Chữ nhật, Tròn, Tam giác, Vuông, SV IT, SV Biz là các lớp cụ thể

1.2. ĐỊNH NGHĨA LỚP TRỪU TƯỢNG

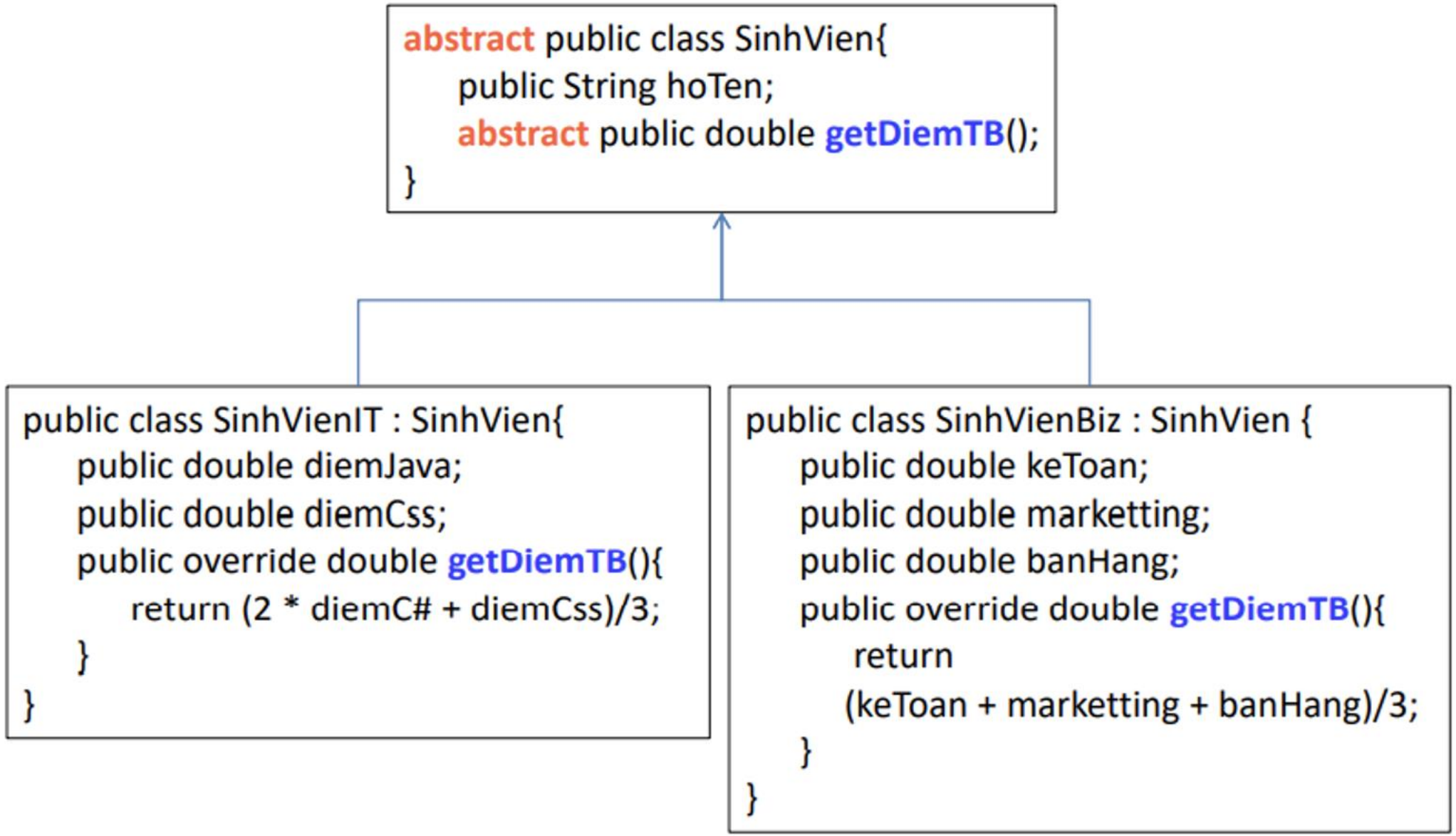
```
abstract public class MyClass{
    abstract public type MyMethod();
}
```

Sử dụng từ khóa abstract để định nghĩa lớp và phương thức trừu tượng

```
abstract public class SinhVien{
    abstract public double getDiemTB();
}
```

```
abstract public class Hinh{
    abstract public double getChuVi();
    abstract public double getDienTich();
}
```

1.2. ĐỊNH NGHĨA LỚP TRỪU TƯỢNG (2)



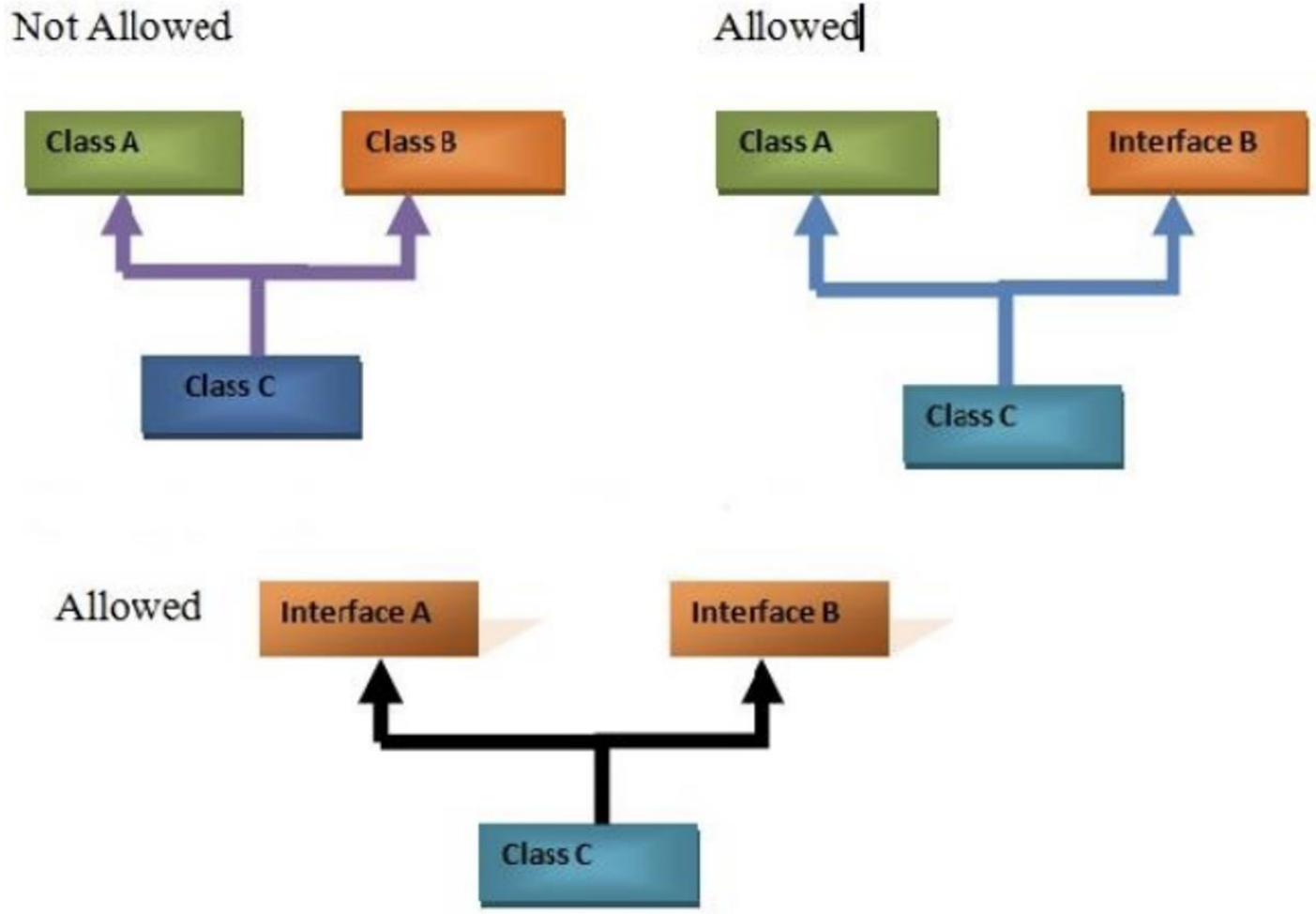
1.2. ĐỊNH NGHĨA LỚP TRỪU TƯỢNG (3)

- Từ khóa abstract được sử dụng để định nghĩa lớp và phương thức trừu tượng
- Phương thức trừu tượng là phương thức không có phần thân xử lý và được khai báo bằng từ khóa abstract.
- Lớp chứa phương thức trừu tượng thì lớp đó phải là lớp trừu tượng.
- Trong lớp trừu tượng có thể định nghĩa các phương thức cụ thể hoặc khai báo các trường
- Không thể sử dụng new để tạo đối tượng từ lớp trừu tượng.

2.1. INTERFACE

❑C# hỗ trợ sử dụng interface giải quyết nhiều vấn đề.

❑Interface được xem như là một lớp, lớp đó có thể được một class hoặc struct khác implement nó



2.1. INTERFACE (2)

Cú pháp khai báo interface

```
<access_modifier> interface <tên_interface> <: tên_base_interface>
{

    // interface member

}
```

- **<access_modifier>** : là public hoặc internal, nếu không ghi rõ mặc định là internal.
- **interface** : từ khóa khai báo một interface.
- **<tên_interface>** : thường bắt đầu bằng chữ **I**, ví dụ: IShape, IAnimal, IStudent,...
- **<: tên_base_interface>** : trường hợp có implement từ interface khác thì dấu 2 chấm **:** biểu thị sự implement, tiếp theo sau là tên base interface.

```
// Khai báo interface IPeople
interface IPeople
{
    // interface member
}

// Khai báo interface IStudent, implement từ interface IPeople
public interface IStudent : IPeople
{
    // interface member
}
```

2.1. INTERFACE (3)

❑Các lưu ý khi khai báo interface

❖Khai báo interface chỉ chứa các khai báo của các non-static function member (phương thức (methods), thuộc tính (properties), sự kiện (events), chỉ mục (indexers))

❖Chỉ khai báo các thành phần, không khai báo code thực thi, không định nghĩa nội dung code

❖Các thành phần không được sử dụng trong interface là: constructor, destructor, field, hằng, thành phần static.

❖Không thể khai báo hay chỉ định phạm vi truy cập (access modifiers) cho các thành phần bên trong interface. Các thành phần này sẽ mặc định là public

2.1. INTERFACE (4)

Khai báo một interface IDung, bao gồm các thành phần ĐÚNG:

```
interface IDung
{
    // method
    0 references
    void xyz();

    // property
    0 references
    string name { get; set; }

    // indexer
    0 references
    double this[int index] { get; set; }

    // event
    event EventHandler OnChanged;
}
```

2.1. INTERFACE (5)

Khai báo một interface ISai, bao gồm các thành phần SAI:

```
interface ISai
{
    // Không được sử dụng hàm [constructor]
    // error CS0526: Interfaces cannot contain constructors
    0 references
    ISai() { }
    // Không được sử dụng hàm [destructor]
    // error CS0575: Only class types can contain destructors
    0 references
    ~ISai() { }
    // Không được chỉ định phạm vi truy cập (access modifiers)
    // error CS0106: The modifier 'public' is not valid for this item
    0 references
    public void xyz();
    // error CS0106: The modifier 'protected' is not valid for this item
    0 references
    protected void abc();
    // Không được khai báo [field]
    // error CS0525: Interfaces cannot contain fields
    int number;
    // Không được khai báo hằng [const]
    // error CS0525: Interfaces cannot contain fields
    public const double PI = 3.14;
    // Không được định nghĩa hàm
    // error CS0531: 'ISai.xyz()': interface members cannot have a definition
    0 references
    void xyz()
    {
        Console.WriteLine("Print xyz");
    }
    // Không được sử dụng thành phần [static]
    // error CS0106: The modifier 'static' is not valid for this item
    0 references
    static void xyz();
}
```

2.2. THỰC THI INTERFACE

❖Chỉ có các class/ struct có thể thực thi interface

❖Khi thực thi interface, các class/ struct phải:

➢Thêm khai báo interface vào khai báo class/ struct

➢Thực thi tất cả các thành phần của interface trong class/ struct

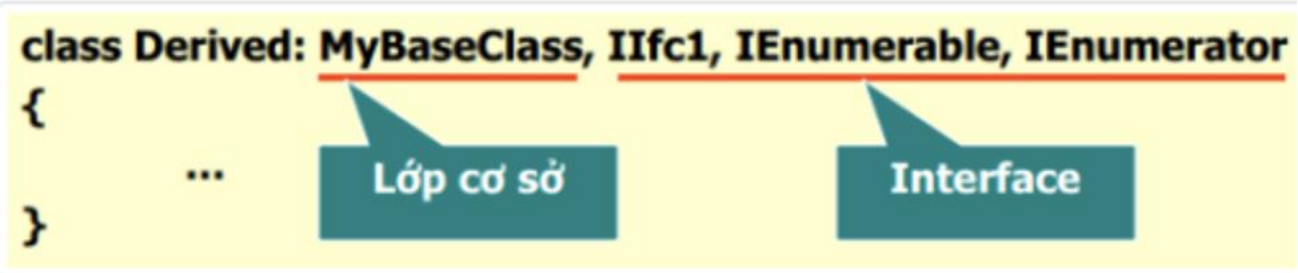
```
interface MyInterface1{
    int DoStuff(int nVar1 int nVar2);
    double DoOtherStuff(string s,long x);
}
```

```
class MyClass : MyInterface1
{
    int DoStuff(int nVar1, long nVar2){
        // implement code
    }
    double DoOtherStuff(string s, long x){
        // implement code
    }
}
```

2.2. THỰC THI INTERFACE (2)

❖Nếu một lớp thực thi interface, nó phải thực thi hết tất cả các thành phần của interface

❖Nếu lớp được dẫn xuất từ lớp cơ sở và có thực thi interface : tên lớp cơ sở phải đặt trước các tên interface trong khai báo lớp



❖Mỗi interface không được phép kế thừa từ một class nào cả

❖Không thể tạo ra một đối tượng từ interface

2.3. INTERFACE LÀ KIỂU THAM CHIẾU

❖Không thể truy xuất interface trực tiếp thông qua các thành phần của đối tượng thuộc lớp

❖Có thể tham chiếu đến interface thông qua ép kiểu của đối tượng thuộc lớp sang kiểu interface

```
interface IIfc1{
        void PrintOut(string s);
}
class MyClass: IIfc1{
        public void PrintOut(string s){
                Console.WriteLine("Calling through: {0}",s);
        }
}
...
MyClass mc=new MyClass();
mc.PrintOut("Object");
IIfc1 ifc=(IIfc1)mc;
ifc.PrintOut("Interface");
...
```

Calling through: Object
Calling through: Interface

Ép kiểu

Toán tử .

2.4. KẾ THỪA INTERFACE

❖Một interface có thể được kế thừa từ 1/nhiều interface

❖Chỉ định 1 interface kế thừa từ các interface khác:

➢Thêm danh sách các interface kế thừa vào khai báo interface

➢Các interface cách nhau bởi dấu phẩy (,)

```
interface IDataRetrieve{
        int GetData();
}
interface IDataStore{
        void SetData(int s);
}
interface IDataIO: IDataRetrieve, IDataStore{
}
class MyData: IDataIO{
        int nPrivateData;
        public int GetData(){
                return nPrivateData;
        }
        public void SetData(int x){
                nPrivateData=x;
        }
}
```

```
class Program{
    static void Main(){
        MyData data=new MyData();
        data.SetData(5);
        Console.WriteLine("{0}",
        data.GetData());
    }
}
```

VÍ DỤ

```
interface ILiveBirth{
        string BabyCalled();
}
class Animal {
}
class Cat : Animal, ILiveBirth{
        string ILiveBirth.BabyCalled(){
                return "kitten";
        }
}
class Dog : Animal, ILiveBirth {
        string ILiveBirth.BabyCalled(){
                return "puppy";
        }
}
class Bird : Animal{
}
```

```
Animal[] animalArray = new Animal[3];
animalArray[0] = new Cat();
animalArray[1] = new Bird();
animalArray[2] = new Dog();
foreach( Animal a in animalArray ){
        ILiveBirth b = a as ILiveBirth;
        if (b != null)
                Console.WriteLine("Baby is called:
                {0}", b.BabyCalled());
}
```

3.1. SỰ GIỐNG NHAU CỦA ABSTRACT CLASS VÀ INTERFACE

- Abstract class và interface đều không thể khởi tạo đối tượng từ chính nó được
- Abstract class và interface đều có chứa abstract method.
- Abstract class và interface đều được kế thừa hoặc thực thi phương thức hoặc properties từ các class dẫn xuất nó.
- Abstract class và interface đều có thể implement từ một hoặc nhiều interface
- Abstract class và interface đều giúp cho code trở nên sáng sủa và gọn gàng, dễ bảo trì và nâng cấp

Abstract class	Interface
– Có thể tạo được đối tượng thông qua lớp dẫn xuất. (Xem lại bài: Lớp trừu tượng Abstract class trong C#)	– Không thể tạo được đối tượng từ nó hoặc từ lớp dẫn xuất.
– Vừa có thể kế thừa class, vừa có thể hiện thực interface. (Trong trường hợp này, tên base class phải để đầu tiên trong danh sách, tiếp theo là tên của các interface)	– Không thể kế thừa class, chỉ có thể hiện thực interface.
– Có thể khai báo field, const, constructor, destructor.	– Không được khai báo field, const, constructor, destructor.
– Có thể chứa phương thức đã định nghĩa hoàn chỉnh.	– Không được chứa phương thức định nghĩa, chỉ khai báo prototype(nguyên mẫu hàm) của phương thức.
– Ở lớp dẫn xuất khi kế thừa phải dùng từ khóa *override* lúc định nghĩa.	– Không dùng từ khóa *override* khi implement (vì định nghĩa mới hoàn toàn).
– Được phép chỉ định access modifiers cho phương thức, thuộc tính... như public, protected, private. Đối với các methods hay properties trừu tượng bắt buộc khai báo với từ khóa **abstract** và access modifiers bắt buộc là **public** hoặc **protected**.	– Không được phép chỉ định access modifiers cho phương thức, thuộc tính. Tất cả giả định là **public** và không thể thay đổi thành access modifiers khác.
– Chỉ dùng để định nghĩa cốt lõi của một lớp và chỉ sử dụng cho những đối tượng có cùng bản chất.	– Dùng để định nghĩa những khả năng bên ngoài của một lớp. Tức là những đối tượng không cùng bản chất với interface, cũng implement được.

BÀI TẬP

Bài 1: Tạo lớp SinhVienPoly gồm 2 thuộc tính họ tên và ngành cùng với phương thức trừu tượng là getDiem(). Thêm phương thức getHocLuc() để xếp loại học lực. Lớp cũng bao gồm một phương thức xuat() để xuất họ tên, ngành, điểm và học lực ra màn hình.

HƯỚNG DẪN:

✓ Xây dựng lớp có mô hình như sau

SinhVien
+ hoTen: String + nganh: String
SinhVien(hoTen, nganh) + getDiem(): double + getHocLuc(): String + xuat(): void

✓ Vì chưa biết sinh viên này học những môn nào nên chưa tính được điểm vì vậy phương thức getDiem() phải là phương thức trừu tượng.

✓ Chú ý lớp SinhVien phải là lớp trừu tượng vì có phương thức getDiem() là phương thức trừu tượng

✓ Phương thức getHocLuc() được viết bình thường vẫn sử dụng phương thức getDiem() để lấy điểm của sinh viên mặc dù hiện tại vẫn chưa biết điểm được tính thế nào. Học lực được tính như sau

- Yếu: điểm < 5
- Trung bình: 5 <= điểm < 6.5
- Khá: 6.5 <= điểm < 7.5
- Giỏi: 7.5 <= điểm < 9
- Xuất sắc: điểm >= 9

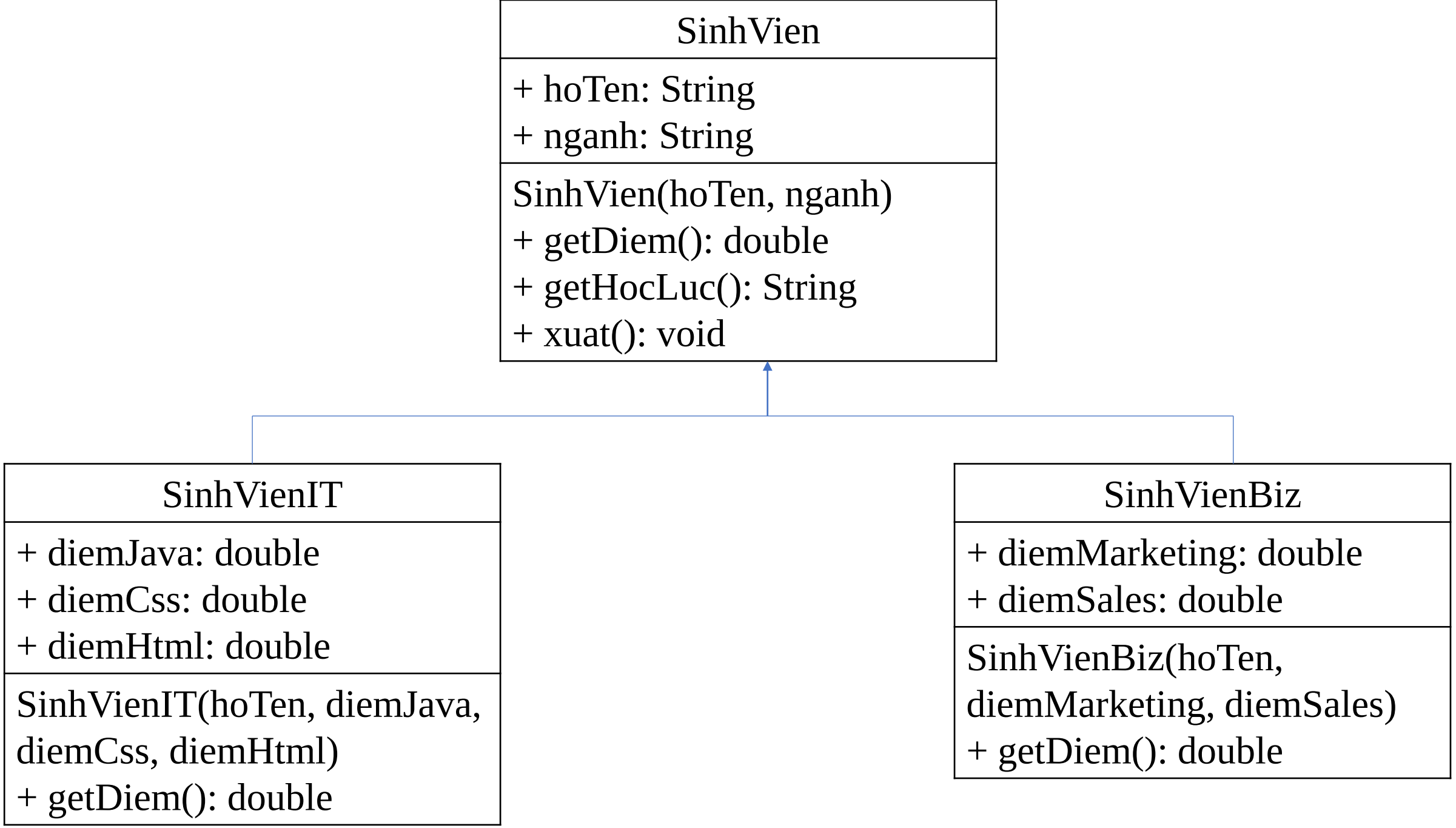
Bài 2: Tạo lớp SinhVienIT và SinhVienBiz kế thừa từ lớp SinhVien.

• SinhVienIT gồm các thuộc tính điểm java, html, css. Ghi đè phương thức getDiem() để tính điểm cho sinh viên IT theo công thức (2*java + html + css)/4

• SinhVienBiz gồm các thuộc tính điểm marketing, sales. Ghi đè phương thức getDiem() để tính điểm cho sinh viên Biz theo công thức (2*marketting + sales)/3

• Tổ chức các lớp theo kiến trúc phân cấp kế thừa như sau

• Ghi đè phương thức getDiem() trên 2 lớp SinhVienIT và SinhVienBiz theo yêu cầu của đề để tính điểm cho các sinh viên của các ngành.



Bài 3: 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
    - o Cho nhập vào một mảng chứa những cuốn sách
    - o Xuất danh sách thông tin những cuốn sách
    - o Lần lượt xuất danh sách ra theo thứ tự được sắp theo tên tác giả, tên sách, năm xuất bản

Bài 4: Tạo một interface có tên là DbAction và khai báo 4 phương thức insert(), update(), delete() và select() theo cú pháp sau:

```
void insert();
void update();
void delete();
void select();
```

Tạo 2 lớp Product và Order thực thi theo interface DbAction. Viết mã cho các phương thức insert(), update(), delete() và select() cho các lớp bằng cách xuất tên của phương thức kết hợp với tên lớp ra màn hình. Ví dụ trong phương thức insert() của lớp Product viết mã là Console.Write("Insert product");

Tạo lớp DbManager chứa phương thức main(). Tạo 2 đối tượng từ 2 lớp Product và Order và gọi phương thức insert() của mỗi đối tượng

```
DbAction db1 = new Product();
DbAction db2 = new Order();
db1.insert();
db2.insert();
```