MỤC LỤC

Chương I: ĐỐI TƯỢNG VÀ LỚP
A. TÓM TẮT LÝ THUYẾT.....7
1. Khai báo lớp.....7
2. Sử dụng các đối tượng của lớp: 9
3. Từ khóa this.....10
4. Sử dụng các từ khóa mức độ truy cập.....11
5. Bộ khởi dựng(constructor):..... **Error! Bookmark not defined.**
6. Bộ sao chép(copy constructor): **Error! Bookmark not defined.**
7. Hàm hủy bỏ (destructor) 12
8. Sử dụng các thành viên tĩnh static: 14
9. Nạp chồng phương thức.....17
10. Đóng gói dữ liệu thông qua các thuộc tính:.....17
B. BÀI TẬP MẪU 19
1. Xây dựng lớp.....19
2. Sử dụng lớp 22
3. Lớp lồng nhau và cách truy xuất.....32
4. Chương trình khai báo nhiều lớp 34
5. Từ khóa this.....39
6. Hàm thiết lập và hàm thiết lập sao chép, hàm sao chép cho lớp vector.....45
7. Sử dụng hàm thiết lập sao chép và hàm hủy bỏ 50
8. Kiểu dữ liệu mảng, các phương thức có giá trị trả về , constructor sao chép.....54
9. Sử dụng các thành phần tĩnh.....59
10. Hàm khởi tạo private để cấm tạo thể hiện của đối tượng.....62
11. Nạp chồng phương thức.....64
12. Gói ghém dữ liệu thông qua các thuộc tính.....74
C. CÂU HỎI LÝ THUYẾT VÀ TRẮC NGHIỆM.....80
D. BÀI TẬP TỰ GIẢI 82
E. BÀI TẬP TỔNG HỢP: 87
Chương II: NẠP CHỒNG TOÁN TỬ
A. TÓM TẮT LÝ THUYẾT.....90
1. Toán tử: 90
2. Chồng toán tử:.....90

3. Cú pháp nạp chồng toán tử: 90
4. Toán tử một ngôi: 92
5. Toán tử hai ngôi: 92
6. Hỗ trợ ngôn ngữ .NET khác: 92
7. Phạm vi sử dụng của các toán tử: 93
8. Một số trường hợp nên sử dụng các toán tử nạp chồng: 94
9. Yêu cầu khi sử dụng toán tử: 94
10. Ưu và nhược điểm của chồng toán tử 95
B. BÀI TẬP MẪU 95
I. BÀI TẬP TRẮC NGHIỆM, LÝ THUYẾT 95
II. CÂU HỎI LÝ THUYẾT VÀ BÀI TẬP MẪU 98
1. Chồng toán tử (+) đối với hai vector 98
2. Chồng toán tử một ngôi 102
3. Xây dựng một lớp sophuc và nạp chồng toán tử +, - và đổi dấu số phức. 104
4. Xây dựng một lớp phân số với tên là phanso và nạp chồng toán tử trên lớp 107
5. Xây dựng một lớp vector và thực hiện các yêu cầu sau: 112
C. CÂU HỎI TRẮC NGHIỆM LÝ THUYẾT 118
D. BÀI TẬP TỰ GIẢI 119
Chương III: KẾ THỪA VÀ ĐA HÌNH:
A. TÓM TẮT LÝ THUYẾT 124
1. Kế thừa: 124
2. Từ khoá base: 125
3. Từ khoá new: 126
4. Tính đa hình : 126
5. Ghi đè: 126
6. Nghiêm cấm kế thừa: 127
7. Cách truy xuất protected: 127
8. Lớp trừu tượng: 127
9. Sự khác nhau giữa phương thức đa hình với phương thức trừu tượng: 128
10. Gốc của tất cả các lớp (Lớp Object*):* 128
11. Kiểu Boxing và Unxing 129
12. Lớp lồng nhau: 130

B. BÀI TẬP MẪU 131
I. Kế thừa 131
1. Xây dựng lớp dẫn xuất thừa kế từ lớp cơ sở. 131
2. Mức độ truy cập Protected 139
3. Cách truy nhập của từ khóa internal và public 146
4. Lớp cơ sở và đối tượng thành phần 152
II. Nghiêm cấm kế thừa 160
III. Tính đa hình. 162
IV. Lớp trừu tượng 169
V. Xây dựng lớp lồng nhau 173
C. MỘT SỐ CÂU HỎI LÝ THUYẾT: 176
D. CÂU HỎI TỰ TRẢ LỜI 179
E. BÀI TẬP TỰ GIẢI: 182
Chương IV: GIAO DIỆN
A. TÓM TẮT LÍ THUYẾT 189
1. Khái niệm về giao diện 189
2. Cài đặt một giao diện 189
3. Truy xuất phương thức của giao diện 190
4. Toán tử is: 190
5. Toán tử as 190
6. Thực thi phủ quyết giao diện 191
7. Thực hiện giao diện một cách tường minh 191
B. BÀI TẬP MẪU 191
I. Sử dụng giao diện 191
II. Mở rộng một giao diện và sự thực thi nhiều giao diện của một lớp 194
III. Xây dựng lớp thực thi phủ quyết một giao diện 197
IV. Xây dựng lớp thực thi giao diện một cách tường minh 202
C. CÂU HỎI VÀ TRẢ LỜI 205
D. CÂU HỎI VÀ BÀI TẬP TỰ LÀM 206

LỜI MỞ ĐẦU

Ngày nay khoa học đang phát triển rất nhanh, đặc biệt là sự bùng nổ của công nghệ thông tin. Tin học luôn thay đổi và phát triển từng giờ, từng ngày. Ngôn ngữ lập trình cũng vậy. Các ngôn ngữ mới luôn ra đời thay thế và khắc phục những hạn chế cho các ngôn ngữ cũ.

Ngôn ngữ lập trình là một công cụ giúp con người thể hiện các vấn đề thực tế trên máy tính một cách hữu hiệu. Với sự phát triển của tin học, các ngôn ngữ đang dần tiến hoá để đáp ứng các thách thức của thực tế.

C# được mệnh danh là ngôn ngữ lập trình hướng đối tượng mạnh nhất trong bộ ngôn ngữ lập trình .NET của hãng Microsoft hiện nay.

Cuốn sách này cung cấp cho các bạn những kiến thức cơ bản liên quan đến việc định nghĩa và cài đặt lập trình hướng đối tượng bằng ngôn ngữ C# trong ứng dụng thực tiễn và áp dụng chúng cho mô hình đối tượng một cách hiệu quả.

Giáo trình bao gồm 4 chương:

Chương 1: Đối tượng và lớp.

Chương 2: Nạp chồng toán tử trên lớp.

Chương 3: Kế thừa và đa hình.

Chương 4: Giao diện

Với mỗi một chương chúng tôi có trình bày tóm tắt nhất về lý thuyết và có bài tập giải mẫu theo từng nội dung. Cuối mỗi chương là hệ thống bài tập tự giải (có hướng dẫn) để giúp các bạn tiện thực hành.

Tuy cuốn sách này phần nào đã hoàn thành xong, nhưng do thời gian và kiến thức có hạn chắc chắn sẽ không tránh khỏi những sai lầm và thiếu sót. Vì vậy chúng tôi rất mong nhận được sự đóng góp của thầy cô và bạn bè để cuốn sách này được hoàn thiện hơn.

Chúng tôi xin chân thành cảm ơn tới cha, mẹ, anh, chị, em; thầy Nguyễn Hữu Đông cùng các thầy cô trong khoa Công nghệ thông tin trường ĐHSPKT Hưng Yên đã tạo điều kiện cho chúng em hoàn thành đề tài này. Xin cám ơn mọi sự giúp đỡ của bạn bè.

CHƯƠNG I
ĐỐI TƯỢNG VÀ LỚP

Mục tiêu: Sau khi tìm hiểu xong chương này người học có thể nắm được các nội dung sau:

- Khai báo lớp
- Khai báo và sử dụng các đối tượng của lớp
- Từ khóa this
- Các thuộc tính truy cập
- Hàm thiết lập
- Hàm thiết lập sao chép
- Hàm hủy bỏ
- Sử dụng các thành viên tĩnh
- Nạp chồng phương thức
- Đóng gói dữ liệu thông qua các thuộc tính

A. TÓM TẮT LÝ THUYẾT

1. Khai báo lớp

Một lớp bao gồm có các thuộc tính và phương thức. Để khai báo một lớp ta sử dụng từ khóa ***class*** với cấu trúc sau đây:

```
[Thuộc tính truy cập]  class <tên lớp>
  {
        Khai báo các thuộc tính của lớp
        Khai báo các phương thức của lớp
  }
```

Các thuộc tính truy cập gồm có các từ khóa sau đây (sẽ trình bày chi tiết ở phần sau): public, private, internal, protected, internal protected.

Trong C#, cho phép chúng ta khai báo các class lồng nhau. Thông thường khai báo lớp lồng nhau khi ứng dụng có quy mô lớn. Class lồng cho phép sử dụng trong nội bộ class chứa nó khi nó có tầm vực public.

Cấu trúc khai báo một class lồng như sau:

```
class class1
  {
      // khai báo thuộc tính
      // khai báo các phương thức
    public class class2
      {
          // khai báo các thành phần dữ liệu
          // khai báo các phương thức
        }
  }
```

2. Sử dụng các đối tượng của lớp:

Để sử dụng lớp ta phải khai báo đối tượng của lớp đó. Khi một đối tượng của lớp được tạo ra thì nó có đầy đủ các thuộc tính, phương thức của lớp và sử dụng thuộc tính và phương thức của lớp. Các thành phần của một lớp chỉ được sử dụng khi có thể hiện của lớp, trừ trường hợp trong lớp có một hàm khởi dựng là static. Để khai báo một đối tượng của lớp ta dùng từ khóa ***new*** và khai báo nó theo cấu trúc sau:

<tên lớp> <tên đối tượng> = ***new*** *<tên lớp> ([các giá trị khởi tạo nếu có])*

Để truy nhập đến một phương thức ta thông qua tên biến đối tượng và toán tử chấm ".":

<tên đối tượng>. *<tên phương thức> ([danh sách các đối số nếu có])*

Đối với các lớp lồng nhau, để truy cập đến những thuộc tính và phương thức của class lồng thì khi khai báo cần chỉ ra lớp chứa đựng nó. Muốn sử dụng biến của lớp chứa thì các biến của lớp dùng để chứa phải khai báo là static và phải khai báo đối tượng chứa nó.

Ví dụ1: Bạn đã khai báo một lớp diem với đầy đủ các thuộc tính và các phương thức (giả sử là có phương thức hien()), bây giờ bạn muốn tạo một đối tuợng tên là A của lớp này và sử dụng phương thức hiện điểm A thì phải khai báo A la một biến đối tượng như sau:

```
diem A = new diem ();
A.hien();
```

Ví dụ 2: Định nghĩa lớp pheptoan và trong lớp này có chứa lớp tinhhieu, muốn sử dụng đối tượng của lớp tinhhieu thì bắt buộc bạn phải khai báo chỉ rõ lớp chứa ở đây là lớp pheptoan như sau:

```
pheptoan.tinhhieu con = new pheptoan.tinhhieu();
```

sau đó việc truy xuất đến các phương thức của lớp thì tiến hành bình thường nhưng lớp ở trong thì không sử dụng được phương thức của lớp chứa nó và chỉ sử dụng được thành phần dữ liệu tĩnh của lớp chứa mà thôi.

3. Từ khóa this

Từ khóa this dùng để tham chiếu đến chính bản thân của đối tượng đó.this là một con trỏ ẩn nằm ngay bên trong của mỗi một phương thức của lớp và bản thân chúng có thể tham chiếu đến các hàm và các biến khác của một đối tượng. Từ khóa this trong C# cũng tương tự như this trong C++.

Có ba cách để sử dụng từ khóa this.

- *Sử dụng this để phân biệt rõ giữa các thành viên thể hiện và các tham* số của phương thức khi ta sử dụng biến thể hiện và tên của tham số trong phương thức trùng nhau. Tuy nhiên nếu muốn có được sự rõ ràng, minh bạch thì có thể dùng tên biến thành viên và tên tham số là khác nhau từ đầu.

Ví dụ: Trong lớp pheptoan có biến thành viên là int y, int y và phương thức

```
public int setXY(int x, int y)
{
    this.x=x;
    this.y=y;
}
```

this.x, this.y là để tham chiếu đến 2 thành phần dữ liệu của lớp x, y. Còn x, y ở bên phải phép gán chính là hai đối số truyền vào của phương thức setXY

- *Sử dụng this để trao đối tượng hiên hành như là một thông số* cho một hàm hành sự khác. Khi đó một đối tượng đương nhiên sẽ trở thành một tham số của phương thức.

Ví dụ: với hai lớp class1 có chứa phương thức thietlapdoituong(), lớp class2 có chứa phương thức

saochepdoituong() và muốn truyền tham số cho nó là đối tượng của lớp class1 thì sử dụng từ khóa thí như sau:

```
public void saochepdoituong (class1 a)
{
    a.thietlapdoituong(this);
}
```

- *Sử dụng this để thao tác với các indexer* thường được sử dụng trong bản dãy, indexer và các tập hợp.

4 . Bộ khởi dựng(constructor/ Phương thức khởi tạo/ thiết lập)

Cú pháp:

public className([ds tham số]){

// Khởi tạo cho các thành phần dữ liệu của lớp

}

trong đó className: Tên lớp

Chú ý: Phương thức khởi tạo là phương thức có tên trùng với tên của lớp và không có kiểu trả về

5. Sử dụng các từ khóa mức độ truy cập

+ **public :** Không có giới hạn, có thể truy xuất mọi nơi trong bản thân lớp khai báo và bên ngoài hay trong nội bộ khối assembly.

+ private: riêng tư chỉ có phạm vi hoạt động trong lớp mà nó khai báo. Các phương thức bên ngoài lớp không thể truy xuất đến nó.

+ protected: Các thành viên trong lớp được khai báo bằng protected thì chỉ có các phương thức bên trong lớp và các lớp dẫn xuất từ lớp đó mới có thể truy cập đến nó.

+ **internal**: Các phương thức, các biến thành viên được khai báo bằng từ khóa Internal có thể được truy cập bởi tất cả những phương thức của bất cứ lớp nào trong cùng một khối hợp ngữ assembly với lớp đó.

+ protected internal: Các biến thành viên được khai báo bằng từ khóa này trong một lớp A bất kì có thể được truy xuất bởi các

phương thức thuộc lớp A và các phương thức của lớp dẫn xuất từ lớp A và bất cứ lớp nào trong cùng một khối hợp ngữ với lớp A.

- Khối hợp ngữ **Assembly** được hiểu là một khối chia xẻ và dùng lại trong CLR. Khối hợp ngữ là tập hợp các tập tin vật lý được lưu trữ trong một thư mục bao gồm các tập tin tài nguyên.

Các thuộc tính truy cập được áp dụng cho thuộc tính, phương thức của lớp và bản thân lớp. Khi định nghĩa thuộc tính truy cập của lớp là internal, protected chỉ được định nghĩa trong lớp lồng nhau mà thôi.

4. Hàm hủy bỏ (destructor)

Dùng để giải phóng vùng nhớ đã cấp phát cho đối tượng khi mà đối tượng không còn được tham chiếu đến. Hàm hủy bỏ là một hàm ***không có giá trị trả về*** có ***tên trùng tên với class và có thêm kí tự "~"ở trước***. Muốn khai báo một destructor chúng ta khai báo nó với cú pháp như sau:

class className{

public ~className() {Ư

```
public classname
{
        public classname()
        {
                // code of constructor
                // các công việc cần thực hiện
        }
           ~ classname()
           {
                // code of descontructor
                // các công việc cần thực hiện
           }
}
```

Tuy nhiên, trong ngôn ngữ C# thì cú pháp khai báo trên là một shortcut liên kết đến một phương thức kết thúc Finalize được kết với lớp cơ sở, do vậy khi viết

```
~ classname()
  {
 // Thực hiện một số công việc
 }
```

Và viết :

```
class1.Finalize()
{
// Thực hiện một số công việc
base.Finalize();
}
```

Thì hai cách viết như thế này sẽ được C# hiểu là như nhau.

Trong C# có một đối tượng hỗ trợ cho công việc dọn rác mỗi khi đối tượng không còn được tham chiếu đến là đối tượng GC (garbage collector). Đối tượng này thực hiện công việc rọn rác qua 3 bước như sau:

- Tìm kiếm những đối tượng không còn được tham chiếu nữa
- Cố gắng các hoạt động để giải phóng đối tượng không còn được tham chiếu
- Thi hành phương thức finalize() để hủy đối tượng

Ngoài ra, trong cơ chế hủy bỏ của C# còn có phương thức Dispose là một phương thức đối lập hoàn toàn so với phương thức Finalize() của đối tượng GC.

Phương thức Dispose là cho phép chương trình thực hiện các công việc dọn dẹp hay giải phóng tài nguyên mong muốn mà không phải chờ cho đến khi phương thức Finalize() được gọi. Khi đó ta không được phép dùng phương thức Finalize() trong đối tượng mà thay vào đó ta sẽ gọi một phương thức tĩnh của lớp GC (garbage collector) là GC.SuppressFinalize(this). Sau đó phương thức Finalize() sử dụng để gọi phương thức Dispose() như sau:

```
public void Dispose()
{
        // Thực hiện công việc dọn dẹp
        // Yêu cầu bộ thu dọc GC trong thực hiện kết thúc
        GC.SuppressFinalize( this );
}
public override void Finalize()
{
        Dispose();
        base.Finalize();
}
```

Câu lệnh using:

Mặc dù phương thức rọn rác của C# là tự động cung cấp nhưng không thể chác rằng phương thức Dispose() sẽ được gọi và việc giải phóng tài nguyên không thể xác định được. Câu lệnh using sẽ đảm bảo rằng phương thức Dispose() sẽ được gọi trong thời gian sớm nhất. Sử dụng using bằng cách gọi câu lệnh using và truyền vào tên đối tượng muốn hủy bỏ:

using (tên đối tượng cần hủy bỏ)

Khi trong lớp không khai báo một destructor nào thì trình biên dịch sẽ gọi tiến trình Garbage collector trong ngôn ngữ C# để giải phóng đối tượng này trong bộ nhớ. Phần lớn trong ngôn ngữ C# thì có cơ chế tự động gom rác mỗi khi biến đó không được tham chiếu đến nên chúng ta không cần quan tâm nhiều đến nó như trong C++.

5. Sử dụng các thành viên tĩnh static:

Thành viên tĩnh không thể hiện gì cho lớp về cả thuộc tính và phương thức mà nó như là một thành phần của lớp.

Sử dụng từ khóa static để khai báo một thành viên tĩnh. Thành phần tĩnh chỉ được sử dụng với lớp, phương thức, thuộc tính, sự kiện và constructor nhưng không thể được sử dụng với những bản dãy, indexer, destructor hay kiểu khác với những lớp.

Sử dụng phương thức tĩnh:

Một phương thức static có phạm vi hoạt động giống như một phương thức toàn cục mà không cần tạo ra bất cứ một thể hiện nào của lớp cả. Toàn cục ở đây hiểu theo nghĩa là toàn cục trong lớp.

Gọi một phương thức static:

Về bản chất thành phần static là một thành phần của lớp không thể hiện trả về vì vậy không có một tham chiếu this. Một hàm static không thể trực tiếp truy xuất đến các thành viên không static mà phải thông qua một đối tượng thể hiện của lớp đó như sau:

Tenlop.tenhamtinh ([danh sach tham so neu co]:

Ví dụ *diem.hien()* là lời gọi đến phương thúc tĩnh có tên là hien() của lớp diem

Sử dụng các phương thức khởi tạo static:

Trong khai báo một static constructor không có từ khóa truy cập. Phương thức tĩnh chỉ có thể truy nhập đến thành phần dữ liệu cũng có tính chất tĩnh mà thôi. Nếu trong khai báo lớp có một hàm static constructor thì hàm này sẽ được gọi trước khi bất cứ một thể hiện nào của lớp được tạo ra.

Việc sử dụng hàm khởi tạo static cần được cân nhắc kỹ lưỡng vì chúng ta không thể theo dõi nó được như trong C++ vì thế thường gây ra những hậu quả khó lường.

- Khởi tạo private constructor:

Việc sử dụng một hàm khởi tạo private trong lớp sẽ có tác dụng ngăn chặn tạo ra bất kì một đối tượng nào của lớp. Hàm khởi tạo private này mặc nhiên, không có tham số gì cả và trống rỗng. Khi đó trong lớp sẽ không có hàm khởi tạo public nên sẽ không khởi tạo được bất cứ một thành viên thể hiện nào.

Ví dụ không muốn tạo ra bất kì một đối tượng nào của lớp diem thì trong khi định nghĩa lớp ta sẽ định nghĩa thêm một hàm khởi tạo tầm vực là private như sau:

```
private diem ()
{
 // không làm gì cả
}
```

Sử dụng các thuộc tính tĩnh:

Trong C# không hề có một biến nào có phạm vi hoạt động toàn cục như trong một số ngôn ngữ lập trình khác (pascal, C, C++, Visual Basic …) việc sử dụng một biến với mục đích "toàn cục" trở nên là một điều không thể. Biến toàn cục trong các ngôn ngữ khác được hiểu là toàn cục trong ứng dụng nhưng đối với C# thì toàn cục theo nghĩa hiểu của nó là toàn cục trong một lớp và không có khái niệm toàn cục trong toàn bộ chương trình. Nếu ta khai báo một biến thành viên tĩnh của lớp thì biến thành viên tĩnh này có tầm vực hoạt động theo ý nghĩa toàn cục đó. Các biến thành viên tĩnh có hoạt động tích cực trong vai trò này.

Lớp tĩnh:

Một lớp có thể được xây dựng là một lớp tĩnh và chỉ chứa các thành phần tĩnh mà thôi và nó không cho phép tạo thể hiện của lớp bằng việc sử dụng từ khóa new. Lớp static thường được tải tự động trong Net.Framework khi chương trình hoặc namespace chứa lớp được tải lên.Việc tạo một static class giống với việc tạo ra một lớp mà chỉ chứa một private constructor. Như vậy là có thể kiểm tra chắc chắn và đảm bảo những thành viên của lớp này không thể được tạo ra.

Khi nào thì sử dụng các thuộc tính tĩnh và khi nào thì sử dụng phương thức tĩnh:

+ Sử dụng phương thức tĩnh khi muốn khởi tạo một giá trị nào đó ngay khi chương trình biên dịch mà không cần tạo thể hiện của lớp hay việc tạo một thiết lập duy nhất mà một sự chuẩn bị (initializer) không thể thực hiện được và chỉ cần thực hiện một lần mà thôi. Trong trường hợp nếu ta muốn lớp mà sau khi ta định nghĩa không thể tao ra một thể hiện nào của nó thì ta sử dụng phương thức khởi tạo private (bên trong thân phương thức trống rỗng).

+ Sử dụng biến static khi bạn muốn tạo ra một biến có vai trò toàn cục trong lớp để theo dõi hoạt động của nó. Ví dụ như bạn tạo ra một biến static để theo dõi xem ở thời điểm hiện tại có bao nhiêu

biến đối tượng được tạo ra chẳng hạn. Khi đó đi cùng với việc tạo đối tượng bạn hãy gọi phương thức chứa thành phần tĩnh này thì bạn có thể theo dõi như dùng biến toàn cục trong các ngôn ngữ lập trình khác.

+ Sử dụng lớp tĩnh để chứa các phương thức mà không liên quan tới một đối tượng đặc biệt. Chẳng hạn như yêu cầu tạo tập hợp phương thức không hành động theo thể hiện dữ liệu và không liên quan đến một đối tượng đặc biệt nào. Một static class không cho phép thừa kế.

6. Nạp chồng phương thức

Chồng phương thức là việc tạo ra nhiều phương thức trùng tên với nhau nhưng nhận các tham số khác nhau hay trả về dữ liệu khác nhau. Việc phân biệt các hàm này dựa vào dấu ấn:

+ Khác nhau các tham số: khác nhau về số lượng tham số

+ Khác nhau về kiểu dữ liệu của tham số, kiểu dữ liệu trả về của phương thức.

Khi nạp chồng một phương thức ta phải thay đổi kí hiệu (dấu ấn) của phương thức, số tham số hay kiểu dữ liệu của tham số hoặc có thể thay đổi cả về giá trị của các tham số. Khi đó thì thực chất không phải là nạp chồng phương thức mà đó là hai phương thức khác nhau có cùng tên nhưng khác nhau về kiểu giá trị trả về. Đó là một điều cần chú ý khi nạp chồng phương thức.

Khi gọi một phương thức có nạp chồng phương thức thì cần chú ý về số tham số, kiểu dữ liệu cho phù hợp với từng phương thức nếu không thì sẽ phát sinh lỗi khi biên dịch chương trình.

7. Đóng gói dữ liệu thông qua các thuộc tính:

Đóng gói dữ liệu với thuộc tính thực chất là một quá trình ta lấy giá trị cho biến thành viên và thiết lập giá trị đó cho biến để nó được truy cập thông qua phương thức của lớp mà không qua đối tượng. Trong C# cung cấp khả năng khai báo hàm chung gọi là thuộc tính cho hàm **get** và **set**

```
public string  (tên thuộc tính)
    {
        get { //Lấy giá tri thuộc tính }
        set { //Trả về giá trị cùng kiểu với thuộc tính đã khai báo}
    }
```

+ Phương thức ***get*** trả về một đối tượng dữ liệu kiểu thuộc tính, phương thức này giống phương thức của đối tượng.

+ Phương thức **s*et:*** thiết lập giá trị của thuộc tính, có kiểu trả về là void. Khi thiết lập một phương thức set phải dùng từ khóa *value* để tượng trưng cho đối được trao và được giữ bởi thuộc tính.

Lợi ích của việc gói ghém dữ liệu là che giấu thông tin mặc dù người sử dụng vẫn thao tác với thuộc tính.

☺**Vài điểm cần chú ý khi định nghĩa một thuộc tính**

+ Ta không khai báo tường minh các tham số trong thủ tục *set***..**

+ Có thể chỉ xây dựng thuộc tính chỉ đọc hoặc chỉ viêt bằng cách bỏ đi một trong hai thủ tục trên.

+ Nếu ta muốn tạo một thuộc tính có public để đọc nhưng lại muốn hạn chế protected trong gán thì phải tạo thuộc tính chỉ đọc public sau đó tạo một hàm *set()* với truy cập protected ở bên ngoài thuộc tính đó.

+ C# cho phép bạn tạo một thuộc tính virtual hay abstract (xem thêm kế thừa và đa hình) cú pháp như sau:

```
public abstract string (tên thuộc tính)
    {
        get;
        set;
    }
```

B. BÀI TẬP MẪU

1. Xây dựng lớp

Ví dụ 1:

Xây dựng lớp diem với các thuộc tính tung độ, hoành độ của điểm đó, phương thức đổi tọa độ giữa dương và âm, phương thức di chuyển theo một giá trị nhập vào từ bàn phím, phương thức hiện điểm lên màn hình.

a, Hướng dẫn:

+ các thuộc tính gồm có:
  int x ; // *tọa độ hoành độ*
  int y ; // *tọa độ tung độ*
+ các phương thức của lớp:
  nhập thông tin
  đổi tọa độ
  phương thức move: di chuyển điểm
  phương thức hien: hiện thông tin lên màn hình

b, Bài giải mẫu:

```
class Diem
{
   public int x, y;
   {
      x = ox;
      y = oy;
   }
   public void nhap()
   {
      Console.WriteLine("Nhap toa do cua diem:");
      x = int.Parse(Console.ReadLine());
      y = int.Parse(Console.ReadLine());
   }
   public void move(int dx, int dy)
   {
      x += dx;
      y += dy;
   }
   public void chuyen()
   {
      x = -x;
      y = -y;
   }
```

```
    public void hien()
    {
        Console.Write("toa do :(");
        Console.Write("{0},{1}", x, y);
        Console.WriteLine(")");
    }
}
```

Trong ví dụ trên, chúng tôi chỉ ra cách khai báo một lớp thì cần phải khai báo tường minh các thuộc tính (thành phần dữ liệu), và cả các phương thức (cái mà một đối tượng của lớp có thể thi hành). Với một phương thức không có giá trị trả về thì khai báo có từ khóa ***void*** còn nếu có giá trị trả về thì phải khai báo có giá trị trả về.

Ví dụ 2:

Xây dựng lớp stack để mô phỏng một stack bao gồm

- Hàm khởi tạo số phần tử tối đa,
- Phương thức isEmpty kiểm tra xem stack có rỗng không
- Phương thức isFull kiểm tra xem stack có đầy không
- Phương thức push và pop để thêm vào, lấy ra một phần tử

a, Hướng dẫn:

Các thuộc tính của lớp stack gồm có:

top: mô tả phần tử ở đầu stack

n: số phần tử tối đa của stack

Các phương thức của lớp gồm có:

public stack(): khởi tạo giá trị của stack với số phần tử tối đa

pubic bool empty(): trả về giá trị kiểu đúng sai khi stack rỗng hay không

public bool full(): trả về kiểu đúng sai khi stack đầy hay không đầy

public void push (int x):thêm một phần tử vào stack

public int pop(): Lấy ra một phần tử từ stack. Đây là hàm vì nó có trả ra giá trị, giá trị chính là phần tử mà ta vừa lấy ra được từ stack.

b, Bài giải mẫu:

```
using System;
namespace Stack
{
   class Stack
   {
      private int top;
      private int []s;
      public bool  empty()
      {
         return (top == -1);
      }
      public bool  full()
      {
         return (top >= s.Length);
      }
      public Stack ()
      {
         s = new int[20];
         top=-1;
      }
      public void push(int x)
      {
         if(!full())
         {
```

```
            top=top+1;
            s[top]=x;
         }
         else
            Console.Write("Stack tran");
      }
      public int pop()
      {
         if (empty())
        {
            Console.Write("Stack can");
             return 0;
         }
         else
            return s[top--];
      }
   }
```

2. Sử dụng lớp

Ví dụ 1:

Xây dựng lớp diem như bài 1 sau đó viết chương trình nhập tọa độ của điểm từ bàn phím, di chuyển một tọa độ, lấy tọa độ đối xứng, hiện tọa độ của điểm lên màn hình.

a, Hướng dẫn:

Thuộc tính và phương thức của lớp diem giống hệt bài trên, khi đó muốn xây dựng chương trình ta chỉ việc sử dụng đối tượng của lớp mà thôi. Muốn vậy phải khai báo đối tượng kiểu lớp bằng sử dụmg từ khóa ***new*** để cấp phát vùng nhớ**.** Để truy xuất đến các phương thức của lớp thì ta truy xuất thông qua các đối tượng của lớp diem

Chẳng hạn như có đối tượng của lớp la A muốn truy xuất tới phương thức nhap() thì ta truy nhập như sau: A.nhap();

b, Chương trình hoàn thiện như sau:

```
using System;
using System.Collections.Generic;
using System.Text;
namespace vidu1
{
class diem
   {
      public int x, y;
      public void move(int dx, int dy)
      {
         x += dx;
         y += dy;
      }
      public void hien()
      {
         Console.Write("toa do :(");
         Console.Write("{0},{1}", x, y);
         Console.WriteLine(")");
      }
      public void chuyen()
      {
         x = -x;
         y = -y;
      }
      public void nhap()
      {
         Console.WriteLine("Nhap toa do cua diem:");
         x = int.Parse(Console.ReadLine());
         y = int.Parse(Console.ReadLine());
      }
   }
   class tester
   {
     static void Main(string[] args)
```

```
    {
        diem b = new diem();
       // bien trong C# luon doc khoi gan gia tri truoc khi su
dung
        b.nhap();
        Console.Write("diem b ");
        b.hien();
        Console.WriteLine("toa do doi xung la:");
        b.chuyen();
        b.hien();
        b.move(2, 6);
      }
    }
}
```

Kết quả sau khi chạy chương trình là

```
Nhap toa do cua diem:
      8
      19
      diem b toa do : (8,19)
      toa do doi xung la:
      toa do(-8,-19)
      diem b sau khi di chuyen la:
      toa do (-6,-13)
```

Ví dụ 2:

Xây dựng chương trình nhập tam giác, tính chu vi, diện tích và in ra màn hình đó là loại tam giác nào: cân, vuông, vuông cân, đều hay thường.

a, Hướng dẫn:

Trong bài này sẽ nói sơ qua về một số thuộc tính truy cập của một số thành phần trong lớp (chi tiết về phần này xin các bạn tham khảo ở phần sau mang tên là sử dụng các từ khóa chỉ mức độ truy cập)

- Thuộc tính của lớp là độ dài ba canh của tam giác

Các phương thức của lớp:

- Phương thức nhập thông tin
- Phương thức hiện thông tin
- Phương thức kiểm tra loại tam giác
- Phương thức tính chu vi tam giác
- Phương thức tính diên tích tam giác

Ở đây, vì phương thức nhập thông tin và phương thức hiện thông tin chúng ta mong muốn sẽ được sử dụng trong bất kì hàm nào nên ta xây dựng có thuộc tính ***public***, còn phương thức cho biết loại tam giác, tính diện tích tam giác và chu vi tam giác thì chúng ta mong muốn có tính chất bảo vệ và không cho các thành phần bên ngoài tác động vào nó thì sẽ xây dựng có thuộc tính truy xuất la ***private***. Vì sao vậy? Muốn xây dựng có thuộc tính truy cập là public có được không? Câu trả lời là có nhưng không có tính chất che chắn dữ liệu và một thành phần có thể làm thay đổi đến nó trong khi tiêu chí xếp loại tam giác, tính diện tích và chu vi tam giác thì đối với tam giác nào cũng giống nhau do đó ta xây dựng nó là private.

b, Giải mẫu:

```
using System;
namespace VD2
{
   class tamgiac
   {
      private int a;
      private int b;
      private int c;
      private int i=0;
      public void nhap()
      {
```

```
        Console.WriteLine("nhap thong so cho tam giac");
        a=Convert.ToInt32(Console.ReadLine());
        b=Convert.ToInt32(Console.ReadLine());
        c=Convert.ToInt32(Console.ReadLine());
    }
    public void hien()
    {
        Console.WriteLine("tam giac ma ban vua nhap la:");
        Console.WriteLine("do dai ba
canh:,{0},{1},{2}",a,b,c);
        Console.WriteLine("chu vi:{0}",chuvi());
        Console.WriteLine("dien tich:{0}",dientich());
        i = loaitg();
        switch(i)
        {
            case 1:
                Console.WriteLine("la tam giac deu");break;
            case 2:
                Console.WriteLine("la tam giac vuong
can");break;
            case 3:
                Console.WriteLine("la tam giac can");break;
            case 4:
                Console.WriteLine("la tam giac vuong");break;
            case 5:
                Console.WriteLine("la tam giac thuong");break;
        }
    }
    private double dientich()
    {
        return 0.25 * Math.Sqrt((a+b+c) * (a+b-c) * (a-b+c) *
(b+c-a));
    }
    private double chuvi()
    {
```

```
            return (a + b + c);
        }
        private int loaitg()
        {
            if((a==b) || (b==c) || (a==c))
            {
                if (a==b && b==c)
                    return 1;
                else
                {
                  if ((a*a==b*b+c*c) || (b*b==a*a+c*c) ||
(c*c==a*a+b*b))
                        return 2;
                    else
                        return 3;
                }
            }
            else
            {
                if (a*a==b*b+c*c ||b*b==a*a+c*c ||c*c==a*a+b*b)
                    return 4;
                    else
                    return 5;
            }
        }
    }
    class tester
    {
        static void Main(string[] args)
        {
            tamgiac tg = new tamgiac();
            tg.nhap();
            tg.hien();
            Console.ReadLine();
        }
```

```
    }
}
```

Kết quả sau khi chạy chương trình như sau:

```
nhap thong so cho tam giac
3
4
5
tam giac ma ban vua nhap la:
do dai ba canh: 3,4,5
chu vi: 12
dien tich:6
la tam giac vuong
```

Các bạn chú ý vào phần in nghiêng. Đó là phương thức hien() của lớp nhưng chúng lại có thể truy nhập đến phương thức tính diện tích và phương thức hiện của lớp. Như vậy là trong cùng một lớp các phương thức cùng cấp có thể truy xuất lẫn nhau.

Ví dụ 3:

Sử dụng lớp stack ở trên để xây dựng một chương trình chuyển đổi cơ số đếm từ hệ 10 sang hệ 2, hê 8, hệ 16 bằng cách sử dụng phép chia liên tiếp.

a, Hướng dẫn:

Muốn đổi một số từ hệ 10 sang hệ 2, 8,16 thì ta lấy số đó chia liên tiếp cho hệ số muốn đổi. Ở mỗi lần chia, được số dư thì ta **push** nó vào stack. Thực hiện tiếp cho đến khi số dư nhỏ hơn số chia thì dừng lại và **pop** lần lượt các phần tử trong stack ra ta được kết quả.

b, Bài giải mẫu:

```
using System;
using System.Collections.Generic;
```

```
using System.Text;
namespace stack
{
   class Stack
   {
      private int top;
      private int[] s;
      public bool empty()
      {
         return (top == -1);
      }
      public bool full()
      {
         return (top >= s.Length);
      }
      public Stack()
      {
         s = new int[20];
         top = -1;
      }
      public void push(int x)
      {
         if (!full())
         {
            top = top + 1;
            s[top] = x;
         }
         else
            Console.Write("Stack tran");
      }
   }
      public int pop()
      {
         if (empty())
         {
```

```
            Console.Write("Stack can");
            return 0;
        }
        else
            return s[top--];
    }
}
class tester
{
    static void Main()
    {
        int n, a, k;
        Console.Write("Nhap vao so can doi:");
        n = int.Parse(Console.ReadLine());
        Console.Write("Nhap vao he so can chuyen:");
        a = int.Parse(Console.ReadLine());
        Stack T = new Stack();
        if (a == 2)
        {
            while (n != 0)
            {
                T.push(n % 2);
                n = n / 2;
            }
            Console.Write("Ket qua chuyen sang he nhi phan:");
            while (!T.empty())
            {
                Console.Write("{0}", T.pop());
            }
        }
        if (a == 8)
        {
            while (n != 0)
            {
```

```
                T.push(n % 8) ;
                n = n / 8;
            }
            Console.Write("Ket qua chuyen sang he bat phan:");
            while (!T.empty())
            {
                Console.Write("{0}", T.pop());
            }
        }
        if (a == 16)
        {
            string st = "0123456789ABCDEF";
            while (n != 0)
            {
              T.push((int)st[n % 16]);
               n = n / 16;
            }
            Console.Write("Ket qua chuyen sang he thap luc
phan:");
            while (!T.empty())
            {
                Console.Write("{0}",(char)T.pop());
            }
        }
        Console.ReadLine();
    }
  }
}
```

Kết quả sau khi chạy chương trình là:

```
Nhap vao so can doi: 8
Nhap vao he can doi: 2
Ket qua: 1000
```

Ở cả hai ví dụ trên có một phương thức public Stack () đây chính là một phương thức thiết lập của lớp stack nhưng không có tham số. Chính phương thức này đã khởi tạo cho chúng ta một stack với số phần tử tối đa là n (n=20).

3. Lớp lồng nhau và cách truy xuất

Xây dựng một lớp đơn giản có sử dụng lớp lồng nhau. Xây dựng lớp tinhtong có phương thức nhập, tính tổng. Trong lớp có chứa lớp nhỏ hơn là lớp tính hiệu của ba số trong đó một số là thành phần dữ liệu của lớp chứa.

a, Hướng dẫn:

Thành phần dữ liệu của lớp chứa (lớp tinhtong) gồm :

public int x;
public int y;
static int c; khai báo là static để lớp ở trong sau này có thể sử dụng được

Phương thức nhap(), tinhtong()

Thành phần dữ liệu của lớp ở trong (lớp tinhieu) gồm

int a;
int b;

Phương thức gồm nhap(), hieu() để tính hiệu trong đó có sử dụng biến c của lớp chứa.

Sử dụng lớp tinhieu: Khai báo một đối tượng của lớp tinhhieu thì phải chỉ rõ cả lớp chứa nó nữa. Ví dụ khai báo một đối tượng con của lớp tinhhieu thì phải khai báo: *tinhtong.tinhhieu con = **new** tinhtong.tinhhieu();*

b, Bài giải mẫu:

```
using System;
namespace loplongnhau
{
   class tinhtong // lop chua
   {
      public int x;
      public int y;
```

```
        static int c;
        public void nhap()
        {
            Console.Write("x =");
            x = Convert.ToInt32(Console.ReadLine());
            Console.Write("y =");
            y = Convert.ToInt32(Console.ReadLine());
            Console.Write("c =");
            c = int.Parse(Console.ReadLine());
        }
        public double tong()
        {
            return (x + y);
        }
        public class tinhhieu // lop long
        {
            int a;
            int b;
            public void nhap()
            {
                Console.WriteLine("nhap thong tin cho doi tuong
lop ben trong:");
                Console.Write("a =");
                a = int.Parse(Console.ReadLine());
                Console.Write("b =");
                b = int.Parse(Console.ReadLine());
            }
           public int hieu()
            {
                return (a - b - c);
            }
        }
        static void Main(string[] args)
        {
            Console.WriteLine("khai bao va su dung lop chua:");
```

```
            tinhtong cha = new tinhtong();
            cha.nhap(); // phuogn thuc nay cua lop cha
            Console.WriteLine("x+y={0}", cha.tong());
            Console.WriteLine("khai bao va su dung lop ben
trong:");
            tinhtong.tinhhieu con = new tinhtong.tinhhieu();
            con.nhap();
            Console.Write("a-b-c={0}",con.hieu());
            Console.ReadKey();
        }
    }
}
```

Kết quả sau khi chạy chương trình**:**

```
Khai bao va su dung lop chua:
x  = 9
```

```
y  = 6
c  = 3
x+y=15
khai bao va su dung lop con:
nhap thong tin cho doi tuong lop con:
a = 8
b = 6
a - b - c = -1
```

4. Chương trình khai báo nhiều lớp

Xây dựng 2 lớp: lớp hocsinh bao gồm họ tên học sinh, mã số học sinh, năm sinh của học sinh và lớp danhsach dùng để chứa toàn bộ học sinh đã nhập, sắp xếp thứ tự lại theo năm sinh.

a, Hướng dẫn:

Các thuộc tính của lớp hocsinh:

```
private string mshv;
private string hoten;
```

private int namsinh;

Các phương thức:

Phương thức nhập thông tin,

Phương thức hiện thông tin

Phương thức so sánh năm sinh của sinh viên trả ra giá trị là 0; 1 hay -1 khi tương ứng với kết quả bằng, lớn hơn, nhỏ hơn.

Các thuộc tính của lớp dachsach chính là một mảng các học sinh ở lớp hocsinh

Phương thức gồm

Nhập: thi hành phương thức nhap() cho từng đối tượng kiểu hocsinh

In danh sách: thi hành phương thức hien() cho từng đối tượng kiểu hocsinh

Sắp xếp danh sách theo thứ tự năm sinh giảm dần

b, Bài giải mẫu:

```
namespace vidu6
{
   class Program
   {
      class Hocsinh
      {
         private string ms;
         private string hoten;
         private int namsinh;
         public Hocsinh()
         {
         }
         public void nhap()
         {
            Console.Write("ma so:"); ms = Console.ReadLine();
            Console.Write("ho ten:"); hoten =
Console.ReadLine();
            Console.Write("nam sinh:");
            namsinh = int.Parse(Console.ReadLine());
         }
```

```
        public void nhap()
        {
            Console.Write("ma so:"); ms = Console.ReadLine();
            Console.Write("ho ten:"); hoten =
Console.ReadLine();
            Console.Write("nam sinh:");
            namsinh = int.Parse(Console.ReadLine());
        }
        public int ssnamsinh(Hocsinh a,Hocsinh b)
        {
            int k;
            if (a.namsinh > b.namsinh)
                k =1;
             else {
                if (a.namsinh == b.namsinh)
                    k= 0;
                else k = -1;
            }
            return k;
        }
        public void print()
        {
            Console.WriteLine("\t{0}\t{1}\t{2}", ms, hoten,
namsinh);
        }
    }
    class Danhsach
    {
        int n, i;
        private Hocsinh[] ds;
        public Danhsach(int n)
        {
            ds = new Hocsinh[n];
        }
        public void nhapds()
```

```
        {
            Console.Write("so hoc sinh = ");
            n = Convert.ToInt32(Console.ReadLine());
            ds = new Hocsinh[n];
            for (i = 0; i < ds.Length; i++)
                ds[i] = new Hocsinh();
            for (i = 0; i < n; i++)
            {
                Console.WriteLine("hoc sinh thu {0}:", i + 1);
                ds[i].nhap();
            }
        }
        public void printds()
        {
            Console.WriteLine("danh sach hoc sinh ban dau:");
            for (i = 0; i < ds.Length; i++)
            {
                Console.WriteLine("\t \ t  \t ");
                Console.Write("\t{0}", i + 1);
                ds[i].print();
            }
        }
        public void sapxep()
        {
            for (i = 0; i < n; i++)
            {
                Hocsinh max = ds[i];
                for (i = 0; i < n; i++)
                {
                    if (max.ssnamsinh(ds[i], ds[i + 1]) == 1)
                    {
                        max= ds[i + 1];
                        ds[i + 1] = ds[i];
                        ds[i] = max;
                    }
```

```
                    for (i = 0; i < n; i++)
                    ds[i].print();
                }
            }
        }
        static void Main(string[] args)
        {
            Danhsach d = new Danhsach(3);
            d.nhapds();
            d.printds();
            Console.WriteLine("danh sach sap sep theo nam
sinh:");
            d.sapxep();
            Console.ReadLine();
        }
    }
}
```

Kết quả sau khi chạy chương trình:

```
So hoc sinh = 3
Hoc sinh thu 1:
Ma so: hs1
Ho ten: nguyen thi mai huong
Nam sinh 1987
Hoc sinh thu 2:
Ma so: hs2
Ho ten: nguyen van hai
Nam sinh 1986
Hoc sinh thu 3:
Ma so: hs3
Ho ten: ta van ngoc
Nam sinh 1989
Danh sach hoc sinh ban dau:
        Mhs  ho ten                 nam sinh
        hs1   nguyen thi mai huong  1987
        hs2   nguyen van hai        1986
        hs3   ta van ngoc           1989
Danh sach sap xep theo nam sinh:
        hs2   nguyen van hai        1986
        hs1   nguyen thi mai huong  1987
        hs3   ta van ngoc           1989
```

5. Từ khóa this

1. *Sử dụng this để phân biệt tham số phương thức và biến thành viên của lớp.*

Xây dựng lớp phuơng trình bậc hai đơn giản và thực hiện tính nghiệm cho phương trình đó. Trong chương trình có xây dựng hàm thiết lập không tham số và có tham số. Sử dụng từ khóa thí trong bài để phân biệt rõ biến thành viên của lớp và tham số của phương thức.

a, Hướng dẫn:

Vì trong bài chỉ yêu cầu xây dựng một lớp đơn giản nên thành phần thuộc tính và phương thức gồm có:

Thuộc tính: Các hệ số của phương trình bậc hai;

- public double a, b, c;

Phương thức :

- Phương thức khởi tạo không tham số: public ptbachai()
- Phương thức khởi tạo ba tham số:public ptbachai(int a, int b, int c)

Vì trong phương thức này ta sử dụng tham số của phương thức trùng với biến thành viên của lớp nên ta phải sử dụng từ khóa thí để phân biệt giữa chúng.

- Phương thức nhập
- Phương thức hiện phương trình
- Phương thức tính nghiệm

b, Bài giải mẫu:

```
using System;
namespace vidu4
{
   class Ptbachai
   {
      public double a, b, c;
      public ptbachai()
      {
         a = 5;
         b = 3;
         c = 8;
      }
      public ptbachai(int a, int b, int c)
      {
         this.a = a;
         this.b = b;
         this.c = c;
      }
      public void nhap()
      {
         Console.WriteLine("nhap he so cho PTB2:");
         a = Convert.ToInt32(Console.ReadLine());
         b = Convert.ToInt32(Console.ReadLine());
         c = Convert.ToInt32(Console.ReadLine());
      }
      private void tinhnghiem()
      {
         double x, y;
         double delta;
         delta=this.b*this.b-4*this.a*this.c;
         if (delta < 0)
         {
            Console.WriteLine("khong co nghiem");
         }
         else
```

```
        {
            if (delta == 0)
            {
                x = -this.b / 2 *this. a;
                Console.WriteLine("nghiem kep x = y ={0}",x);
                    }
            else
            {
                x = (-this.b - Math.Sqrt(delta)) / 2 *this.a;
                y = (-this.b + Math.Sqrt(delta)) / 2 *this.a;
                Console.WriteLine("PT co hai nghiem phan
biet:");
                Console.WriteLine("x = {0}; y = {1}", x, y);
            }
        }
    }
    public void hien()
    {
        Console.WriteLine("phuong trinh bac hai:");
        Console.WriteLine("{0}*x2 + {1}*x + {2}", this.a,
this.b, this.c);
    }
    class tester
    {
        static void Main(string[] args)
        {
            ptbachai x = new ptbachai();
            x.hien(); x.tinhnghiem();
            ptbachai y = new ptbachai();
            y.nhap();
            y.hien();
            y.tinhnghiem();
            Console.ReadKey();
        }
    }
```

```
    }
}
```

Trong phương thức khởi tạo ba tham số có câu lệnh this.a=a thì this.a chính là biến thành viên của lớp còn a ở đây lại là tham số của phương thức mà bạn nhập vào khi chạy chương trình và nó được truyền vào bằng câu lệnh gán. Các biến public được truy xuất tại mọi nơi trong chương trình. Biến khai báo bằng từ khóa này được hiểu như là biến công cộng của lớp và có thể truy xuất bất kì nếu muốn.
Kết quả sau khi chạy chương trình:

```
phuong trinh bac hai:
5*x2 + 3*x + 8
khong co nghiem
nhap he so cho PTB2
5
8
- 4
phuong trinh bac hai:
5*x2 + 8*x + -4
PT co hai nghiem phan biet:
x = -50;  y = 10
```

2. *Sử dụng từ khóa thí để trao đối tượng hiện hành như một tham số, minh học*
sử dụng biến khai báo private

Xây dựng một lớp nhanvien nhập họ tên, chức vụ nhân viên, tiền lương nhân viên, hiện thông tin có liên quan. Xây dựng một lớp dùng để tính thuế mà nhân viên phải nộp cho nhà nước từ tiền lương đó. (Số tiền này hiển thị cùng các thông tin liên quan đến nhân viên trong phương thức hiển thị của lớp nhanvien).

a, Hướng dẫn:

Các thành phần của lớp nhanvien chúng ta vẫn xây dựng bình thường như các ví dụ trước. Chỉ khác ở chỗ trong phương thức hiện có tham chiếu tới đối tượng của lớp Thue dùng để tính thuế phải nộp như sau: Thue.Thuephainop(this) trong đó Thuephainop là phương thức tính thuế của class Thue có đối số là đối tượng lớp nhanvien.

b, Bài giải mẫu:

```
using System;
class Nhanvien
{
   private string hoten;
   private string chucvu;
   private decimal tienluong;
   public Nhanvien()
   {
   }
   public Nhanvien(string hoten, string chucvu, decimal
tienluong)
   {
      Console.WriteLine("Su dung PTTL 3 tham so:");
      this.hoten = hoten;
      this.chucvu = chucvu;
      this.tienluong = tienluong;
   }
   public void nhap()
   {
      Console.Write("Ho ten:");
      hoten = Console.ReadLine();
      Console.Write("Chuc vu:");
      chucvu = Console.ReadLine();
      Console.Write("Tien luong:");
      tienluong = decimal.Parse(Console.ReadLine());
   }
```

```
    public void dipslayNV()
    {
        Console.WriteLine("Ho ten:\t{0}", hoten);
        Console.WriteLine("Ma chuc vu:\t{0}", chucvu);
        Console.WriteLine("Tien luong la:\t{0:C}", tienluong);
        // Trao doi tuong cua lop Thue nhu mot doi so
        Console.WriteLine("Thue phai nop:\t{0:C} ",
Thue.Thuephainop(this));
    }
    // xay dung phuong thuc chi doc gia tri cua doi tuong
    public decimal luong
    {
        get { return tienluong; }
    }
}
class Thue
{
    public static decimal Thuephainop(Nhanvien E)
    {
        return 0.03m * E.luong;
        // E.luong: doc gia tri cua truong luong
    }
}
class MainClass
{
    static void Main()
    {
        Nhanvien E1 = new Nhanvien("Nguyen Tien Quynh
Anh","thu ki", 2200000);
        E1.dipslayNV();
        Nhanvien E2 = new Nhanvien();
        E2.nhap();
        E2.dipslayNV();
        Console.ReadKey();
    }
```

```
}
```

Kết quả khi chạy chương trình:

```
Su dung PTTL 3 tham so:
Hien thi thong tin:
Ho ten:  Nguyen Tien Quynh Anh
Chuc vu:   thu ki
Tien luong: $2,200,000.00
Thue phai nop:   $ 66,000.00
Nhap thong tin:
Ho ten: Chu Thi Minh Ha
Chuc vu:  ke toan
Tien luong:   2546000
Hien thi thong tin:
Ho ten: Chu Thi Minh Ha
Chuc vu:  ke toan
Tien luong la:   $2546000
Thue phai nop:   $76,380.00
```

6. Hàm thiết lập và hàm thiết lập sao chép, hàm sao chép cho lớp vector

Xây dựng một lớp vector đơn giản có sử dụng các hàm thiết lập không tham số, một tham số và hai tham số.Trong lớp có sử dụng hàm thiết lập sao chép, hàm sao chép cho đối tượng của lớp (phân biệt hai cách dùng của constructor)

a, Hướng dẫn:

Vì vecto là một đối tượng mà các thành phần dữ liệu không tĩnh nên phải khai báo vùng nhớ dùng để chứa các tọa độ là một mảng một chiều: float[] v

Đối với các hàm thiết lập không tham số và một tham số thì không có gì đáng ngại cả. Chỉ có hàm thiết lập hai tham số thì cần chú ý một chút: Trong hàm thiết lập hai tham số thì một tham số là số chiều của vecto còn tham số còn lại là một mảng một chiều mà

ta sẽ đọc các phần tử của mảng vào cho vecto nên phương thức này được khai báo như sau:public vecto(int size, float[] u) trong đó float [] u chính là mảng có chứa sẵn các phần tử.

Với hàm thiết lập sao chép thì chính là việc sao chép lại giá trị của các thành phần dữ liệu đã có trong một đối tượng cùng lớp vecto và đưa vào vùng nhớ của vecto đã được cấp phát bộ nhớ.

```
public vecto(int size, float[] u)
    {
       int i; n = size;
       v = new float[n];
       for (i = 0; i < n; i++)
       {
          v[i] = u[i];
       }
    }
```

b, Chương trình mẫu:

```
using System;
namespace vidu7
{
   class Vecto
   {
      int n = 20;
      float[] v;
      public vecto()
      {
         int i;
         Console.WriteLine("\t Su dung ham thiet lap khong
tham so:");
         Console.Write("\t So chieu cua vecto la:");
         i = Convert.ToInt32(Console.ReadLine());
         n = i;
         v = new float[n];
         Console.WriteLine("\t nhap thong so cho vecto:");
```

```
        for (i = 0; i < n; i++)
        {
            Console.Write("\t toa do thu {0}:", i);
            Console.Write("\t v[{0}]=", i + 1);
            v[i] = int.Parse(Console.ReadLine());
        }
    }
    public vecto(int size)
    {
        int i;
        Console.WriteLine("\t Su dung ham thiet lap mot tham
so:");
        n = size;
        v = new float[n];
        Console.WriteLine("\t nhap thong so cho vecto:");
        for (i = 0; i < n; i++)
        {
            Console.Write("\t toa do thu {0}:", i);
            Console.Write("\t v[{0}]=", i + 1);
            v[i] = int.Parse(Console.ReadLine());
        }
    }
    public vecto(int size, float[] u)
    {
        int i;
        Console.WriteLine("\t Su dung ham thiet lap hai tham
so:");
        n = size;
        v = new float[n];
        Console.WriteLine("\t tao mot vecto tu mot mang mot
chieu:");
        for (i = 0; i < n; i++)
        {
            v[i] = u[i];
        }
```

```
        }
        public vecto(vecto u)
        {
            int i;
            Console.WriteLine("\t su dung ham thiet lap sao
chep:");
            // xin cap phat vung nho co kich thuoc bang voi doi
tuong cu
            v = new float[n = u.n];
            // thuc hien gan cac vung nho dong cua doi tuong cu
sang doi tuong moi
            for (i = 0; i < n; i++)
            {
                v[i] = u.v[i];
            }
        }
        public void saochep(vecto b)
        {
            n = b.n;
            Console.WriteLine("\t Su dung ham sao chep");
            for (int i = 0; i < n; i++)
            {
                this.v[i] = b.v[i];
            }
        }
        public void hien()
        {
            Console.Write(\t "(");
            for (int i = 0; i < v.Length; i++)
                if (i == v.Length - 1)
                    Console.Write("{0}", v[i]);
                else
             Console.Write("{0},", v[i]);
              Console.WriteLine(")");
        }
```

```
    class tester
      {
         static void Main(string[] args)
         {
      vecto v1 = new vecto();
                Console.Write("\t v1:");v1.hien();
                vecto v2 = new vecto(4);
                Console.Write("\t v2:");v2.hien();
                float[] a ={ 12, 8, 11, 3, 20 , 85 };
                vecto v3 = new vecto(4, a);
                Console.Write("\t v3:"); v3.hien();
                vecto v4 = new vecto(v3);
                Console.Write("\t v4:"); v4.hien();
                if (v3.n==v2.n)
                   v3.saochep(v2);
                else
                {
                   Console.WriteLine("\t khong tien hanh sao
chep. Gia tri vecto van la:");
                }
                Console.Write("\t v3:"); v3.hien();
                Console.ReadLine();
            }
      }
   }
}
```

Kết quả sau khi chạy chương trình:

```
Su dung ham thiet lap khong tham so:
So chieu cua vecto la: 2
Nhap thong so cho vecto:
v[1]=10
v[2]=89
v1:       (10,89)
Su dung ham thiet lap một tham so:
Nhap thong so cho vecto:
v[1]=4
v[2]=5
```

7. Sử dụng hàm thiết lập sao chép và hàm hủy bỏ

Xây dựng lớp thoigian lấy về thời gian hiện hành. Có sử dụng phương thức khởi tạo, phương thức sao chép và phương thức hủy bỏ.

a, Hướng dẫn:
Để lấy được thời gian hiện hành thì ta sử dụng hàm now.
Các thuộc tính của lớp gồm ngày, tháng, năm, giờ, phút, giây được lấy theo thời gian hệ thống thông qua hàm System.DateTime.Now
Các phương thức của lớp gồm:

- public dipslay() dùng để hiển thị thông tin
- public ThoiGian (System.DateTime tg): thiết lập sao chép thời gian từ một đối tượng đã có
- public ThoiGian(int Date,int Month,int Year,int Hour, int Minute,int Second): PT thiết lập 6 tham số khởi tạo giá trị cho biến đối tượng.
- public ThoiGian(int Date,int Year,int Hour,int Second): PT thiết lập 4 tham số khởi tạo giá trị cho biến đối tượng.

- public void saochep(ThoiGian TG) dùng để sao chép đối tượng.
- ~ ThoiGian(): là phương thức hủy bỏ đối tượng của lớp.Phương thức này sẽ được gọi tự động bởi trình thu gom rác của C#

b, Bài giải mẫu

```
using System;
namespace saochep
{
   public class ThoiGian
   {
      private int Nam ;
      private int Thang = 7;
      private int Ngay = 30;
      private int Gio = 5;
      private int Phut = 19;
      private int Giay = 23;
      System.DateTime now = System.DateTime.Now;
      public void dipslay()
      {
         System.Console.WriteLine("\t Ngay:{0}/{1}/{2}",
Ngay, Thang, Nam);
       System.Console.WriteLine("\t Gio:{0}:{1}:{2}", Gio,
Phut, Giay);
      }
      public ThoiGian( System.DateTime tg)
      // hàm thiết lập sao chép
      {
         Nam = tg.Year;
         Thang = tg.Month;
         Ngay = tg.Day;
         Gio = tg.Hour;
         Phut = tg.Minute;
         Giay = tg.Second;
      }
```

```
        public void saochep(ThoiGian TG)
        {
            this.Gio=TG.Gio;
            this.Phut=TG.Phut;
            this.Giay=TG.Giay;
            this.Ngay = TG.Ngay;
            this.Thang = TG.Thang;
            this.Nam = TG.Nam;
        }
       public ThoiGian( int Date,int Month,int Year,int Hour,
int Minute,int Second)
            // ham thiet lap 6 tham so
        {
            Nam = Year;
            Thang = Month;
            Ngay = Date;
            Gio = Hour;
            Phut = Minute;
                        Giay = Second;
        }
        public ThoiGian(int Date, int Year, int Hour, int
Second)
        {
            Nam = Year;
            Ngay = Date;
            Gio = Hour;
            Giay = Second;
        }
        ~ThoiGian()
        {
            Console.WriteLine("\t\t doi tuong da duoc huy");
            GC.SuppressFinalize(this);
            Console.ReadKey();
        }
    }
```

```
  public class Tester
   {
      static void Main()
      {
         System.DateTime currentTime =
System.DateTime.Now;
         Console.WriteLine("\t Thoi gian hien hanh la:\t\t");
         ThoiGian tghh = new ThoiGian( currentTime );
         tghh.dipslay();
         Console.WriteLine("\t t1 khoi tao tu ham thiet lap sao
chep:\t");
         ThoiGian t1 =tghh;
         t1.dipslay();
         Console.WriteLine("\t t2 goi ham thiet lap 6 tham
so:\t");
         ThoiGian t2 = new ThoiGian(12,12,2007,4,56,39);
         t2.dipslay();
         Console.WriteLine("\t t1 sao chep tu doi tuong t2:\t\t");
         t1.saochep(t2);
         t1.dipslay();
         Console.WriteLine("\t t3 goi tu ham thiet lap 4 tham
so:\t");
         ThoiGian t3 = new ThoiGian(15, 2008, 12, 40);
         t3.dipslay();
         Console.ReadKey();
      }
   }
}
```

Kết quả sau khi chạy chương trình:

```
Thoi gian hien hanh la:
Ngay: 8/6/2007
Gio: 2:29:26
T1 khoi tao tu ham thiet lap sao chep:
Ngay: 8/6/2007
Gio: 2:29:26
T2 goi tu ham thiet lap 6 tham so:
Ngay: 12/12/2007
```

Trong ví dụ trên, hàm hủy bỏ đối tượng bằng phương thức SuppressFinalize của đối tượng GC (Garbage Collector). Dòng thông báo "doi tuong da duoc huy" xuất hiện ngay sau khi ta gõ một phím bất kì cho thấy phương thức hủy bỏ đã được goi tự đông trước khi kết thúc chương trình và giải phóng vùng nhớ cho các đối tượng. Trong phương thức thiết lập 4 tham số, các tham số không được khởi tạo sẽ lấy giá trị mặc định của biến hay giá trị khởi gán ban đầu nên có kết quả như trên.

8. Kiểu dữ liệu mảng, các phương thức có giá trị trả về , constructor sao chép

Xây dựng lớp Matrix cho các ma trận các phương thức nhập ma trận từ bàn phím, hiện ma trận ra màn hình, cộng hai ma trận, trừ hai ma trận, nhân hai ma trận.

a, Hướng dẫn:

Ma trận được lưu trữ các phần tử trong mảng hai chiều nên phải khai báo một mảng int[,] để lưu trữ các thành phần đó.

Các phương thức thao tác với ma trận:

- Phương thức nhập: sẽ tiến hành nhập các phần tử của ma trận theo từng hàng
- Phương thức hiện ma trận lên màn hình

- Phương thức chuyển vị hai ma trận có giá trị trả về là một ma trận. Ở đây chỉ là việc đổi vị trí: cv.m[i,j]=this.m[j,i]
- Phương thức tong(matran b) làm công việc cộng hai ma trận: tmp.m[i, j] = this.m[i, j] + b.m[i, j]
- Phương thức tich(int k:): tính tích của ma trận với một số
- Phương thức tich(matran b): tính tích của hai ma trận (đây là phương thức chồng với phương thức tich của ma trận với một số)
- Phương thức hieu(matran b): tính hiệu của hai ma trận

Để tính được tổng, hiệu, tích của hai ma trận thì chúng phải tương thích nhau về điều kiện cụ thể hai ma trận cùng có số hàng và số cột thì mới cộng, trừ được, ma trận này phải có số cột bằng số hàng của ma trận kia thi mới thực hiện được phép nhân hai ma trận. Trong những trường hợp khác thì sinh lỗi. Vì vậy cần có công việc kiểm tra điều kiện tương thích. Và các phương thức này đều trả ra giá trị là một đối tượng lớp ma trận

b, Bài giải mẫu:

```
using System;
namespace matran
{
   class Matrix
   {
      int i, j;
      int[,] m;// mảmg để lưu trữ phần tử của ma trận
      public Matrix()
      {
         m = new int[10, 20];
      }
      public Matrix(int h, int c)
      {
         m = new int[h, c];
      }
      public void nhap()
      {
         int i, j;
         for (i = 0; i < m.GetLength(0); i++)
         {
            Console.WriteLine("Nhap cac phan tu cho hang thu {0} :", i + 1);
            for (j = 0; j < m.GetLength(1); j++)
            {
               Console.Write("m[{0},{1}]= ", i, j);
               m[i, j] = Convert.ToInt32(Console.ReadLine());
            }
         }
```

```
    }
    public Matrix chuyenvi()
    {
        Matrix cv = new Matrix(this.m.GetLength(0), this.m.GetLength(1));
        for (i=0;i< this.m.GetLength(0);i++)
        {
            for (j=0;j<this.m.GetLength(1);j++)
                cv.m[i,j]=this.m[j,i];
        }
        return cv;
    }
    public Matrix tong(Matrix b)
    {
        Matrix tmp = new Matrix(this.m.GetLength(0), this.m.GetLength(1));
        for (i = 0; i < this.m.GetLength(0); i++)
        {
            for (j = 0; j < this.m.GetLength(1); j++)
                tmp.m[i, j] = this.m[i, j] + b.m[i, j];
        }
        return tmp;
    }
    public Matrix hieu(Matrix b)
    {
        Matrix tmp = new Matrix(this.m.GetLength(0), this.m.GetLength(1));
        for (i = 0; i < this.m.GetLength(0); i++)
        {
            for (j = 0; j < this.m.GetLength(1); j++)
```

```
{
   Matrix tmp = new Matrix(this.m.GetLength(0), b.m.GetLength(1));
   for (i = 0; i < this.m.GetLength(1); i++)
                for (j = 0; j < b.m.GetLength(0); j++)
      {
         tmp.m[i, j] = 0;
         for (int k = 0; k < tmp.m.GetLength(1); k++)
            tmp.m[i, j] = tmp.m[i, j] + this.m[j, k] * b.m[k, j];
      }
   return tmp;
}
public bool tuongthichcongtru(Matrix b)
{
   return (this.m.GetLength(0) == b.m.GetLength(1));
}
public bool tuongthichnhan(Matrix b)
{
   return (this.m.GetLength(1) == b.m.GetLength(0));
}
```

```
                Console.WriteLine("hai ma tran khong tuong thich hang va cot");
               }
                Console.ReadKey();
            }
        }
    }
}
```

Kết quả sau khi chạy chương trình:

```
nhap ma tran a:
m[0,0]=1
m[0,1]=2
m[1,0]=3
m[1,1]=1
ma tran a:
        1       2
        2       1
nhap ma tran b:
m[0,0]=0
m[0,1]=1
m[0,2]=3
m[1,0]=5
m[1,1]=2
m[1,2]=3
ma tran b la:
        0   1    3
        5   2    3
tich cua ma tran a va b la:
       10   5    9
        5   5   12
ma tran chuyen vi cua a la;
        10      5
        5       5
        9      12
```

9. Sử dụng các thành phần tĩnh

Xây dựng lớp Nhanvien để quản lý nhân viên bao gồm mã nhân viên, chức vụ của nhân viên và họ tên nhân viên. Trong lớp sử dụng thành phần tĩnh như biến toàn cục trong phạm vi lớp theo dõi

khi một đối tượng được tạo ra (bằng cách cho biết số thứ tự của nhân viên mới đó).

a, Hướng dẫn:

Thành phần dữ liệu của lớp gồm có: Mã nhân viên, chức vụ và họ tên của nhân viên đều là dữ liệu kiểu string.

Để theo dõi đối tượng mỗi khi một nhân viên được tạo ra thì dùng một biến static int demnhanvien có mức độ truy cập là public để có thể truy cập bất cứ khi nào.

Phương thức của lớp bao gồm: Các phương thức nhập, hiện, thiết lập thông thường. Ngoài ra còn có phương thức

```
public static int sttnhanvien()
{
   return ++demnhanvien;
}
```

để tăng số lượng nhân viên hiện thời. Số thứ tự của nhân viên mặc định chính bằng số lượng nhân viên hiện thời.

b, Bài giải mẫu:

```
using System;
public class nhanvien
{
   public string manv;
   public string hoten;
   public string chucvu;
   public static int demnhanvien;
   public nhanvien() // pt thiet lap
   {
   }
   public nhanvien(string manv, string chucvu, string hoten)
   {// phương thức thiết lập
      this.manv= manv;
      this.chucvu = chucvu;
      this.hoten = hoten;
   }
   public void nhap()
   {
      Console.Write("\t Nhap vao ma cua nhan vien: ");
      string nvid = Console.ReadLine();
      Console.Write("\t Nhap vao chu vu cua nhan vien:");
      string cv=Console.ReadLine();
      Console.Write("\t Nhap vao ho ten cua nhan vien: ");
      string name = Console.ReadLine();
```

```
        manv = nvid;
        chucvu = cv;
    }
  public void hienthi()
    {
        Console.WriteLine("\t Ma nhan vien \t:   {0}", this.manv);
        Console.WriteLine("\t Chuc vu \t:   {0}", this.chucvu);
        Console.WriteLine("\t Ho ten cua nhan vien \t: {0}", this.hoten);
        // truy nhập đến thành phần dữ liệu tĩnh thông qua tên lớp
        Console.WriteLine("\t So thu tu \t: {0}", nhanvien.demnhanvien);
    }
    public static int sttnhanvien()
    {
        return ++demnhanvien;
    }
}
class tester
{
    static void Main()
    {
        Console.Write("\t So luong nhan vien hien thoi \t:");
        string n = Console.ReadLine();
        nhanvien.demnhanvien = Int32.Parse(n);
        Console.WriteLine("\t tao mot doi tuong:");
        nhanvien nv1 = new nhanvien();
        nv1.nhap();
        // gọi phương thức tĩnh thông qua tên lớp
        nhanvien.sttnhanvien();
        nv1.hienthi();
        Console.WriteLine("\t them mot doi tuong:");
        nhanvien nv2 = new nhanvien("gd","giam doc","Nguyen Ngoc Minh");
        nhanvien.sttnhanvien();
        nv2.hienthi();
        Console.ReadKey();
    }
}
```

Kết quả khi chạy chương trình:

```
So luong nhan vien hien thoi     :445
Tao mot doi tuong:
Nhap vao ma cua nhan vien: nvkh
```

```
Nhap vao chuc vu cua nhan vien: Nhan vien ke hoach
Nhap vao ho ten cua nhan vien: Nguyen Van Hai Lam
Thong tin ve nhan vien:
Ma nhan vien  :nvkh
Chuc vu  :Nhan vien ke hoach
Ho ten          :Nguyen Van Hai Lam
So thu tu :446
Them mot doi tuong:
Thong tin ve nhan vien:
Ma nhan vien  :gd
Chuc vu  :giam doc
Ho ten          :Nguyen Ngoc Minh
So thu tu :447
```

10. Hàm khởi tạo private để cấm tạo thể hiện của đối tượng

Chương trìnhmẫu một ví dụ đơn giản như sau:

```
namespace Static_class
{
   public class vatnuoi
   {
      static vatnuoi()
      {
```

```
            // vì trong lớp không có hàm khởi tạo public
            //vatnuoi cho = new vatnuoi();   // loi
             vatnuoi.dem = 15;
             Console.WriteLine("So luong vat nuoi luc dau la:{0}",vatnuoi.dem);
             Console.WriteLine("Them moi mot con vat nuoi: meo tam the");
             Console.WriteLine("So luong vat nuoi hien thoi la:{0}",
vatnuoi.Soluong());
             Console.ReadLine();
        }
    }
}
```

Trong chương trình nếu bạn cố tình khai báo:*vatnuoi cho = new vatnuoi();* thì sẽ phát sinh lỗi như sau trong quá trình chạy:"Static_class.vatnuoi.vatnuoi()' is inaccessible due to its protection level": không tạo ra được đối tượng lớp vatnuoi bởi nó có mức bảo vệ. Điều này chứng tỏ rằng chính phương thức khởi tạo tĩnh đã ngăn cấm tạo đối tượng thể hiện của lớp. Do vậy ta chỉ có thể truy cập tới các phương thức tĩnh của lớp mà thôi.

Kết quả khi chạy chương trình:

```
Loi goi nay duoc goi dau tien tu phuong thuc khoi tao static
So luong vat nuoi luc dau la: 15
Them mot con vat nuoi: meo tam the
So luong vat nuoi hien thoi la: 16
```

Từ kết quả trên cho thấy Một phương thức khởi tạo tĩnh sẽ được gọi ngay khi chương trình biên dịch mà không cần tạo đối tượng thể hiện nào của lớp.

11. Nạp chồng phương thức

1. Xây dựng lớp phanso và thao tác với lớp với các phương thức bao gồm:

- Phương thức khởi tạo phân số có tử bằng 0 và mẫu bằng 1
- Phương thức khởi tạo (int ts, int ms) để khởi tạo phân số có tử số bằng ts và mẫu số bằng ms
- Phương thức nhập thông tin cho phân số
- Phương thức cộng hai phân số
- Phương thức cộng phân số với một số
- Phương thức trừ hai phân số
- Phương thức trư 1 phân số cho một số
- Phương thức in kết quả ra màn hình

a, Hướng dẫn:

Thuộc tính của lớp: int ts,ms: tử số và mẫu số của phân số

Các phương thức của lớp:

- Khởi tạo không tham số: pulic phanso() và gán giá trị của tử =0, mẫu số=1

- Khởi tạo hai tham số: public phanso(int tu,int mau)
- Cộng phân số với phân số
- cộng phân số với một số
- Trừ phân số cho phân số
- Trừ phân số cho một số
- Phương thức hiện kết quả lên màn hình

Vì trong bài này có nhiều phương thức chồng nhau (sự chồng phương thức) nên các bạn chú ý cách gọi các phương thức.

b, Bài giải mẫu:

```
using System;
using System.Collections.Generic;
using System.Text;
namespace phanso
{
   class phanso
   {
      public int ts;
      public int ms;
      //ham thiet lap ko tham so
      public phanso()
      {
         ts = 0;
         ms = 1;
      }
      //phuong thuc thiet lap hai tham so
      public phanso(int t, int m)
      {
         ts = t;
         ms = m;
      }
{
      phanso kq = new phanso();
      kq.ts = a.ts * b.ms - a.ms * b.ts;
      kq.ms = a.ms * b.ms;
      Console.WriteLine(":{0}/{1}", kq.ts, kq.ms);
   }
   private void hieu(phanso a, int c)
   {
      phanso kq = new phanso();
      kq.ts = a.ts - a.ms * c;
      kq.ms = a.ms;
      Console.WriteLine("{0}/{1}", kq.ts, kq.ms);
   }
   private void hien(phanso p)
   {
      Console.WriteLine("phan so:{0}/{1}",p.ts,p.ms);
```

```
            c = Convert.ToInt32(Console.ReadLine());
            Console.Write("tong cua phan so thu nhat voi c la:");
            KQ.cong(P, c);
            Console.Write("hieu cua phan so thu nhat voi c la:");
            KQ.hieu(P, c);
            Console.Write("tong cua phan so thu hai voi c la:");
            KQ.cong(Q, c);
            Console.ReadKey();
           }
      }
   }
}
```

Kết quả sau khi chạy chương trình là:

```
goi phuong thuc thiet lap khong tham so:
phan so: 0/1
nhap P:
tu so = 5
mau so = 6
phan so : 5/6
nhap Q:
tu so = 3
mau so = 6
phan so: 3/6
tong cua P & Q la: 48/36
```

Trong ví dụ trên có tới 6 phương thức chồng nhau từng đôi một đó là hàm thiết lập không tham số public phanso() và hàm thiết lập hai tham số public phanso(int t, int m) dùng để khởi tạo các giá trị ban đầu cho biến đối tượng phân số; hai phương thức cộng (giữa hai phân số: private void cong(phanso a,phanso b), giữa phân số với một số private void cong(phanso a, int c)) và hai phương thức trừ (giữa phân số với một số private void hieu(phanso a, phanso b), giữa phân số và một số private void hieu(phanso a, int c)). Các phương thức trùng nhau về tên nhưng khác nhau về số lượng tham số, khác nhau về thành phần dữ liệu của từng đối số của phương thức. Ngoài ra thì chồng phương thức còn khác nhau về kiểu dữ liểu trả về của phương thức nữa. Tất nhiên là việc sử dụng chúng cần phải phân biệt rõ từng phương thức dựa vào sự khác nhau của từng phương thức. Nếu không thì sẽ phát sinh lỗi do trình biên dịch.

2. Xây dựng một lớp vecto và hiện các phép toán trên vector

a, Hướng dẫn:

Thuộc tính:

- Cần cấp phát một vùng nhớ động để lưu các thành phần dữ liệu của một vecto.

Phương thức:

- Phương thức khởi tạo
- Phương thức sao chép
- Phương thức tính tổng, hiệu, tích hai vecto (chú ý về phương thức trả ra giá trị)
- Phương thức nhập, hiện vecto

b, Bài giải mẫu:

```
using System;
class vecto
{
   //Khai bao mang v chua các thành phần tọa độ của vector
   int[] v;
   //Phuong thuc khoi tao khong tham so
   public vecto()
   {
      v=new int[10];
   }
   public vecto(int n)// phuong thuc khoi co tham so khoi tao vecto n chieu
   {
      v=new int[n];
   }
   public double dodai()
   {
      return Math.Sqrt(this.tichvohuong(this));
   }
   public void nhap()
   {
      Console.WriteLine("nhap cac thanh phan cua vecto :");
      for (int i = 0; i < v.Length; i++)
      {
         Console.Write("v[{0}]=", i + 1);
         v[i] = int.Parse(Console.ReadLine());
      }
```

```
   }
   // Cong hai vecto
   public vecto cong(vecto u)
   {
      vecto tmp=new vecto(v.Length);
           if (v.Length == u.v.Length)
        for (int i = 0; i < v.Length; i++)
```

```
    {
        vecto tmp = new vecto(v.Length);
        for (int i = 0; i < v.Length; i++)
            tmp.v[i] = v[i] * k;
        return tmp;
    }
    public bool vuonggoc(vecto a)
    {
        return (this.tichvohuong(a) == 0);
    }
    // Tich vo huong hai vecto
    public int tichvohuong(vecto u)
    {
        int tmp = 0;
        if (v.Length == u.v.Length)
          for (int i = 0; i < v.Length; i++)
              tmp = tmp + v[i] * u.v[i];
        else
           Console.WriteLine("Khong thuc hien duoc.Hai vecto khong cung
chieu");
        return tmp;
    }
        }
    public void print(vecto c)
    {
        Console.Write("(");
        for (int i = 0; i < v.Length; i++)
            if (i == v.Length - 1)
                Console.Write("{0}", v[i]);
            else
                Console.Write("{0},",v[i]);
        Console.WriteLine(")");
    }
}

class Tester
{
    static void Main()
    {
        vecto A= new vecto(3);
```

```
        int k;
        A.nhap();
        B.nhap();
        Console.Write("vecto A"); A.print(A);
        Console.WriteLine("Do dai = {0}", A.dodai());
        Console.Write("vecto B"); B.print(B);
        Console.WriteLine("Do dai = {0}", B.dodai());
        if (A.vuonggoc(B) == true)
           Console.Write("Hai vec to vuong goc");
        else
           Console.WriteLine("Hai vecto khong vuong goc !");
        Console.Write("Tong hai vecto la:");
        S = A.cong(B);
        Console.Write(" S = "); S.print(S);
        Console.Write("Hieu hai vecto la:");
        S = A.hieu(B);
        Console.Write("S = "); S.print(S);
        Console.WriteLine("Tich vo huong cua hai vecto: S = {0}",
A.tichvohuong(B));
        Console.Write("Ban hay nhap vao mot so k =");
        k = int.Parse(Console.ReadLine());
        Console.Write("Tich cua vecto A voi k la:");
        S = A.tich(k);
        Console.Write("S = "); S.print(S);
        Console.Write("Tich cua vecto B voi k la:");
        S = B.tich(k);
        Console.Write("S = "); S.print(S);
        Console.ReadKey();
    }
}
```

Kết quả sau khi chạy chương trình:

```
Nhap cac thanh phan cua vecto
V[1]=4
V[2]=7
V[3]=6
Vecto A(4,7,6)
Do dai = 10,0498756211209
Nhap cac thanh phan cua vecto
V[1]=1
V[2]=0
V[3]=9
```

```
Vecto B(1,0,9)
Do dai = 9.05538513813742
Hai vecto khong vuong goc
Tong cua hai vecto la: S=(5,7,15);
Hieu cua hai vecto la: S=(3,7,-3)
Tich vo huong cua hai vecto la: S=58
Ban hay nhap vao so k: 3
Tich cua A voi k la: S=(12,21,18)
Tich cua B voi k la: S=(3,0,27)
```

12. Gói ghém dữ liệu thông qua các thuộc tính

Xây dựng một thuộc tính **ten** cho lớp **tentoi** và hiện thị họ tên ra màn hình

a, Hướng dẫn:

Để xây dựng một thuộc tính ten ban đầu ta đi khai báo thuộc tính họ trong lớp **tentoi** bằng từ khóa public để trong lớp **tester** có thể gọi nó ra thực hiện.

Khai báo một thuộc tính có tên bất kì(ví dụ hoten) bằng từ khóa private để nó chỉ có thể sử dụng trong lớp tentoi với mục đích ghi giá trị cho thuộc tính ten.

Sử dụng hàm set và get để ghi và đọc giá trị cho thuộc tính.

b, Bài giải mẫu:

```
using System;
namespace hoten
{
   public class tentoi
   {
      public string ho;
      private string hoten;
      public string ten
         {
            get
            {
               return hoten;
            }
            set
            {
               hoten = value;
            }
         }
      class tester
      {
         static void Main()
         {
```

```
            tentoi T = new tentoi();
            T.ten ="Hoa";
            T.ho = "Nguyen thi";
            Console.WriteLine(" Ten cua toi la {0} {1}",T.ho,T.ten);
            Console.WriteLine(" Xin chao tat ca cac ban!");
            Console.ReadKey();
        }
    }
  }
}
```

Kết quả sau khi chạy chương trình:

```
Ten cua toi la Nguyen thi Hoa
Xin chao tat ca cac ban!
```

Bài tập 4:

Xây dựng một chương trình thực hiện yêu cầu sau:

- Nhập vào thông tin của các học sinh với số lượng học sinh tùy ý người sử dụng. Thông tin của học sinh bao gồm: Họ tên, điểm toán, điểm lí, điểm hóa.
- Hiển thị danh sách các sinh viên phải thi lại. Nếu không có sinh viên nào thi lại thì đưa ra thông báo "Không có học sinh nào phải thi lại"

a, Hướng dẫn:

- Theo yêu cầu của bài toán thông tin về học sinh rất nhiều bao gồm họ tên, điểm toán, điểm lí, điểm hóa vì vậy để hiển thị danh sách hoc sinh ra bằng cách truy xuất tới nó từ bên ngoài lớp thì ta phải khai báo từ khóa truy xuất đến các thuộc tính có phạm vi hoạt động ngoài lớp vì vậy sẽ làm chương trình sẽ giảm tính rõ ràng, dữ liệu không được bảo vệ vì vậy ta cần phải đóng gói dữ liệu để tránh được những điều trên.

Trong bài tập này ta cần phải xây dựng 2 lớp

- class hocsinh

+ Khai báo các biến thành viên gồm s(tên), dt(điểm toán), dh(điểm hóa), dl(điểm lí) chỉ được sử dụng trong lớp hocsinh nên nó được khai báo bằng từ khóa private. Các dữ liệu này được đóng gói.

+ Phương thức nhập, hiển thị, hàm đóng gói dữ liệu (điểm toán, lí, hóa) được lớp dshs truy xuất đến nên nó được khai báo bằng từ khóa public.

- class dshs

+ Phương thức nhập và hiển thị các thông tin cho học sinh theo danh sách bằng cách gọi phương thức nhập và hiển thị ở lớp hocsinh.

+ Thành phần dữ liệu: (n: số học sinh, ds kiểu mảng) được khai báo bằng từ khóa private vì nó chỉ được sử dụng trong lớp dshs.

b, Bài giải mẫu:

```
using System;
class hocsinh
{
   private string s;
   private double dt, dl, dh;
   public void nhap()
   {
      Console.Write("Nhap ho ten: ");
      s = Console.ReadLine();
      Console.Write("Diem toan: ");
      dt = double.Parse(Console.ReadLine());
      Console.Write("Diem li: ");
      dl = double.Parse(Console.ReadLine());
      Console.Write("Diem hoa: ");
      dh = double.Parse(Console.ReadLine());
   }
   public void hien()
   {
      Console.Write("Thong tin can hien thi\n");
      Console.WriteLine("Ho ten:{0}", s);
      Console.WriteLine("Diem toan:{0}",dt);
      Console.WriteLine("Diem li:{0}", dl);
      Console.WriteLine("Diem hoa:{0}", dh);
   }
   public Double tdl
   {
      set
      { dl = value; }
```

```
    public Double tdh
    {
        set
        { dh = value; }
        get
        { return dh; }
    }
}
class dshs
{
    private int n;
    private hocsinh[] ds;
    public void nhap()
    {
        Console.Write("Nhap so hs=");
        n = Convert.ToInt32(Console.ReadLine());
        ds = new hocsinh[n];
        for (int i = 0; i < n; i++)
```

```
    static void Main()
    {
        dshs a = new dshs();
        a.nhap();
        a.hien();
        Console.ReadLine();
    }
}
```

Kết quả sau khi chạy chương trình:

```
Nhap so hs=2
Nhap thong  tin cho tung hoc sinh
Nhap thong tin cho hoc sinh thu 1
Nhap ho ten: Nguyen thi Trang
Diem toan: 8
Diem hoa: 9
Diem li: 10
Nhap thong tin cho hoc sinh thu 2
Nhap ho ten: Nguyen hai hau
Diem toan:6
Dem hoa: 7
Diem li: 4
Danh sach cac hoc sinh thi lai
Ho ten: Nguyen hai hau
Diem toan: 6
Diem hoa: 7
Diem li: 4
```

C. CÂU HỎI LÝ THUYẾT VÀ TRẮC NGHIỆM

Câu 1: Sự khác nhau giữa khai báo thành phần dữ liệu public va không public là như thế nào? Lấy ví dụ minh họa.

Câu 2: Tham số this được sử dụng như thế nào? Khi nào thì sử dụng this?

Cau 3: Trình bày về cách thức sử dụng các thành phẫn static. Các thành phần static được sử dụng khi nao? Muốn lớp không tạo được thể hiện của đối tượng thì phải làm gì?

Câu4: Câu nào trong những câu sau đúng:

a, Một lớp luôn luôn có lớp cha.

b, Một lớp luôn có một hoặc nhiều lớp con.

c, Thể hiện của một lớp cơ sở cũng là thể hiện của một lớp dẫn xuất.

d, Thể hiện của lớp dẫn xuất cũng là thể hiện của lớp cơ sở.

Câu 5: Câu nào trong những câu sau đúng

Một lớp trừu tượng là:

a, Một lớp mà tất cả thể hiện là không xác định.

b, Một lớp khong có định nghĩa phương thức và các thuộc tính.

c, Một lớp kế thừa từ nhiều lớp khác nhau.

d, Một lớp mà không thể có thể hiện cho nó.

Câu 6: Chỉ ra câu trả lời đúng nhất lỗi của các khai báo sau:

```
class A
        {
        public void nhap(int a, int b)
        {
  //nhap a,b
        }
        public int tong(int c);
          }
        int nhap( ):A
        class main()
              {
}
```

a, Dòng 3 bi lỗi
b, Dòng 7 bị lỗi.
c, Dòng 7 và Dòng 9 bị lỗi

Câu 7: Đoạn chương trình sau đây có lỗi. Hãy tìm ra lỗi và sửa lại cho đúng:

```
using System;
using System.Console;
   class VD1
   {
      public string first;
   }
   class tester
   {
      static void Main(string[] args)
      {
         VD1 vd=new VD1();
         Write("Nhap vao mot chuoi:");
         vd.first=ReadLine();
         Write("Chuoi nhap vao: {0}",vd.first);
      }
   }
```

Hướng dẫn: - Không có khai báo using System.Console;
- Không tồn tại câu lệnh Write, ReadLine độc lập

Câu 8: Tìm và sửa lỗi trong đoạn chương trình sau:

```
using System;
class Tester
{
static void Main()
      {
            Display();
      }
public static void Display()
      {
```

```
                System.Console.WriteLine("Xin chao tat ca moi nguoi");
                return 0;
        }
    }
```

Hướng dẫn: Sai ở phương thức public static void Display()

Câu 9: Hãy sửa lỗi sai trong đoạn code sau cho đúng nhất:

```
using System;
class Class1
{
    public static void Getnumber(ref int x, ref int y)
{
    x = 5;
    y = 7;
}
public static void Main()
    {
        int a = 0; int b = 0;
        Getnumber(a,b);
        System.Console.WriteLine("a={0} \nb={1}", a, b);
        System.Console.ReadKey();
    }
}
```

Hướng dẫn: Sai ở cách truyền tham số cho phương thức Getnumber(), cần phải có đủ hai tham chiếu ref.

D. BÀI TẬP TỰ GIẢI

Bài 1

Xây dựng lớp hình tròn, thuộc tính là bán kính của hình tròn, các phương thức tính chu vi và diện tích của hình tròn đó

Hướng dẫn:

Thành phần dữ liệu của lớp hình tròn chỉ gồm bán kính hình tròn đó.

Phương thức:

- public nhap(): nhập bán kính cho hình tròn
- public double chuvi(): tính chu vi hình tròn
- public double hien(): hiển thị thông tin về hình tròn (bk, cv, dt)

Hai phương thức tính chu vi và diện tích là hai phương thức có trả ra giả trị.

Bài 2

Xây dựng lớp hình trụ với thuộc tính gồm bán kính hình tròn, chiều cao hình trụ và các phương thức thiết lập. Sau đó tính chu vi, diện tích xung quanh, diện tích toàn phần và thể tích của hình trụ đó

Hướng dẫn:

Thành phần dữ liệu gồm có: bán kính hình tròn và chiều cao hình trụ

Phương thức:

- nhập thông tin cho đối tượng
- hiển thị thông tin liên quan tới đối tượng
- tính chu vi
- tính diện tích xung quanh
- tính diện tích toàn phần
- tính thể tích của hình trụ ()

Bài 3

Xây dựng lớp Point cho các điểm trong không gian ba chiều (x,y,z). Lớp chứa một constructtor mặc định, một hàm Nagate biến đổi điểm thành đại lượng có dấu âm, một hàm Norm dùng để tính khoảng cách từ điểm tới gốc tọa độ và một hàm in kết quả lên màn hình.

Hướng dẫn:

Thuộc tính: ba thành phần dữ liệu x, y, z

Phương thức:

- constructor mặc định dùn để khởi tạo giá trị cho đối tượng
- Nagate(): biến đổi tọa độ của điểm
- Norm(): tính khoảng cách từ điểm đó tới gốc tọa độ
- Nhập thông tin
- Hiển thị thông tin

Bài 4

Xây dựng lớp dãy số gồm mô tả các dãy số gồm các phương thức sau:

- phương thức nhập dùng để nhập thông tin từ bàn phím
- phương thức print dùng để hiển thị thông tin lên bàn phím
- phương thức khởi tạo dayso(int n) để khởi tạo một mảng gồm n phần tử

Bài 5

Viết chương trình nhập vào một ngày tháng sau đó in lên màn hình. Sau đó xây dựng chương trình tính số ngày cách nhau giữa hai mốc thời gian được nhập vào từ bàn phím

Hướng dẫn:

Thành phần dữ liệu gồm có int day, int month, int year miêu tả ngày, tháng, năm của đối tượng. Ngaoì ra, cần khai báo hai mảng hằng số là mảng tháng và mảng ngày để theo dõi là tháng nào với tương ứng bao nhiêu ngày trong tháng đó:

unsafe public string(char *) = (“mot”, “hai”, “ba”, “bon”, “nam”, “su”, “bay”, “tam”, “chin”, “muoi”, “muoi mot”, “muoi hai”);
ngaythang = new int [] { 31, 28, 31, 30, 31, 30, 31, 31, 30 ,31, 30, 31}

Các phương thức thiết lập: không tham số, một tham số, hai tham số hay ba tham số. Trong lớp cần xây dựng phương thức kiểm tra tính hợp lệ của dữ liệu nhập vào.

Phương thức nhập và hiện thông tin của đối tượng. Trong phương thức hiện cần chú ý in rõ là năm nhuận hay không: Một năm nhuận nếu năm chia hết cho 4 hoặc chia hết cho 400, nhưng không chia hết cho 100. Trong năm nhuận thì tháng hai co tới 29 ngày.

Bài6

Viết lớp ConvertUtil thực hiện chuyển một số từ hệ cơ số 10 sang hệ cơ số 2, 8, 16 và cơ số a bất kỳ.

Hướng dẫn:

Trong bài này bạn nhất thiết phải sử dụng stack để thực hiện chuyển đổi. Việc chuyển đổi từ hệ cơ số 10 sang các cơ số bất kì thông qua phép chia liên tiếp giống với bài toán ví dụ ở trên. Bạn hãy tham khảo và tự hoàn thiện ý tưởng.

Bài 7

Viết chương trình xây dựng lớp Timer với các hàm thiết lập và các tham số mặc định, thành phần dữ liệu: hour, minute, second. Các phương thức Time(int = 0, int = 0, int = 0) phương thức thiết lập với ba tham số ngầm định dùng để khởi tạo dữ liệu cho đối tượng, Settime() dùng để thiết lập dữ liệu cho đối tượng, các phương thức in giờ dưới dạng chuẩn và in giờ dạng giờ quân đội.

Hướng dẫn:

Thành phần dữ liệu gồm có: hour, minute, second

Phương thức:

- Các phương thức khởi tạo cho đối tượng
- Phương thức kiểm tra tính hợp lệ của dư liệu nhập vào: 0<= hour <= 24; 0<= minute <= 60; 0<= second <= 60;
- Phương thức nhập thông tin
- Phương thức in giờ quân đôi: (ví dụ 16:4:30)
- Phương thức in giờ chuẩn: (ví dụ: 4:4:30 PM)

Bài 8

Xây dựng lớp Employee bao gồm tên và chứng minh nhân dân. Ngoài phương thức thiết lập, một copy constructor, một destructor và còn có phương thức nhập, hiện họ tên và chứng minh thư lên màn hình.

Bài 9

Xây dựng lớp sophuc thực hiện các phép toán trên số phức với các thuộc tính phần thực phần ảo và các phương thức bao gồm

- Phương thức khởi tạo (float pt, float pa) dùng để khởi tạo các thành phần cho số phức với phần thực bằng pt, phần ảo bằng pa

- Phương thức khởi tạo để phần thực bằng 0, phần ảo bằng 0
- Phương thức cộng hai số phức
- Phương thức trừ hai số phức
- Phương thức hiển thị số phức ra mà hình

Bài 10

Xây dựng lớp hình trụ Cyliender dữ liệu gồm bán kính chiều cao của hình trụ và phương thức tính chu vi, thể tích của hình trụ đó. Trong lớp ngoài các phương thức tính toán còn có các phương thức thiết lập và sao chép.

Hướng dẫn:

Các thuộc tính của lớp gồm:

- float bk: bán kính của hình tròn đáy trụ
- float cc: chiều cao của hình trụ

Các phương thức gồm có:

- phương thức thiết lập khởi tạo dữ liệu cho đối tượng thuộc lớp hinhtru này
- phương thức sao chép đối tượng của lớp hình trụ
- phương thức tính chu vi hình trụ
- phương thức tính thể tích của hình trụ

Bài 11:

Xây dựng lớp ma trận vuông với các hàm tính định thức, ma trận nghịch đảo, ma trận chuyển vị, phương thức ma trận, xuất ma trận lên màn hình. Cộng, trừ, nhân hai ma trận.

Hướng dẫn:

Ma trận vuông là một mảng hai chiều có số hàng bằng số cột. Để nhập thông tin cho ma trận ta nhập cho từng phần tử theo hàng. Hiện thông tin cũng hiện theo hàng và cột của ma trận.

Áp dụng các kiến thức về ma trận để tính định thức, ma trận nghịch đảo, ma trận chuyển vị cho ma trận, cộng, trừ, nhân hai ma trận.

(xem thêm phần bài tập giải mẫu ở trên về ma trận)

Bài 12:

Xây dựng lớp sinh viên quản lý họ tên, ngày sinh, điểm ba môn học trở nên, đưa ra số lượng sinh viên được làm đồ án tốt nghiệp

hay thi tốt nghiệp, bao nhiêu sinh viên không được thi tốt nghiệp với các tiêu chuẩn xét:

- làm đồ án nếu điểm trung bình =>7 và không có môn nào dưới 5
- thi tốt nghiệp nếu điểm tb < 7 và không có môn nào dưới 5
- thi lại nếu có ít nhất một môn

(Xem lại ví dụ trong phần bài giả mẫu. Định nghĩa thêm trong lớp một phương thức kiểm tra để đưa ra danh sách sinh viên được làm đồ án, thi tốt nghiệp hay không được thi tốt nghiệp)

E. BÀI TẬP TỔNG HỢP:

Bài 1:

Cho một dãy các hình chữ nhật. Bằng lập trình hướng đối tượng, hãy viết chương trình thực hiện các chức năng sau:

- Nhập vào một dãy hình chữ nhật
- Tìm và in ra hình chữ nhật có chu vi lớn nhất và hình chữ nhật có diện tích lớn nhất

Bài 2:

Cho một dãy các điểm. Hãy tìm ra tam giác có chu vi nhỏ nhất, tam giác có diện tích lớn nhất trong các tam giác được tạo ra từ dãy điểm trên.

Hướng dẫn:

Để giải quyết bài toán trên thì ta cần định nghĩa hai lớp: lớp diem với các phương thức nhập, hiện thông tin, tính độ dài đoạn thẳng (độ dài giữa hai điểm). Lớp tamgiac với mỗi tam giác có 3 điểm chính là đối tượng lớp điểm. Định nghĩa các phương thức tính chu vi tam giac, phương thức tính diện tích tam giác, phương thức hiện thông tin. Thi hành các phương tính chu vi, diện tích của đối tượng tam giác. So sanh chu vi, diện tích để đưa ra tam giác có chu vi nhỏ nhất, diện tích lớn nhất.

Bài 3:

Bằng phương pháp lập trình hướng đối tượng, hãy viết chương trình quản lý nhân sự cho một tổ dân cư (khu phố), chi tiết về cả

ngày thang năm sinh, nơi sinh và chỗ ở hiện tại. Đưa ra thông tin về gia đình có người cao tuổi nhất hay người ít tuổi nhất.

Hướng dẫn:

Hướng giả quyết giống bài 2:
Gia đình là một mảng các phần tử kiểu người
Khu phố lại là một mảng các đối tương kiểu gia đình.
Vì vậy cần định nghĩa ba lớp:

+ Lớp nguoi:

Mô tả rõ họ tên, ngày tháng năm sinh, nơi sinh, nghề nghiệp, chỗ ở của đối tượng. Các phương thức nhập, hiện thông tin.

+ Lớp giadinh

Mô tả gia đình này có bao nhiêu người, ai cao tuổi nhất trong gia đình, ai kém tuổi nhất trong gia đình. Nhập thông tin cho gia đinh chính là nhập thông tin cho từng thành viên. Phương thức hiển thị tóm tắt thông tin về gia đình.

+ Lớp khupho

Khu phố có bao nhiêu gia đình, ai là người cao tuổi nhất trong khu phố, ai là người kém tuổi nhất. (So sánh người cao tuổi, kém tuổi nhất trong giữa các gia đình để đưa ra kết quả cho khu phố). Phương thức hiển thị thông tin.

Có thể xây dựng thuộc tính chỉ đọc cho các thuộc tính trong lớp nguoi và lớp giadinh

CHƯƠNG II

NẠP CHỒNG TOÁN TỬ TRÊN LỚP

Mục tiêu: Sau khi tìm hiểu xong chương này người hcọ có thể nắm được các nội dung sau:

- Đưa ra được các nguyên tắc định nghĩa chồng toán tử.
- So sánh sự giống và khác nhau giữa việc xây dựng một hàm chồng toán tử và hàm thông thường.
- Biết được các yêu cầu cần thiết khi sử dụng toán tử.

● Chồng được các hàm toán tử tiêu biểu tượng trưng cho các phép toán(+, - ,= ,!=, = =, []…)

● Chuyển đổi kiểu ngầm định trong kiểu dữ liệu đối tượng.

A. TÓM TẮT LÝ THUYẾT

1. Toán tử:

Toán tử được kí hiệu bằng một biểu tượng dùng để thực hiện một hành động. Các kiểu dữ liệu cơ bản của C# như kiểu nguyên hỗ trợ rất nhiều các toán tử gán, toán tử toán học, toán tử logic..

2. Chồng toán tử:

Toán tử được định nghĩa chồng bằng cách định nghĩa một hàm toán tử, hàm toán tử là các phương thức tĩnh, giá trị trả về của nó thể hiện kết quả của một phép toán và những tham số là toán hạng. Khi chúng ta tạo một toán tử cho một lớp là chúng ta nạp chồng toán tử(overloaded) cũng giống như là chúng ta nạp chồng bất cứ phương thức thành viên nào.

Việc chồng toán tử thực chất cũng giống như ta xây dựng một hàm thông thường và kết quả thu được như nhau nhưng chồng toán tử giúp người lập trình có thể sử dụng các kí hiệu toán học thường gặp(+ , +, *, /) , dễ nhớ và gần gũi hơn

3. Cú pháp nạp chồng toán tử:

Hàm toán tử bao gồm từ khóa operator theo sau là kí hiệu của toán tử được định nghĩa chồng. Trong đó từ khoá operator là một bổ xung phương thức.

C# cho phép nạp chồng toán tử như sau:

```
Type  Operator  operator_symbol(parameter_list);
```

Trong đó :

- Type là kiểu giá trị trả về của hàm.
- parameter_list là danh sách các đối số nếu có

- Operator_symbol là các toán tử. Trong C# có các hàm toán tử có thể nạp chồng và các phương thức thay thế như sau:

Toán tử	Tên phương thức thay thế	Toán tử	Tên phương thức thay thế
+	Add	>	Compare
-	Subtract	<	Compare
*	Multiply	!=	Compare
/	Divide	>=	Compare
%	Mod	<=	Compare
^	Xor	*=	Multiply
&	BitwiseAnd	- =	Subtract
|	Bitwiseor	^=	Xor
&&	Add	<<=	leftshift
||	Or	%=	Mod
=	Assign	+=	Add
<<	leftshift	&=	BitwiseAnd
>>	Rightshift	|=	Bitwiseor
= =	Equals	/=	Divide
--	Decrement	-	Negate
++	Increment		

Ví dụ 1: Sử dụng nạp toán tử (+) để cộng hai phân số

```
Public static phanso operator +(phanso rhs ,phanso lhs)
        {
  //các câu lệnh
              }
```

Giải thích: rhs là phân số bên phải, lhs là phân số bên trái.

Ví dụ 2: Sử dụng nạp chồng toán tử(-) một ngôi để đổi dấu cho một số như sau:

```
Public static So operator –(So s)
        {
}
```

4. Toán tử một ngôi:

Các toán tử dùng cho một đối tượng hay tham chiếu đến đối tượng trong lớp. Ví dụ các toán tử một ngôi như: + (+a) , - (-a) , ++ (++a) …

5. Toán tử hai ngôi:

Các toán tử như toán tử (= =) so sánh giữa hai đối tượng. toán tử (!=) so sánh không bằng giữa hai đối tượng, "<" so sánh nhỏ hơn, ">" so sánh lớn hơn… Đây là các toán tử phải có các cặp toán hạng hay ta còn gọi là toán tử hai ngôi.Trong C# còn có toán tử ba ngôi.

Chú ý: Trong ngôn ngữ C# không tạo được toán tử **nonstatic**, do vậy toán tử nhị phân phải lấy hai toán hạng.

6. Hỗ trợ ngôn ngữ .NET khác:

C# cung cấp khả năng cho phép nạp chồng toán tử cho các lớp mà chúng ta xây dựng. Trong khi đó các ngôn ngữ .NET khác không cho phép điều này.

+ Để đảm bảo lớp hỗ trợ các phương thức thay thế cho phép các ngôn ngữ khác có thể gọi . Để làm được điều này thì khi nạp chồng toán tử nào đó thì phải xây dựng cho nó một phương thức có chức năng tương tự như toán tử đã chồng.

Ví dụ:

Khi chúng ta nạp chồng toán tử (+) thì phải cung cấp một phương thức Add( ) cũng làm chức năng cộng hai đối tượng.

Khi sử dụng chồng toán tử (= =) thì nên cung cấp thêm phương thức Equals() bởi đối tượng và hướng chức năng này đến toán tử (= =) cho phép lớp của ta đa hình và cung cấp khả năng hữu ích cho ngôn ngữ .Net khác.

+ Toán tử so sánh : Có 6 toán tử so sánh ứng với 3 cặp. Giữa các cặp này luôn luôn có kết quả đối nghịch nhau: Nếu toán hạng đầu trả về giá trị true thì toán hạng kia trả về giá trị false. C# luôn luôn yêu cầu bạn nạp chồng cả hai toán tử đó. Nếu bạn nạp chồng toán tử "= =" thì phải nạp chồng toán tư"!=" nếu không trình biên dịch sẽ báo lỗi.

Có một hạn chế là toán tử so sánh phải trả về kiểu *bool*. và đó cũng là điểm khác nhau giữa các toán hạng này và toán hạng số học.

Cấu trúc của phương thức có chức năng tương tự như toán tử so sánh đã chồng được xây dựng như sau:

```
public override bool Equals(object obj)
    {
        Console.WriteLine("phuong thuc equals");
         if (!(obj is phanso))
      {
                return false;
       }
         return this = = (phanso)obj;
      }
```

Trong đó toán tử is dùng để kiểm trình kiểu đối tượng lúc chương trình đang thực hiện có thích ứng với toán hạng hay không. Nếu kiểm tra thấy kiểu đối tượng thích ứng với toán hạng thì kết quả trả ra là true và ngược lại là false.

+ Toán tử chuyển đổi : Dùng từ khóa implicit và explicit nó có tác dụng chuyển đổi một số từ kiểu có kích thước nhỏ sang kích thước lớn và (ngược lại) mà không bị mất thông tin.

7. Phạm vi sử dụng của các toán tử:

Phạm trù	**Toán tử**	**Hạn chế**
Nhị phân toán học	+, *, /, -, %	Không
Thập phân toán học	+, -, ++, --	Không
Nhị phân bit	&, |, ^, <<, >>	Không
Thập phân bit	!, ~, true, false	Không
So sánh	= =,!=,>=,<, <=, >	Phải nạp chồng theo từng cặp.

8. Một số trường hợp nên sử dụng các toán tử nạp chồng:

1. Trong toán học, mọi đối tượng toán học như: tọa độ, vector, ma trận, hàm số v…v… Nếu bạn viết chương trình làm những mô hình toán học hay vật lý, bạn nhất định sẽ mô tả những đối tượng này.

2. Những chương trình đồ họa sẽ sử dụng các đối tượng toán học và toạ độ khi tính toán vị trí của trên màn hình.

3. Một lớp mô tả số lượng tiền.

4. Việc sử lý từ hay chương trình phân tích văn bản có lớp để mô tả các câu văn, mệnh đề và bạn phải sử dụng các toán hạng để liên kết các câu lại với nhau.

9. Yêu cầu khi sử dụng toán tử:

+ Định nghĩa những toán tử trong kiểu dữ liệu giá trị, kiểu dữ liệu xây dựng sẵn. + Cung cấp phương thức nạp chồng toán tử chỉ bên trong của lớp nơi mà những phương thức được định nghĩa.

+ Sử dụng tên và các kí hiệu được mô tả trong trong Common Language speci-fication (CLS). Tức là ta không được chồng một toán tử mới

+ Sử dụng chồng toán tử trong trường hợp kết quả trả về rõ ràng.

Ví dụ:

Khi ta sử dụng toán tử "or" hoặc "and" giữa một giá trị thời gian này với một thời gian khác thì kết quả trả về sẽ không rõ ràng vì vậy trong trường hợp này ta không nên sử dụng chồng toán tử.

+ Nạp chồng toán tử có tính chất đối xứng. Nghĩa là:

Khi dùng toán tử (= =) thì phải dùng toán tử (!=)

Khi dùng toán tử (<) thì phải dùng toán tử (>).

Khi dùng toán tử (> =) thì phải dùng toán tử (<=).

+ Phải cung cấp các phương thức thay thế cho toán tử được nạp chồng. Bởi vì hầu hết các ngôn ngữ khác không cho phép chồng toán tử nên khi ta sử dụng thêm phương thức có chức năng tương tự như toán tử giúp cho chương trình có thể phù hợp với nhiều ngôn ngữ khác nhau.

10. Ưu và nhược điểm của chồng toán tử

Ưu điểm :

+ Nạp chồng toán tử là một phương thức ngắn gọn giúp mã nguồn dễ nhìn hơn, dễ quản lí,sáng sủa hơn.

+ Nạp chồng toán tử là đường dẫn cho những đối tượng

Nhược điểm:

+ Ta phải sử dụng nạp chồng toán tử một cách hạn chế và phù hợp với các toán tử của lớp được xây dựng sẵn, không sử dụng toán tử quá mới hay quá riêng rẽ. Nếu sử dụng toán tử một cách lạm dụng thì sẽ làm chương trình nhầm lẫn

B. BÀI TẬP MẪU

I. BÀI TẬP TRẮC NGHIỆM, LÝ THUYẾT

Câu 1: Phương thức thay thế:

Nếu ta không xây dựng phương thức thay thế cho các toán tử được nạp chồng thì chương trình có biên dịch hay không.

***Trả lời*:**

Nếu ta không xây dựng phương thức cho toán tử được nạp chồng thì

- Chương trình sẽ biên dịch nếu trong ngôn ngữ C# và các ngôn ngữ cho phép chồng toán tử.
- Chương trình sẽ không biên dịch nếu nó sử dụng trong ngôn ngữ lập trình khác không cho phép sử dụng chồng toán tử.

Bởi vì: Các phương thức có chức năng tương tự như toán tử đã chồng có tác dụng thay thế hàm toán tử khi sử dụng tất cả các ngôn ngữ không nên trong ngôn ngữ không cho phép sử dụng hàm toán tử thì vẫn sử dụng được với các toán tử thông qua các phương thức.

Câu 2: Hạn chế khi sử dụng toán tử:

Xây dựng một lớp Employee với các thông tin cán bộ: Họ tên, giới tính, tiền lương, quê quán. Vậy để gọi một phương thức gia tăng mức lương của nhân viên ta có thể sử dụng toán tử gia tăng (++) trong lớp Employee hay không.

Trả lời:

Ta không thể sử dụng toán tử gia tăng (++) để gia tăng mức lương của cán bộ. Bởi vì bên trong của lớp **Employee** còn có nhiều thuộc tính khác như họ tên, giới tính…Nên ta không sử dụng toán tử gia tăng duy nhất một thuộc tính lương được .

Câu 3: Chuyển đổi kiểu

Cho các dòng lệnh sau:

```
Int      a = 3;
Uint     b = 5;
Double   c= 9;
Long          x = a + b;
Double  y =c + a;
```

Hãy giải thích khi trình biên dịch gặp một toán hạng.

Trả lời:

- Xét dòng lệnh : x = a + b;

Có nghĩa là cộng hai số có kiểu integer (a và b). Bình thường kết quả trả ra sẽ là kiểu Integer nhưng chương trình sẽ ép cho nó trả ra kết quả là kiểu long và điều này trong ngôn ngữ C# cho phép.

- Xét dòng lệnh : y = c + a;

Ta thấy trong nạp chồng này có kiểu double và kiểu Integer, cộng chúng lại và trả ra kiểu double. Vậy chúng ta phải đổi a(có giá trị kiểu int) sang kiểu double sau đó mới cộng với c(double)

Như vậy việc nạp chồng toán tử ở đây như một phiên bản của toán tử nhận hai số double như hai tham số và trình biên dịch phải chắc chắn rằng nó có thể ép kiểu kết quả về một kiểu thích hợp nếu cần.

Câu 4: Cho đoạn chương trình sau:

```
Int   myInt = 12;
Long      myLong;
mylong = myInt;          (1)  //Ngầm định
myInt = (Int)myLong;(2)  // Tường minh
```

Bạn hãy chọn ra một phương án đúng nhất trong các phương án sau:

a, Cách chuyển đổi (1) đúng.

b, Cách cuyển đổi (2) đúng.

c, Cả hai cách đều đúng.

d,Cả hai cách đều đúng nhưng ta phải xây dựng thêm các hàm chuyển đổi.

e, Chỉ có phương án (1) là sai vì nó gây mất thông tin**.**

Trả lời: Phương án (d) là đúng nhất.

Giải thích :

- Giả sử khi ta chuyển đổi một số nguyên sang một phân số thì có thể chuyển đổi số nguyên đó có mẫu là 1.

Ví dụ: a = 12 được chuyển thành 12/1

- Ngược lại khi chuyển đổi một phân số sang một số nguyên thì sẽ làm mất thông tin (vd: 9/4 = 2). Vì vậy ta phải xây dựng một hàm chuyển đổi có sử dụng từ ngữ ngầm định (implicit) để đảm bảo không bị mất thông tin.

Trường hợp tường minh (explicit) không đảm bảo an toàn dữ liệu sau khi chuyển đổi do đó việc này sẽ được thực hiện một cách công khai.

Câu 5: Toán tử (→)

Cho đoạn chương trình sau:

```
Using System
{
Struct point
    {
        Public  int x,y;
    }
Class main class
    {
        Unsafe static void main
            Point pt =new point()
            Point *pp = &pt;
            pp → x =123;
            pp→ y=456;
        Console.WriteLine("{0} {1}",pt.x, pt.y);
    }
}
```

Output 12345 (1)

123 456 (2)

123 (3)

Hãy chọn một kết qủa đúng nhất

Trả lời:

Đáp án (2) là đúng.

Câu 6: Hãy chỉ ra cách khai báo đúng một hàm chồng toán tử của lớp phân số trong các cách khai báo sau đây:

public static ps operator -(real ps1) (1)
public static ps operator =(real ps1) (2)
public static ps operator = =(ps ps1, ps ps2) (3)
public static ps operator - -(ps ps1,ps ps2) (4)

Trả lời:

Ta biết rằng các toán tử được định nghĩa chồng phải bảo toàn số ngôi của chính toán tử đó theo cách hiểu thông thường.

Cụ thể:

+ Nếu ta chồng toán tử một ngôi "-(real ps1) "tương ứng với việc đảo dấu của ps1

+ Nếu ta chồng toán tử hai ngôi"-(ps ps1, ps ps2) tương ứng với việc ta thực hiên chông toán tử trừ đối với ps1 và ps.

Nhưng không thể định nghĩa toán tử gán như (+=, = =,- =,=…) bằng cách chồng toán tử một ngôi. Điều đó chứng tỏ cách khai báo (2) là sai.

Vậy trong các dòng khai báo ở bài tập này có dòng (1), (2), (3) là đúng

II. CÂU HỎI LÝ THUYẾT VÀ BÀI TẬP MẪU

1. Chồng toán tử (+) đối với hai vector

a, Hướng dẫn:

- Xây dựng một vector có ba đối là x, y, z (kiểu double).
- Để sử dụng trong lớp vector nên ta khai báo bằng thuộc tính public. Vì sử dụng thuộc tính public có phạm vi hoạt động trong toàn bộ chương trình.

Public x,y,z;

- Sử dụng từ khoá this được dùng để tham chiếu đến thể hiện hiện hành của một đối tượng xem là con trỏ dùng để tham chiếu

đến tất cả các phương thức không có thuộc tính tĩnh trong lớp vector. Các phương thức khác trong lớp có thể tham chiếu đến những phương thức khác và các biến thành viên thông qua từ khóa this.

- Sử dụng chồng toán tử (+) để cộng hai vector

Vậy chương trình được tóm tắt như sau:

```
Struct   vector
      {
           public   double   x,y,z;   // biến thành viên truy cập public
        public   vector(double  x,double  y, double z )
            {
             // Dùng từ khóa this để tạo đối tượng cho một  vector
             }
        public   vector(vector  rhs)
            {
               }
        public   override  string  ToString()
             {
       // Dùng hàm ToString để chuyển sang xâu
 }
        public  static vector  operator  + (vector lhs, vector rhs)
            {
             // Dùng chồng toán tử cộng để cộng hai vector
             // lhs là vector bên trái.
             // rhs là vector bên phải.
        }
```

b, Bài giải mẫu:

```
using System;
namespace vt
{
   struct Vector
   {
      public double x, y, z;
      public Vector(double x, double y, double z)
```

```
        {
            this.x = x;
            this.y = y;
            this.z = z;
        }
       public Vector(Vector rhs)
        {
            x = rhs.x;
            y = rhs.y;
            z = rhs.z;
        }
        public override string ToString()
       {
         return "( " + x + " , " + y + " , " + z + " )";
        }
        public static Vector operator * (Vector lhs,Vector  rhs)
        {
         Vector result = new Vector(lhs);
         result.x *= rhs.x;
         result.y *= rhs.y;
         result.z *= rhs.z;
         return result;
         }
        public static Vector operator +(Vector lhs, Vector rhs)
        {
            Vector result = new Vector(lhs);
            result.x += rhs.x;
            result.y += rhs.y;
            result.z += rhs.z;
            return result;
        }
    }
    public class tester
    {
        static void Main()
```

```
        {
            Vector vect1, vect2, vect3, vect4;
            vect1 = new Vector(3.0, 3.0, 1.0);
            vect2 = new Vector(2.0, -4.0, -4.0);
            vect3 = vect1 + vect2;
        vect4 = vect1 + vect2;
          Console.WriteLine("vect1 = " + vect1.ToString());
         Console.WriteLine("vect2 = " + vect2.ToString());
          Console.WriteLine("Tong cua vector1 và vector2 la:");
            Console.WriteLine("vect3 = " + vect3.ToString());
          Console.WriteLine("Tich cua vector1 và vector2 la:");
            Console.WriteLine("vect4 = " + vect3.ToString());
            Console.ReadLine();
        }
    }
}
```

Kết quả sau khi thực hiện chương trình:

```
vect1 = ( 5, 10, 4)
vect2 = ( 3, 6, 9)
Tong cua vector1 và vector2 la:
vect3 = ( 8, 16, 13)
Tich cua vector1 và vector2 la:
Vect4 = ( 15, 16, 36)
```

Nhận xét:

Từ chương trình trên ta cần chú ý đến việc xây dựng một biến rhs có kiểu là một vector bởi vì để cộng 2 hay nhiều vector thì ta phải cộng hai vector trước sau đó lại cộng tiếp với một vector sau (đóng vai trò là rhs), cứ như thế đến khi nào hết.

Từ bài toán cộng hai vector này, tương tự ta có thể xây dựng một chương trình cộng nhiều vector với nhau.

2. Chồng toán tử một ngôi

Xây dựng một lớp có tên SO. Cung cấp giá trị cho số thứ nhất và số thứ hai, sau đó

- Dùng chồng toán tử một ngôi để tăng giá trị của số thứ nhất lên 1 đv.
- Dùng chồng toán tử hai ngôi để cộng hai số sau khi tăng số thứ nhất lên 1 đơn vị.

a, Hướng dẫn:

Xây dựng lớp **SO** bình thường (tạo đối tượng cho lớp, chuyển thành xâu bằng phương thức ToString để hiện ra màn hình ,…).

Để tăng giá trị lên hay giảm giá trị của một số ta dùng toán tử tăng giảm như: "- -"là toán tử giảm, "++"là toán tử tăng.

+ Xây dựng một hàm chồng toán tử (+) để tạo một biến có kiểu **SO** và biến vừa tạo ra cho phép truy nhập tới biến value chứa giá trị của **SO**. Nó được khai báo cụ thể như sau:

```
public static SO operator +(SO  s)
  {   }
```

+ Xây dựng hàm chồng toán tử 1 ngôi với toán tử tăng (++) như sau:

```
public static SO operator ++(SO   s)
        {   }
```

Hàm này có tác dụng cho phép có thể tăng giá trị của biến lên 1 đơn vị, khi ta gọi trong chương trình chính

```
Console.WriteLine("Chuyen so thu nhat la: {0}", ++so1)
      {   }
```

Sau đó hàm chồng toán tử một ngôi sẽ được thực hiện.

+ Xây dựng hàm chồng toán tử 2 ngôi với toán tử (+) để cộng số thứ nhất với số thứ hai:

```
public static SO operator +(SO so1, SO so2 ) {}
```

+ Xây dựng lớp **Test** chứa chương trình chính gọi các hàm đã xây dựng ở lớp **SO** ra thực hiện và đưa kết quả mà đề bài yêu cầu ra màn hình

b, Bài giải mẫu:

```
using System;
```

```
namespace motngoi
{
   struct so
   {
      int value;
      public so(int value)
      {
         this.value = value;
      }
      public override string ToString()
      {
         return (value.ToString());
      }
      public static so operator +(so s)
      {
         return (new so(-s.value));
      }
      public static so operator +(so so1, so so2)
      {
         return (new so(so1.value + so2.value));
      }
      public static so operator ++(so s)
      {
         return (new so(s.value + 1));
      }
   }
      class Test
      {
         public static void Main()
         {
            so so1 = new so(11);
            Console.WriteLine("So thu nhat la: {0}", so1);
            so so2 = new so(125);
            Console.WriteLine("So thu hai la: {0}", so2);
            Console.WriteLine("Chuyen so thu nhat la: {0}",
```

```
++so1);
            Console.WriteLine("tong hai so sau khi chong toan
tu mot ngoi    doi voi so thu nhat");
            Console.WriteLine("tong: {0}", so1 + so2);
            Console.ReadLine();
         }
      }
   }
```

Kết quả sau khi thực hiện chương trình:

```
So thu nhat la: 11
So thu hai la: 125
So thu nhat duoc chuyen thanh la:12
```

```
Tong hai so sau khi chong toan tu mot ngoi doi voi so thu nhat
Tong :137
```

Nhận xét:

- Trong bài tập này cần chú trọng đến cách chồng toán tử một ngôi. Cần nắm được những toán tử một ngôi được đa năng hóa.
- Chú ý đến dòng lệnh sau:

```
public static so operator +(so s)
 {
      return (new so(-s.value));
    }
```

3. Xây dựng một lớp sophuc có hai phần thực (real) và ảo (imaginary). Nạp chồng toán tử +, - và đổi dấu số phức.

a, Hướng dẫn:

+ Để xây dựng một lớp số phức và thực hiện cộng trừ hai số phức thì ta có thể dùng chồng toán tử như bài tập trên.

+ Nhưng để đổi dấu của một số phức (một số, một phân số) thì làm thế nào?. Điều này có thể thực hiện bằng cách sử dụng chồng toán tử (-) để đổi dấu.

Chồng toán tử "-"được định nghĩa như sau:

```
public static sophuc operator -(sophuc a)
  {
        sophuc tmp =new sophuc();
        tmp.real = - a.real;
        tmp.imaginary = a.imaginary;
        return tmp;
  }
```

Để gọi hàm chồng toán tử "-" ra thực hiện bằng cách gọi nó ra thực hiện, viết dòng lệnh trong chương trình chính như sau:

```
Console.WriteLine("{0}",-a);
```

b, Bài giải mẫu:

```
using system;
namespace  sp
{
    public class sophuc
     {
      private int real;
      private int imaginary;
      public sophuc() : this(0, 0)
        {
            //cung cap gia tri ban dau
         }
       public sophuc(int r, int i)
        {
            real = r;
                  imaginary = i;
         }
       public override string ToString()
       {
            return(System.String.Format("{0} + {1}i", real,
imaginary));
        }
       public static sophuc operator+(sophuc a, sophuc b)
       {
```

```
            return new sophuc (a.real + b.real, a.imaginary +
b.imaginary);
          }
            // su dung chong toan tu doi hai so:
        public static sophuc operator-(sophuc a, sophuc  b)
        {
           return new sophuc (a.real - b.real, a.imaginary -
b.imaginary);
         }
        //Su dung toan tu doi voi mot so
        public static sophuc operator -(sophuc a)
       {
                    sophuc tmp =new sophuc();
                    tmp.real = - a.real;
                    tmp.imaginary = a.imaginary;
                    return tmp;
      }
     }
     class Testsophuc
    {
           static void Main()
             {
             sophuc a = new sophuc(6, 64);
             sophuc b = new sophuc(2, 5);
             Console.WriteLine("So phuc thu nhat  a = {0}",
a.ToString());
             Console.WriteLine("So phuc thu hai la b = {0}",
b.ToString());
             sophuc tong= a + b;
             Console.WriteLine("Tong hai so phuc = {0}",
tong.ToString());
             sophuc hieu = a - b;
             Console.WriteLine("Hieu hai so phuc la = {0}",
hieu.ToString());
             //Chuyển dấu cho số phức thứ nhất.
```

```
            Console.WriteLine("so phuc thu nhat doi dau thanh
{0}",-a);
            Console.ReadLine();
        }
    }
}
```

Kết quả sau khi thực hiện chương trình:

```
So phuc thu nhat a=6 + 64i
So phuc thu hai la b=2 + 5i
Tong hai so phuc = 8 + 69i
Hieu hai so phuc  = 4 + 59i
So phuc thu nhat doi dau thanh  -6 + 64i
```

Nhận xét:

+ Trong bài tập này cần chú ý đến việc chồng toán tử (-) để đổi dấu cho số phức còn trong bài tập số 5 thì dùng chồng toán tử (+) để tăng thêm giá trị cho một số.

+ Ta cũng có sử dụng toán tử (-) để giảm giá trị của một số hay(một phân số, một số phức).

Điều quan trọng ta cần phải biết được toán tử một ngôi nào được sử dụng và được sử dụng khi nào là thích hợp để đem lại hiệu quả cao.

4. Xây dựng một lớp phân số với tên là phanso và đa năng hoá toán tử trên lớp. Thực hiện các nhiệm vụ sau:

+ Hiển thị các phân số ban đầu và phân số sau khi đã sử lí trên màn hình.

+ Cộng hai phân số.

+ Cộng một phân số với một số.

+ So sánh hai phân số và hiển thị kết quả trên màn hình.

a, Hướng dẫn:

Chúng ta xây dựng một lớp phân số như sau:

```
public class phanso
{
    // Xây dựng một phân số có tử số và mẫu số
   public phanso(int tuso1,int mauso1)
    //Xây dựng một phân số chỉ chứa một toán hạng.
   public phanso(int tuso2)
    //Chuyển một số nguyên thành một phân số có mẫu
là 1
   public static implicit operator phanso(int theInt)
    //Chuyển một phân số thành một số nguyên.
   public static explicit operator int(phanso thephanso)
    // So sánh hai phân số bằng toán tử (= =) và (! =).
   public static bool operator = =(phanso lhs, phanso
rhs)
      public static bool operator !=(phanso lhs, phanso
rhs)
   //Sử dụng chồng toán tử (+): cộng hai phân số
   public static phanso operator +(phanso lhs,phanso rhs)
 }
```

b,Bài giải mẫu:

```
using system;
namespace lop_phan_so
{
   public class phanso
    {
     //Biến thành viên lưu trữ tử số và mẫu số
      private int tuso1;
      private int mauso1;
      public phanso(int tuso1,int mauso1)
         {
            this.tuso1 = tuso1;
            this.mauso1 = mauso1;
         }
         public phanso(int tuso2)
         {
```

```
            this.tuso1= tuso2;
            mauso1 = 1;
         }
         public static implicit operator phanso(int theInt)
         {
            return new phanso(theInt);
         }
         public static explicit operator int(phanso thephanso)
         {
            return thephanso.tuso1 / thephanso.mauso1;
          }
          public static bool operator = =(phanso lhs, phanso rhs)
         {
            if (lhs.tuso1 = = rhs.tuso1 && lhs.mauso1 = =
rhs.mauso1)
            {
            // Khi hai phân số bằng nhau
               return true;
            }
            // Khi hai phân số không bằng nhau
            return false;
         }
        public static bool operator !=(phanso lhs, phanso rhs)
        {
         Console.WriteLine("su dung toan tu !=");
         return !(lhs = = rhs);
         }
         public override bool Equals(object obj)
        {
             Console.WriteLine("phuong thuc equals");
              if (!(obj is phanso))
            {
                return false;
            }
            return this = = (phanso)obj;
```

```
        }
        public static phanso operator +(phanso lhs,phanso rhs)
        {
            if(lhs.mauso1 = = rhs.mauso1)
            {
                Console.WriteLine("-Sử dụng toán tử + với hai
phân số");
                return new phanso(lhs.tuso1 +
rhs.tuso1,lhs.mauso1);
            }
            Console.WriteLine("Sử dụng toán tử+cho một phân số
với một số");
            int fist = lhs.tuso1 * rhs.mauso1;
            int second = rhs.tuso1 * lhs.mauso1;
            return new phanso(fist+ second, lhs.mauso1 *
rhs.mauso1);
        }
        public override string  ToString()
        {
            string s = tuso1.ToString() + "/" + mauso1.ToString();
            return s;
        }
    public class tester
    {
        static void Main()
        {
            Console.WriteLine("-Phân số thứ nhất là:");
            phanso ps1=new phanso(2,4);
            Console.WriteLine("ps1:{0}",ps1.ToString());
            Console.WriteLine("-Phân số thứ hai là:");
            phanso ps2 = new phanso(1,4);
            Console.WriteLine(" ps2:{0} ",ps2.ToString());
            phanso ps3 = ps1 + ps2;
            Console.WriteLine(" ps1+ps2=ps3 ");
            Console.WriteLine(" ps2:{0} ",ps2.ToString());
```

```
            phanso ps3 = ps1 + ps2;
            Console.WriteLine(" ps1+ps2=ps3 ");
            Console.WriteLine(" ps3:{0} ",ps3.ToString());
            phanso ps4 = ps3 + 5;
            Console.WriteLine(" ps4=ps3+5:{0}",ps4.ToString());
            phanso ps5 = new phanso(2, 4);
            Console.WriteLine(" phanso(2,4)");
            Console.WriteLine("-So sánh ps1 va ps5 ");
            if (ps5 = = ps1)
            {
                Console.WriteLine("
ps5:{0}==ps1:{1}",ps5.ToString(), ps1.ToString());
                Console.WriteLine("->Hai phân số này bằng nhau");
            }
            Console.WriteLine("-So sanh ps2 va ps3 ");
            if (ps2 = = ps3)
             {
                Console.WriteLine("->Hai phân số này bằng nhau
");
                Console.WriteLine("ps2:{0}==ps3:{1}",
ps2.ToString(), ps3.ToString());
              }
                Console.WriteLine("ps2:{0}!=ps3:{1}",
ps2.ToString(), ps3.ToString());
                Console.WriteLine("->Hai phân số này không bằng
nhau");
                Console.ReadLine();
                }
            }
        }
}
```

Kết quả sau khi thực hiện chương trình:

```
-Phan so thu nhat la: ps1 = 2/4
- Phan so  thu hai la: ps2 = 1/4
- Su dung toan tu + voi hai phan so
        ps1 + ps2 = ps3
        ps3 = 3/4
- Su dung toan tu + voi mot phan so va mot so
        ps4 = ps3 + 5
```

Chú ý:

+ Nếu như trong chương trình trên ta không dùng phương thức **Equals** thì chương trình vẫn chạy bình thường mà không báo lỗi. Nhưng nó lại rất cần thiết vì phương thức Equals cung cấp một đối tượng cho phép các ngôn ngữ khác có thể gọi.

+ Ta có thể nhập giá trị của phân số ngoài màn hình mà không cấp giá trị trước trong chương trình.

5. Xây dựng một lớp vector và thực hiện các yêu cầu sau:

- Có thể nhập n vector từ bàn phím.
- Nhập số chiều tùy ý.
- Đưa ra tổng các vector.
- Đưa ra hiệu các vector.

a, Hướng dẫn:

- Trong chương trình này ngoài lớp tester ta đi xây dựng lớp vector và DSVT , nhập n vector với số chiều tùy ý thì ta cần khai báo một biến tĩnh n và sử dụng từ khóa public để khai báo. Biến n được khai báo như sau: public static int n;

n là một biến tĩnh bởi vì nó có thể thay đổi tùy ý.

- Để nhập được số vector tùy ý ta cần phải khai báo số vector dưới dạng mảng trong lớp DSVT như sau: private vector[] ds

Tương tự để nhập được số chiều của các vector một cách tùy ý ta cũng khai báo số chiều của vector trong lớp vector tương tự như trên: private int[]a.

-Trong lớp vector xây dựng các hàm và các phương thức

public vector()
public void nhap()
public void hien()
public vector tongvector(vector T)
public vector truvector(vector Tr)
public vector(vector lhs)
public vector coppy
public static vector operator -(vector lhs, vector rhs)
public static vector operator +(vector lhs,vector rhs)

- Trong lớp DSVT

public void nhap():gọi phương thức nhập của lớp vector ra thực hiện ds[i].nhap()

public vector tong(): Gọi hàm coppy() ra thực hiện và sử dụng vòng lặp for duyệt từ vector đầu tiên đến các vector cuối cùng và trừ chúng lần lượt dựa theo phần tử (lhs và rhs

public vector tru(): Làm tương tự như cộng}

b, Bài giải mẫu:

```
using System;
namespace vectorchong
{
   class vector
   {
      public static int n;
      private int[] a;
      public vector()
        {
         a = new int[n];
        }
      public void nhap()
        {
         int i;
```

```
        Console.WriteLine("Phan tu cua vector:");
        for (i = 0; i < n; i++)
          {
            Console.Write("a[{0}]=",i);
            a[i] = int.Parse(Console.ReadLine());
           }
      }
    public void hien()
      {
        int i;
        for (i = 0; i < n; i++)

        Console.Write("{0}", a[i]);
        Console.WriteLine();
      }
    public vector tongvector(vector T)
      {
        vector tmp = new vector();
        int i = 0;
        for (i = 0; i < n; i++)
            tmp.a[i] = a[i] + T.a[i];
        return tmp;
      }
    public vector truvector(vector Tr)
      {
        vector tmp = new vector();
        int i = 0;
        for (i = 0; i < n; i++)
            tmp.a[i] = a[i] - Tr.a[i];
        return tmp;
      }
    public vector(vector lhs)
      {
        int i;
        for (i = 0; i < n; i++)
```

```
            this.a[i] = lhs.a[i];
        }
        public vector coppy()
        {
          int i;
          vector tmp = new vector();
          for (i = 0; i < n; i++)
             tmp.a[i] = a[i];
          return tmp;
        }
      public static vector operator +(vector lhs,vector rhs)
        {
          vector tmp = new vector();
          int i;
          for (i = 0; i < n; i++)
             tmp.a[i] = lhs.a[i] + rhs.a[i];
          return tmp;

        }
     public static vector operator -(vector lhs, vector rhs)
          {
             vector tmp = new vector();
             int i;
             for (i = 0; i < n; i++)
                tmp.a[i] = lhs.a[i] - rhs.a[i];
             return tmp;
          }
        }
    }
  class DSVT
       {
          private int SVT;
          private vector[] ds;
          public void nhap()
           {
```

```
            int i;
            Console.Write("nhap so chieu cua vector n=");
            vector.n = int.Parse(Console.ReadLine());
            Console.Write("Nhap so vector SVT =");
            SVT = int.Parse(Console.ReadLine());
            ds = new vector[SVT];
            for (i = 0; i < ds.Length; i++)
            {
          ds[i] = new vector();
          Console.WriteLine("Nhap thong tin cho Vector thu {0}", i
+ 1);
          ds[i].nhap();
            }
          }
          public vector tong()
          {
            vector tmp = ds[0].coppy();
            int i;
            for (i = 1; i < SVT; i++)
               tmp = tmp + ds[i];
            return tmp;
          }
          public vector tru()
          {
            vector tmp = ds[0].coppy();
            int i;
            for (i = 1; i < SVT; i++)
            tmp = tmp - ds[i];
            return tmp;
          }
        }
        class tester
        {
          private int x;
          static void Main()
```

```
        {
            DSVT t = new DSVT();
            t.nhap();
            t.tong();
            Console.WriteLine("- Tong cua cac vector da nhap la:");
            t.tong().hien();
            Console.WriteLine("- Hieu cua cac vector da nhap la:");
            t.tru().hien();
            Console.ReadLine();
        }
    }
}
```

Kết quả sau khi thực hiện chương trình:

```
Nhap so chieu cua vector n=3
Nhap so vector SVT = 3
Nhap thong tin cho vector thu 1
Phan tu cua vector:
        a[0] = 2
        a[1] = 4
        a[2] = 2
Nhap thong tin cho vector thu 2
Phan tu cua vector:
        a[0] =5
        a[1] = 6
        a[2] = 2
Nhap thong tin cho vector thu 3
Phan tu cua vector
        a[0] =1
        a[1] =5
        a[2] =3
- Tong cua các vector da nhap la:
        8       15      7
- Hieu cua cac vector da nhap la:
        -4      -7      -3
```

Nhận xét:

- Qua chương trình trên ta cần chú ý làm thế nào để nhập vector có số chiều, số phần tử tùy ý từ bàn phím. Từ đó có thể nhập n ma trận hay một số phức,… từ bàn phím.
- Sử dụng hàm copy để cộng trừ nhân các vector ta nhập một cách đơn giản hơn.
- Sử dụng hàm chồng toán tử cộng, trừ hai vector

C. CÂU HỎI TRẮC NGHIỆM LÝ THUYẾT

Câu 1: Hãy cho biết định nghĩa nào sau đây sai

- Chồng toán tử định nghĩa như là một hàm thành phần.
- Chồng toán tử định nghĩa như là một hàm bạn.
- Chồng toán tử định nghĩa như là một phương thức.

Hãy giải thích vì sao chọn như vậy.

Câu 2: Cho đoạn chương trình sau đây:

```
using System;
class Mainclass
  {
     static void main()
         {
             int  a = 5;
             Console.WriteLine(-a);
             Console.WriteLine(a-1);
             Console.WriteLine(a-.5);
}
}
output:
      -5    (1)
      -6    (2)
      -5,5 (3)
```

Hãy cho những trường hợp sau đây trường hợp nào đúng

- Dòng (1), (2), (3) đều đúng

- Dòng (2), (3) là sai.
- Dòng (2), (3) là đúng.
- Dòng (1) đúng, dòng (2) và (3) sai.

Câu 3*: Khi nào ta sử dụng toán tử chuyển đổi? Thế nào là chuyển đổi tường minh và chuyển đổi ngầm định. Nếu không sử dụng toán tử chuyển đổi trong trường hợp vừa nêu thì làm điều gì sẽ xẩy ra?*

Câu 4*: Trong C++ thì hàm chồng toán tử giống như một hàm bạn. Vậy trong C# thì điều này có đúng không? Nếu đúng thì hãy đưa ra một ví dụ cụ thể. Ngược lại thì hãy giải thích tại sao*

Câu 5*: Việc chồng toán tử trong ngôn ngữ C# có điều gì khác so với chồng toán tử ở các ngôn ngữ khác hay không?Nếu có, thì sự khác biệt đó có ưu điểm gì?*

D. BÀI TẬP TỰ GIẢI

Bài 1: Cho đoạn chương trình sau hãy đưa ra kết quả của chương trình để hiểu rõ về phương thức ToString:

```
using System;
namespace phuongthuctostring
{
   using System;
   namespace Wrox.ProCSharp.OOCSharp
   {
      class MainEntryPoint
     {
       static void Main(string[] args)
        {
           Money cash1 = new Money();
           cash1.Amount = 40M;
           Console.WriteLine("cash1.ToString() returns: " +
cash1.ToString());
           cash1 = new BetterMoney();
           cash1.Amount = 40M;
           Console.WriteLine("cash1.ToString() returns: " +
cash1.ToString());
```

```
            Console.ReadLine();
        }
    }
    class Money
    {
        private decimal amount;
        public decimal Amount
        {
            get
            {
                return amount;
            }
            set
            {
                amount = value;
            }
        }
    }
    class BetterMoney : Money
    {
        public override string ToString()
        {
            return "$" + Amount.ToString();
        }
    }
  }
}
```

Bài 2:

Tạo kiểu dữ liệu **Date** biểu diễn ngày tháng năm. Cài đặt thuật toán để tính một ngày trước hoặc sau một ngày xác định nào đó tính khoảng cách theo ngày giữa hai ngày xác định, tính thứ trong tuần của ngày. Sử dụng toán tử vào ra cho
ngày.

Hướng dẫn:

- Đưa ra ngày xác định nào đó có thể nhập từ bàn phím hoặc cung cấp giá trị cho nó bằng từ khóa new

- Để hiển thị được thời gian của một ngày trước và sau của một ngày xác định nào đó bằng cách lấy ngày xác định trừ đi 1 hoặc cộng thêm 1 vào ngày hiện tại. Để làm được điều này ta dùng hàm chồng toán tăng giảm (- -) và (++) một ngôi.

- Để hiển thị ngày,tháng, năm dưới dạng xâu "ngày / tháng / năm" thì ta phải dùng phương thức ToString().

Bài 3:

Xây dựng lớp biểu diễn đa thức với các toán tử cộng, trừ, nhân, chia và đảo dấu. Định nghĩa toán tử để thực hiện các yêu cầu của bài toán..

Hướng dẫn:

- Xây dựng một đa thức có dạng f(x) = Ax + By + Cz + D và đa thức có dạng f'(x) = A'x + B'y +C'z + D' = 0.
- Cung cấp giá trị cho các hằng số cho hai đa thức
- Sử dụng phương thức ToString để tạo xâu đa thức
- Sử dụng chồng toán tử để thực hiện yêu cầu của bài toán.

operator -	có một đối để đảo dấu đa thức.
operator +	có hai đối để cộng hai đa thức.
operator -	có hai đối để trừ hai đa thức.
operator *	có hai đối để nhân hai đa thức.
operator /	có hai đối để chia hai đa thức.

Bài 4:

Xây dựng một lớp biểu diễn các vector n chiều với các toán tử cộng, trừ, tích có hướng hai vector và tích có hướng một vector với một số thực. Định nghĩa toán tử cho phép truy nhập các thành phần vector.

Bài 5:

Xây dựng một lớp mở rộng của lớp phanso đã xây dựng ở phần bài tập mẫu. Các yêu cầu cần bổ xung thêm:

+ Nhập phân số từ bàn phím

+ Dùng nạp chồng toán tử để nhân, chia hai phân số hoặc nhân, chia một phân số với một số hoặc ngược lại

Hướng dẫn:

- Để nhân, chia hay nhân hai phân số với nhau hoặc nhân chia một phân số với một số ta khai báo chồng toán tử bình thường tương tự như đối với cộng hoặc trừ

Cụ thể với toán tử(*) :

```
public static phanso operator *( phanso  lhs, phanso  rhs )
{
    int fistproduct = lhs.tuso1 * rhs.tuso2;
    int  secondproduct = lhs.mauso1 * rhs.mauso2;
    return  new  phanso(fistproduct, secondproduct);
}
```

Bài tập 6: Xây dựng lớp MaTran cho để thực hiện các phép toán trên ma trận, bao gồm:

- Hàm tạo ma MaTran(int sh, int sc) dùng để khởi tạo một ma trận có số hàng là sh và số cột là sc.
- Phương thức "nhap" để nhập ma trận từ bàn phím
- Phương thức "cong" để cộng hai ma trận
- Phương thức "hieu" để thực hiện phép hiệu hai ma trận
- Phương thức "doiDau" để đổi dấu các phần tử của ma trận
- Phương thức "chuyenVi" để chuyển vị một ma trận
- Phương thức "tich" để nhân hai ma trận
- Phương thức "tuongThich" để kiểm tra xem hay ma trận có tương thích với nhau hay không?

Hướng dẫn:

- Để khai báo một ma trận thì khai báo như sau:

```
struct  MT
{
    int     a[sh][sc];   //Mảng a chứa các phần tử của ma trận.
    int  n;              //Cấp ma trận.
}
```

- Để thực hiện nhập, cộng hiệu, đổi dấu, tích thì xem lại cách giải của các bài tập trong bài tập mẫu .
- Muốn đổi dấu các phần tử của ma trận ta dùng chồng toán tử (-) như bài tập 6 (bài tập mẫu).

Bài 7:

Xây dựng lớp SoPhuc để mở rộng hơn bài tập 6(bài tập mẫu) thực hiện các phép toán về số phức, bao gồm:

- Hàm tạo SoPhuc(float pt, float pa) để khởi tạo phần thực bằng pt và phần ảo bằng pa
- Hàm tạo SoPhuc() để khởi tạo phần thực bằng 0 và phần ảo bằng 0
- Phương thức "chia" để thực hiện phép chia hai số phức
- Phương thức "nghichDao" để tính nghịch đảo của một số phức
- Phương thức "soSanhBang" để so sánh xem hai số phức có bàng nhau hay không?
- Phương thức "lonHon", "nhoHon", … để thực hiện phép toán so sánh lớn hơn, nhỏ hơn…
- Phương thức "nhap" dùng để nhập số phức từ bàn phím
- Phương thức "print" dùng để in số phức ra màn hình.

Hướng dẫn:

+ Xem lại bài tập 6 (bài tập mẫu) để ta có thể xây dựng một lớp số phức .

+ Nhập số phức từ bàn phím như bài tập 9 (bài tập mẫu)

+ Chia hai số phức bằng cách ta chia phần thực và phần ảo riêng như cộng hai số phức sau đó cộng lại thì sẽ thu được kết quả .

+ Nghich đảo một số phức cũng giống nư nghịch đảo một phân số.

+ Ta có thể so sánh số phức bằng toán tử so sánh (= =) và (!=) như bài tập 8 trong phần bài tập mẫu.

CHƯƠNG III

KẾ THỪA VÀ ĐA HÌNH

Mục tiêu: Sau khi tìm hiểu chương này người học có thể nắm được các nội dung sau:

- Cài đặt được sự thừa kế
- Sử dụng các thành phần của lớp cơ sở
- Định nghĩa lại các hàm thành phần
- Truyền thông tin giữa các hàm thiết lập của lớp dẫn xuất và cơ sở
- Sự tương thích giữa các đối tượng của lớp dẫn xuất và lớp cơ sở
- Chỉ ra đặc điểm của phương thức trừu tượng và tính đa hình, cài đặt bằng bài toán cụ thể
- Phân biệt các từ khóa new, virtual, override…
- Lớp trừu tượng, lớp cô lập, lớp báo

A. TÓM TẮT LÝ THUYẾT

1. Kế thừa:

Khái niệm: là tiếp thu tiếp nhận những cái đã được xây dựng.

Nếu lớp B là lớp được kế thừa từ lớp A thì toàn bộ các thành phần của lớp A cũng có mặt trong lớp B mà không cần phải khai báo trong lớp B. Lớp A được gọi là lớp cơ sở hay lớp cha, còn lớp B được gọi là lớp dẫn xuất hay lớp con.

Kế thừa có hai loại: Đơn kế thừa và đa kế thừa.

Ở chương này chúng ta chỉ nghiên cứu về đơn kế thừa còn đa kế thừa sẽ được thể hiện rõ hơn ở chương giao diện.

Cách khai báo lớp dẫn xuất kế thừa một lớp cơ sở : Tên lớp dẫn xuất : Tên lớp cơ sở.

Ví dụ : Lớp A kế thừa lớp B thì khai báo như sau : Class A **:**B

- Đơn kế thừa là một lớp dẫn xuất chỉ được kế thừa từ một lớp cơ sở

Ví dụ khai báo đơn kế thừa

```
class Class1
    {
    public Class1()
```

```
        {
            Code of contructor
        }
    }
class class2:Class1   // Khai báo lớp Class2 kế thừa lớp Class1
    {
        public class2()
        {
            Code of contructor
        }
}
```

- Đa kế thừa là mỗi Class được phép kế thừa nhiều lớp cơ sở.

Ví dụ khai báo đa kế thừa

```
class Class1
        {
        public Class1()
        {
            Code of contructor
        }
    }
interface IClass1
{
    // member of IClass1
}
class class2:Class1, IClass1
    {
        public class2()
        {
            Code of constructor
        }
}
```

2. Từ khoá Base:

Sử dụng để truy nhập các thành viên của lớp cơ sở từ lớp dẫn xuất.

3. Từ khoá New:

Khi khai báo phương thức trong lớp kế thừa mà có tên trùng với tên của phương thức(không có từ khoá *abstract* hay *virtual*) trong lớp cơ sở thì phải sử dụng từ khoá *new* để che giấu phương thức.

Ví dụ:

```
class diem
{
    public void nhap()
}
class tamgiac:diem
{
    public new void nhap()
}
```

4. Tính đa hình :

Bao gồm những đặc điểm sau

+ Nó được hiểu như là khả năng sử dụng nhiều hình thức của một thực thể mà không cần quan tâm đến từng chi tiết cụ thể.

+ Phương thức là đa hình có từ khoá Virtual ở trước và phương thức này phải được ghi đè ở lớp dẫn xuất.

+ Cho phép cài đặt phương thức của lớp dẫn xuất trong khi thi hành

Ví dụ:Khai báo phương thức đa hình

```
public virtual float dientich()
```

5. Ghi đè:

Ghi đè phương thức khi có nhu cầu cho phép người lập trình thay đổi hay mở rộng các thành viên (phương thức) cùng tên của lớp cơ sở trong lớp dẫn xuất thì bạn sử dụng từ khoá *override*.

Ví dụ:

```
class hinh
 {
    public virtual float dientich()
  }
class hinhcn:hinh
```

```
{
   public override float dientich() // phương thức ghi đè
}
```

6. Nghiêm cấm kế thừa:

Một lớp được cho là nghiêm cấm các lớp khác kế thừa khi nó được khai báo với từ khoá là *sealed* ngay trước từ khoá *Class*. Do vậy bất kì lớp nào khai báo kế thừa lớp đó đều phát sinh lỗi khi biên dịch.

Ví dụ : Khai báo lớp nghiêm cấm kế thừa

```
   sealed class sinhvien
```

7. Cách truy xuất protected:

Bạn cần một lớp dẫn xuất để thâm nhập vào các thành phần của lớp cơ sở. Mặc dù bạn không muốn khai báo các thành phần của lớp cơ sở đó là chung (public). Cách tốt nhất là dùng cách truy xuất có bảo vệ protected trong định nghĩa của lớp cơ sở. Chúng đều được xử lí chung như bên trong phạm vi lớp dẫn xuất nhưng nó vẫn được coi là riêng ở mọi nơi khác. Cách truy nhập protected chỉ mở rộng cho lớp con.

8. Lớp trừu tượng:

Là lớp có ít nhất một phương thức trừu tượng. Khi xây dựng lớp trừu tượng thì mọi thành viên được định nghĩa trong lớp trừu tượng đều sử dụng từ khoá abstract.

Phương thức trừu tượng không có sự thực thi. Phương thức này chỉ đơn giản tạo ra một tên phương thức và kí hiệu phương thức. Nó không định nghĩa phần thân, thay vào đó chúng được cài đặt trong phương thức ghi đè của lớp dẫn xuất.

Để khai báo lớp trừu tượng bạn sử dụng từ khoá **abstract** trước từ khoá **class** như cú pháp sau:

```
abstract class baseclass
{
   // Code of members
}
```

9. Sự khác nhau giữa phương thức đa hình với phương thức trừu tượng:

Phương thức đa hình phần thân được định nghĩa tổng quát, ở lớp dẫn xuất có thể thay đổi phương thức đó. Phương thức trừu tượng phần thân không được định nghĩa và nó được cài đặt trong phương thức của lớp dẫn xuất.

10. Gốc của tất cả các lớp (Lớp Object*):*

Trong C #, các lớp kế thừa tạo thành cây phân cấp và lớp cao nhất (hay lớp cơ bản nhất) chính là lớp Object. Bất cứ kiểu dữ liệu nào thì cũng được dẫn xuất từ lớp System.Object. Lớp Object cung cấp một số các phương thức dùng cho lớp dẫn xuất có thể thực hiện việc phủ quyết. Sau đây là bảng tóm tắt các phương thức của lớp Object.

Phương	**Chức năng**
Equal( )	So sánh bằng nhau giữa hai đối tượng
GetHashCode( )	Cho phép những đối tượng cung cấp riêng những hàm băm cho sử dụng tập hợp.
GetType( )	Cung cấp kiểu của đối tượng
ToString( )	Cung cấp chuỗi thể hiện của đối tượng
Finalize( )	Don dẹp các tài nguyên
Memberwise	Tạo một bản sao từ đối tượng.

Ví dụ minh hoạ việc kế thừa phương thức ToString() của lớp Objec t

```
using System;
namespace Object
{
   public class Object
   {
      private int value;
      public Object(int val)
      {
         value=val ;
      }
```

```
        public virtual string ToString()
        {
            return value.ToString();
        }
    }
class tester
    {
        static void Main(string[] args)
        {
            int i = 5;
            Console.WriteLine("Gia tri cua i:{0}", i.ToString());
            Object A = new Object(7);
            Console.WriteLine("Gia tri cua A:{0}", A.ToString());
            Console.ReadLine();
        }
    }
}
```

Kết quả sau khi thực hiện chương trình:

```
Giá trị của i: 5
Giá trị của A: 7
```

11. Kiểu Boxing và Unxing

Boxing là tiến trình chuyển đổi một kiểu giá trị thành kiểu Object. Nó là ngầm định khi ta cung cấp một giá trị tham chiếu đến giá trị này và giá trị được chuyển đổi ngầm định.

Ví dụ: Minh hoạ Boxing

```
using System;
class Boxing
{
  public static void Main()
  {
```

```
        int i = 123;
        Console.WriteLine("The object value:{0}", i);
        Console.ReadLine();
    }
}
```

Unxing : trả kết quả của một đối tượng về kiểu giá trị, ta thực hiện mở tường minh nó. Ta thiết lập theo 2 bước sau:

1. Chắc chắn rằng đối tượng là thể hiện của một giá trị đã được box
2. Sao chép giá trị từ thể hiện này thành giá trị của biến.

Vd: Minh hoạ boxing và Unxing

```
using System;
public class Unxing
{
        public static void Main()
        {
      int i = 123;
      object o=i;
      int j = (int)o;
      Console.WriteLine("j :{0}", j);
      Console.ReadLine();
        }
}
```

12. Lớp lồng nhau:

Lớp được khai báo trong thân của một lớp được gọi là lớp nội hay lớp lồng, lớp kia là lớp ngoài. Lớp lồng có khả năng truy cập tới các thành viên của lớp ngoài. Một phương thức của lớp lồng có thể truy xuất đến biến thành viên **private** của lớp ngoài. Hơn nữa lớp nội ẩn trong lớp ngoài so với các lớp khác, nó có thể là thành viên kiểu **private** của lớp ngoài.Lớp lồng được khai báo với từ khoá **internal** trước từ khoá class.

Ví dụ

```
public class A
{
```

```
        internal class B
    }
```

B. BÀI TẬP MẪU

I. Kế thừa

1. Xây dựng lớp dẫn xuất thừa kế từ lớp cơ sở.

Bài 1:

Xây dựng lớp có tên là lương để tính lương cho cán bộ với các thông tin sau: Họ tên, lương cơ bản, hệ số lương,với các phương thức :Khởi tạo không tham số dùng để khởi tạo lương cơ bản là 450, hệ số lương là 2,3.Phương thức thiết lập 2 tham số dùng để khởi tạo lương cơ bản,hệ số lương.

Sau đó kế thừa lớp có tên là lương được xây dựng ở trên dùng để tính lương mới cho các cán bộ với việc bổ sung thêm hệ số phụ cấp, lương được tính lại như sau: Lương=lương cơ bản*hệ số lương*hệ số phụ cấp.

a, Hướng dẫn:

Bài toán này được xây dựng gồm có 2 đối tượng:**class luong** và **class luongmoi.** Nhìn vào bài toán trên ta thấy cả hai lớp có những thuộc tính và phương thức giống nhau như: lương cơ bản, hệ số lương, các phương thức khởi tạo, tính lương.Theo như lập trình cấu trúc thì bạn phải đi xây dựng tất cả các thuộc tính và phương thức cho cả hai lớp trên. Ngược lại với lập trình hướng đối tượng thì việc giải quyết lại đơn giản hơn. Bạn chỉ cần xây dựng một lớp class luong, còn lớp class luongmoi sẽ kế thừa lại các thuộc tính và phương thức giống lớp class luong chứ không phải xây dựng lại. Nó chỉ xây dựng các thuộc tính và phương thức riêng của nó mà lớp Class luong không có.

```
class luong  //Lớp cơ sở
{
    Thuộc tính:
            private string hoten;
            private double hesoluong;
        private static int luongcoban;
```

***Lưu ý**: Lương cơ bản là thuộc tính không thay đổi.Vậy ta phải khai báo*

với từ khoá Static

Phương thức:

public void nhap ()

public luong() *// Phương thức thiết lập không tham số*

public luong(double lcb, float hsl) *// Phương thức thiết lập 2 tham số*

public double tinhluong()

public void hien()

}

class luongmoi *//Lớp dẫn xuất*

{

Thuộc tính:

private double hesophucap;

Phương thức:

public new void nhap ()

public luongmoi() // *Phương thức thiết lập không tham số* public luongmoi(int lcb,double hsl, double hspc)

// *Phương thức thiết lập các tham số*

public new double tinhluong()

public new void hien()

}

b, Bài giải mẫu:

```
using System;
   class luong
   {
      private string hoten;
      private double hesoluong;
      private static int luongcoban;
      public void nhap()
      {
         Console.Write("Nhap ho ten:") ;
         hoten=Console.ReadLine();
```

```
            Console.Write("He so luong:");
            hesoluong=double.Parse(Console.ReadLine());
         }
        public luong ()
        {
            luongcoban=450;
            hesoluong=2.34;
        }
        public luong (double lcb,float hsl)
        {
            luongcoban=lcb ;
            hesoluong =hsl ;
        }
      public void hien()
        {
            Console.WriteLine("Ho ten :{0}",hoten);
            Console.WriteLine("He so luong:{0}",hesoluong );
        }
        public double tinhluong()
        {
             return luongcoban*hesoluong ;
        }
    }
   class luongmoi:luong
    {
        private double hesophucap;
        public  luongmoi():base()      // Gọi phương thức thiết lập
không tham số
        {                                    của lớp cơ sở
            hesophucap=0.4;
        }
         public luongmoi(int lcb,float hsl,double
hspc):base(lcb,hsl)
        {                                  // Gọi phương thức thiết lập 2
tham số
```

```
            hesophucap = hspc;              của lớp cơ sở
        }
        public new void nhap()
        {
            base.nhap();
            Console.Write("He so phu cap:");
            hesophucap =double.Parse(Console.ReadLine());
         }
        public new void hien()
        {
            base.hien();
            Console.WriteLine("He so phu cap= {0}", hesophucap);
        }
        public new double tinhluong()
        {
            return base.tinhluong()*hesophucap ;
        }
    }
  class tester
    {
        static void Main()
        {
            luongmoi A = new luongmoi();
            A.nhap();
            A.hien();
            Console.WriteLine("Luong moi:{0}", A.tinhluong());
            Console.ReadLine();
        }
    }
```

Kết quả sau khi thực hiện chương trình:

```
Họ tên : Nguyễn Văn A
Hệ số lương : 3.6
Hệ số phụ cấp : 4.3
Lương mới  : 6 966
```

Kết luận: Mục đích xây dựng bài toán trên là giúp bạn biết cách xây dựng lớp dẫn xuất kế thừa từ một lớp cơ sở, cách gọi hàm thiết lập, phương thức của lớp cơ sở thông qua từ khoá **Base.** Và trả lời một số câu hỏi sau: Khi nào thì xây dựng một lớp kế thừa từ lớp khác ? Và mục đích của việc kế thừa ?

- Một lớp được kế thừa từ lớp khác khi một lớp đó có các thuộc tính và phương thức giống với một lớp nào đó đã được xây dựng.
- Mục đích của việc kế thừa giúp chúng ta không phải đi xây dựng những gì đã có.Do vậy bạn đã tiết kiệm thời gian viết mã cốt và sử dụng tài nguyên có hiệu quả. Hơn nữa điều đó còn giúp cho chương trình sáng sủa, và tránh được các lỗi khi lập trình.

Bài 2

Xây dựng phương trình bậc hai sau đó kế thừa giải phương trình trùng phương

a, Hướng dẫn:

Ta thấy phương trình trùng phương là trường hợp đặc biệt của phương trình bậc hai. Bài toán giải phương trình bậc hai là một bài toán phổ biến và đều được lập trình rất nhiều trong các ngôn ngữ. Vậy trên cơ sở của phương trình bậc hai chúng ta có thể kế thừa lại để lập trình giải phương trình trùng phương.

Bài toán được chia làm hai đối tượng Ptbachai và pttp

Các phương thức và thuộc tính của phương trình bậc hai

```
class ptbachai
{
    protected  double a,b,c;
    public void nhap()
    public double delta()
    public void giai(out double x1,out double x2,out bool kt)
    public void hien()
}
class pttp:ptbachai
```

```
    {
        public new void hien()
    }
```

b, Bài giải mẫu:

```
using System;
   class ptb2
   {
      protected double  a,b,c;
      public void nhap()
      {
         Console.Write (" Nhap vao he so a =" );
         a =double.Parse(Console.ReadLine());
         Console.Write(" Nhap vao he so b =" );
         b = double.Parse(Console.ReadLine());
         Console.Write(" Nhap vao he so c =" );
         c =  double.Parse(Console.ReadLine());
      }
      public double  delta()
      {
         return (b*b-4*a*c);
      }
      public void giai(out double x1,out double x2,out bool kt)
      {
         kt = true;
if (delta() < 0) { kt = false; x1 = x2 = 0; }
         else
         {
            if (delta() == 0)
               x1 = x2 = -b / (2 * a);
            else
            {
               x1 = (-b + Math.Sqrt(delta())) / (2 * a);
               x2 = (-b - Math.Sqrt(delta())) / (2 * a);
            }
         }
```

```
        }
        public void hien()
        {
            double x1, x2;
            bool kt;
            giai(out x1,out x2,out kt);
            if (kt == false) Console.WriteLine("Phuong trinh vo
nghiem");
            else
                if (x1 == x2) Console.WriteLine("Phuong trinh co
nghiem kepx1=x2={0}", x1);
                else
                {
                    Console.WriteLine("Phuong trinh co 2 nghiem phan
biet");
                    Console.WriteLine("x1={0}", x1);
                    Console.WriteLine("x2={0}", x2);
                }
          }
    }
class pttp:ptb2
    {
        public new void hien()
        {
            double x1, x2;
            bool kt;
            base.giai(out x1,out x2,out kt);
            if (kt==false )
                Console.Write("Phuong trinh vo nghiem");
            else
            {
                if ((x1 == x2 & x1 < 0))
                    Console.Write("Phuong trinh vo nghiem");
               if((x1 == x2 & x1==0)|| (x1 == 0 & x2 < 0))
                    Console.Write("Phuong trinh co 1 nghiem
```

```
y1={0}",x1);
            if ((x1 == x2 & x1 > 0))
            {
               Console.WriteLine("Phuong trinh co 2 nghiem
y1,y2");
               Console.WriteLine("y1={0}", Math.Sqrt(x1));
               Console.WriteLine("y2={0}", -Math.Sqrt(x1));
            }
            if (x1 == 0 & x2 > 0)
            {
               Console.WriteLine("Phuong trinh co 3 nghiem ");
               Console.WriteLine("y1={0}", x1);
               Console.WriteLine("y2={0}", -Math.Sqrt(x2));
               Console.WriteLine("y3={0}", Math.Sqrt(x2));
            }
            if (x1 < 0 & x2 == 0)
               Console.WriteLine("Phuong trinh co 1 nghiem y1=
{0}",x2);
            if (x1 > 0 & x2 == 0)
            {
               Console.WriteLine("Phuong trinh co 3 nghiem");
               Console.WriteLine("y1={0}", x2);
               Console.WriteLine("y2={0}", -Math.Sqrt(x1));
               Console.WriteLine("y3={0}", Math.Sqrt(x1));
            }
            if (x1 < 0 & x2 > 0)
               Console.WriteLine("Phuong trinh co 2 nghiem");
            if (x1 > 0 & x2 < 0)
            {
               Console.WriteLine("Phuong trinh co 2 nghiem");
               Console.WriteLine("y1={0}", -Math.Sqrt(x1));
               Console.WriteLine("y2={0}", Math.Sqrt(x1));
            }
    if (x1 > 0 & x2 > 0 & x1!= x2 )
            {
```

```
                Console.WriteLine("Phuong trinh co 4 nghiem phan
biet");
                Console.WriteLine("y1={0}", -Math.Sqrt(x1));
                Console.WriteLine("y2={0}", Math.Sqrt(x1));
                Console.WriteLine("y3={0}", -Math.Sqrt(x2));
                Console.WriteLine("y4={0}", Math.Sqrt(x2));
             }
            }
      }
   }
   class tester
   {
      static void Main()
      {
         pttp D = new pttp();
         D.nhap();
         D.hien();
         Console.ReadLine();
      }
   }
```

Kết quả sau khi chạy chương trình:

```
Nhập vào hệ số a =3
Nhập vào hệ số b =2
Nhập vào hệ số c =1
Phương trình vô nghiệm
```

2. Mức độ truy cập Protected

Bài 1

Xây dựng lớp hình vuông, tính diện tích cho lớp đó. Dùng kĩ thuật thừa kế để tính thể tích cho một hình lập phương.

a, Hướng dẫn:

Bài toán gồm có 2 đối tượng: Hình vuông và hình lập phương.

Ta thấy hình lập phương là trường hợp đặc biệt của hình vuông, do vậy nó không những kế thừa các phương thức và gọi hàm thiết lập của lớp cơ sở mà nó còn thừa kế cả thuộc tính của lớp cơ sở. Thông thường thuộc tính được khai báo là Private, tức là thuộc tính riêng tư của lớp và không cho phép các lớp khác truy cập. Như vậy lớp hình lập phương muốn kế thừa được thuộc tính của lớp hình vuông thì ta phải khai báo thuộc tính trong lớp hình vuông là thuộc tính Protected.

Các thuộc tính và phương thức của hình vuông

```
class hinhvuong     // Lớp cơ sở
{
      protected float a;
      public hinhvuong()
      public hinhvuong(float x)
      public void nhap()
      public void hien()
      public float dientich()
}
```

Các thuộc tính và phương thức của hình lập phương.

```
class hinhlp     // Lớp dẫn xuất
{
    public hinhlp()
    public hinhlp(float x)
    public void hien()
    public float thetich()
}
```

a, Bài giải mẫu:

```
using System;
  class hinhvuong
  {
    protected  float a;
    public hinhvuong()
    {
```

```
        a = 2;
      }
      public hinhvuong(float x)
      {
        a = x;
      }
       public void nhap()
      {
        Console.Write("Nhap canh cua hinh vuong:");
        a=float.Parse(Console.ReadLine());
      }
      public float dientich()
      {
        return a*a;
      }
      public void hien()
      {
        Console.WriteLine("Thong tin can hien thi");
        Console.WriteLine("Canh hinh vuong = {0}",a);
        Console.WriteLine ("Dien tich cua hinh vuong=
{0}",dientich());
      }
   }
   class hinhlp :hinhvuong
   {
      public hinhlp(): base()
      { }
      public hinhlp(float x): base(x)
      { }
      public float thetich()
      {
        return base.dientich() * a;
      }
      public new void hien()
      {
```

```
        base.hien();
        Console.WriteLine("The tich cua hinhlp={0}", thetich());
    }
}
class tester
{
    static void Main()
    {
        hinhlp  B = new hinhlp ();
        B.nhap();
        B.hien();
        Console.ReadKey();
    }
}
```

Kết quả sau khi chạy chương trình:

```
Nhập cạnh của hình vuông: 5
Thông tin hiển thị
Cạnh hình vuông = 5
Diện tích hình vuông = 25
Thể tích hình lập phương = 125
```

***Lưu ý*:** Nhiều khi bạn đọc không chú ý đến kiểu dẫn xuất protected. Mặc dù lớp hình lập phương có thuộc tính giống với lớp hình vuông nhưng các bạn lại cho rằng nó là thuộc tính Private không truy xuất được. Do đó lại xây dựng thuộc tính cho lớp hình lập phương điều này là không cần thiết.

Bài tập 2:

Xây dựng lớp Stack với các thuộc tính và phương thức cần thiết. Sau đó kế thừa để chuyển một số nguyên dương bất kì sang hệ đếm cơ số 2,8

a, Hướng dẫn:

Bài toán gồm có 2 đối tượng: Stack và Đổi số

Bài toán này được xây dựng nhằm củng cố thêm cho bạn đọc về cách xây dựng lớp kế thừa, và thuận tiện khi sử dụng tính kế thừa.

Ta sử dụng cơ chế pop và push của Stack để chuyển đổi. Mỗi khi ta lấy số cần chuyển chia lần lượt cho hệ cần đổi thì ta lần lượt push số dư của từng phép chia vào Stack. Sau khi thực hiện chia xong thì ta pop các số dư trong Stack ra. Việc push và pop thực hiện theo cơ chế vào trước ra sau.Như vậy ta sẽ được kết quả của việc chuyển đổi.

Các thuộc tính và phương thức của lớp Stack

```
class Stack          //Lớp cơ sở
{
    private  int top;
 int  [] s;
    public  stack()
    public Stack(int n)
    public bool full()          // Kiểm tra Stack đầy
    public bool empty()     //  Kiểm tra Stack trống
    public  void push(int x)
    public  void pop()
}
  class Doiso:Stack      //Lớp dẫn xuất
{
   private int a,n;         //a là hệ cần đổi, n là số cần đổi
   public void nhap()
   public void doi()
   public void hien()
}
```

b, Bài giải mẫu:

```
using System;
namespace stack
{
   class Stack
   {
      private int top;
      private int []s;
      public bool  empty()
      {
```

```
        return (top == -1);
      }
      public bool  full()
      {
        return (top >= s.Length);
      }
      public Stack ()
      {
        s = new int[20];
        top=-1;
      }
     public void push(int x)
      {
        if(!full())
        {
          top=top+1;
          s[top]=x;
        }
        else
          Console.Write("Stack tran");
      }
      public int pop()
      {
        if(empty())
        {
          Console.Write("Stack can");
          return 0;
        }
        else
          return s[top--];
      }
   }
class doiso : Stack
   {
      private int a, n;
```

```
    public void nhap()
    {
        Console.Write("Nhap vao so can doi:");
        n = Convert.ToInt32 (Console.ReadLine());
        Console.Write("Nhap vao he can doi:");
        a = Convert.ToInt32(Console.ReadLine());
    }
    public void doi()
    {
        int du;
        while (n != 0)
        {
            du = n % a;
            push(dư);
            n = n / a;
        }
    }
    public void hien()
    {
      while (!empty())
        {
            Console.Write("{0}", pop());
        }
    }
}
class tester
{
    static void Main()
    {
        doiso d = new doiso();
        d.nhap();
        d.doi();
        d.hien();
        Console.ReadKey();
    }
```

```
    }
}
```

Kết quả sau khi chạy chương trình:

```
Nhập vào số cần đổi: 8
Nhập vào hệ cần đổi: 2
Kết quả chuyển đổi: 1000
```

3. Cách truy nhập của từ khóa internal và public

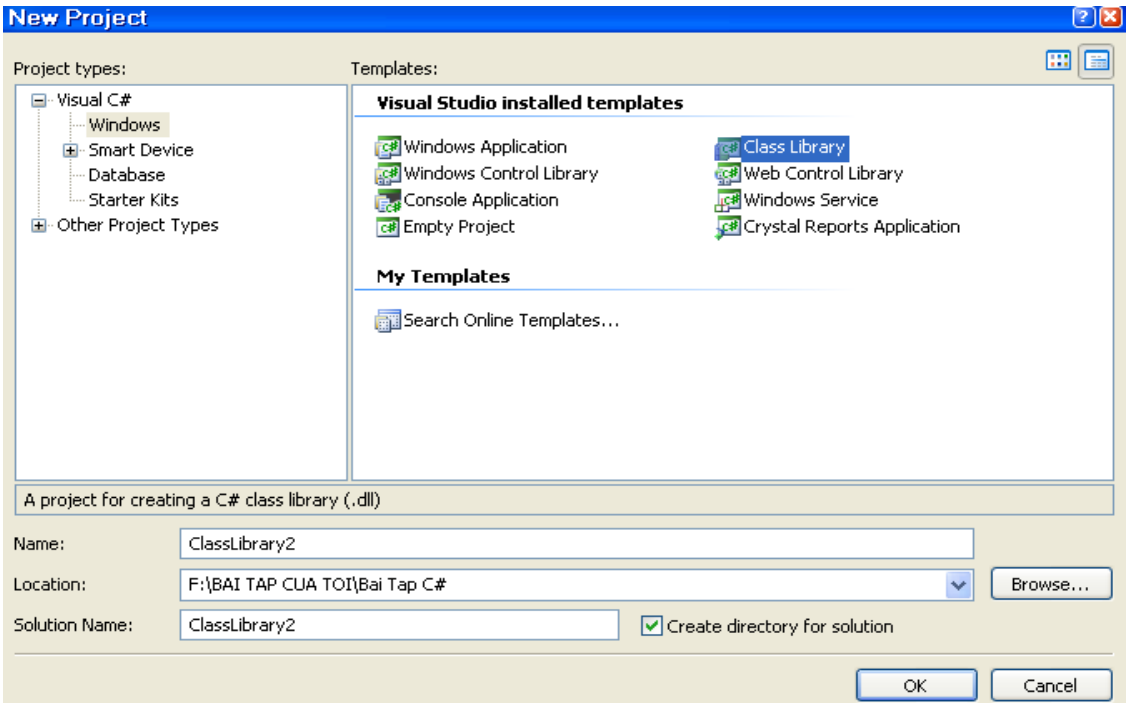
Ở chương lớp và đối tượng chúng ta thường truy xuất các thuộc tính và phương thức với hai từ khóa là private và public. Từ khóa internal cũng nhắc đến nhưng nó ít được sử dụng, vậy từ khóa này được dùng để làm gì và khi nào thì cần dùng đến nó. Để làm sáng tỏ từ khóa này chúng ta đi xây dựng chương trình cài đặt dưới đây.

Trong ngôn ngữ C#, chúng ta thấy có các thư viện sẵn có như: using System, using System.text….Các thư viện này được xây dựng sẵn và chúng ta chỉ việc khai báo để sử dụng.Trên thực tế chúng ta có thể xây dựng các lớp sau đó đóng gói như các thư viện có sẵn trên. Điều này được thể hiện rõ trong phần kế thừa.

Ví dụ: Bạn có lớp cở sở A, Lớp B kế thừa các phương thức của A, Lớp C cũng có nhu cầu kế thừa và sử dụng các phương thức của A. Như vậy ở hai chương trình khác nhau, cùng là lớp A nhưng chúng ta đều phải xây dựng nó ở cả hai chương trình. Vấn đề đặt ra ở đây là bạn xây dựng một lớp cơ sở A và có thể gọi nó ra sử dụng ở bất cứ chương trình nào đó kế thừa nó mà không phải xây dựng lại lớp cơ sở đó.

Cách xây dựng một lớp và đóng gói thành thư viện

Bước 1. Vào Microsoft visual studio 2005→ vào File chọn New chọn Project→ chọn Class Library như hình vẽ nhấn OK

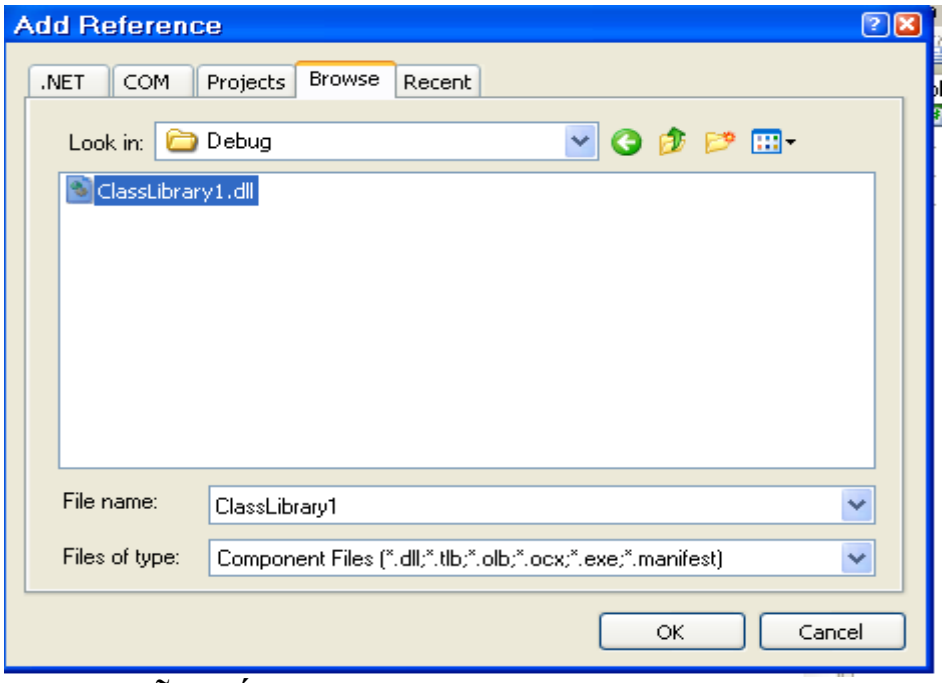


Bước 2. Xây dựng lớp cơ sở A, lưu ý là lớp này không xây dựng hàm main. Lớp này có tên Class Library1.

Bước 3. Sau khi xây dựng xong bước 2, ta tiến hành đóng gói tạo thư viện.Cách đóng gói thực hiện như sau:Vào Build chọn Build Class Library1. Như vậy là bạn đã tạo ra một tệp Class Library1.dll trong debug của thư mục bin của Class Library1.

Bước 4: Đăng kí tệp Class Library1.dll như một tệp của Windown bằng cách:

- Tạo một chương trình mới vào project chọn Add Reference.... xuất hiện hộp thoại như hình sau:



Chọn đường dẫn đến tệp ClassLibrary1.dll vừa tạo sau đó nhấn OK.

Bước 5: Khai báo thư viện vừa tạo trong project chứa lớp dẫn xuất.

Ví dụ: Khai báo thư viện ClassLibrary1

using system ClassLibrary1

Sau khi khai báo xong bạn chỉ việc kế thừa lại lớp cơ sở đó bình thường.

Bài 1: Ví dụ minh họa sử dụng với từ khóa public

Xây dựng lớp Cơ sở A

```
using System;
  public class ClassA
   {
      internal int x;
      public ClassA()
      {
         x = 2;
      }
      public ClassA(int x)
      {
         this.x = x; }
       public void nhap()
```

```
        {
            Console.Write("Nhap vao gia tri cua x=");
            x = int.Parse(Console.ReadLine());
        }
      public void hien()
     {
            Console.WriteLine("Thong tin can hien thi");
            Console.WriteLine("Gia tri cua x={0}", x);
        }
    }
```

Xây dựng lớp B kế thừa từ lớp cơ sở A

```
using System;
using ClassLibrary1;
    class ClassB:ClassA
    {
        internal int y;
        public ClassB(): base()
        {
            y = 5;
        }
        public ClassB(int x, int y): base(x)
        {
            this.y= y;
        }
        public new void nhap()
        {
            base.nhap();
            Console.Write("Nhap gia tri cua y=");
            y = int.Parse(Console.ReadLine());
        }
       public new void hien()
        {
            base.hien();
            Console.WriteLine("Gia tri cua y={0}", y);
```

```
        Console.WriteLine("Tong x + y  = {0}", tong());
        }
       public int tong()
        {
           return x + y ; }
        }
    }
 class tester
    {
       static void Main(string[] args)
       {
          ClassB D = new ClassB();
          D.nhap();
          D.hien();
          Console.ReadLine();
       }
    }
```

Lưu ý: Lớp B muốn truy xuất hay kế thừa các thành viên của lớp A thì Lớp A phải xây dựng là lớp có từ khóa truy cập là public. Vì ta xây dựng chương trình trên nó là hai khối Asembly khác nhau. Bạn cũng có thể sử dụng thư viện lớp A cho bất cứ lớp dẫn xuất nào cần kế thừa nó. Nếu ta thay đổi từ khóa public của ClassA là internal thì chương trinh sẽ báo lỗi vì từ khóa internal chỉ cho phép truy xuất trong khối Assembly.

Bài 2: Ví dụ về cách truy cập của từ khóa internal

```
using System;
namespace BTle
{
   internal class ClassA
   {
      internal int x;
      public ClassA()
      {
```

```
            x = 3;
        }
        public ClassA(int x)
        {
            this.x = x;
        }
    }
}
class ClassB:ClassA
    {
        int y;
        public ClassC(): base()
        {
          y = 6;
        }
        public ClassC(int x, int y): base(x)
        {
            this.y = y;
        }
        public new void nhap()
        {
            base.nhap();
            Console.Write("Nhap gia tri cua y=");
            y = int.Parse(Console.ReadLine());
        }
        public new void hien()
        {
            base.hien();
            Console.WriteLine("Gia tri cua y={0}", y);
            Console.WriteLine("Tong x + y  = {0}", tong());
        }
        public int tong()
        {
            return x + y ;
        }
```

```
}
class tester
{
   static void Main(string[] args)
   {
      ClassBD = new ClassB();
      D.nhap();
      D.hien();
      Console.ReadLine();
   }
}
```

Kết quả chương trình

```
Nhập vào giá trị của x=5
Nhập vào giá trị của y=4
Thông tin cần hiển thị
Giá trị của x=5
Giá trị của y=4
Tổng x+y=9
```

Lưu ý: Trong lập trình các bạn chỉ cần quan tâm đến cách truy nhập và đặc điểm của 3 từ khóa truy cập sau: private, public, protected. Còn lại các từ khóa khác chỉ cần tham khảo để biết về cách thức truy cập của nó.

4. Lớp cơ sở và đối tượng thành phần

Lớp cơ sở A thường được xử lí giống như một thành phần kiểu đối tượng của lớp dẫn xuất B. Nhưng ta có thể thay việc dùng A là lớp cơ sở của B bằng cách khai báo một thành phần kiểu A trong lớp B.

Bài 1:

Xây dựng lớp Stack với các thuộc tính và phương thức cần thiết. Sau đó dùng lớp Stack đó để kiểm tra một xâu kí tự nhập vào

từ bàn phím gồm các kí tự "(" và ")".Kết quả kiểm tra: Xâu hợp lệ "((( )))"

Xâu không hợp lệ "((( ))".

a, Hướng dẫn:

Bài toán có 2 đối tượng class stack, class KTchuoi. Bài toán này chúng ta không xây dựng kế thừa lớp Stack như bài trên mà chúng ta sẽ xây dựng trong lớp KTchuoi có một thành phần kiểu Stack. Việc xây dựng đối tượng kiểu thành phần của một lớp cũng giống như kế thừa ta có thể truy nhập được các thuộc tính và phương thức của lớp là đối tượng thành phần đó.Và để làm rõ điều này chúng ta tham khảo chương trình mẫu ở dưới.

Các phương thức và thuộc tính của lớp Stack xây dựng tương tự như bài 3

Các phương thức và thuộc tính của lớp KTchuoi

```
Class KTchuoi
 {
   Stack a=new Stack();
   public bool kiemtra(string s)
 }
```

b, Bài giải mẫu:

```
using System;
class stack
{
            int top;
            int[] p;
  public stack(int n)
   {
       p=new int[n];
       top = -1;
   }
   public stack()
   {
      p=new int[20];
      top = -1;
   }
```

```
    public bool isempty()
    {
        if (top == -1)
            return true;
        else
            return false;
    }
    public bool full()
    {
        if (top == p.Length)
            return true;
        else
            return false;
    }
    public void push(int a)
    {
        if (full() == true)
            Console.Write("stack day !");
        else
            p[++top] = a;
    }
    public int pop()
    {
        if (isempty())  return 0;
        else return p[top--];
    }
   public void hien()
    {
        while (!isempty())
        {
            Console.Write("{0}",pop());
        }
    }
}
class KTchuoi
```

```
{
   stack a = new stack();
   public bool kiemtra(string s)
   {
      int t;
      for (int i = 0; i < s.Length; i++)
      {
         if (s[i] == '(')
            a.push(Convert.ToInt32(s[i]));
            if (s[i] == ')')
               if (a.isempty()) return false;
               else   t=a.pop();
              }
           if (a.isempty())
          return true;
          else return false;
   }
}
   class Tester
   {
      static void Main()
      {
         string f;
         Console.Write("Ban nhap vao mot chuoi :");
         f = Console.ReadLine();
         KTchuoi a = new KTchuoi();
         if (a.kiemtra(f) == true)
            Console.Write("Chuoi hop le !");
         else Console.Write("Chuoi khong hop le !");
         Console.ReadKey();
      }
   }
```

Kết quả sau khi chạy chương trình:

```
Ban nhap vao mot chuoi: ((())
Chuoi khong hop le !
```

Lưu ý: Hai cách sử dụng kế thừa và khai báo là đối tượng thành phần của lớp đều giống nhau là có thể kế thừa các thuộc tính và phương thức của lớp đã có. Nhưng không phải bài toán nào cũng có thể sử dụng được cả hai cách trên, mà tuỳ theo yêu cầu mà ta có thể chọn phương pháp phù hợp.

Bài 2:

a) Cho lớp '***HocVien'*** được khai báo như sau:

```
class HocVien{ //lớp mô tả một học viên
    private String mahv;  //mã hiệu của học viên
    private String hoTen; //họ tên của học viên
    private int namSinh; //năm sinh của học viên
```

Hãy xây dựng cho lớp HocVien trên các phương thức sau:

- Phương thức tạo dựng với 2 tham số xâu và 1 tham số thực, phương thức này lần lượt lấy các giá trị truyền vào của tham số để gán cho các thuộc tính mahv, hoTen, namSinh.
- Phương thức '*print*' để in ra màn hình thông tin của học viên theo định dạng: 'mahv, hoTen, namSinh'.
- Phương thức 'compare' thực hiện so sánh hai học viên, kết quả trả về một trong các giá trị {-1, 0, 1} lần lượt tương ứng với các trường hợp: học viên thứ nhất có tuổi nhỏ, bằng, lớn hơn học viên thứ hai.

b) Cho lớp '*DanhSach*' được khai báo như sau:

```
class DanhSach{ //lớp biểu diễn cho một danh sách các học viên
    private HocVien ds[]; //mảng để lưu danh sách học viên
    public DanhSach(int n){
 //phương thức tạo dựng để khởi tạo một mảng gồm n phần tử
     ds=new HocVien[n];
```

Hãy xây dựng cho lớp '*DanhSach*' trên các phương thức sau:

+ Phương thức 'printList' để in danh sách học viên ra màn hình, mỗi học viên trên một dòng.

+ Phương thức 'sortList' để sắp xếp danh sách học viên theo thứ tự tăng dần của năm sinh.

a.Hướng dẫn giải

Bài toán bao gồm có 2 đối tượng là Học viên và Danh sách. Lớp Danh sách thực chất là một mảng các đối tượng học viên. Như vậy ở bài toán này chúng ta sử dụng kế thừa thì rất khó giải quyết, vì vậy ta nên xây dựng lớp Danh sách có một đối tượng thành phần là Học viên. Các thuộc tính và phương thức của hai lớp này các bạn đã thấy rất rõ ở đề bài.

b.Chương trình mẫu

```
using System;
class Hocvien
 {
    private string mshv;
    private string hoten;
    private int namsinh;
    public Hocvien()
    {
    }
    public void nhap()
    {
       Console.Write("masv:"); mshv = Console.ReadLine();
       Console.Write("hoten:"); hoten = Console.ReadLine();
       Console.Write("namsinh:");
       namsinh = int.Parse(Console.ReadLine());
    }
    public void print()
    {

Console.WriteLine("\t{0}\t{1}\t{2}",mshv,hoten,namsinh);
    }
    public int compare(Hocvien a,Hocvien b)
    {
       int k;
       if (a.namsinh < b.namsinh)
```

```
            k =-1;
          else {
            if (a.namsinh == b.namsinh)
              k= 0;
            else k = 1;
          }
          return k;
        }
    }
class Danhsach
    {
        private int n;
        private Hocvien[] ds;
        public Danhsach(int n)
        {
            ds = new Hocvien[n];
        }
         public void nhapds()
        {
            Console.Write("nhap so hoc vien:");
            n = int.Parse(Console.ReadLine());
            ds = new Hocvien[n];
            for (int i = 0; i < ds.Length; i++)
              ds[i] = new Hocvien();
            for (int i = 0; i < n; i++)
            {
                Console.WriteLine(" Hoc vien thu{0}", i + 1);
                ds[i].nhap();
            }
        }
        public void printlist()
        {
            Console.WriteLine("Danh sach cac hoc vien la");
            Console.WriteLine("\tstt\tmshv\thoten\t\tnamsinh");
            (int i = 0; i < ds.Length; i++)
```

```
            {
                Console.Write("\t{0}", i + 1);
                ds[i].print();
            }
        }
        public void sortList()
        {
            for (int i = 0; i < ds.Length-1; ++i)
            {
                Hocvien min = ds[i];
                for (int j = i+1; j < ds.Length; ++j)
                {
                    if (min.compare(ds[i], ds[j]) == 1)
                    {
                        min = ds[i];
                        ds[i] = ds[j];
                        ds[j] = min;
                    }
                    else
                    {
                        min = ds[i];
                    }
                }
            }
            for (int i = 0; i < n; i++)
                ds[i].print();
        }
    }
class tester
{
    static void Main(string[] args)
    {
        Danhsach d = new Danhsach(3);
        d.nhapds();
        d.printlist();
```

```
        Console.WriteLine("Danh sach sau khi sap sep theo nam
sinhla:\n");
        d.sortList();
        Console.ReadLine();
      }
   }
```

Kết quả sau khi chạy chương trình:

```
Danh sách các học viên
STT  Mahv     Họ tên                    Năm sinh
1      sv01      Nguyễn Hương Chanh      1987
2        sv02     Nguyễn Thị Lan          1984
3        sv03     Trần Thị Lương          1987
4        sv04     Phương Mai              1985
Danh sách sau khi sắp xếp
  Nguyễn Thị Lan              1984
  Phương Mai                  1985
  Nguyễn Hương Chanh          1987
  Trần Thị Lương              1987
```

II. Nghiêm cấm kế thừa

Bài 1.

Xây dựng một lớp học sinh bao gồm các thuộc tính: họ tên, điểm toán, điểm lí, phương thức nhập, hiện. Lớp học sinh được khai báo là lớp nghiêm cấm kế thừa. Sau đó xây dựng một lớp sinh viên kế thừa các thuộc tính và phương thức của lớp học sinh, ngoài ra lớp sinh viên còn có thêm thuộc tính điểm hoá. Hãy thực hiện chương trình trên và xem chương trình sẽ xảy ra điều gì?

a, Hướng dẫn:

Ta thấy lớp học sinh là lớp nghiêm cấm kế thừa vậy ta phải khai báo từ khoá Sealed trước từ khoá Class như sau:

```
sealed Class hocsinh
{
```

```
        private string hoten;
        private float dtoan, dli;
        public void nhap()
        public void hien()
    }
    class sinhvien:hocsinh
    {
        private float dhoa;
        public void nhap()
        public void hien()
    }
```

b, Bài giải mẫu:

```
using System;
   sealed class hocsinh
   {
      private string hoten;
      private float dtoan,dli;
      public void nhap()
      {
         Console.Write("Nhap ho ten hoc sinh: ");
         hoten=Console.ReadLine();
         Console.Write("Diem toan: ");
         dtoan=float.Parse(Console.ReadLine());
         Console.Write("Diem li: ");
         dli=float.Parse(Console.ReadLine());
      }
      public void hien()
      {
         Console.WriteLine("Thong tin can hien thi");
         Console.WriteLine("Ho ten :{0}",hoten );
         Console.WriteLine("Diem toan: {0}",dtoan );
         Console.WriteLine("Diem li: {0}",dli );
      }
   }
   class sinhvien:hocsinh
```

```
{
   private float dhoa;
  public new void nhap()
   {
      base.nhap();
      Console.Write("Nhap diem hoa:");
      dhoa=float.Parse(Console.ReadLine());
   }
   public new void hien()
   {
      base.hien();
      Console.WriteLine("Diem hoa: {0}",dhoa );
   }
}
class tester
{
   static void Main(string[] args)
   {
      sinhvien a = new sinhvien();
      a.nhap();
      a.hien();
      Console.ReadLine();
   }
}
```

Kết quả sau khi chạy chương trình:
Chương trình đưa ra dòng thông báo lỗi: "*Cannot derive from sealed type hocsinh"*
Kết luận: Lớp sinhvien không thể kế thừa các thuộc tính và phương thức từ lớp Sinh viên.Bởi vì lớp học sinh được bảo vệ cấm kế thừa bằng từ khoá Sealed.

III. Tính đa hình.

Bài 1

Xây dựng một lớp hình, kế thừa lớp hình đó để tính diện tích cho các hình: Hình tròn, hình chữ nhật..

a, Hướng dẫn:

Bài toán này gồm có 3 đối tượng: class Hinh, class hinhtron, class hinhcn.

Ta thấy: hình chữ nhật, hình tròn… đều có chung một phương thức là tính diện tích. Nhưng cách tính diện tích của mỗi hình lại không giống nhau, mỗi một hình nó cách tính riêng. Một phương thức mà có nhiều thể hiện khác nhau được coi là phương thức đa hình. Như vậy phương thức tính diện tích được coi là phương thức đa hình.

Ta xây dựng một lớp cơ sở class Hình có phương thức tính diện tích, tức lớp class Hình chỉ xây dựng làm thế nào để tính diện tích. Cách thức tính như thế nào thì tuỳ thuộc vào hai lớp dẫn xuất class hinhtron và class hinhcn có thể kế thừa lại bằng cách ghi đè.

Các phương thức và thuộc tính của lớp class Hinh

```
class Hinh
{
   //Phương thức tính diện tích được khai báo là phương thức đa hình
   public  virtual float getArea()
}
```

Các thuộc tính và phương thức của Class hinhtron

```
class hinhtron
{
   private float r;
   public hinhtron(float r)
   public override float getArea() //Phương thức ghi đè
}
```

Các thuộc tính và phương thức của lớp Class hinhcn

```
class hinhcn
{
    private float cd,cr;
    public hinhcn(float cd,float cr)
    public override float getArea() //phương thức ghi đè
}
```

b, Bài giải mẫu:

```
using System;
   class hinh
   {
      public virtual double  getArea()
      {
         return getArea();
      }
   }
   class hinhtron:hinh
   {
      private float r;
      public hinhtron(float r)
      {
         this.r=r;
      }
      public override double getArea()
      {
         return r*r*3.14;
      }
   }
class hinhcn:hinh
   {
      private float cd,cr;
      public hinhcn(float cd,float cr)
      {
         this.cd =cd;
         this.cr =cr;
      }
      public override double getArea()
      {
         return cd*cr;
      }
    }
  class tester
```

```
    {
        static void Main(string[] args)
        {
            hinh H = new hinh();
            hinhcn hcn = new hinhcn(2,3);
            hinhtron ht = new hinhtron(3);
            H = ht;
            Console.WriteLine("Dien tich hinh tron= {0}",
H.getArea());
            H = hcn;
            Console.WriteLine ("Dien tich hinh chu nhat= {0}",
H.getArea());
            Console.ReadLine();
        }
    }
```

Kết quả sau khi chạy chương trình:

```
Diện tích hình tròn = 28.26
Diện tích hình chữ nhật= 6
```

Kết luận: Khi kế thừa một phương thức đa hình ta phải ghi đè bằng từ khoá override. Bài toán trên người lập trình sử dụng phương thức thiết lập để lấy dữ liệu đầu vào. Bạn đọc có thể nâng cấp chương trình với các phương thức nhập, hiện để có thể tính toán với bất kì mọi dữ liệu nhập.

Bài 2:

Một cái chuồng gồm 10 cái lồng dùng để nhốt mèo hoặc chó (mỗi lồng chỉ nhốt nhiều nhất một con).Viết chương trình quản lí việc nhốt mèo hoặc chó trong các lồng.

a, Hướng dẫn:

Ta xây dựng các lớp mèo và chó dẫn xuất từ lớp động vật. Do vậy, khi cần quản lí thêm việc nhốt con gà chẳng hạn chỉ cần bổ

sung vào chương trình lớp con gà mà chương trình vẫn hoạt động bình thường mà không cần phải sửa đổi.
Với bài toán này chúng ta sẽ xây dựng lớp động vật có tính đa hình sau đó kế thừa xây dựng lớp chó, mèo. Nếu bài toán này không xây dựng đa hình thì phải giải quyết như thế nào? Ta thấy lớp chó, mèo đều thuộc lớp động vật nhưng chúng không có đặc điểm gì giống nhau để có thể kế thừa cho nhau.Vì vậy chúng ta cần xây dựng một lớp chung cho hai loài này là lớp động vật.
Bài toán gồm có 3 đối tượng: class Dongvat, class cho, class meo

```
class Dongvat
{
    private int vt;
    private string ten;
    public virtual void nhap(int v)
    public virtual void hien()
}
class cho:Dongvat
{
    public override void nhap(int v)
    public override void hien()
}
class meo:Dongvat
{
    public override void nhap(int v)
    public override void hien()
}
```

b, Bài giải mẫu:

```
using System;
namespace chomeo
{
   class Dongvat
   {
      private int vt;
      private string ten;
      public virtual void nhap(int v)
```

```
    {
        vt=v;
        Console.WriteLine ("Vi tri cua chuong: {0}",vt);
        Console.Write ("Nhap vao ten dong vat:");
        ten=Console.ReadLine();
      }
  public virtual void hien()
    {
        Console.WriteLine("Vi tri cua chuong:{0}",vt);
        Console.WriteLine("Ten dong vat:{0}",ten );
    }
 }
 class cho:Dongvat
 {
    public override void  nhap(int v)
    {
        Console.WriteLine("\nNhap thong tin cho con cho");
        base.nhap(v);
    }
    public override void  hien()
    {
        Console.WriteLine("Day la con cho");
      base.hien();
    }
 }
 class Meo:Dongvat
 {
    public override void  nhap(int v)
    {
        Console.WriteLine("\nNhap thong tin cho con meo");
       base.nhap(v);
    }
    public override void  hien()
    {
        Console.WriteLine("Day la con meo");
```

```
        base.hien();
      }
   }
class tester
   {
      static void Main(string[] args)
      {
         Dongvat[] ds = new Dongvat[20];
         int i;
         for (i = 0; i < ds.Length; ++i)
            ds[i] = null;
         ConsoleKeyInfo kt;
        i=0;
         do
         {
            Console.Write ("Chuong thu {0} ban nuoi cho(c)hay
meo(m): ",i);
            kt = Console.ReadKey();
            switch (char.ToUpper(kt.KeyChar))
            {
               case 'C':
                  ds[i] = new cho();
                  ds[i].nhap(i);
                  break;
               case 'M':
                  ds[i] = new Meo();
                  ds[i].nhap(i);
                  break;
            }
            i++;
         }
         while ((int)kt.KeyChar != 13 && i < 20);
         int vt;
         Console.Write("Nhap vao vi tri chuong can kiem tra");
         vt = int.Parse(Console.ReadLine());
```

```
            if (ds[vt] == null)
                Console.WriteLine("Chuong nay khong nuoi con gi");
            else
                ds[vt].hien();
            Console.ReadKey();
        }
    }
}
```

Kết quả sau khi chạy chương trình:

```
Chuồng thứ 0 bạn nuôi gì chó(c) hay mèo(m):c
Nhập thông tin cho chó
Vị trí của chuồng là: 0
Nhập vào tên động vật: Nic
Chuồng thứ 1 bạn nuôi gì chó(c)hay mèo(m): m
Nhập thông tin cho mèo
Vị trí chuồng là: 1
Nhập vào tên động vật: Mướp
Nhập vào vị trí chuồng cần kiểm tra: 1
Đây là con mèo
Vị trí chuồng là: 1
Tên động vật: Mướp
```

IV. Lớp trừu tượng

Bài 1.

Ta thấy lớp chó mèo được xây dựng ở bài 2 kế thừa lớp động vật có các phương thức mang tính đa hình. Để bạn đọc có thể phân biệt được giữa phương thức đa hình với phương thức trừu tượng, chúng ta xây dựng lại bài toán trên theo kiểu lớp trừu tượng

a, Hướng dẫn:

Ta thấy khi quản lí việc nhốt chó mèo, ta có một mảng các chuồng. Và các chuồng đó được dùng để nhốt động vật, và chúng ta

không quan tâm xem chuồng nhốt con vật gì. Việc nhốt con vật gì là do mỗi chuồng. Bài toán đặt ra là quản lí chó mèo, từ 2 loài này ta thấy chúng đều có chung một nguồn gốc là động vật. Động vật mang tính chất chung, không thể hiện rõ là con vật gì. Vì thế ta xây dựng lớp động vật là lớp cơ sở trừu tượng của hai con vật trên.

Các thuộc tính và phương thức của đối tượng chó, mèo xây dựng giống bài 2 phần tính đa hình.Nhưng chỉ khác ở lớp động vật:

```
public abstract  class Dongvat
{
      protected int vt;
      protected string ten;
      public abstract void nhap(int v);
      public abstract void hien();
}
```

b, Bài giải mẫu:

```
using System;
using System.Collections.Generic;
using System.Text;
namespace truu_tuong
{
   public abstract class Dongvat
   {
      protected int vt;
      protected string ten;
      public abstract void nhap(int v);
      public abstract void hien();
   }
class cho:Dongvat
   {
      public override void  nhap(int v)
      {
         vt=v;
         Console.WriteLine ("\nNhap thong tin cho con cho");
         Console.WriteLine ("Vi tri cua chuong:{0}",vt);
         Console.Write("Nhap ten cua con cho:");
```

```
            ten=Console.ReadLine();
        }
        public override void  hien()
        {
            Console.WriteLine("Day la chuong nuoi cho");
            Console.WriteLine("Vi tri cua chuong:{0}",vt);
            Console.WriteLine("Ten cua con cho :{0}",ten );
        }
    }
    class meo:Dongvat
    {
        public override void  nhap(int v)
        {
            vt=v;
            Console.WriteLine ("\nNhap thong tin cho con meo");
            Console.WriteLine ("Vi tri cua chuong:{0}",vt);
            Console.Write("Nhap ten cua con meo:");
            ten=Console.ReadLine();
        }
        public override void  hien()
        {
            Console.WriteLine("Day la chuong nuoi meo");
            Console.WriteLine("Vi tri cua chuong:{0}",vt);
            Console.WriteLine("Ten cua con meo :{0}",ten );
        }
    }
class tester
    {
        static void Main(string[] args)
        {
            Dongvat[] ds = new Dongvat[20];
            int i;
   for (i = 0; i < ds.Length; ++i)
                ds[i] = null;
            ConsoleKeyInfo kt;
```

```
            i = 0;
            do
            {
                Console.Write("Chuong thu {0} ban nuoi cho(c)hay
meo(m): ", i);
                kt = Console.ReadKey();
                switch (char.ToUpper(kt.KeyChar))
                {
                    case 'C':
                        ds[i] = new cho();
                        ds[i].nhap(i);
                        break;
                    case 'M':
                        ds[i] = new meo();
                        ds[i].nhap(i);
                        break;
                }
                i++;
            }
            while ((int)kt.KeyChar != 13 && i < 20);
            int vt;
            Console.Write("Nhap vao vi tri chuong can kiem tra");
            vt = int.Parse(Console.ReadLine());
            if (ds[vt] == null)
                Console.WriteLine("Chuong nay khong nuoi con gi");
            else
                ds[vt].hien();
            Console.ReadKey();
        }
    }
}
```

Kết quả sau khi chạy chương trình:
Giống bài trên

V. Xây dựng lớp lồng nhau

Bài 1.

Xây dựng lớp hình chữ nhật với các thuộc tính và phương thức cần thiết. Sau đó xây dựng lớp hình hộp bên trong lớp hình chữ nhật và tính thể tích cho hình hộp đó.

a, Hướng dẫn:

Khi giải quyết một bài toán bạn mong muốn một lớp dẫn xuất có thể truy xuất đến các thành phần thuộc tính của lớp cơ sở. Thông thường bạn sẽ kế thừa bằng cách khai báo thuộc tính của lớp cơ sở là protected. Nhưng bài toán này bạn không cần phải làm điều đó mà vẫn truy nhập được thuộc tính Private của lớp cơ sở. Chúng ta sẽ xây dựng một lớp cơ sở, thay vì xây dựng lớp dẫn xuất ta sẽ xây dựng một lớp lồng bên trong của lớp cơ sở trên. Và lớp này cũng sẽ trở thành một đối tượng thành phần của lớp ngoài.

Bài toán gồm có 2 đối tượng class hinhcn, class hinhhop

Cấu trúc xây dựng lớp lồng của bài toán này như sau:

```
class hinhcn
{
    private int cdai, crong;
    public hinhcn()
    public hinhcn(int cd, int cr)
    public void nhap()
    public int dientich()
    Public void hien()
    internal class hinhhop
    {
        private int cao;
        public hinhhop()
        public hinhhop(int cd, int cr, int c)
        public void nhap(hinhcn f)
        public int thetich(hinhcn f)
        public void hien(hinhcn f)
    }
}
```

b, Bài giải mẫu:

```
using System;
namespace loplong
{
   class hinhcn
   {
      private int cdai,crong;
      public hinhcn()
      {
         cdai=5;
         crong=3;
      }
      public hinhcn(int cd, int cr)
      {
         cdai = cd;
         crong = cr;
      }
      public void nhap()
      {
         Console.Write("nhap chieu dai:");
         cdai=int.Parse(Console.ReadLine());
         Console.Write("Nhap chieu rong:");
         crong=int.Parse(Console.ReadLine());
      }
      public int dientich()
      {
         return cdai*crong ;
      }
      public void hien()
      {
         Console.WriteLine("Thong tin can hien thi");
         Console.WriteLine("Chieu dai={0}",cdai );
         Console.WriteLine("Chieu rong={0}",crong);
         Console.WriteLine("Dien tich hinh cn ={0}",dientich());
```

```
    }
    internal class hinhhop
    {
        int cao;
        public hinhhop()
        {
            cao=4;
        }
        public hinhhop(int cd,int cr,int c)
        {
            cao = c;
        }
        public int thetich(hinhcn f)
        {
          return f.dientich()*cao;
        }
     public void nhap(hinhcn f)
       {
            f.nhap();
            Console.Write("Nhap chieu cao=");
            cao = int.Parse(Console.ReadLine());
        }
        public void hien(hinhcn f)
        {
            f.hien();
            Console.WriteLine("The tich hinh hop:{0}", thetich(f));
        }
    }
}
class tester
{
    static void Main(string[] args)
    {
        hinhcn f = new hinhcn();
        hinhcn.hinhhop A = new hinhcn.hinhhop();
```

```
            A.nhap(f);
            A.hien(f);
            Console.ReadLine();
        }
    }
}
```

Kết quả sau khi chạy chương trình:

```
Nhập chiều dài: 10
Nhập chiều rộng: 7
Nhập chiều cao:  5
Thông tin hiển thị
Chiều dài = 10
Chiều rộng =7
Diện tích hình chữ nhật = 70
Thể tích hình hộp =350
```

Kết luận

Việc xây dựng lớp lồng có ưu điểm là có thể truy xuất các thành viên private của lớp ngoài.Tuy nhiên nếu có nhiều lớp được xây dựng lồng nhau thì chương trình rất rối, bạn đọc sẽ khó hiểu hơn.Bài toán này chỉ lên xây dựng khi có ít lớp, nếu bài toán có nhiều lớp thì chúng ta lên xây dựng một lớp cơ sở sau đó kế thừa để xây dựng các lớp khác. Khi xây dựng lớp lồng bên trong cần chú ý khai báo từ khoá Internal trước từ khoá Class.

C. MỘT SỐ CÂU HỎI LÝ THUYẾT:

Câu 1: Có cần thiết phải chỉ định từ khóa **override** trong phương thức phủ quyết của lớp dẫn xuất hay không?

Trả lời: Có, chúng ta phải khai báo rõ ràng từ khóa override với phương thức phủ quyết phương thức ảo (của lớp cơ sở) bên trong

lớp dẫn xuất.

Câu 2: Lớp trừu tượng là thế nào? Có thể tạo đối tượng cho lớp trừu tượng hay không?

Trả lời: Lớp trừu tượng không có sự thực thi, các phương thức của nó được tạo ra chỉ là hình thức, tức là chỉ có khai báo, do vậy phần định nghĩa bắt buộc phải được thực hiện ở các lớp dẫn xuất từ lớp trừu tượng này. Do chỉ là lớp trừu tượng, không có sự thực thi nên chúng ta không thể tạo thể hiện hay tạo đối tượng cho lớp trừu tượng này.

Câu 3: Có phải khi tạo một lớp thì phải kế thừa từ một lớp nào không?

Trả lời: Không nhất thiết như vậy, tuy nhiên trong C#, thì tất cả các lớp được tạo điều phải dẫn xuất từ lớp Object. Cho dù chúng có được khai báo tường minh hay không. Do đó Object là lớp gốc của tất cả các lớp được xây dựng trong C#. Một điều thú vị là các kiểu dữ liệu giá trị như kiểu nguyên, thực, ký tự cũng được dẫn xuất từ Object

Câu 4: Lớp lồng bên trong một lớp là như thế ?

Trả lời: Lớp lồng bên trong một lớp hay còn gọi là lớp nội được khai báo với từ khóa internal, chứa bên trong phạm vi của một lớp. Lớp nội có thể truy cập được các thành viên private của lớp mà nó chứa bên trong.

Câu 5: Chương trình sau đây có lỗi, hãy sửa lỗi biên dịch và chạy chương trình.Lệnh nào gây ra lỗi và nguyên nhân gây lỗi.

```
class A
{
  private int x;
   public A()
```

```
        {
            x=2;
        }
        public A(int x)
        {
            this.x=x;
        }
    }
class B:A
    {
        private int y;
        public B():base()
        {
            y=3;
        }
        public B(int x,int y):base(x)
        {
            this.y=y;
        }
        public int tong()
        {
            return x + y;
        }
    }
    class teater
    {
        static void Main(string[] args)
        {
            B h = new B(2,3);
            Console.WriteLine("Gia tri cua tong:{0}", h.tong());
            Console.ReadLine();
        }
    }
```

Trả lời 5: Chương trình trên khi biên dịch có 1 lỗi ".A.x is inaccessible due to its protection level".

Câu lệnh gây ra lỗi là: **return** x + y;
Nguyên nhân gây ra lỗi: Lớp B kế thừa thuộc tính x của lớp A. Mà ta thấy thuộc tính x của lớp A là private nên nó chỉ được sử dụng trong lớp và không cho phép lớp khác kế thừa.
Cách sửa: Lớp B muốn kế thừa thuộc tính x của lớp A thì trong lớp A ta xây dựng thuộc tính x là protected.
Kết quả chương trình: Giá trị của tổng=5

D. CÂU HỎI TỰ TRẢ LỜI

Câu 1: Theo bạn phương thức đa hình và phương thức trừu tượng :

a. Cùng là một phương thức ảo.

b. Là phương thức ảo có ý nghĩa sử dụng giống nhau nhưng cách xây dựng hai phương thức lại khác nhau.

c. Hai phương thức này thực chất là một phương thức

Câu 2: Khi khai báo phương thức trong lớp kế thừa mà có tên trùng với tên của phương thức trong lớp cơ sở thì:

a. Phải sử dụng từ khoá *new* để che giấu phương thức.

b. Không cần thiết phải sử dụng từ khoá new

c. Thay đổi bằng cách khai báo tham số sử dụng của phương thức.

Câu 3: Một phương thức ảo trong lớp cơ sở có nhất thiết phải được phủ quyết trong lớp dẫn xuất hay không?

Câu 4: Lớp trừu tượng có cần thiết phải xây dựng hay không? Hãy cho một ví dụ về một lớp trừu tượng cho một số lớp.

Câu 5: Lớp cô lập là gì? Có thể khai báo protected cho các thành viên của nó được không?

Câu 6: Lớp Object cung cấp những phương thức nào mà các lớp khác thường xuyên kế thừa để sử dụng.

Câu 7: Thế nào là boxing và unboxing? Hãy cho biết hai ví dụ về quá trình này?

Câu 8: Định nghĩa nào sau đây định nghĩa rõ lớp trừu tượng hợp lệ

```
class A
 {
    abstract void unfinished() {}
 }
 class B
 {
    abstract void unfinished();
 }
 abstract class C
 {
    abstract void unfinished();
 }
 public class abstract D
 {
    abstract void unfinished();
 }
```

Câu 9: Chương trình sau đây có lỗi, hãy sửa lỗi biên dịch và chạy chương trình.Lệnh nào gây ra lỗi và nguyên nhân gây lỗi.

```
using System;
   abstract public class Dongvat
   {
      protected string name;
      public abstract void Who();
      public Dongvat(string name)
      {
         this.name=name ;
      }
   }
   class cho:Dongvat
   {
      private string color;
      public cho(string name,string color):base(name)
      {
```

```
            this.color=color ;}
        {
            Console.WriteLine ("Toi la con cho {0} mau {1}",name
,color );
        }
    }
    class meo:Dongvat
    {
        private int trongluong;
        public meo(string name,int trongluong):base(name)
        {
            this.trongluong=trongluong ;
        }
        public override void  Who()
        {
            Console.WriteLine ("Toi la con
meo{0}nang{1}",name,trongluong);
        }
    }
    class tester
    {
        static void Main(string[] args)
        {
            Dongvat[] ds = new Dongvat[3];
           ds[0] = new cho("John", "vang");
            ds[1] = new meo("Muop", 2);
            ds[2] = new Dongvat("Noname");
         for (int i = 0; i < 3; i++)
            {
                ds[i].Who();
            }
            Console.ReadLine();
        }
    }
}
```

Câu 10: Phân biệt mục đích sử dụng của các từ khoá sau: new, virtual, override.

E. BÀI TẬP TỰ GIẢI:

Bài 1

Xây dựng lớp Stack với các thao tác cần thiết. Từ đó hãy dẫn xuất từ lớp Stack để đổi một số nguyên dương n sang hệ đếm 16.

Hướng dẫn

Bài toán gồm có 2 đối tượng: Stack và Đổi số

Các thuộc tính và phương thức của lớp Stack

```
class Stack          //Lớp cơ sở
{
   private  int top;
            int  [] s;
    public  stack()
    public Stack(int n)
    public bool full()        // Kiểm tra Stack đầy
    public bool empty()     //  Kiểm tra Stack trống
    public  void push(int x)
    public  void pop()
}

class Doiso          //Lớp dẫn xuất
{
   private int a,n;          //a là hệ cần đổi, n là số cần đổi
   public void nhap()
   public void doi()
   public void hien()
}
```

Lưu ý: Bài toán này yêu cầu chỉ cho phép nhập hệ cần đổi là 16. Còn lại các giá trị khác không cho phép nhập. Vậy bạn phải khống chế các giá trị của hệ số a sao cho thoả mãn đề bài.

Bài 2 :

Xây dựng lớp hình tròn với các phương thức cần thiết.Sau đó kế thừa để tính thể tích hình cầu.

Hướng dẫn

Bài toán gồm hai đối tượng class hinhtron, class hinhcau

```
class hinhtron
{
    private float r;
    public hinhtron()
    public hinhtron(float x)
    public void nhap()
    public void hien()
    public float dientich()
}
class hinhcau
{
    public float thetich()
}
```

Lưu ý: Công thức tính thể tích hình cầu $V=(4*\Pi*r^3)/3$

Bài 3:

Viết chương trình quản lí nhân sự và tính lương cho nhân viên trong công ty.

1.Quản lí thông tin nhân viên (Họ tên, ngày sinh, địa chỉ)

2. Tính lương cho nhân viên. Biết trong công ty có ba loại nhân viên và cách tính lương như sau:

Nhân viên sản xuất: số sản phẩm*20 000 đ

Nhân viên công nhật: số ngày công*50 000 đ

Nhân viên quản lí : hệ số lương * lương cơ bản.

Hướng dẫn

Bài toán được chia thành 5 đối tượng: Class nhanvien, Class NVSX, Class NVCN, Class NVQL, Class DSNV.

Các thuộc tính và phương thức của Class Nhanvien

+ Họ tên, địa chỉ, ngày sinh

+ Nhap(), xuat(), tinhluong()

Các thuộc tinh và phương thức của Class NVSX

+ Kế thừa thuộc tính của Class nhanvien

+ sosanpham
+ Phương thức Nhap(), xuat(), tinhluong().

Các thuộc tính và phương thức của Class NVCN
+Kế thừa thuộc tính của Class nhanvien
+ songaycong
+ Phương thức Nhap(), xuat(), tinhluong()

Các thuộc tính và phương thức của Class NVQL
+Kế thừa thuộc tính của Class nhanvien
+ hesoluong, luongcoban
+ Phương thức Nhap(), xuat(), tinhluong()

Các thuộc tính và phương thức của Class DSNV
+ soluongnv, Nhan vien*[] ds
+ Phương thức: Nhap ds(), xuat ds(), xuatluong()

Lưu ý: Bài toán này các lớp đều có phương thức tính lương, xong mỗi lớp có cách tính lương là khác nhau. Do vậy ta nên xây dựng phương thức tính lương ở lớp cơ sở (Class nhanvien) là phương thức ảo, sau đó ở các lớp dẫn xuất ta kế thừa lại bằng cách ghi đè phương thức đó..

Bài 4:

Xây dựng lớp "MaTranVuong" bằng cách kế thừa từ lớp "MaTran" được xây dựng và bổ xung thêm các phương thức sau:

- "getDeterminant" để trả về định thức của ma trận.
- "duongCheoChinh" để tính tổng các phần tử trên đường chéo chính
- "duongCheoPhu" để tính tổng các phần tử trên đường chéo phụ
- "duongCheoChinhK" để tính tổng các phần tử trên đường chéo chính thứ k
- "duongCheoPhuK" để tính tổng các phần tử trên đường chéo phụ thứ k

Bài 5:

a) Xây dựng lớp trừu tượng "Hinh" như sau:

```
abstract class Hinh
{
  abstract public float getArea();
```

}

b) Xây dựng lớp "HinhVuong", "HinhTron", "HinhTamGiac", "HinhBinhHanh", "HinhThoi"… bằng cách kế từ lớp Hinh và ghi đè phương thức getArea để tính diện tích của các hình tương ứng.

Hướng dẫn

Bài toán gồm có 6 đối tượng: class Hinh, class HinhVuong, class HinhTron, class HinhBinhHanh, class Hinhthoi

Các thuộc tính và phương thức của các lớp:

```
abstract public class Hinh
{
    abstract public float getArea();
}
class Hinhvuong:Hinh
{
    float canh;
    public Hinhvuong()
    public Hinhvuong(foat a)
    public override float getArea()
    public void nhap()
    public void hien()
}
class Hinhtron:Hinh
{
    private float r;
    public Hinhtron()
    public Hinhtron(float r)
    public override float getArea ()
}
class Hinhtamgiac:Hinh
{
    private float cao,day
    public Hinhtamgiac()
    public Hinhtamgiac(float cao, float day)
    public override float getArea()
}
```

```
class Hinhbinhhanh:Hinh
{
    private float a,b // a,b là 2 đường chéo của hình bình hành
    public Hinhbinhhanh()
    public Hinhbinhhanh(float a, float b)
    public override float getArea()
}
class HinhThoi:Hinh
{
    private float a,b // a,b là 2 đường chéo của hình thoi
    public HinhThoi()
    public HinhThoi(float a, float b)
    public override float getArea()
}
```

Lưu ý: Các công thức tính diện tích của các hình

Tamgiac=1/2(day*cao)

Hinhbinhhanh=a*b // *a,b là 2 đường chéo*

Hinhthoi=1/2(a*b)

Bài 6:

Xây dựng các lớp sau:

+ Lớp NGƯỜI gồm:
  - Các thuộc tính
    Char hoten;
    int ns;
  - Các phương thức
    Hai hàm khởi tạo,Phương thức in()

+ Lớp MÔNHỌC gồm:
  - Các thuộc tính:
    Char mon; *// Tên môn học*.
    int st; // số tiết
  - Các phương thức
    Hai hàm khởi tạo,Phương thức in()

+Lớp GIÁOVIÊN
  - Các thuộc tính
    Kế thừa từ lớp người

Thêm một số thuộc tính
Char bomon ; // *Bộ môn công tác*
MÔNHỌC mh; // *Môn học đang dạy*

- Các phương thức

Hai hàm khởi tạo ,phương thức int()

Để đưa ra thông tin về giáo viên (họ tên, năm sinh, bộ môn, môn dậy,số tiết của môn học đang dậy đó).

Gợi ý: Trong lớp GIÁOVIÊN có thể gọi 2 phương thức in()

+ GIÁOVIÊN.in() // *Được xây dựng trong lớp GIÁOVIÊN*
+ NGƯỜI.in() // *Thừa kế từ lớp người*.

Bài 7.

Viết ứng dụng đơn giản phục vụ cho học sinh phổ thông kiểm tra kiến thức về hình học (trong không gian R^2), sau đây là yêu cầu về ứng dụng:

- Cho phép tính khoảng cách giữa hai điểm
- Cho phép tính độ dài của một đoạn thẳng
- Cho phép kiểm tra sự giao nhau của hai đoạn thẳng
- Cho phép kiểm tra vị trí tương đối của hai đường tròn
- Cho phép kiểm tra một điểm nằm trên, trong hay bên ngoài đường tròn.
- Cho phép tính diện tích, chu vi của hình tròn.
- Cho phép tính diện tích, chu vi của tam giác.
- Cho phép kiểm tra một điểm nằm trên, trong hay ngoài tam giác.
- Tính diện tích, chu vi, kiểm tra sự giao nhau của hình chữ nhật.
- Tính diện tích, chu vi, kiểm tra sự giao nhau của hình vuông.

Bài 8

Viết chương trình tính giá điện của một căn hộ :

Phương thức thiết lập 2 tham số dung khởi tạo giá điện(=750 vnd) và số điện đã dùng.

Phương thức nhập cho từng lớp

Phương thức hiện cho từng lớp

Sau đó kế thừa lớp cơ sở để tính tiền điện mới cho hợp với việc nếu số điện dùng quá 100 số điện thì giá điện kể từ số 100 trở đi là

1000 vnd. Còn nếu số điện sử dụng lớn hơn 50 và nhỏ hơn 100 thì giá điện là 800, Số diện sử dụng nhỏ hơn 50 thì giá điện là 750.

CHƯƠNG IV
GIAO DIỆN

Mục đích: Sau khi nghiên cứu chương này người học nắm được các nội dung sau:

- Hiểu thế nào là một giao diện.
- Cách thức xây dựng và truy cập một giao diện
- Sự thực thi nhiều giao diện của một lớp
- Sự khác nhau giữa giao diện và lớp trừu tượng

A. TÓM TẮT LÍ THUYẾT

1. Khái niệm về giao diện

Giao diện là tập các hàm khai báo sẵn mà không cài đặt. Các lớp thực thi giao diện có nhiệm vụ cài đặt các hàm này.

Mục đích của việc cài đặt giao diện: Cho phép một lớp dẫn xuất có thể kế thừa thực thi nhiều giao diện. Điều này đã khắc phục được hạn chế của lớp trừu tượng là một lớp dẫn xuất chỉ có thể thực thi một lớp trừu tượng.

2. Cài đặt một giao diện

Cú pháp để định nghĩa một giao diện như sau:

[thuộc tính] [bổ sung truy cập] interface <tên giao diện> [: danh sách cơ sở]

```
{
   <phần thân giao diện>
}
```

Ví dụ: Xây dựng một giao diện có tên là Hinh và giao diện này có hai phương thức đọc và ghi đối tượng.

```
interface Hinh
 {
  void read();
  void write();
 }
```

Mở rộng giao diện:

Chúng ta mở rộng một giao diện đã có bằng cách thêm các phương thức, các thành viên hay bổ sung cách làm việc cho các thành viên.

Thực thi nhiều giao diện:

Trong ngôn ngữ C# cho phép chúng ta thực thi nhiều hơn một giao diện.

Ví dụ: Lớp Document có thể thực thi cả hai giao diện Istorable và Icompressible. Ta xây dựng như sau: public class Document:Istorable, Icompressible

3. Truy xuất phương thức của giao diện

Chúng ta có thể truy cập những thành viên của giao diện như thể các thành viên của lớp.

Ví dụ: Lớp Hình vuông thực thi giao diện có tên là Hinh ở ví dụ trên. Như vậy lớp Hình vuông có thể truy cập 2 thành viên read() và write() của giao diện Hinh.

```
Hinhvuong H=new Hinhvuong();
  H.read();
  H.write();
```

4. Toán tử is:

Dùng để kiểm tra một đối tượng xem nó có hỗ trợ giao diện hay không.

Cú pháp của toán tử is: <biểu thức> is <Kiểu dữ liệu>

5. Toán tử as

Toán tử as kết hợp toán tử is và phép gán bằng cách đầu tiên kiểm tra hợp lệ phép gán rồi sau đó phép gán được thực hiện. Nếu

không hợp lệ thì toán tử as trả về giá trị Null.Sử dụng toán tử as để loại bỏ việc xử lí các ngoại lệ.
Cú pháp của toán tử as: <biểu thức> as <Kiểu dữ liệu>

6. Thực thi phủ quyết giao diện

Khi thực thi một lớp chúng ta có thể tự do đánh dấu bất kì hay tất cả các phương thức thực thi giao diện như một phương thức ảo.
Ví dụ: Phủ quyết phương thức read() của giao diện Hinh

Public virtual void Read()

7. Thực hiện giao diện một cách tường minh

Một lớp có thể cài đặt nhiều giao diện nên có thể xẩy ra trường hợp đụng độ về tên khi hai giao diện có cùng một tên hàm. Để giải quyết xung đột này ta khai báo cài đặt một cách tường minh hơn. Khi đó ta phải thêm tên giao diện vào trước tên phương thức.

Lựa chọn các phương thức phơi bày của giao diện

Người thiết kế có thêm thuận lợi khi một giao diện được thi công trong suốt quá trình ấy sự thi công tường minh không được thể hiện ở phía Client.Giả sử như đối tượng Document thi công giao diện Istorable nhưng chúng ta không muốn các phương thức Read() và Write() của giao diện istorable được xem như là Public trong lớp Document. Chúng ta có thể sử dụng phần thực hiện tường minh để chắc rằng chúng không sẵn có trong suốt quá trình phân bố.

Thành viên ẩn

Ngôn ngữ C# có một khả năng mới là, các thành viên của giao diện có thể được ẩn đi.

B.BÀI TẬP MẪU

I. Sử dụng giao diện

Các bạn có thể tham khảo chương trình sau để hiểu rõ về giao diện và cách sử dụng nó.

```
using System;
// khai báo giao diện interface IStorable
interface IStorable
```

```
{
   void Read();
   void Write(object obj);
   int Status
   {
   get;
   set;
   }
}
   // thực thi phương thức Write public void Write( object o)
   public void Write( object o)
   {
      Console.WriteLine("Impleting the Write Method for
IStorable");
   }
// tạo một lớp thực thi giao diện IStorable public class Document
: IStorable
public class Document : IStorable
{
   public Document( string s)
   {
      Console.WriteLine("Creating document with: {0}", s);
   }
   // thực thi phương thức Read()
   public void Read()
   {
      Console.WriteLine("Implement the Read Method for
IStorable");
   }
 // thực thi phương thức Write public void Write( object o)
   public void Write( object o)
   {
      Console.WriteLine("Impleting the Write Method for
IStorable");
   }
```

```
    // thực thi thuộc tính public int Status
    public int Status
    {
        get
        {
            return status;
        }
        set
        {
            status = value;
        }
    }
     private int status = 0;
}
public class Tester
 {
    static void Main()
    {
        // truy cập phương thức trong đối tượng Document
        Document doc = new Document("Test Document");
        doc.Status = -1;
        doc.Read();
        Console.WriteLine("Document Status: {0}", doc.Status);
        // gán cho một giao diện và sử dụng giao diện
        IStorable isDoc = (IStorable)doc;
        isDoc.Status = 0;
        isDoc.Read();
        Console.WriteLine("IStorable Status: {0}", isDoc.Status);
        Console.ReadKey();
    }
}
```

Kết quả sau khi chạy chương trình:

```
Creating document with: Test Document
Implement the Read Method for Istorable
Document Status : -1
Implement the Read Method for Istorable
Istorable Status : -1
```

II.Mở rộng một giao diện và sự thực thi nhiều giao diện của một lớp

Bài 1.

Xây dựng giao diện nhân viên bao gồm hai phương thức: Nhập và hiển thị thông tin của cán bộ. Ta có thể xây dựng thêm một giao diện nhân viên mới mở rộng từ giao diện nhân viên. Giao diện nhân viên mới này xây phương thức cách tính lương cho mỗi loại nhân viên.

Hãy xây dựng lớp nhân viên sản xuất thực thi hai giao diện trên. Biết cách tính lương nhân viên như sau:

Lương=Hsl*Lương cơ bản

a, Hướng dẫn:

Ở chương 3 chúng ta đã biết đến sự kế thừa của một lớp dẫn xuất, xong nó chỉ kế thừa được từ một lớp cơ sở. Như bài toán này chúng ta không thể xây dựng Class Nhanviensx kế thừa từ hai lớp là Nhanvien và Nhanvienmoi. Mục đích chúng ta muốn lớp Class Nhanviensx kế thừa và thực thi cả hai Nhanvien và Nhanvienmoi, do đó chúng ta phải xây chúng là hai giao diện.

Bài toán bao gồm có hai giao diện và một lớp thực thi giao diện: Interface Nhân viên, interface Nhân viên mới, Class Nhân viên sx

Các phương thức và thuộc tính

```
interface Nhanvien
{
   void nhap();
   void hien();
}
```

```
    interface Nhanvienmoi:Nhanvien // Mở rộng giao diện Nhân viên
    {
      double luong();
    }
class Nhanviensx:Nhanvien,Nhanvienmoi // Thực thi hai giao diện
    {
       private string ten;
       private static int lcb;
       private double hsl;
       public Nhanviensx()
                public virtual void nhap()
              public virtual void hien()
                public virtual double luong()
             }
```

b, Bài giải mẫu:

```
using System;
   interface Nhanvien
   {
      void nhap();
      void hien();
   }
    interface Nhanvienmoi:Nhanvien
    {
      double luong();
    }
    class Nhanviensx:Nhanvien,Nhanvienmoi
    {
       private string ten;
       private static int lcb;
       private double hsl;
       public Nhanviensx()
       {
          hsl = 2.5;
          lcb=450;
```

```
        }
        public Nhanviensx(int hsl)
        {
            this.hsl=hsl ;
        }
        public virtual void nhap()
        {
            Console.Write("Nhap vao ho ten:");
ten=Console.ReadLine();
            Console.Write("Nhap vao he so luong:");
            hsl=float.Parse(Console.ReadLine());
        }
      public virtual void hien()
        {
            Console.WriteLine("Thong tin can hien thi");
            Console.WriteLine("Ho ten nhan vien:{0}",ten );
            Console.WriteLine("He so luong:{0}",hsl );
        }
      public virtual double luong()
        {
            return hsl*lcb ;
        }
    }
class tester
    {
        static void Main(string[] args)
        {
            Nhanviensx A = new Nhanviensx();
            A.nhap();
            A.hien();
            Console.WriteLine("Luong cua nhan vien nay la:{0}",
A.luong());
            Console.ReadLine();
        }
    }
```

Kết quả sau khi chạy chương trình:

```
Nhập vào họ tên: Nguyễn Văn A
Nhập vào hệ số lương: 5
Thông tin cần hiển thị
Họ tên nhân viên: Nguyễn Văn A
Hệ số lương: 5
Lương của nhân viên này là: 2250
```

III. Xây dựng lớp thực thi phủ quyết một giao diện

Bài 1

a) Xây dựng lớp "DaySo" để mô tả một dãy số, gồm các phương thức sau:

- Phương thức "nhap" dùng để nhập dãy số từ bàn phím
- Phương thức "print" dùng để in dãy số ra màn hình
- Hàm tạo DaySo(int n) dùng để khởi tạo một mảng gồm n phần tử

b) Xây dựng giao diện Sort như sau:

```
interface Sort
{
   public void Sort();
}
```

c) Xây dựng các lớp "QuickSort", "SelectionSort", "InsertSort" bằng cách kế thừa từ lớp DaySo và triển khai giao diện Sort để thực hiệp việc sắp xếp: nổi bọt, chọn trực tiếp, chèn trực tiếp

a, Hướng dẫn:

Giả sử bạn muốn thiết kế phương thức sắp xếp chung để sắp xếp các phần tử. Phần tử ở đây có thể là mảng các đối tượng như dãy số, ma trận, lớp hoc viên. Các đối tượng này có thể được sắp xếp bằng các thuật toán khác nhau. Vậy bạn cần xác định một phương thức sắp xếp chung nhằm thực hiện sắp xếp các đối tượng. Cách thức sắp xếp các đối tượng như thế nào do từng thuật toán tự quyết định.Ta thấy bài toán vừa yêu cầu kế thừa vừa thực thi giao

diện. Đây chính là tiến bộ của sử dụng giao diện so với lớp trừu tượng.

Các đối tượng của bài toán bạn đọc có thể xác định thấy rõ ngay ở phần đề bài.

Các thuộc tính và các phương thức

```
class dayso
{
    public int[] a;
    protected int n;
    public dayso()
    public dayso(int n)
    public void nhap()
    public void print()
}
interface sort
{
    void sapxep();
}
class quicksort:dayso, sort
{
     public void sapxep()
}
class selectsort:dayso,sort
{
     public void sapxep()
}
class inserttionsort:dayso,sort
{
    public void sapxep()
}
```

Lưu ý: Lớp dayso có 2 thuộc tính là n và mảng a hai thuộc tính này nó được các lớp dẫn xuất kế thừa và gọi ra để thực thi. Do vậy hai thuộc tính này phải được khai báo là Public.

b, Bài giải mẫu:

```
using System;
```

```
class dayso
{
   public int[] a;
   protected int n;
   public dayso()
   {
      a = new int[5];
   }
   public dayso(int n)
   {
      a = new int[n];
   }
   public void nhap()
   {
      Console.Write("Nhap vao so phan tu cua day so:");
      n = int.Parse(Console.ReadLine());
      Console.WriteLine("Nhap vao thong tin cua day so:");
      for (int i = 0; i < n; i++)
      {
         Console.Write("a[{0}]=", i);
         a[i] = int.Parse(Console.ReadLine());
      }
   }
   public void print()
   {
      for (int i = 0; i < n; i++)
         Console.Write("{0}\t",a[i]);
         Console.WriteLine();
   }
}
interface sort
{
     void sapxep();
}
class quicksort:dayso,sort
```

```
{
    public void sapxep()
   {
      int i, j,x,tmp;
      x=a[(n)/2];
      i = n-1; j = n;
do
      {
         while (a[i] < x) i++;
         while (a[j] > x) j--;
         if (i <= j)
         {
            tmp = a[i];
            a[i] = a[j];
            a[j] = tmp;
            i++; j--;
         }
      } while (i < j);
   }
}
class selectsort:dayso,sort
{
   public void sapxep()
   {
      int min,tmp,j;
      for (int i = 0; i < n - 1; i++)
      {
         min=i;
         for (j = i + 1; j < n; j++)
            if (a[j] < a[min])
               min = j;
               tmp = a[min];
               a[min] = a[i];
               a[i] = tmp;
        }
```

```
    }
}
class inserttionsort:dayso,sort
{
   public void sapxep()
   {
      int i,pos,k;
      for (i = 0; i < n; i++)
      {
         k = a[i]; pos = i-1;
         while ( pos >=0 && a[pos] > k)
        {
            a[pos + 1] = a[pos];
            pos--;
         }
         a[pos + 1] = k;
      }
   }
}
class Tester
{
   static void Main()
   {
      //sort a = new quicksort();
      //selectsort a = new selectsort();
      inserttionsort a = new inserttionsort();
      a.nhap(); Console.Clear();
      Console.WriteLine("Day truoc khi sap xep :");
      a.print();
      Console.WriteLine("Day sau khi sap xep   :");
      a.sapxep();
      a.print();
      Console.ReadKey();
   }
}
```

Kết quả sau khi chạy chương trình:

```
Dãy trước khi sắp xếp:
2    5    3    2
Dãy sau khi sắp xếp:
2    2    3    5
```

Lưu ý: Ta có thể phủ quyết tất cả các phương thức của giao diện trong lớp thực thi cũng có thể chỉ phủ quyết một vài hoặc không phủ quyết một phương thức nào.

IV. Xây dựng lớp thực thi giao diện một cách tường minh

Ví dụ: Xây dựng lớp Document, thực thi hai giao diện Italk, Istorable . Giao diện Italk có phương thức Read() và Italk. Giao diện Istorable có phương thức Read() và Write().

a, Hướng dẫn:

Bài toán gồm có :1 lớp Class Document, và 2 giao diện Italk, Istorable.

Ta thấy lớp Document thực thi hai giao diện, và cả hai giao diện trên đều có phương thức Read (). Vậy làm thế nào để máy tính cũng như bạn đọc có thể biết rằng lớp Document thi hành phương thức Read() là của giao diện nào. Để giải quyết vấn đề này bằng cách thực hiện giao diện một cách tường minh.

Xây dựng các giao diện và lớp

```
interface Istorable
{
    void Read();
    void Write();
}
interface Italk
{
    void Read();
    void Italk();
```

```
    }
     class Document:Italk,       Istorable
    {
          public Document(string s)
          void Istorable.Read() // Cài đặt phương thức Read() của Istorable
          public void Write()
          puclic void Italk()
          void Italk.Read() // Cài đặt phương thức Read() của ITalk
    }
```

Lưu ý: Chúng ta phải ép kiểu thành giao diện để gọi Phương thức Read() của Istorable và Italk. Thực thi giao diện một cách tường minh cho phương thức Read() bằng cách gọi cả tên của giao diện.

b, Bài giải mẫu:

```
using System;
namespace bt
{
  interface Istorable
  {
     void Read();
     void Write();
  }
   interface ITalk
   {
      void ITalk();
      void Read();
   }
class Document:Istorable,ITalk
   {
      public Document(string s)
      {
         Console.WriteLine("Tao van ban {0}",s);
      }
      void Istorable.Read()
      {
```

```
            Console.WriteLine("Thuc thi Istorable.Read()");
        }
        public void Write()
        {
            Console.WriteLine("Thuc thi Istorable.Write()");
        }
        void ITalk.Read()
        {
            Console.WriteLine("Thuc thi ITalk.Read()");
        }
        public void ITalk()
        {
            Console.WriteLine("Thuc thi ITalk.ITalk");
        }
    }
    class tester
    {
        static void Main(string[] args)
        {
            Document thedoc = new Document("Bai tap C#");
            Istorable isdoc = thedoc as Istorable;
            if (isdoc != null)
            {
                isdoc.Read();
            }
            ITalk itdoc = thedoc as ITalk;
            if (itdoc != null)
            {
                itdoc.Read();
            }
            thedoc.Write();
            thedoc.ITalk();
            Console.ReadLine();
        }
    }
```

```
}
```

Kết quả sau khi chạy chương trình:

```
Tạo văn bản  Bài tập C#
Thực thi Istorable.Read()
Thực thi Italk.Read()
Thực thi Istorable.Write()
Thực thi Italk.ITalk
```

C. CÂU HỎI VÀ TRẢ LỜI

Câu 1: So sánh giữa lớp và giao diện?

Trả lời: Giao diện khác với lớp ở một số điểm sau: giao diện không cung cấp bất cứ sự thực thi mã nguồn nào cả. Điều này sẽ được thực hiện tại các lớp thực thi giao diện. Một giao diện đưa ra chỉ để nói rằng có cung cấp một số sự xác nhận hướng dẫn cho những điều gì đó xảy ra và không đi vào chi tiết. Một điều khác nữa là tất cả các thành viên của giao diện được giả sử là public ngầm định. Nếu chúng ta cố thay đổi thuộc tính truy cập của thành viên trong giao diện thì sẽ nhận được lỗi.

Giao diện chỉ chứa những phương thức, thuộc tính, sự kiện, chỉ mục. Và không chứa dữ liệu thành viên, bộ khởi dựng, và bộ hủy. Chúng cũng không chứa bất cứ thành viên static nào cả.

Câu 2: Sự khác nhau giữa giao diện và lớp trừu tượng?

Trả lời: Sự khác nhau cơ bản là sự kế thừa. Một lớp có thể kế thừa nhiều giao diện cùng một lúc, nhưng không thể kế thừa nhiều hơn một lớp trừu tượng.

Câu 3: Các lớp thực thi giao diện sẽ phải làm gì?

Trả lời: Các lớp thực thi giao diện phải cung cấp các phần thực thi chi tiết cho các phương thức, thuộc tính, chỉ mục, sự kiện được khai báo trong giao diện.

Câu 4: Có bao nhiêu cách gọi một phương thức được khai báo trong giao diện?

Trả lời: Có 4 cách gọi phương thức được khai báo trong giao diện:

- Thông qua lớp cơ sở tham chiếu đến đối tượng của lớp dẫn xuất
- Thông qua một giao diện tạo từ lớp cơ sở tham chiếu đến đối tượng dẫn xuất
- Thông qua một đối tượng dẫn xuất
- Thông qua giao diện tạo từ đối tượng dẫn xuất

Câu 5: Các thành viên của giao diện có thể có những thuộc tính truy cập nào?

Trả lời: Mặc định các thành viên của giao diện là public. Vì mục tiêu của giao diện là xây dựng cho các lớp khác sử dụng. Nếu chúng ta thay đổi thuộc tính này như là internal, protected hay private thì sẽ gây ra lỗi.

Câu 6: Chúng ta có thể tạo thể hiện của giao diện một cách trực tiếp được không?

Trả lời: Không thể tạo thể hiện của giao diện trực tiếp bằng khai báo new được. Chúng ta chỉ có thể tạo thể hiện giao diện thông qua một phép gán với đối tượng thực thi giao diện.

D. CÂU HỎI VÀ BÀI TẬP TỰ LÀM

Câu hỏi:

Câu 1: Toán tử is được dùng làm gì trong giao diện?

Câu 2: Toán tử as có lợi hơn toán tử is về mặt nào khi được sử dụng liện quan đến giao diện ?

Câu 3: Giao diện là kiểu dữ liệu tham chiếu hay kiểu giá trị?

Câu 4: Khi thực thi giao diện với cấu trúc. Thì truy cập các thành viên của giao diện thông qua đối tượng hay thông qua tham chiếu giao diện là tốt nhất?

Câu 5: Số giao diện có thể được kế thừa cho một lớp?

Câu 6: Việc thực thi giao diện tường minh là thực thi như thế nào? Trong trường hợp nào thì cần thực hiện tường minh?

Bài tập:

Bài 1. Trong giao diện sau sai ở đâu?

```
public interface House
```

```
{
   @Deprecated
   void open();
   void openFrontDoor();
   void openBackDoor();
}
```

Bài 2. Hoàn thành chương trình sau:

```
interface Closable
{
   void close();
}
class File implements Closable
 {
   @Override
   public void close()
    {
      //... close this file...
   }
}
```

Điều gì sẽ xảy ra? Bạn có thể giải thích tại sao?

Bài 3: Hãy viết một giao diện khai báo một thuộc tính ID chứa chuỗi giá trị. Viết một lớpEmployee thực thi giao diện đó.

Bài 4: Đoạn mã nguồn sau đây có lỗi hãy sử lỗi và hãy cho biết tại sao có lỗi này. Sau khi sửa lỗi hãy viết một lớp Circle thực thi giao diện này?

```
public interface osi
   {
      long with;
      long height;
      double area();
      int side();
   }
```

Bài 5: Chương trình sau đây có lỗi hãy sửa lỗi, biên dịch và chạy lại chương trình. Hãy giải thích tại sao chương trình lại lỗi.

```
using system;
interface Ipoint
   {
      int x
      {
         get;
         set;
      }
      int y
      {
         get;
         set;
      }
   }
   class MyPoint
   {
      private int myX;
      private int myY;
      public MyPoint(int x,int y)
      {
         myX=x;
         myY=y;
      }
      public int x
      {
         get { return myX ;}
         set { myX=value ;}
      }
      public int y
      {
         get
           {return myY;}
         set
       {myY=value ;}
      }
```

```
}
class tester
{
    private static void printpoint(Ipoint p)
    {
        Console.WriteLine("x={0},y={1}", p.x, p.y);
    }
    static void Main()
    {
        MyPoint p = new MyPoint(2, 3);
        Console.Write("My point ");
        printpoint(p);
        Ipoint p2 = new Ipoint();
        printpoint(p2);
        Console.ReadLine();
    }
}
```

Bài 6*:* Trong các cách khai báo sau cách khai báo giao diện nào là đúng. Cách khai báo nào sai thì hãy sửa lại cho đúng.

```
interface A
{
    void print () {};
}
interface B
{
    void print();
}
abstract interface C
{
    print();
}
abstract interface D
{
    abstract void print() {}
```

```
}
```

Bài 7: Xây dựng một giao diện IDisplay có khai báo thuộc tính Name kiểu chuỗi. Hãy viết hai lớp Dog và Cat thực thi giao diện IDisplay, cho biết thuộc tính Name là tên của đối tượng.