Chương 3: Lập Trình Hướng Đối Tượng với C#

Gv: Đặng Hữu Nghị

3.1. Đối tượng và lớp

- 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

}

3.1. Đối tượng và lớp

- Các thuộc tính truy cập gồm có các từ khóa sau đây: public, private, internal, protected, internal protected
  - **public :** Không có giới hạn, có thể truy xuất mọi nơi trong bản thân lớp khai báo và bên ngoài
  - **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

3.1. Đối tượng và lớp

- **internal**: giới hạn truy cập trong cùng một project
- **protected internal**: giới hạn truy cập trong project hoặc lớp dẫn xuất

3.1. Đối tượng và lớp

- Trong C#, cho phép chúng ta khai báo các class lồng nhau.

```
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
        }
}
```

3.1. Đối tượng và lớp

- Để sử dụng lớp ta phải khai báo đối tượng của lớp đó
- 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])

3.1. Đối tượng và lớp

- Ví dụ:

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.

- 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

```
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 :");
        Console.Write("X = ");
        x = int.Parse(Console.ReadLine());
        Console.Write("Y = ");
        y = int.Parse(Console.ReadLine());
    }
}
```

```
class Program
    {
        static void Main(string[] args)
        {
            diem d = new diem();
// bien trong C# luon doc khoi gan gia tri truoc khi su dung
            d.nhap();
            Console.Write("diem b ");
            d.hien();
            Console.WriteLine("toa do doi xung la:");
            d.chuyen();
            d.hien();
            Console.WriteLine("toa do sau di chuyen la:");
            d.move(2, 6);
            d.hien();
            Console.ReadLine();
        }
    }
}
```

```
namespace C3_Bai1
{
   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:");
           Console.Write("X = ");
            x = int.Parse(Console.ReadLine());
           Console.Write("Y = ");
                    y = int.Parse(Console.ReadLine());
     } }
```

```
class Program
   {
      static void Main(string[] args)
      {
         diem d = new diem();
// bien trong C# luon doc khoi gan gia tri truoc khi su dung
         d.nhap();
         Console.Write("diem b ");
         d.hien();
         Console.WriteLine("toa do doi xung la:");
         d.chuyen();
         d.hien();
         Console.WriteLine("toa do sau di chuyen la:");
         d.move(2, 6);
         d.hien();
         Console.ReadLine();
      }
   }
}
```

Chương trình hoàn thiện

3.1. Đối tượng và lớp

- **Từ khóa this**
  - Từ khóa this dùng để tham chiếu đến chính bản thân của đối tượng đó.
  - Vi 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;
}
```

3.2. Method - Phương thức

Mẫu tổng quát của phương thức

<phạm vi> <Kiểu dữ liệu> TênPhươngThức([tham số])

{ <Các lệnh> }

Phạm vi

- Xác định phạm vi hay cách phương thức được gọi (sử dụng)
- Các từ khoá phạm vi : private, public, static, internal

2.5. Method - Phương thức

Kiểu dữ liệu của phương thức (đầu ra), gồm 2 loại:

- void: Không trả về giá trị
- float / int / long / string / kiểu cấu trúc / … : Trả về giá trị có kiểu dữ liệu tương ứng với kết quả xử lý

2.5. Method - Phương thức

- Tên phương thức : Đặt tên theo qui ước sao cho phản ánh đúng chức năng thực hiện của phương thức
- Danh sách các tham số (nếu có) : đầu vào của phương thức (trong một số trường hợp có thể là đầu vào và đầu ra của phương thức nếu kết quả đầu ra có nhiều giá trị - Tham số này gọi là tham chiếu)

3.2. Method - Phương thức

Khi hàm xử lý biến toàn cục thì không cần tham số

```
static int a, b;
static void Nhap()
{
	Console.Write("Nhap a: ");
	a = int.Parse(Console.ReadLine());
	Console.Write("Nhap b: ");
	b = int.Parse(Console.ReadLine());
}
static void Xuat()
{
	Console.WriteLine("a = {0}; b = {1}", a, b);
}
static void Main(string[] args)
{
	Nhap();
	Xuat();
}
```

3.2. Method - Phương thức

Phương thức không trả về giá trị

static void <TênPhươngThức> ([danh sách các tham số])

{

Khai báo các biến cục bộ

Các câu lệnh hay lời gọi đến phương thức khác.

}

- Gọi hàm: <TênPhươngThức>(danh sách tên các đối số);
- Những phương thức loại này thường rơi vào những nhóm chức năng: Nhập / xuất dữ liệu, thống kê, sắp xếp, liệt kê

3.2. Method - Phương thức

- Ví dụ: Viết chương trình nhập số nguyên dương n và in ra màn hình các ước số của n

```
static void LietKeUocSo(uint n)
{
  for (int i = 1; i <= n; i++)
      if (n % i == 0)
            Console.Write("{0}\t", i);
}
static void Main(string[] args)
{
  uint n;
  Console.Write("Nhap so nguyen duong n: ");
  n=uint.Parse(Console.ReadLine());
  Console.Write("Cac uoc so cua {0}: ", n);
  LietKeUocSo(n);
  Console.ReadLine();
}
```

3.2. Method - Phương thức

Phương thức có trả về kết quả

static <KDL> <TênPhươngThức> ([tham số])
{
<KDL> kq;
Khai báo các biến cục bộ
Các câu lệnh hay lời gọi đến phương thức khác.
return kq;
}

- Gọi hàm:

  <KDL> Tên biến = TênPhươngThức(tên các đối số);

- Những phương thức này thường rơi vào các nhóm: *Tính tổng, tích, trung bình, đếm, kiểm tra, tìm kiếm*

3.2. Method - Phương thức

- Viết chương trình nhập số nguyên dương n và tính. Tính tổng các số từ 1 đến n

```
static ulong TongS(uint n)
{
    ulong kq = 0;
    for (uint i = 1; i <= n; i++)
          kq + = i;
    return kq;
}
static void Main(string[] args)
{
    ulong S;
    uint n;
     Console.Write("Nhap vao so nguyen n: ");
     n = uint.Parse(Console.ReadLine());
     S = TongS(n);
     Console.Write("Tong tu 1 den n: " + S);
     Console.ReadLine();
}
```

3.2. Method - Phương thức

Tham số là tham chiếu

- Tham số lưu kết quả xử lý của hàm: out
  - thường dùng cho trường hợp nhập dữ liệu, kết quả hàm có nhiều giá trị
- Tham số vừa làm đầu vào và đầu ra: ref
- Dùng từ khóa ref hoặc out trước kiểu dữ liệu của khai báo tham số và trước tên đối số khi gọi phương thức.

3.2. Method - Phương thức

Tham số là tham chiếu

- Dùng từ khóa **ref** bắt buộc phải khởi gán giá trị ban đầu cho đối số trước khi truyền vào khi gọi phương thức
- Nếu dùng **out** thì không cần thiết

3.2. Method - Phương thức

Ví dụ: Hoán vị 2 số nguyên a, b cho trước. Đánh giá kết quả khi viết chương trình với hai trường hợp sau

1. Trường hợp không dùng tham chiếu
2. Trường hợp dùng tham chiếu: ref

3.2. Method - Phương thức

- **Không dùng tham chiếu**

```
static void HoanVi(int a, int b)
{
    int tam = a;
   a = b;
   b = tam;
   Console.WriteLine("Trong HoanVi: a = " + a + ";b = " + b);
}
static void Main(string[] args)
{
    int a = 5, b = 21;
   Console.WriteLine("Truoc HoanVi: a = {0}; b = {1}", a, b);
   HoanVi(a, b);
   Console.WriteLine("Sau HoanVi: a = " + a + ";b = " + b);
}
```

3.2. Method - Phương thức

- **Dùng tham chiếu**

```
static void HoanVi(ref int a, ref int b)
{
    int tam = a;
   a = b;
   b = tam;
   Console.WriteLine("Trong HoanVi: a = " + a + ";b = " + b);
}
static void Main(string[] args)
{
    int a = 5, b = 21;
   Console.WriteLine("Truoc HoanVi: a = {0}; b = {1}", a, b);
   HoanVi(ref a, ref b);
   Console.WriteLine("Sau HoanVi: a = " + a + ";b = " + b);
}
```

3.2. Method - Phương thức

- **Sử dụng tham chiếu out**

```
static void Nhap(out int a, out int b)
{
        Console.Write("Nhap a: ");
        a = int.Parse(Console.ReadLine());
        Console.Write("Nhap b: ");
        b = int.Parse(Console.ReadLine());
}
static int Tong(int a, int b)
{
        return a + b;
}
static void Main(string[] args)
{
        int a, b;
        Nhap(out a, out b);
        s=Tinh(a, b);
        Console.WriteLine(“{0}+{1}={2}”, a, b, s);
}
```

3.3. Phương thức khởi tạo (Constructor)

- Cú pháp:

```
public <Tên lớp> ([ds tham số])
{
        // Khởi tạo cho các thành phần dữ liệu
        của lớp
}
```

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

3.3. **Phương thức khởi tạo (Constructor)**

```
class KhachHang
{
    private int mMaKH;
    private string mTenKH;
    public KhachHang()
    {
      mKH = 0;
      mTenKH = “ABC”;
    }
}
```

3.3. Phương thức khởi tạo (Constructor)

- Constructor có thể có tham số

```
class KhachHang
{
    private int mMaKH;
    private string mTenKH;
    public KhachHang() {
        mKH = 0; mTenKH = “ABC”;
    }
    public KhachHang(int MaKH, string TenKH)
    {
      mKH = MaKH;
      mTenKH = TenKH;
    }
}
```

3.4. 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:*

3.4. Hàm hủy bỏ (destructor)

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

3.5. Thành viên tĩnh

- Bình thường các thuộc tính, phương thức sẽ có đặc điểm:
  - Chỉ có thể sử dụng sau khi khởi tạo đối tượng.
  - Dữ liệu thuộc về riêng mỗi đối tượng
  - Được gọi thông qua tên của đối tượng.

3.5. Thành viên tĩnh

- Đặc điểm của thành viên tĩnh:
  - Được khởi tạo **1 lần duy nhất** ngay khi biên dịch chương trình.
  - Có thể **dùng chung** cho mọi đối tượng.
  - **Được gọi thông qua tên lớp**.
  - Được **huỷ khi kết thúc** chương trình.

3.5. Thành viên tĩnh

- Có 4 loại thành viên tĩnh chính:
  - Biến tĩnh (static variable).
  - Phương thức tĩnh (static method).
  - Lớp tĩnh (static class).
  - Phương thức khởi tạo tĩnh (static constructor).
- Để khai báo 1 **thành viên tĩnh** ta sử dụng từ khoá **static** đặt trước tên biên, tên phương thức hoặc tên lớp.

3.5.1. Biến tĩnh

- Cú pháp:

<phạm vi truy cập> static <kiểu dữ liệu> <tên biến> = <giá trị khởi tạo>;

- **Biến tĩnh** là:
  - Một biến dùng chung cho mọi đối tượng thuộc lớp.
  - Nó được khởi tạo vùng nhớ 1 lần duy nhất ngay khi chương trình được nạp vào bộ nhớ để thực thi và huỷ khi kết thúc chương trình

```
using System;
namespace C3_Bai_2
{
   class Cat
   {
       private int weight;
       public int Weight
       {
          get { return weight; }
          set { weight = value; }
       }
       private int height;
       public int Height
       {
          get { return height; }
          set { height = value; }
       }
       public static int Count = 0;
       public Cat()
       {
          weight = 20;
          height = 500;
          Count++;
       }
   }
```

```
class Program
{
    static void Main(string[] args)
    {
       Console.WriteLine(" So luong meo ban dau: " + Cat.Count);
       Cat BlackCat = new Cat(); // Tạo ra con mèo đầu tiên

       Console.WriteLine(" So luong meo hien tai: " + Cat.Count);

       Cat WhiteCat = new Cat(); / / Tạo ra con mèo thứ 2
       Console.WriteLine(" So luong meo hien tai: " + Cat.Count);
       Console.ReadKey();
    }
 }
}
```

3.5.2. Phương thức tĩnh

- Cú pháp:

<phạm vi truy cập> static <kiểu trả về> <tên phương thức>
{
// nội dung phương thức
}

- Phương thức (Hàm) tĩnh được sử dụng với 2 mục đích chính:
  - Hàm tĩnh là 1 hàm dùng chung của lớp. Được gọi thông qua tên lớp và không cần khởi tạo bất kỳ đối tượng nào, từ đó tránh việc lãng phí bộ nhớ.
  - Hỗ trợ trong việc viết các hàm tiện ích để sử dụng lại.

Ví dụ: Viết 1 hàm tiện ích đó là tính luỹ thừa của 1 số nguyên để hỗ trợ tính toán

3.5.2. Phương thức tĩnh

```
class TienIch
{
        public static long LuyThua(int CoSo, int SoMu)
        {
            long KetQua = 1;
            for (int i = 0; i < SoMu; i++)
            {
                KetQua *= CoSo;
            }

            return KetQua;
        }
    }
```

- Gọi phương thức thông qua tên lớp và không cần khởi tạo đối tượng.

```
Console.WriteLine(TienIch.LuyThua(2, 3));
```

3.5.3. Lớp tĩnh

- Cú pháp

```
<phạm vi truy cập> static class <tên lớp>
{
        // các thành phần của lớp
}
```

- Đặc điểm:
  - Chỉ chứa các thành phần tĩnh (biến tĩnh, phương thức tĩnh).
  - **Không** thể khai báo, khởi tạo 1 đối tượng thuộc lớp tĩnh.

3.5.3. Lớp tĩnh

- Lớp tĩnh thường được dùng với mục đích khai báo 1 lớp tiện ích chứa các hàm tiện ích hoặc hằng số vì:
  - Ràng buộc các thành phần bên trong lớp phải là **static**.
  - Không cho phép tạo ra các đối tượng dư thừa làm lãng phí bộ nhớ.
  - Mọi thứ đều được truy cập thông qua tên lớp.

3.5.3. Lớp tĩnh

- Trong C# có rất nhiều lớp tiện ích sử dụng lớp tĩnh, phương thức tĩnh để khai báo. Ví dụ điển hình đó là lớp **Math**.
- Lớp **Math** chứa:
  - Các hằng số như **PI**, **E**.
  - Các phương thức hỗ trợ tính toán như: **sin**, **cos**, **tan**, **sqrt**, **exp**, . . .

3.6. 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 dư liệu khác nhau.
- Việc phân biệt các hàm này dựa vào:
  - Số lượng tham số
  - kiểu dư liệu của tham số, kiểu dư liệu trả về của phương thức.

3.6. Nạp chồng phương thức

```
public void In()
{
        // Các câu lệnh
}

public void In(string s)
{
        // Các câu lệnh
}

public void In(int s)
{
        // Các câu lệnh
}
```

3.6. Nạp chồng phương thức

- Ví dụ:
  - Xây dựng lớp Số phức gồm:
    - Các thuộc tính: thuc và ao là 2 số thực
    - Các phương thức:
      - Phương thức khởi tạo không tham số để khởi tạo giá trị 0 cho phần thực và phần ảo
      - Phương thức khởi tạo 2 tham số để khởi tạo giá trị cho phần thực và phần ảo
      - Hàm Cong() để cộng 1 số phức với 1 số phức
      - Hàm Cong() để cộng 1 số phức với 1 số thực
      - Hàm In() để in 1 số phức theo dạng (thuc, ao)
  - Sau đó cho khai báo và khởi tạo 2 số phức x, y và nhập vào 1 số thực t. Tính và in ra x+y, x+t

```
using System;

namespace Vidu
{
        class Sophuc
        {
                public double thuc, ao;
                public Sophuc()
                {
                        thuc = 0;ao = 0;
                }
                public Sophuc(double t,double a)
                {
                        thuc = t; ao = a;
                }
```

```
        public static Sophuc Cong(Sophuc a, Sophuc b)
        {
                Sophuc t = new Sophuc();
                t.thuc = a.thuc + b.thuc;
                t.ao = a.ao + b.ao;
                return t;
        }
        public static Sophuc Cong(Sophuc a, double b)
        {
                Sophuc t = new Sophuc();
                t.thuc = a.thuc + b;
                t.ao = a.ao;
                return t;
        }
        public void In()
        {
                Console.Write("(" + thuc + ", " + ao + ")");
        }
} //Kết thúc định nghĩa lớp Sophuc
```

```
class Program
{
        static void Main(string[] args)
        {
                Sophuc x = new Sophuc(1, 2);
                Sophuc y = new Sophuc(3, 4);
                Sophuc z = new Sophuc();
                z = Sophuc.Cong(x, y);
                Console.Write("Tong cua hai so phuc "); x.In();
                Console.Write(" va ");y.In(); Console.Write(" La: "); z.In();
                Console.WriteLine();
```

```
                double t;
                Nhaplai:
                try
                {
                        Console.Write("Nhap vao mot so thuc: ");
                        t = double.Parse(Console.ReadLine());
                }
                catch
                {
                        Console.WriteLine("Ban nhap sai dinh dang");
                        goto Nhaplai;
                }
                z = Sophuc.Cong(x, t);
                Console.Write("\nTong cua so phuc "); x.In();
                Console.Write(" va so thuc " + t + " la: "); z.In();
                Console.ReadKey();
        }
    }
}
```

3.7. 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**

3.7. Đóng gói dữ liệu thông qua các thuộc tính

public <kiểu dữ liệu>(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 dư liệu là che giấu thông tin mặc dù người sử dụng vẫn thao tác với thuộc tính

Ví dụ 1

```
using System;
namespace hoten
{
        public class tentoi

            public string ho;
            private string hoten;
            public string ten
            {
                        get { return hoten; }
                        set { hoten = value; }
            }
        }
        class Program
        {
            static void Main(string[] args)
            {
                        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();
            }
        } }}
```

```
public string ten
 {
            get ;
            set ;
 }
```

Ví dụ 2

```
int  tuoi = 0;
public int Tuoi
{
    set
    {
        tuoi = value;
       if (_tuoi < 0)
          tuoi = 0;//Không có tuổi nhỏ hơn 0
    }
    get
    {
       return  tuoi;
    }
}
this.Tuoi = -8; //Thực hiện set => _tuoi = 0;
int layTuoi = this.Tuoi;//thực hiện get => trả về giá trị _tuoi (0)
```

3.8. Nạp chồng toán tử trên lớp

- 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
- Hàm toán tử bao gồm từ khóa **operator** theo sau là ki hiệu của toán tử được định nghĩa chồng

public static <kiểu dữ liệu> operator<Toán tử>([<danh sách tham số>])

Ví dụ:

```
public static Sophuc operator +(Sophuc a, Sophuc b)
 {
        Sophuc t = new Sophuc();
        t.thuc = a.thuc + b.thuc;
        t.ao = a.ao + b.ao;
        return t;
 }
 public static Sophuc operator+(Sophuc a, double b)
 {
        Sophuc t = new Sophuc();
        t.thuc = a.thuc + b;
        t.ao = a.ao;
        return t;
 }
```

3.8. Nạp chồng toán tử trên lớp

```
public class Point
{
        public int x;
        public int y;
        public Point (){ }
        public Point(int xx,int yy)
        {
                x = xx ;
                y = yy;
        }
        public static Point operator + (Point p1, Point p2) {
                Point result = new Point();
                result.x = p1.x  + p2.y;
                result.y = p1.x  + p2.y;
                return result;
        }
}
```

```
using System;

namespace NapChongToanTu
{
        class Program
        {
                static void Main(string[] args)
                {
                        Phanso ps1 = new Phanso(4, 9);
                        Phanso ps2 = new Phanso(7, 5);
                        Phanso ps3 = ps1 + ps2;
                        ps1.InPhanso();
                        Console.Write(" + ");
                        ps2.InPhanso();
                        Console.Write(" = ");
                        ps3.InPhanso();
                        Console.ReadKey();
                }
        }
```

```
class Phanso
  {
      private int tu, mau;
      public Phanso(int tuso,int mauso)
      { tu = tuso;mau = mauso;   }
      public static Phanso operator + (Phanso ps1, Phanso ps2)
      {
          int ts, ms;
          ts = ps1.tu * ps2.mau + ps1.mau + ps2.tu;
          ms = ps1.mau * ps2.mau;
          Toigian(ref ts, ref ms);
          return new Phanso(ts, ms);
      }
      private static void Toigian(ref int ts, ref int ms)
      {
          int t = ts; int m = ms;
          while (t != m)
                                  if (t > m)  t -= m;
                                  else        m -= t;
          ts = ts / t; ms = ms / t;
      }
     public void InPhanso()
      {  Console.Write(tu + "/" + mau);  }
   }}
```

3.9. Thừa kế

- Cú pháp để tạo lớp kế thừa trong C# là:

[<Quyền truy cập>] class <Tên lớp cha>
{
  ...
}
[<Quyền truy cập>] class <Tên lớp con> : <Tên lớp cha>
{
  ...
}

<Tên lớp cha>: lớp cơ sở - Base class

<Tên lớp con>: lớp dẫn xuất – Derived class

```
using System;
namespace C3_KeThua02
{
    class Chunhat
    {
        protected float dai, rong;
        public Chunhat(float d = 0, float r = 0)
        { dai = d; rong = r;}
        public float Dientich()
        { return dai * rong; }
    }
    class Vuong : Chunhat
    {
        public Vuong(float s)
        { dai = rong = s;}
    }
    class Program
    {
        static void Main(string[] args)
        {
            Chunhat cn = new Chunhat(4, 3);
            Vuong v = new Vuong(5);
            Console.WriteLine("Dien tich hinh chu nhat la: " + cn.Dientich());
            Console.WriteLine("Dien tich hinh vuong la: " + v.Dientich());
            Console.ReadKey();
        } }}
```

3.9. Thừa kế

- **Từ khoá Base:**

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

Ví dụ:

```
public class nhan_vien
{
    string ho_ten;
    int tuoi;
    //Phương thức khởi tạo không có tham số
    public nhan_vien(){
        ho_ten = "";
        tuoi = 0;
    }
    //Phương thức khởi tạo có tham số
    public nhan_vien(string pho_ten, int ptuoi){
        ho_ten = pho_ten;
        tuoi = ptuoi;
    }
}
```

3.9. Thừa kế

```
public class nv_van_phong : nhan_vien
{
    int so_ngay_vang;
    public nv_van_phong(string photen,int ptuoi,int pngayvang) : base(photen,ptuoi)
   {
        so_ngay_vang = pngayvang;
    }
}
```

3.9. Thừa kế

- **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

3.9. Thừa kế

Vi dụ 1:

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

3.9. Thừa kế

Vi dụ 2:

```
class Animal
{
        public Animal()
        {
                Console.WriteLine("Animal constructor");
        }
        public void Talk()
        {
                Console.WriteLine("Animal talk");
        }
}
```

3.9. Thừa kế

```
class Dog : Animal
 {
        public Dog() {
                Console.WriteLine("Dog constructor");
                }
        public new void Talk()
                {
                        Console.WriteLine("Dog talk");
                }
  }
```

3.9. Thừa kế

```
class Program
 {
     static void Main(string[] args)
                 {
                         Animal a1 = new Animal();
                         a1.Talk();
                         Dog d1 = new Dog();
                         d1.Talk();
                 }
 }
```

3.9. Thừa kế

- Ví dụ:

```
using System;
namespace C3_BaseNew
{
    class A
    {
        protected int x;
        public A()
        {  x = 3;  }
        public A(int x)
        {  this.x = x;   }
        public void nhap()
        {
            Console.Write("Nhap gia tri cua x: ");
            x = int.Parse(Console.ReadLine());
        }
        public void hien()
        { Console.WriteLine("Gia tri cua x la {0}", x); }
    }
```

3.9. Thừa kế

```
class B: A
   {
      int y;
      public B():base()
      {    y = 6;  }
      public B(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 la {0}", y);
          Console.WriteLine("Tong cua x + y la {0}", x + y);}
   }
```

3.9. Thừa kế

```
class Program
{
    static void Main(string[] args)
    {
        B a = new B();
        a.hien();
        B b = new B(10, 15);
        b.hien();
        B c = new B();
        c.nhap();
        c.hien();
        Console.ReadKey();

    }
  }
}
```

3.10. 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
  - Vi dụ: Khai báo phương thức đa hình

    public **virtual** float dientich()

3.10. Tính đa hình

- **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*.
  - Vi dụ:

```
class hinh
{
        public virtual float dientich()
}
class hinhcn : hinh
{
        public override float dientich() // phương thức ghi đè
}
```

3.10. Tính đa hình

- **Ví dụ**: Xây dựng một lớp hình, kế thừa lớp hình đó để tính diện tích cho các hình: Hình tròn, hình chữ nhật
  - Ta thấy: hình chư nhật, hình tròn… đều có chung một phương thứclà tính diện tích. Nhưng cách tích 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 tinh 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

3.10. Tính đa hình

```
using System;

namespace C3_DaHinh01
{
    class Hinh
    {
        public virtual double Dientich()
        {
            return Dientich();
        }
    }
```

3.10. Tính đa hình

```
class Hinhtron : Hinh
    {
        private float r;
        public Hinhtron(float r)
        {
            this.r = r;
        }
        public override double Dientich()
        {
            return r * r * 3.14;
        }
    }
```

3.10. Tính đa hình

```
class Hinhcn : Hinh
    {
        private float cd, cr;
        public Hinhcn(float cd, float cr)
        {
            this.cd = cd;
            this.cr = cr;
        }
        public override double Dientich()
        {
            return cd * cr;
        }
    }
```

3.10. Tính đa hình

```
class Program
    {
        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.Dientich());
            H = hcn;
            Console.WriteLine("Dien tich hinh chu nhat= {0}",H.Dientich());
            Console.ReadLine();
        }
    }
}
```

3.11. 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

3.11. Lớp trừu tượng

- 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
}
```

3.11. Lớp trừu tượng

- **Sự khác nhau giữa phương thức đa hình với phương thức trừu tượng:**
  - Giống nhau: đều sử dụng từ khoá override
  - Khác nhau:
    - Phương thức đa hình (Virtual method) phần thân được định nghĩa tổng quát, ở lớp dẫn xuất có thể thay đổi phương thức đó.
    - Phương thức trừu tượng (Abstract method) phần thân không được định nghĩa và nó được cài đặt trong phương thức của lớp dẫn xuất.

3.11. Lớp trừu tượng

```
abstract class Shape
{
        // Khai cac field
        protected float m_Height = 5;
        protected float m_Width = 10;
        //Khai bao cac method
        public abstract void Dientich();
        public abstract void Chuvi();
        public  void PrintHeight()
        {
                Console.WriteLine("Height = {0}",m_Height);
        }
        public void PrintWidth()
        {
                Console.WriteLine("Width = {0}",m_Width);
        }
}
```

3.11. Lớp trừu tượng

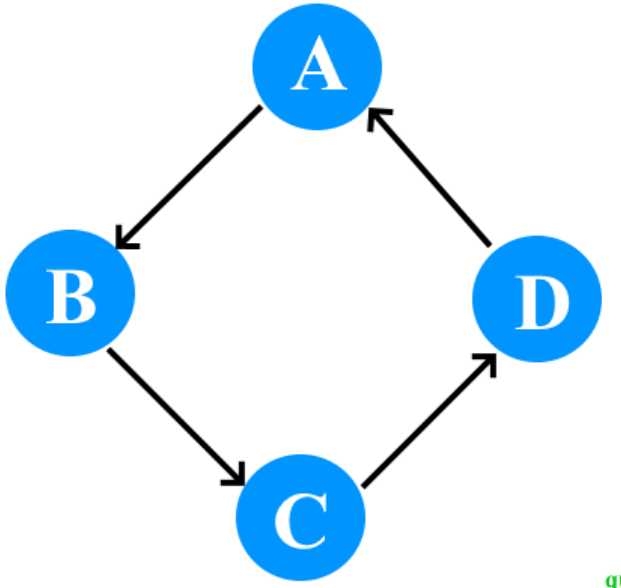
```
class Rectangle:Shape
{
        public Rectangle( ){
                m_Height = 20;
                m_Width = 30;
        }
        public override void Dientich() {
          Console.WriteLine(“Dien tich: {0}",m_Height * m_Width );
        }
        public override void Chuvi() {
        Console.WriteLine(“Chu vi= {0}",(m_Height+m_Width)*2);
        }
}
```

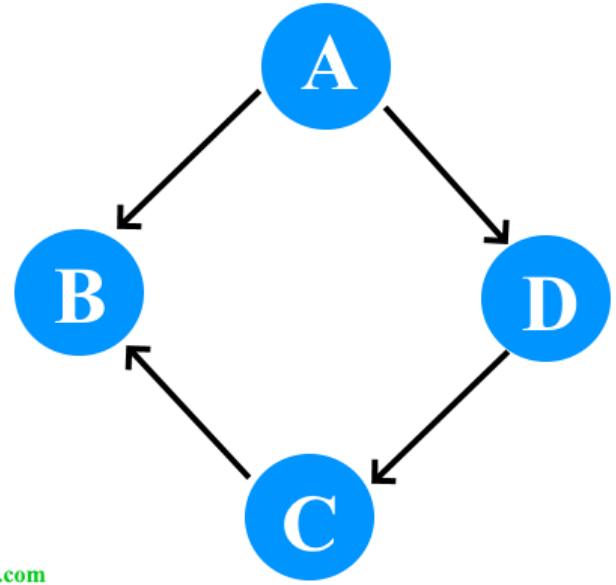
3.11. Lớp trừu tượng

```
class Program
   {
          static void Main(string[] args)
          {
                 Rectangle objRec =  new Rectangle();
                 objRec.Dientich();
                 objRec.Chuvi();
          }
   }
```

3.12. Giao diện – Interface

- **C#** không hỗ trợ đa kế thừa, 1 lớp kế thừa từ nhiều lớp. Tại sao lại như vậy, ta xét thử 2 trường hợp kế thừa dưới đây.
- Giả sử ta có 4 lớp A, B, C, D.





3.12. Giao diện – Interface

- **Trường hợp 1**: Lớp **B** kế thừa từ lớp **A**, lớp **C** kế thừa từ lớp **B**, lớp **D** kế thừa từ lớp **C** nhưng lớp **A** lại kế thừa từ lớp **D**.
- **Trường hợp 2**: Lớp **B** và **D** kế thừa từ lớp **A**, lớp **C** kế thừa từ lớp **D**, lớp **B** kế thừa từ lớp **C**.

3.12. Giao diện – Interface

- Nhưng nếu không hỗ trợ đa kế thừa thì có bị hạn chế gì không, đa kế thừa giúp ích gì?
- Ta xét thử ví dụ sau. Giả sử mình có 2 lớp
  - Động vật trên cạn
  - Động vật dưới nước
  - Tiếp theo mình tạo 1 lớp **Ếch**. Vì ếch là loài lưỡng cư nên nó phải được thừa hưởng thuộc tính và phương thức của cả 2 lớp **Động vật trên cạn** và **Động vật dưới nước**.

3.12. Giao diện – Interface

- Để khắc phục điều này C# đưa ra phương pháp gọi là Giao diện (interface)
- Một lớp có thể kế thừa một lớp khác đồng thời kế thừa từ nhiều interface khác nhau

3.12. Giao diện – Interface

- Không được tạo đối tượng.
- Không thể định nghĩa các phương thức.
- Interface có thể được kế thừa các interface khác.

Ví dụ:

```
namespace Vi_du
{
        interface IDongVatTrenCan
        {
                void Jump();
        }
        interface IDongVatDuoiNuoc
        {
                void Swim();
        }
```

Ví dụ:

```
class Ech : IDongVatTrenCan, IDongVatDuoiNuoc
{
        public void Jump()
        {
                Console.WriteLine("Ech nhay cao 20 cm");
        }
        public void Swim( )
        {
                Console.WriteLine("Ech boi rat xa");
        }
 }
```

Ví dụ:

```
class Program
{
    static void Main(string[] args)
    {
            Ech E1 = new Ech();
            E1.Jump();
            E1.Swim();
            Console.ReadLine();
    }
 }
}
```

3.12. Giao diện – Interface

Ví dụ 2:

```
interface ITest
   {
          void Print();

   }
class Base:ITest
   {
          public void Print()
          {
           Console.WriteLine("Print method called");
          }
  }
```

3.12. Giao diện – Interface

```
static void Main(string[] args)
{
      Base obj = new Base();
      obj.Print();
     //Gọi phương thức Print() bằng  interface  ITest
      ITest ib = (ITest) obj ;
      ib.Print();
    //Gọi phuong thức Print() bằng cách ép kiểu Interface ITest về
    lớp Base
      Base ojB = (Base) ib;
      ojB.Print();
}
```

3.12. Giao diện – Interface

- Ví dụ 3

```
// interface đơn giản với 1 member là method Paint
interface IControl
{
    void Paint();
}

// interface ITextBox kế thừa IControl
// và bổ xung thêm method SetText
interface ITextBox : IControl
{
    void SetText(string text);
}
```

3.12. Giao diện – Interface

```
// interface IListBox cũng kế thừa từ IControl
// bổ xung method riêng SetItems
interface IListBox : IControl
{
    void SetItems(string[] items);
}

// interface IComboBox kế thừa cả hai interface
// các class, struct thực thi interface này cần thực thi
// 3 phương thức Paint, SetText và SetItems
interface IComboBox : ITextBox, IListBox { }
```

3.12. Giao diện – Interface

```
// class TextBox thực thi interface ITextBox
class TextBox : ITextBox
{
    public void Paint() { /* Code thực thi Paint */}
    public void SetText(string text) { /* Code thực thi SetText */}
}
```

3.12. Giao diện – Interface

```
class ComboBox : IComboBox
{

    public void SetText(string text)
    {
        /* Code thực thi SetText */
    }

    public void Paint()
    {
        /* Code thực thi Paint */
    }

    public void SetItems(string[] items)
    {
        /* Code thực thi SetItems */
    }
}
```

3.12. Giao diện – Interface

- Ví dụ 4:

```
using System;

namespace Interface
{
    public interface itfBay //khai báo interface cho việc bay
    {
        void KhaNangBay();
    }
    public interface itfChay //khai báo interface cho việc chạy
    {
        void KhaNangChay();
    }
```

```
public class lopChim : itfBay, itfChay
{
    public void KhaNangBay()
    {
        Console.WriteLine("Chim bay: Bay bằng 2 cánh");
    }
    public void KhaNangChay()
    {
        Console.WriteLine("Chim chạy: Chạy bằng 2 chân");
    }
}
```

```
public class LopMayBay : itfBay, itfChay
{
    public void KhaNangBay()
    {
        Console.WriteLine("Máy bay : Bay bằng động cơ");
    }
    public void KhaNangChay()
    {
        Console.WriteLine("Máy bay chạy : Chạy bằng 2 bánh");
    }
}
```

```
class Program
{
        static void Main(string[] args)
        {
            lopChim objchim = new lopChim();
            objchim.KhaNangBay();
            objchim.KhaNangChay();
            LopMayBay objmaybay = new LopMayBay();
            objmaybay.KhaNangBay();
            objmaybay.KhaNangChay();
            System.Console.ReadLine();
        }
}
}
```

Bài tập

Xây dựng lớp người Nguoi gồm có:

Dữ liệu: họ tên

Phương thức nhập, phương thức ảo in ra, phương thức ảo được khen thưởng

Cài đặt lớp sinh viên SinhVien kế thừa lớp Nguoi và bổ sung:

Dữ liệu: điểm trung bình

Phương thức: định nghĩa lại phương thức nhập, ghi đè phương thức ảo in, phương thức ảo được khen thưởng nếu điểm trung bình từ 9 trở lên

Cài đặt lớp giảng viên GiangVien kế thừa lớp Nguoi và bổ sung:

Dữ liệu: số bài báo

Phương thức: định nghĩa lại phương thức nhập, ghi đè phương thức ảo in, ghi đè phương thức ảo được khen thưởng nếu có số bài báo từ 5 trở lên

Chương trình chính: nhập mảng các n người (n < 100), in ra danh sách này.

```
using System;
namespace Phuong_thuc_ao_01
{
    class Program
    {
        class Nguoi
        {
            protected string hoten;
            public virtual void nhap()
            {
                Console.Write("Nhap ho ten: ");
                hoten = Console.ReadLine();
            }
            public virtual bool Duockhenthuong()
            {
                return Duockhenthuong();
            }
            public virtual void In()
            {
                Console.WriteLine(hoten);
            }

        }
```

```
class Sinhvien : Nguoi
{
        private float diemtb;
        public void nhap()
        {
            base.nhap();
            Console.Write("Nhap diem: ");
            diemtb = float.Parse(Console.ReadLine());
        }
        public override void In()
        {
            base.In();
            Console.WriteLine("Diem trung binh cong {0}", diemtb);
        }
        public override bool Duockhenthuong()
        {
            if (diemtb >= 9)
                return true;
            return false;
        }

}
```

```
class Giangvien : Nguoi
{
        private int sobaibao;
        public void nhap()
        {
            base.nhap();
            Console.Write("Nhap so bai bao: ");
            sobaibao = int.Parse(Console.ReadLine());
        }
        public override void In()
        {
            base.In();
            Console.WriteLine("So bai bao {0}", sobaibao);
        }
        public override bool Duockhenthuong()
        {
            if (sobaibao >= 5)
                return true;
            return false;
        }
}
```

```
static void Main(string[] args)
{
        Nguoi[] ng = new Nguoi[100];   char chon, loai;   int N = 0;

        do
        {
           Console.Write("Ban nhap thong tin giang vien hay sinh vien (s/g):");
           loai = char.Parse(Console.ReadLine());
           if(loai == 's')
           {
              Sinhvien sv = new Sinhvien();   sv.nhap();  ng[N++] = sv;
           }
           else
           {
              Giangvien gv = new Giangvien();  gv.nhap();   ng[N++] = gv;
           }
           Console.Write("Ban co muon nhap tiep hay khong (c/k):");
           chon = char.Parse(Console.ReadLine());
           if (chon == 'k' || chon == 'K' || N > 100) break;
        } while (true);
        Console.WriteLine("Nhung nguoi duoc khen thuong la:");
        for (int i = 0; i < N; i++)
           if(ng[i].Duockhenthuong())   ng[i].In();
        Console.ReadKey();
} }}
```

Bài tập

Xây dựng lớp người NGUOI gồm có:

Dữ liệu: họ tên, mã số, lương

Phương thức ảo nhập, phương thức ảo xuất, phương thức ảo tính lương

Cài đặt lớp người trong biên chế BC kế thừa lớp NGUOI và bổ sung: Dữ liệu: hệ số lương, phụ cấp; Phương thức: định nghĩa lại phương thức nhập và tính lương.

Cài đặt lớp người làm hợp đồng HD kế thừa lớp NGUOI và bổ sung:Dữ liệu: tiền công lao động, số ngày làm việc trong tháng, hệ số vượt giờ; Phương thức: định nghĩa lại phương thức nhập và tính lương.

Chương trình chính: nhập mảng các n người (n < 100), in ra danh sách này.