



LẬP TRÌNH .NET 2

NỘI DUNG BUỔI 2:

- XÂY DỰNG ĐỐI TƯỢNG, LỚP VÀ CÁC THÀNH PHẦN CỦA LỚP
- KẾ THỪA VÀ ĐA HÌNH, LỚP TRỪU TƯỢNG

Sự tiến hoá của các phương pháp lập trình

- **Lập trình không có cấu trúc:** viết tất cả mã lệnh vào 1 hàm **main** duy nhất và chạy

Nhược điểm của phương pháp lập trình ko cấu trúc:

Chỉ sử dụng biến toàn cục dẫn đến rất tốn bộ nhớ.

Vì có những đoạn chương trình cần sử dụng lại nhiều lần nên dẫn đến lạm dụng lệnh goto.

Khó hiểu, khó bảo trì, không thể tái sử dụng.

Khó phát triển các ứng dụng lớn.

- **Lập trình có cấu trúc** (lập trình thủ tục): chia chương trình lớn ra thành các chức năng, mỗi chức năng được đưa vào 1 hàm. Khi cần dùng đến chức năng nào thì ta sẽ gọi hàm tương ứng.

Nhược điểm của phương pháp lập trình cấu trúc:

Dữ liệu và xử lý tách rời.

Khi cấu trúc dữ liệu thay đổi sẽ dẫn đến thuật toán bị thay đổi.

Không tự động khởi tạo, giải phóng dữ liệu động.

Không mô tả được đầy đủ, trung thực hệ thống trong thực tế.

lập trình hướng đối tượng (OOP - Object-Oriented Programming)

KHÁI NIỆM LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

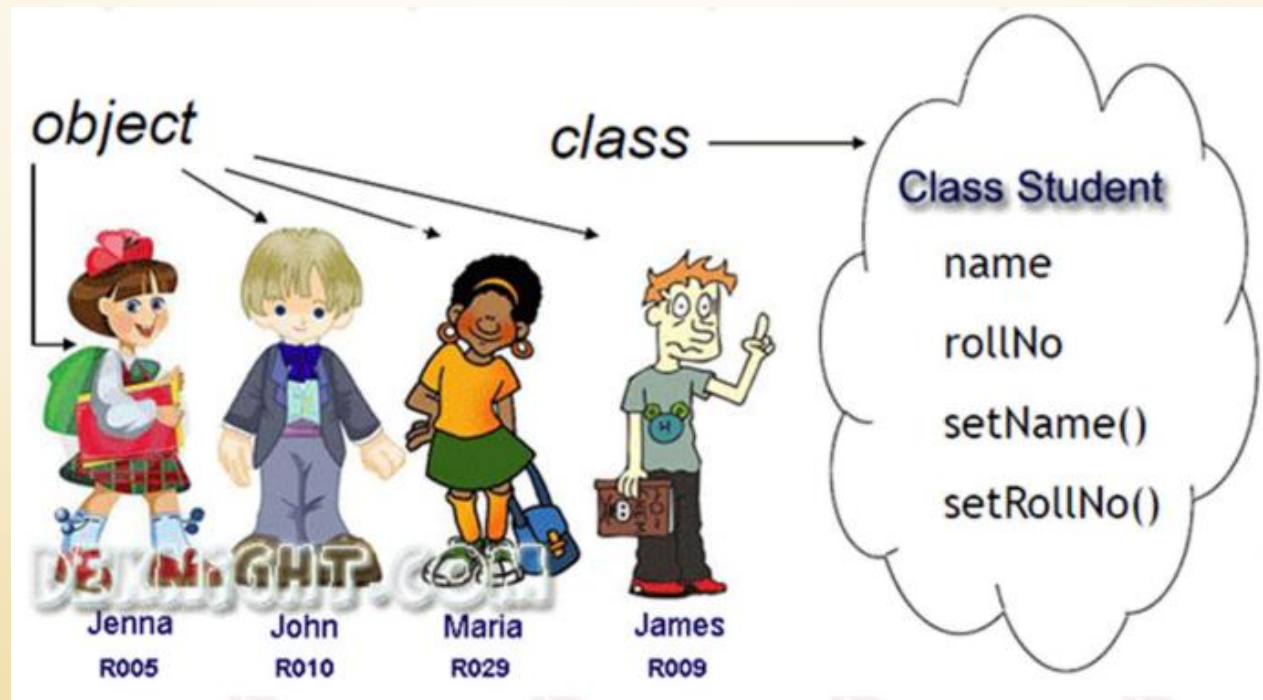
- Lập trình hướng đối tượng là phương pháp lập trình lấy đối tượng làm nền tảng để xây dựng chương trình.
- Một định nghĩa khác về lập trình hướng đối tượng đó là phương pháp lập trình dựa trên kiến trúc **lớp** (class) và **đối tượng** (object).

MỘT SỐ KHÁI NIỆM CƠ BẢN

- **ĐỐI TƯỢNG**
- **Đối tượng** được hiểu như là 1 thực thể: người, vật hoặc 1 bảng dữ liệu, . .
- Một đối tượng bao gồm 2 thông tin: **thuộc tính** và **phương thức**.
- **Thuộc tính** chính là những thông tin, đặc điểm của đối tượng. Ví dụ: một người sẽ có họ tên, ngày sinh, màu da, kiểu tóc, . . .
- **Phương thức** là những thao tác, hành động mà đối tượng đó có thể thực hiện. Ví dụ: một người sẽ có thể thực hiện hành động nói, đi, ăn, uống, . . .

MỘT SỐ KHÁI NIỆM CƠ BẢN

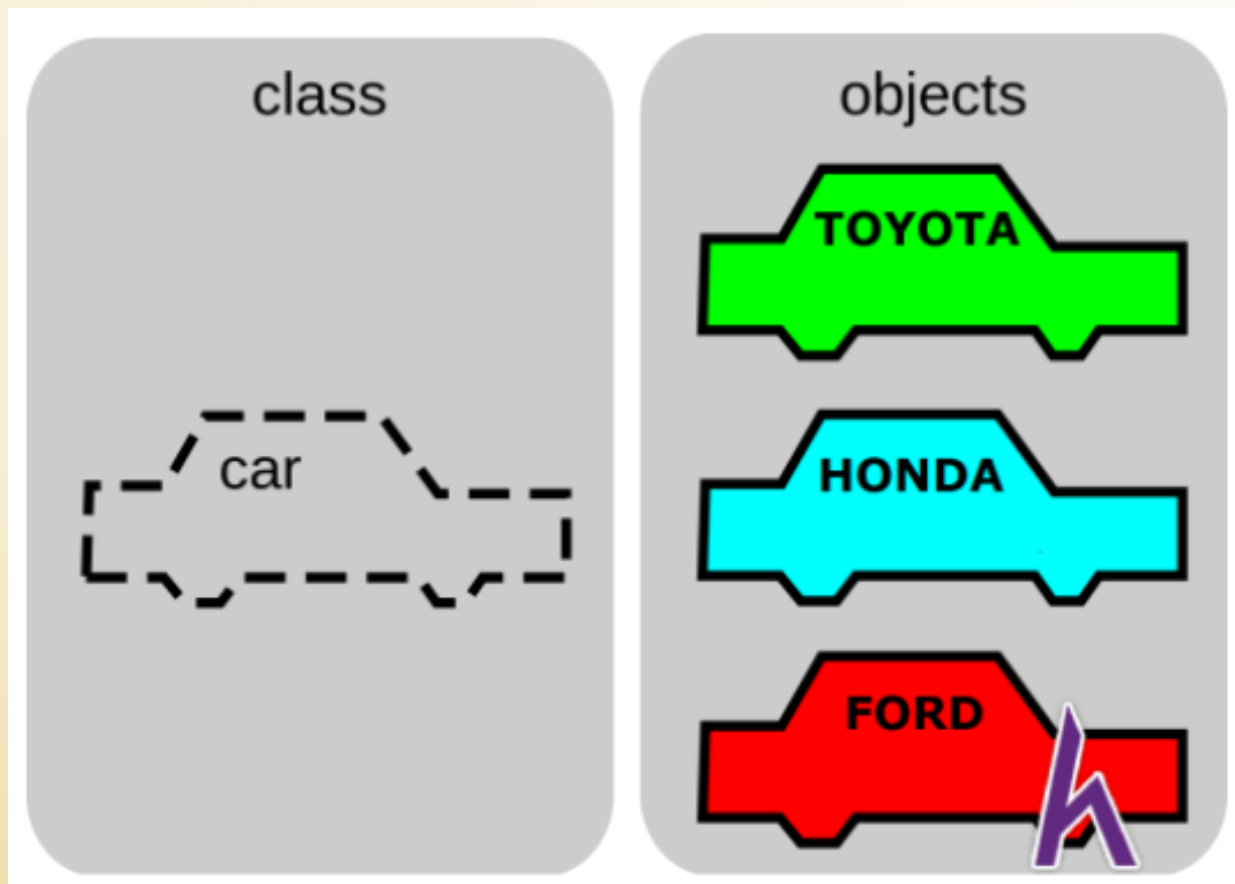
- **Lớp**
- Các đối tượng có các đặc tính tương tự nhau được gom lại thành 1 **lớp đối tượng**.
- Bên trong lớp cũng có 2 thành phần chính đó là thuộc tính và phương thức.
- Ngoài ra, lớp còn được dùng để định nghĩa ra kiểu dữ liệu mới.



PHÂN BIỆT ĐỐI TƯỢNG VÀ LỚP

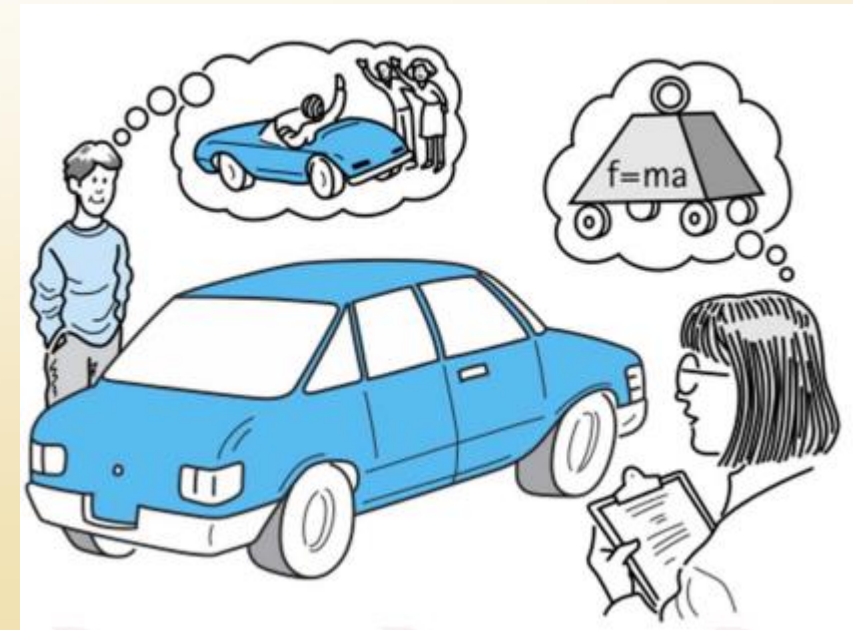
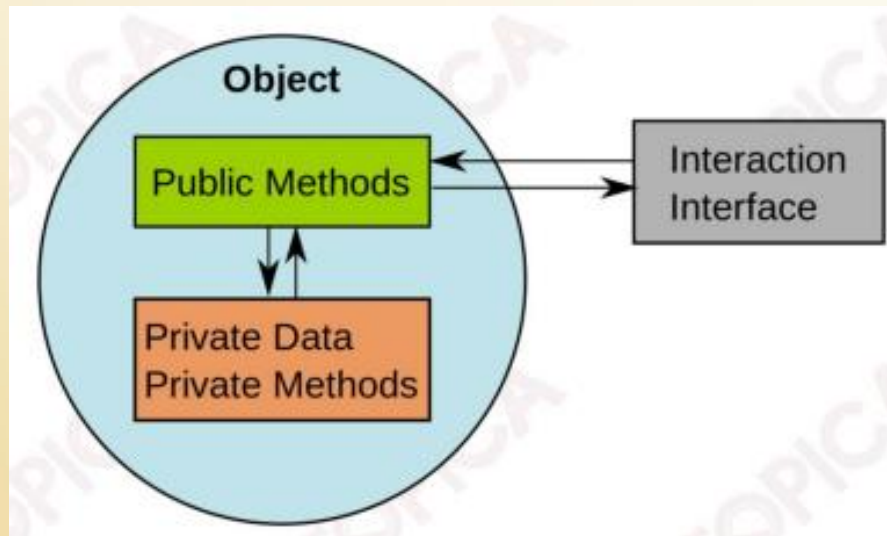
- **Lớp** là một khuôn mẫu còn **đối tượng** là một thể hiện cụ thể dựa trên khuôn mẫu đó.
- ví dụ :
- Nói về con mèo thì lớp chính là loài mèo. Loài mèo có:
 - Các thông tin, đặc điểm như 4 chân, 2 mắt, có đuôi, có chiều cao, có cân nặng, màu lông . . .
 - Các hành động như: kêu meo meo, đi, ăn, ngủ, . . .
- Như vậy mọi động vật thuộc loài mèo sẽ có những đặc điểm như trên.
- Đối tượng chính là một con mèo cụ thể nào đó, như con mèo con mà mình đang nuôi.

PHÂN BIỆT ĐỐI TƯỢNG VÀ LỚP



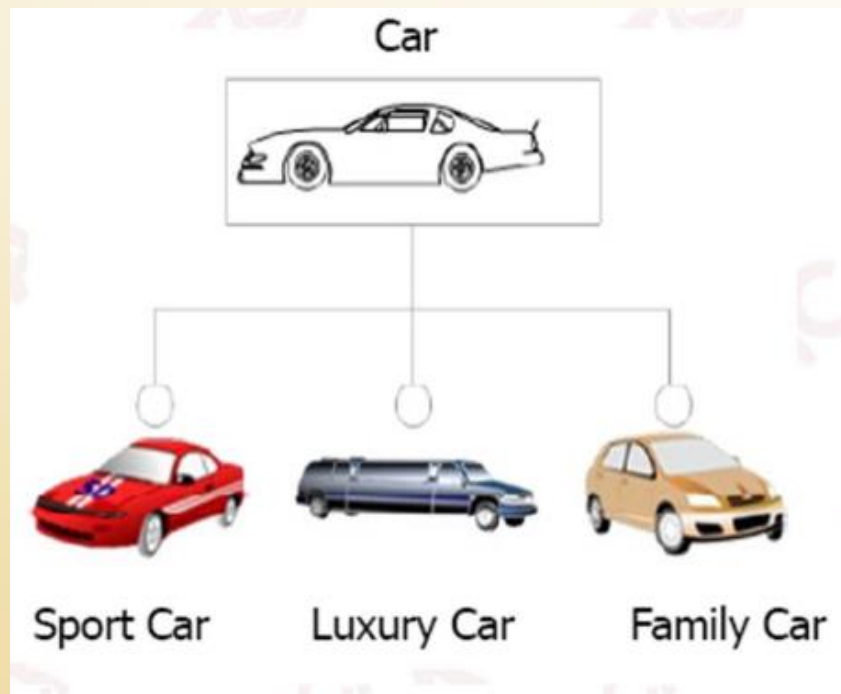
CÁC ĐẶC ĐIỂM CỦA LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

- **Tính đóng gói:** Khả năng truy xuất các thành phần của đối tượng mà vẫn đảm bảo che giấu các đặc điểm riêng tư của đối tượng
- **Tính trừu tượng:** Một đặc tả trừu tượng cho biết một đối tượng sẽ làm gì mà không cần bận tâm vào việc đối tượng làm như thế nào?



CÁC ĐẶC ĐIỂM CỦA LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

- **Tính kế thừa:** Là tính chất cho phép các đối tượng có thể chia sẻ, mở rộng các thuộc tính và phương thức mà không cần định nghĩa lại
- **Tính đa hình:** Thể hiện khi với cùng một phương thức nhưng có thể có các cách xử lý khác nhau



CLASS TRONG C# LÀ GÌ?

- **Class** trong C# chính là cách thể hiện khái niệm về lớp trong lập trình hướng đối tượng.
- Một **class** trong C# có các thành phần như:
- Thuộc tính: là các thành phần dữ liệu hay còn gọi là các biến.
- Phương thức: là các hàm thành phần thể hiện các hành vi của một đối tượng thuộc lớp.
- Phương thức khởi tạo.
- Phương thức hủy bỏ.
- **Class** trong C# thực chất là một kiểu dữ liệu mới do người dùng tự định nghĩa.

KHAI BÁO LỚP TRONG C#

- Cú pháp:
- < Từ khóa khai báo phạm vi > class <Tên lớp>
- {
- //Khai báo các sự kiện (Events)
- //Khai báo các thuộc tính (Properties)
- //Khai báo các phương thức (Methods)
- //Khai báo các biến thành viên (Fields)
- }
- Trong đó:
- Từ khóa khai báo phạm vi: Được sử dụng để chỉ ra phạm vi hoạt động của lớp.
- Các từ khóa phạm vi: public, protected, private, . . .

KHAI BÁO LỚP TRONG C#

- Ví dụ:

- `public class Person`
- `{`
- `private string name;`
- `private int age;`
- `public Person()`
- `{`
- `name = "No name";`
- `age = 0;`
- `}`
- `public Person(string ten, int tuoi)`
- `{`
- `name = ten;`
- `age = tuoi;`
- `}`
- `}`

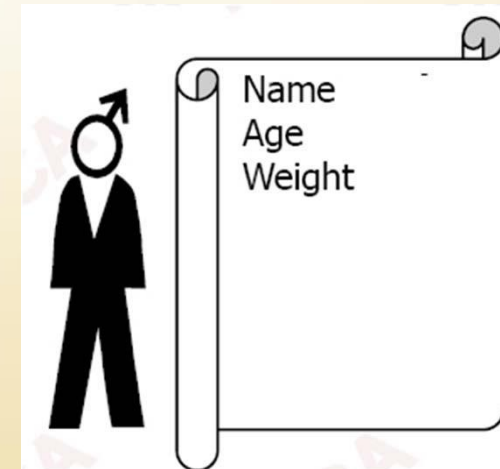
TẠO ĐỐI TƯỢNG TỪ LỚP

- Một đối tượng là một thể hiện của lớp, với các giá trị cụ thể của các trường dữ liệu thành viên.
- Cú pháp tạo đối tượng:
 - `<Tên lớp> <Tên đối tượng> = new <Tên lớp>();`
 - Ví dụ: Tạo ra một đối tượng của lớp Person
 - `Person person1 = new Person();`
 - `Person person2 = new Person();`
 - Tạo ra đối tượng Person và khởi tạo giá trị cho các trường dữ liệu thành viên:
 - `Person p1 = new Person("Dinh Manh Hung", 20);`
 - `Person p2 = new Person("Le Thu Ha", 19);`

XÂY DỰNG CÁC THÀNH PHẦN TRONG LỚP

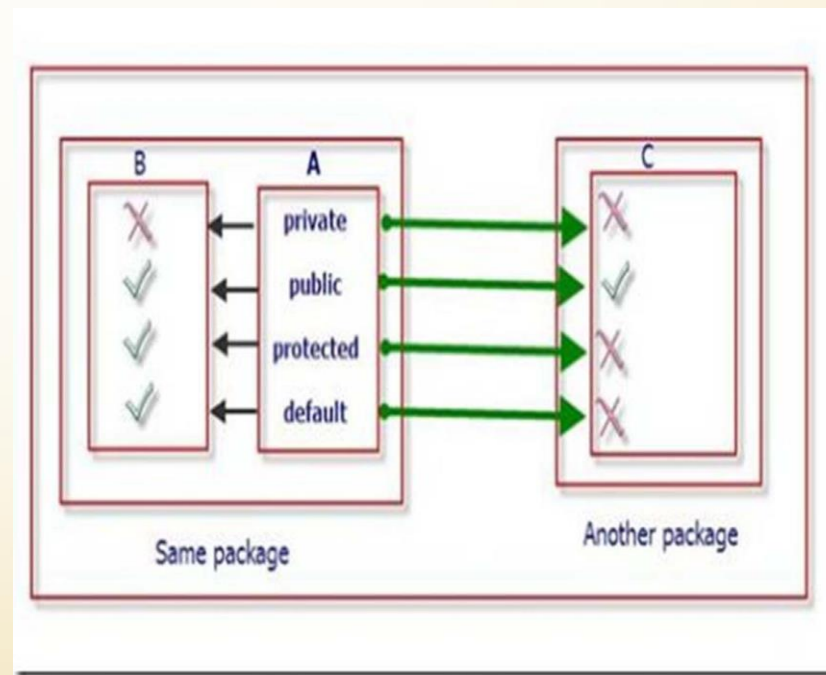
- Biến thành viên (Field): Biến thành viên chính là các trường dữ liệu của lớp, được sử dụng để lưu trữ dữ liệu của lớp.
- Biến thành viên của một lớp được khai báo theo cú pháp sau:
 - $\langle \text{Phạm vi} \rangle \langle \text{Kiểu dữ liệu} \rangle \langle \text{Tên biến} \rangle = \langle \text{Giá trị} \rangle$
 - Trong đó:
 - Phạm vi: Chỉ ra phạm vi hoạt động của biến thành viên trong lớp và có thể sử dụng các từ khóa public, private, protected.
 - Ví dụ lớp Person:

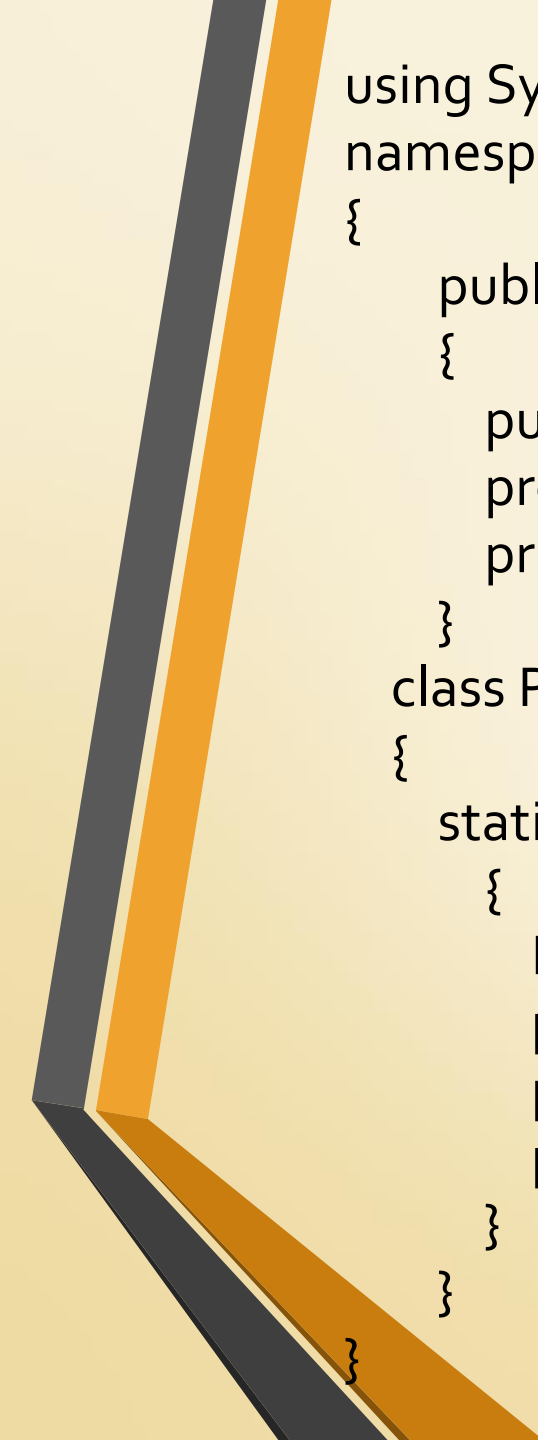
```
public class Person
{
    public string name;
    protected int age;
    private float weight;
}
```



CÁC LOẠI PHẠM VI TRUY CẬP VÀ Ý NGHĨA

Phạm vi truy cập	Ý nghĩa
public	Không hạn chế. Thành phần mang thuộc tính này có thể được truy cập ở bất kỳ nơi nào.
private	Thành phần mang thuộc tính này là thành phần riêng tư chỉ có nội bộ bên trong lớp chứa nó mới có quyền truy cập.
protected	Tương tự như private ngoài ra còn có thể truy cập từ lớp dẫn xuất lớp chứa nó.
internal	Chỉ được truy cập trong cùng 1 Assembly (nói cách khác là cùng project). Thuộc tính này thường được dùng cho class .
protected internal	Tương tự như internal ngoài ra còn có thể truy cập từ lớp dẫn xuất lớp chứa nó





```
using System;
namespace oop
{
    public class Person
    {
        public string name;
        protected int age;
        private float weight;
    }
    class Program
    {
        static void Main(string[] args)
        {
            Person p1 = new Person();
            p1.name = "Hung"; //Hợp lệ
            p1.age = 20; //Không hợp lệ
            p1.weight = 55; //Không hợp lệ
        }
    }
}
```

CÁC LOẠI PHẠM VI TRUY CẬP VÀ Ý NGHĨA

- Ví dụ: truy cập các trường của lớp Person

- `public class Person`

- `{`

- `public string name;`

- `protected int age;`

- `private float weight;`

- `}`

- `static void Main(string[] args)`

- `{`

- `Person p1 = new Person();`

- `p1.name = "Hung"; //Hợp lệ`

- `p1.age = 20; //Không hợp lệ`

- `p1.weight = 55; //Không hợp lệ`

- `}`

- `}`

XÂY DỰNG CÁC THÀNH PHẦN TRONG LỚP

- Thuộc tính (Property): Khi các trường dữ liệu thành viên được khai báo trong lớp thì ta không thể truy xuất đến trường dữ liệu đó từ bên ngoài lớp. Để đảm bảo tính riêng tư của trường dữ liệu nhưng vẫn có thể truy xuất đến từ bên ngoài lớp ta xây dựng các thuộc tính (property).
- Các thuộc tính chính là các thành phần trong một lớp và sử dụng để truy xuất đến các trường dữ liệu thành viên private của lớp.

- Cú pháp:

<public> <Kiểu dữ liệu> <Tên thuộc tính>

{

 get { return <Tên biến thành viên private của lớp>; }

 set { <Tên biến thành viên private của lớp> = value; }

}

- Nếu một thuộc tính có đầy đủ các get và set thì ta nói rằng thuộc tính có tính chất đọc/ghi dữ liệu.

Nếu thuộc tính chỉ chứa get ta nói thuộc tính có tính chất chỉ đọc.

Nếu thuộc tính chỉ chứa set ta nói thuộc tính có tính chất chỉ ghi.

XÂY DỰNG CÁC THÀNH PHẦN TRONG LỚP

- Ví dụ: Lớp Person với thuộc tính Weight

- `public class Person`

- `{`

- `public string name; protected int age;`

- `private float weight;`

- `public float Weight`

- `{`

- `get { return weight; }`

- `set { weight = value; }`

- `}`

- `public Person(string ten, int tuoi, float cannang)`

- `{`

- `name = ten; age = tuoi; weight = cannang;`

- `}`

- `}`

XÂY DỰNG CÁC THÀNH PHẦN TRONG LỚP

- Kiểm tra tính hợp lệ của các thuộc tính: Khi xây dựng các thuộc tính ta có thể sử dụng cấu trúc if để kiểm tra tính hợp lệ của dữ liệu.

```
public class Person
{
    public string name;
    private float weight;
    public float Weight
    {
        get { return weight; }
        set
        {
            if (value > 0)
                weight=value;
            else
                Console.WriteLine("Du lieu khong hop le");
        }
    }
}
```

XÂY DỰNG CÁC THÀNH PHẦN TRONG LỚP

- Phương thức: Phương thức của một lớp thực chất là các hàm nhằm xử lý dữ liệu của lớp. Cú pháp tạo phương thức như sau:
 - <Phạm vi> <Kiểu dữ liệu phương thức> <Tên phương thức> (đối số)
 - {
 - //Các lệnh xử lý dữ liệu
 - }
- Hàm tạo: là hàm đặc biệt trong một lớp, hàm sẽ được tự triệu gọi thực thi mỗi khi tạo ra một đối tượng trong lớp, hàm tạo không có kiểu dữ liệu trả về và có tên trùng với tên lớp.
 - public class Staff
 - {
 - private string name;
 - private float salary;
 - public Staff(string ten, float luong)
 - {
 - name = ten; salary = luong;
 - }
 - }

XÂY DỰNG CÁC THÀNH PHẦN TRONG LỚP

Ví dụ: Phương thức tính lương trong lớp Staff

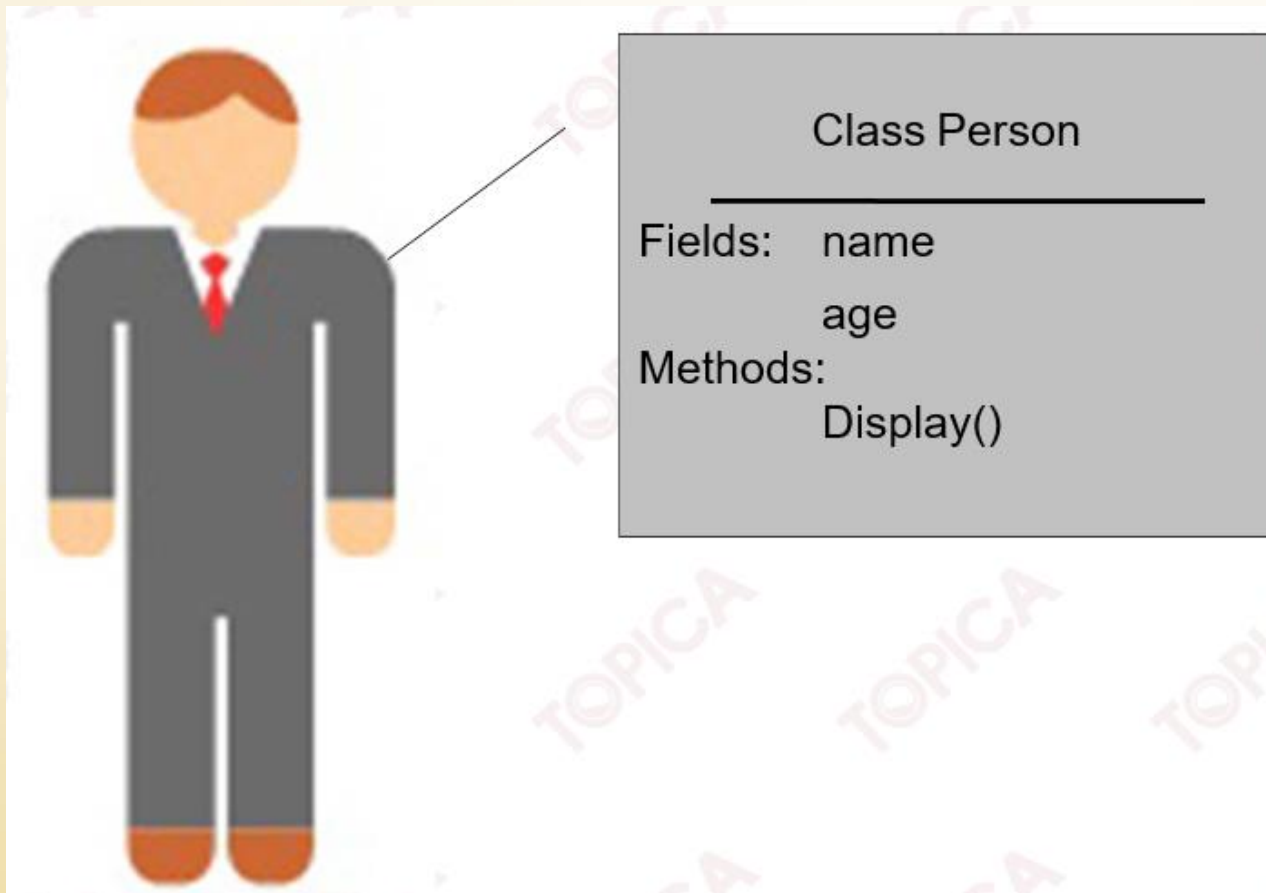
```
public class Staff
{
    private int workingYears;
    private float salary;
    public float TotalSalary()
    {
        float totalsalary = 0;
        if (workingYears < 5) totalsalary = salary;
        else if (workingYears < 10)
            totalsalary = salary + (10 * salary) / 100;
        else totalsalary = salary + (15 * salary) / 100;
        return totalsalary;
    }
}
```

TÍNH KẾ THỪA (INHERITANCE)

- **Tính kế thừa** là một đặc trưng cơ bản của lập trình hướng đối tượng, tính kế thừa cho phép các lập trình viên sử dụng lại các thuộc tính, phương thức của một lớp mà không cần phải định nghĩa lại. Kế thừa giúp tái sử dụng và mở rộng code một cách hiệu quả.
- Một lớp A có thể cho lớp B kế thừa các thuộc tính và phương thức của nó, khi đó lớp A gọi là lớp cơ sở (lớp cha), lớp B gọi là lớp dẫn xuất (lớp con).
- **Cú pháp xây dựng lớp kế thừa trong C#:**
 - <Phạm vi> class <Tên lớp dẫn xuất> : <Tên lớp cơ sở>
 - {
 - //Mô tả các Field, Property, Method,...
 - }

TÍNH KẾ THỪA

- Ví dụ: Xây dựng lớp Person với các thuộc tính Tên (name) và Tuổi (age), hàm tạo và phương thức Display để hiển thị các thông tin.

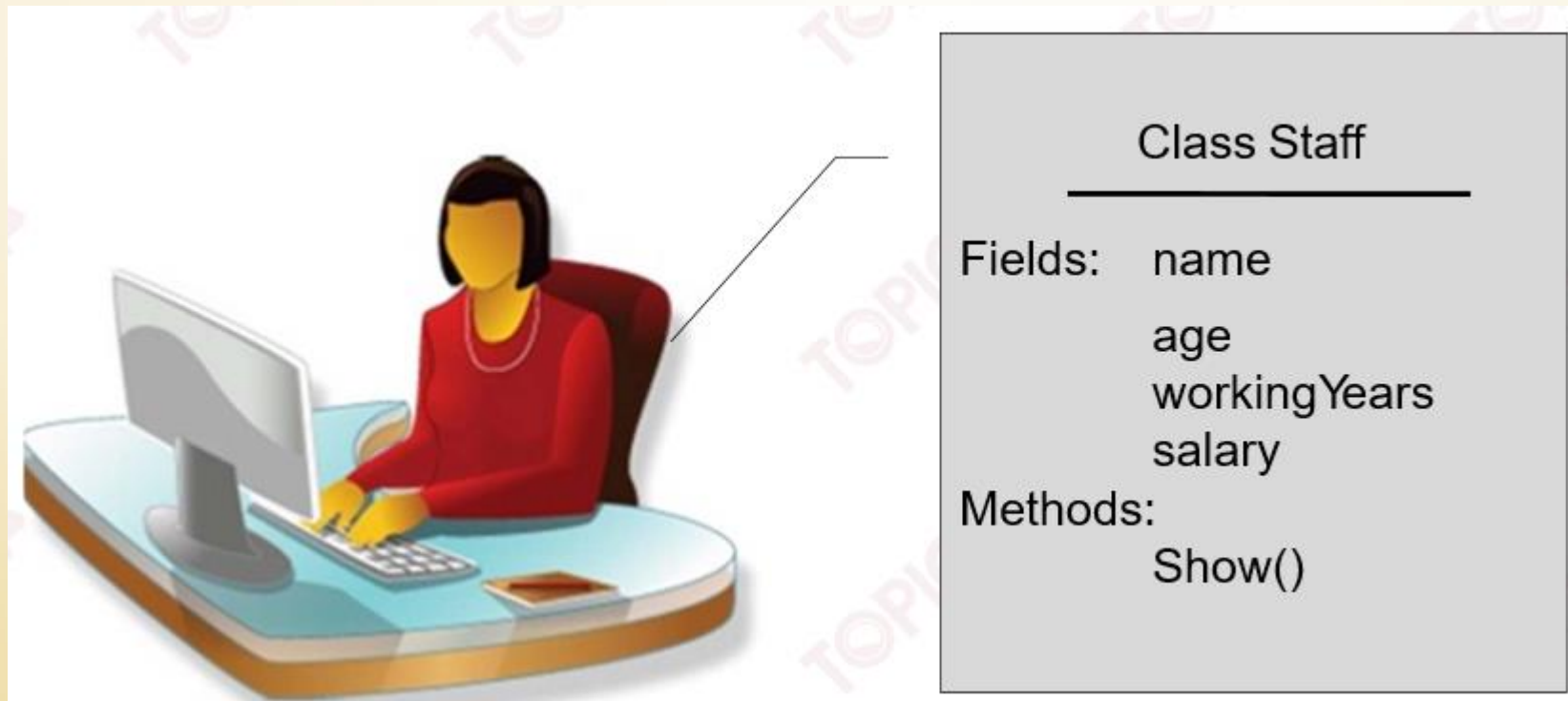


TÍNH KẾ THỪA

```
public class Person
{
    private string name;
    private int age;
    public Person(string ten, int tuoi)
    {
        name = ten;
        age = tuoi;
    }
    public void Display()
    {
        Console.WriteLine("Ho ten: {0}",name);
        Console.WriteLine("Tuoi: {0}",age);
    }
}
```

TÍNH KẾ THỪA

- Xây dựng lớp Staff (Nhân viên) kế thừa từ lớp Person, lớp Staff có thêm các thuộc tính workingYears, salary và phương thức Show() để hiển thị các thông tin về nhân viên, lớp Staff cũng kế thừa hàm tạo và phương thức Display từ lớp Person.



TÍNH KẾ THỪA

```
public class Staff:Person
{
    int workingYears;
    double salary;
    public Staff(string ten, int tuoi, int sonamcongtac, double luong):base(ten, tuoi)
    {
        workingYears = sonamcongtac;
        salary = luong;
    }
    public void Show()
    {
        Display(); //Sử dụng phương thức từ lớp Cha
        Console.WriteLine("So nam cong tac: {0}", workingYears);
        Console.WriteLine("Luong: {0}", salary);
    }
}
```



```
class Program
```

```
{
```

```
    static void Main(string[] args)
```

```
    {
```

```
        Console.WriteLine("Hello World!");
```

```
        Person P1 = new Person("Nguyen Van A", 31);
```

```
        P1.Display();
```

```
        Staff st1 = new Staff("Nguyen Van B", 20, 3, 5000000);
```

```
        st1.Show();
```

```
    }
```

```
}
```

TÍNH ĐA HÌNH (POLYMORPHISM)

- Từ khóa base: Từ khóa base được sử dụng để truy cập đến lớp cha từ lớp con. Ví dụ hàm tạo của lớp Staff kế thừa từ hàm tạo của lớp Person:

```
public Staff(string ten, int tuoi, int sonamcongtac, double  
    luong):base(ten, tuoi)  
{  
    this.workingYears = sonamcongtac ;  
    this.salary = luong;  
}
```

Từ khóa this: Từ khóa this được sử dụng để tham chiếu đến lớp hiện tại.

TÍNH ĐA HÌNH

- Ghi đè (overriding): Được sử dụng để định nghĩa lại phương thức của lớp cơ sở (lớp cha) trong lớp dẫn xuất (lớp con). Chỉ ghi đè các phương thức abstract hoặc virtual.
- Tính đa hình là ý tưởng “sử dụng chung một giao diện cho nhiều phương thức khác nhau” dựa trên phương thức ảo.
- Tham chiếu thuộc lớp cơ sở có thể trỏ đến đối tượng thuộc lớp dẫn xuất và có thể truy cập đến phương thức virtual đã định nghĩa trong lớp dẫn xuất.
- Nếu tham chiếu trỏ đến đối tượng thuộc lớp cơ sở thì phương thức ảo của lớp cơ sở được thực hiện.
- Nếu tham chiếu trỏ đến đối tượng thuộc lớp dẫn xuất thì phương thức ảo đã được định nghĩa lại trong lớp dẫn xuất sẽ được thực hiện.
- Ví dụ: Xây dựng lớp NhanVien và lớp NhanVienVP kế thừa từ lớp NhanVien.

TÍNH ĐA HÌNH

- Ví dụ:
- Xây dựng lớp NhanVien:
- Thuộc tính:
private string hoten;
private int tuoi;
private float hesoluong;
private float luongcoban

Phương thức:

```
public NhanVien(string ht, int t, float hsl)
• public virtual double tinhluong()
• {
•     return hesoluong*luongcoban;
• }
public virtual double tinhluong(int phucap)
{
    return hesoluong * luongcoban + (hesoluong * luongcoban * phucap) / 100;
}
public virtual void Show()
```


Xây dựng lớp NhanVien và lớp NhanVienVP kế thừa từ lớp NhanVien, thêm 2 thuộc tính int so_ngay_nghi;

float so_tien_giam

Phương thức:

```
public override double tinhluong()
```

```
{
```

```
    return base.tinhluong() - so_ngay_nghi*so_tien_giam;
```

```
}
```

```
public override double tinhluong(int phucap)
```

```
{
```

```
    return base.tinhluong(phucap) - so_ngay_nghi*so_tien_giam;
```

```
}
```

```
public override void Show()
```

TÍNH ĐA HÌNH

- Lớp nhân viên
- class NhanVien
- {
- private string hoten;
- private int tuoi;
- private float hesoluong;
- private float luongcoban=3000000;
- public NhanVien(string ht, int t, float hsl)
- {
- hoten =ht;
- tuoi = t;
- hesoluong = sl;
- }
- public virtual double tinhluong()
- {
- return hesoluong*luongcoban;
- }

```
public virtual double tinhluong(int phucap)
{
    return hesoluong * luongcoban +
    (hesoluong * luongcoban * phucap) / 100;
}
public virtual void Show()
{
    Console.WriteLine("Ho
ten: {0}", hoten);

    Console.WriteLine("Tuoi:
{0}",tuoi);
    Console.WriteLine("He
so luong: {0}", hesoluong);
    Console.WriteLine("Luong
co ban: {0}", luongcoban);
}
}
```

TÍNH ĐA HÌNH

Lớp NhanVienVP kế thừa từ lớp NhanVien

```
class NhanVienVP:NhanVien
```

```
{
```

```
    int so_ngay_nghi;
```

```
    float so_tien_giam=10000;
```

```
    public NhanVienVP(string hoten, int tuoi, float hsl, int songaynghi) : base(hoten, tuoi, hsl)
```

```
{
```

```
        so_ngay_nghi = songaynghi;
```

```
}
```

```
    public override double tinhluong()
```

```
{
```

```
        return base.tinhluong() - so_ngay_nghi*so_tien_giam;
```

```
}
```

TÍNH ĐA HÌNH

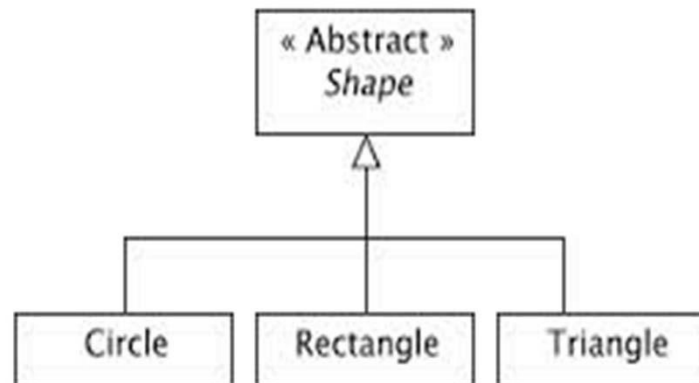
```
public override double tinhluong(int phucap)
{
    return base.tinhluong(phucap) - so_ngay_nghi*so_tien_giam;
}
public override void Show()
{
    base.Show();
    Console.WriteLine("So ngay nghi: {0}", so_ngay_nghi);
    Console.WriteLine("So tien luong: {0,10:0}", this.tinhluong());
}
}
```

TÍNH ĐA HÌNH

- Khởi tạo các đối tượng thuộc lớp NhanVien và NhanVienVP, gọi thực thi các phương thức Show() trong các lớp.
 - `NhanVien nv1 = new NhanVien("Hung",20,2.39F);`
 - `nv1.Show(); //Gọi phương thức Show() của lớp NhanVien`
`NhanVienVP nv2 = new NhanVienVP("Nam", 25,2.39F,3);`
`nv2.Show(); // Gọi phương thức Show() của lớp NhanVienVP`
`NhanVien nv3 = new NhanVienVP("Dung", 25, 2.39F, 3);`
`nv3.Show();//Gọi phương thức Show() của lớp NhanVienVP`

LỚP TRỪ TƯỢNG

- Lớp trừu tượng (abstract class) thực chất là một lớp cơ sở mà các lớp khác có thể dẫn xuất từ nó.
- Lớp trừu tượng là một lớp chưa hoàn chỉnh, trong lớp trừu tượng thường có chứa các phương thức trừu tượng, không thể tạo ra các đối tượng cụ thể từ lớp trừu tượng.
- Một phương thức trừu tượng là một phương thức chỉ có khai báo, mà không có phần thực thi, phần thực thi của phương thức trừu tượng sẽ được triển khai trong lớp dẫn xuất.



LỚP TRỪU TƯỢNG

- Cách khai báo lớp trừu tượng trong C#
 - <Phạm vi> abstract class <Tên lớp>
 - {
 - //Khai báo các thành viên của lớp trừu tượng
 - }
- **Chú ý:**
 - Các phương thức trừu tượng không được khai báo là private.
 - Một lớp trừu tượng không thể được niêm phong (Sealed).
 - Một thành viên trừu tượng không thể là static.
 - Ví dụ: Xây dựng lớp trừu tượng Hình, sau đó xây dựng 2 lớp kế thừa từ lớp Hình là lớp HìnhTron và HìnhChuNhat.

LỚP TRỪU TƯỢNG

- Lớp trừu tượng Hình:

- abstract class Hình

- {

protected double PI = 3.14159;

abstract public double TinhDienTich();

abstract public double TinhChuVi();

- }

LỚP TRỪU TƯỢNG

- Lớp HìnhTron kế thừa từ lớp Hình, và triển khai ghi đè 2 phương thức trừu tượng của lớp Hình.

```
• class HìnhTron: Hình
• {
•     private double bankinh; public HìnhTron(double r)
•     {
•         bankinh = r;
•     }
•     public override double TinhDienTich()
•     {
•         return PI * bankinh * bankinh;
•     }
•     public override double TinhChuVi()
•     {
•         return 2*PI * bankinh;
•     }
• }
```

LỚP TRỪU TƯỢNG

```
• class HìnhChuNhat : Hình
• {
•     private double dai, rong;
•     public HìnhChuNhat(double d, double r)
•     {
•         dai = d;
•         rong = r;
•     }
•     public override double TínhDienTich()
•     {
•         return dai * rong;
•     }
•     public override double TínhChuVi()
•     {
•         return dai + rong;
•     }
• }
```