BÀI 2
LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG VỚI C#

Giảng viên: ThS. Phan Thanh Toàn

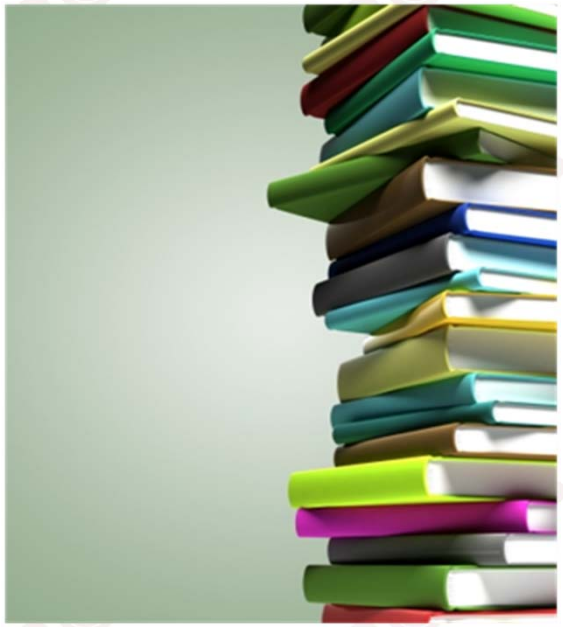
MỤC TIÊU BÀI HỌC

- Liệt kê được các tính chất cơ bản của lập trình hướng đối tượng.
- Phân biệt được các khái niệm cơ bản như lớp, đối tượng, thuộc tính, phương thức.
- Vận dụng ngôn ngữ C# vào triển khai, xây dựng lớp và tạo lập đối tượng.
- Phân biệt các khái niệm cơ bản như trừu tượng, đa hình, kế thừa...

CÁC KIẾN THỨC CẦN CÓ

Để học được môn học này, sinh viên phải học xong các môn học:

- Lập trình cơ bản;
- Lập trình hướng đối tượng;
- Cơ sở dữ liệu;
- Hệ quản trị cơ sở dữ liệu SQL Server.



HƯỚNG DẪN HỌC

- Đọc tài liệu tham khảo;
- Thảo luận với giáo viên và các sinh viên khác về những vấn đề chưa hiểu rõ;
- Trả lời các câu hỏi của bài học.

CẤU TRÚC NỘI DUNG

2.1. TỔNG QUAN VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

2.1.1. Các khái niệm cơ bản

2.1.2. Các đặc trưng của lập trình hướng đối tượng

2.1.1. CÁC KHÁI NIỆM CƠ BẢN

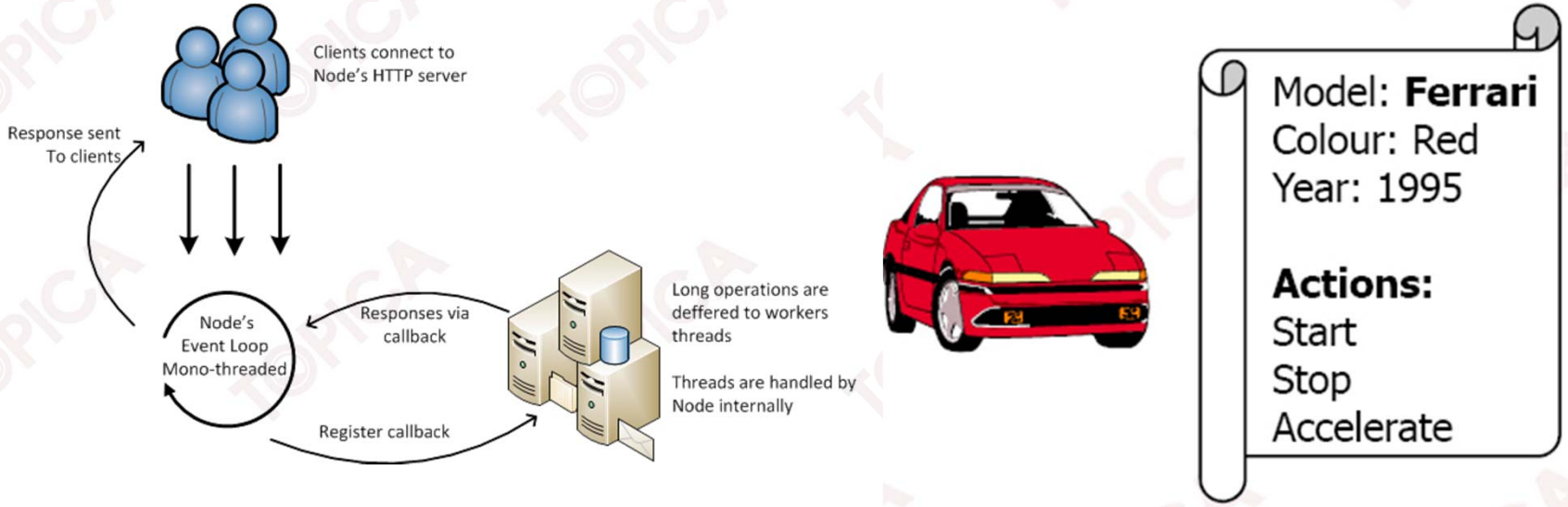
- Lập trình hướng đối tượng là gì? (Object-Oriented Programming: OOP): Lập trình hướng đối tượng là một phương pháp lập trình mới nhằm làm cho chương trình trở lên linh hoạt, tin cậy và dễ phát triển, dễ bảo trì và nâng cấp.
- Sự trừu tượng dữ liệu (Data abstraction): là phương pháp biểu diễn dữ liệu giúp người sử dụng có thể thao tác trên dữ liệu một cách dễ dàng mà không cần quan tâm đến các chi tiết của dữ liệu.

  Ví dụ: Kiểu dữ liệu số thực dấu chấm phẩy động trong các ngôn ngữ lập trình đã được trừu tượng hóa, khi lập trình người lập trình không cần quan tâm đến cách biểu diễn nhị phân chính xác của số thực dấu chấm phẩy động và các chi tiết khác.

2.1.1. CÁC KHÁI NIỆM CƠ BẢN (tiếp theo)

- Đối tượng: là tất cả các thực thể cần quản lý trong chương trình. Mỗi đối tượng sẽ gồm có 3 thành phần chính là: thuộc tính, phương thức và sự kiện.
- Thuộc tính: Được sử dụng để mô tả mặt tĩnh của đối tượng, các đối tượng được phân biệt với nhau qua thuộc tính của đối tượng.
- Phương thức: Được sử dụng mô tả mặt động của đối tượng, phương thức được sử dụng mô tả sự hoạt động và chức năng của đối tượng.

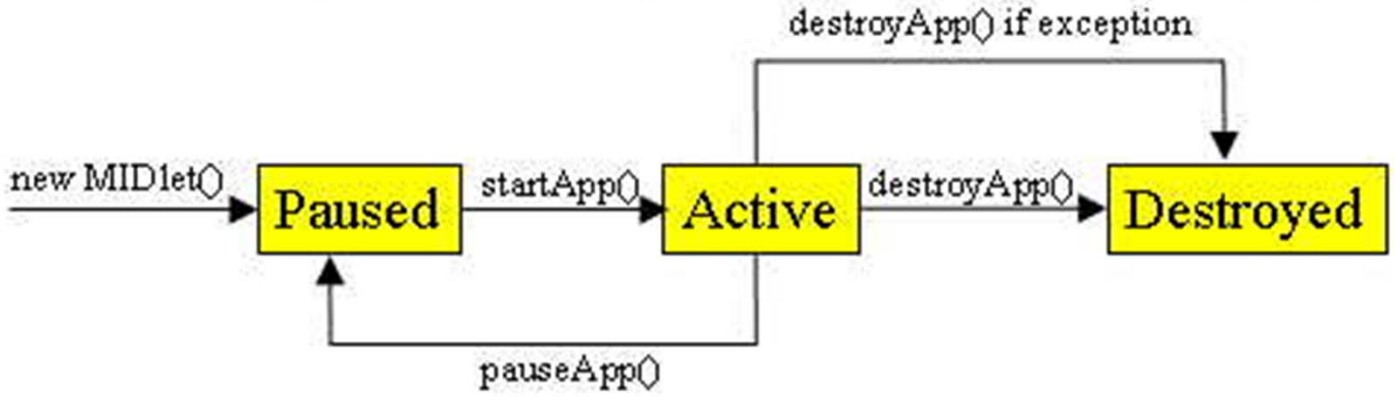
  Ví dụ: Đối tượng xe (car) với các thuộc tính và phương thức.



- Sự kiện: Được sử dụng để gửi thông tin từ đối tượng ra bên ngoài.

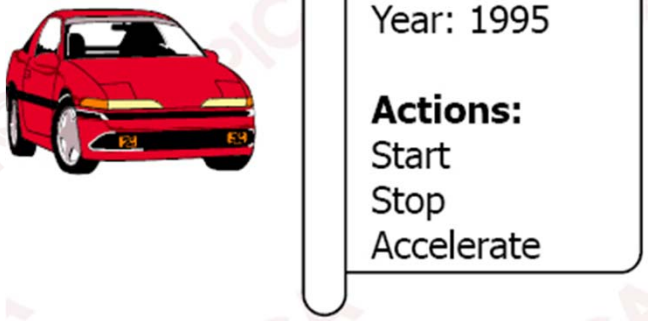
2.1.1. CÁC KHÁI NIỆM CƠ BẢN (tiếp theo)

- Vòng đời của đối tượng:
  - Khai báo;
  - Khởi tạo;
  - Sử dụng;
  - Hủy đối tượng.

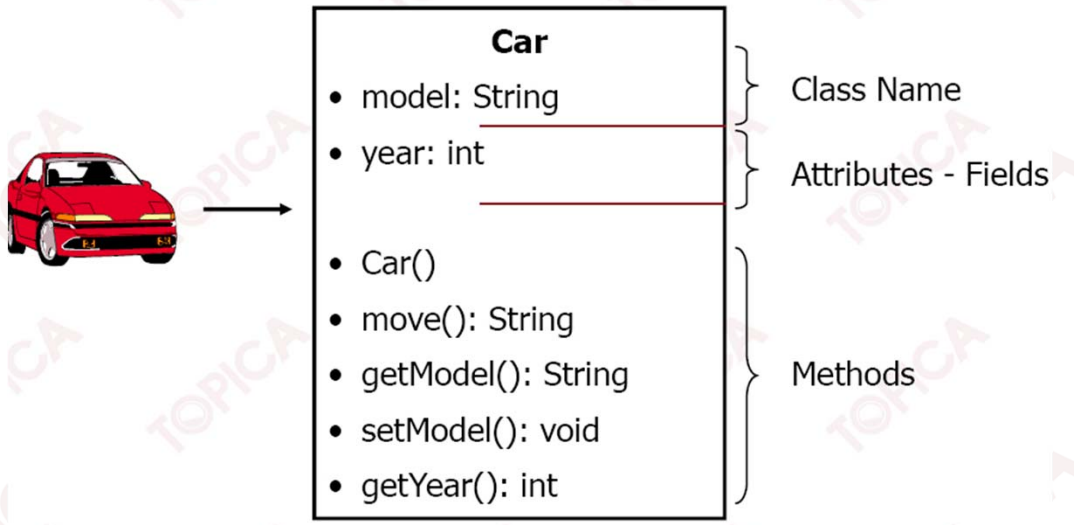


2.1.1. CÁC KHÁI NIỆM CƠ BẢN (tiếp theo)

- Lớp (class): là khái niệm dùng để mô tả nhóm đối tượng có những thuộc tính, phương thức giống nhau.
- Một đối tượng là một thể hiện của lớp với các giá trị cụ thể của các thuộc tính.



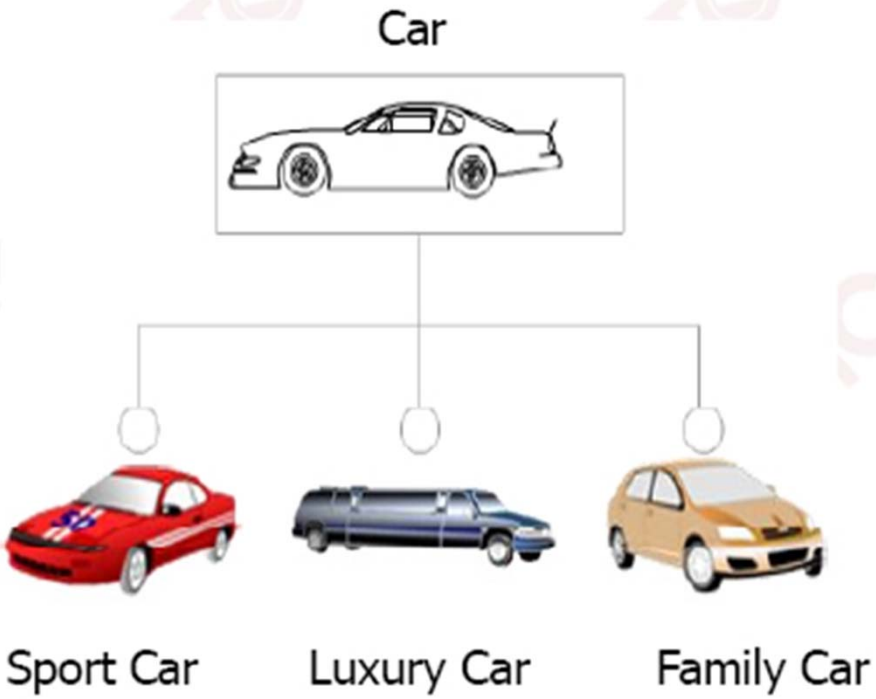
Đối tượng xe

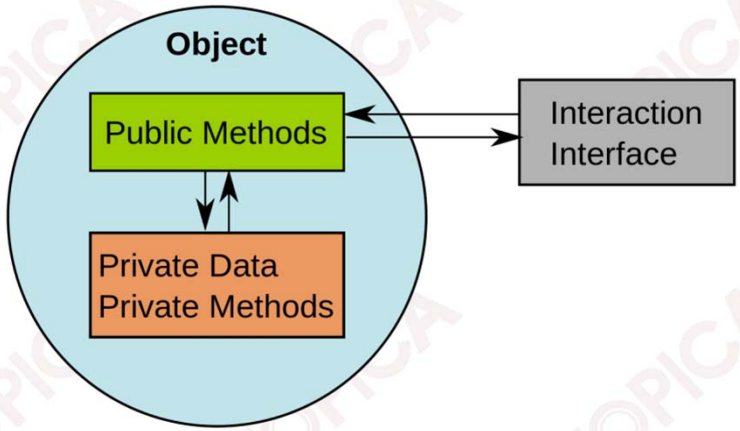


Lớp đối tượng xe

2.1.2. CÁC ĐẶC TRƯNG CƠ BẢN CỦA LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

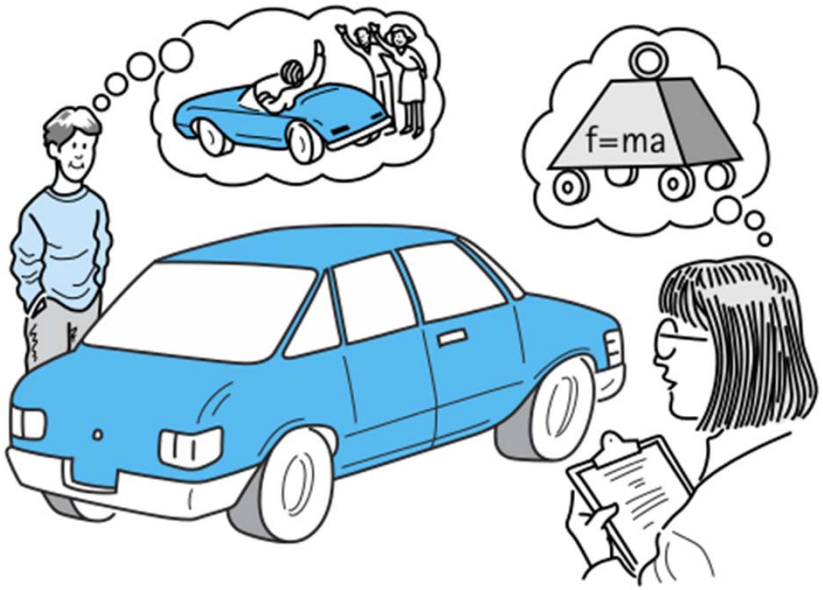
- Tính kế thừa: là tính chất cho phép các đối tượng có thể chia sẻ, mở rộng các thuộc tính và phương thức mà không cần định nghĩa lại.
- Tính bao đóng (Encapsulation): Khả năng truy xuất các thành phần của đối tượng mà vẫn đảm bảo che dấu các đặc tính riêng tư của đối tượng.





2.1.2. CÁC ĐẶC TRƯNG CƠ BẢN CỦA LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG (tiếp theo)

- 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?
- Tính đa hình: Thể hiện khi với cùng một phương thức nhưng có thể có các cách xử lí khác nhau.



2.2. XÂY DỰNG LỚP TRONG C#

2.2.1. Khái niệm

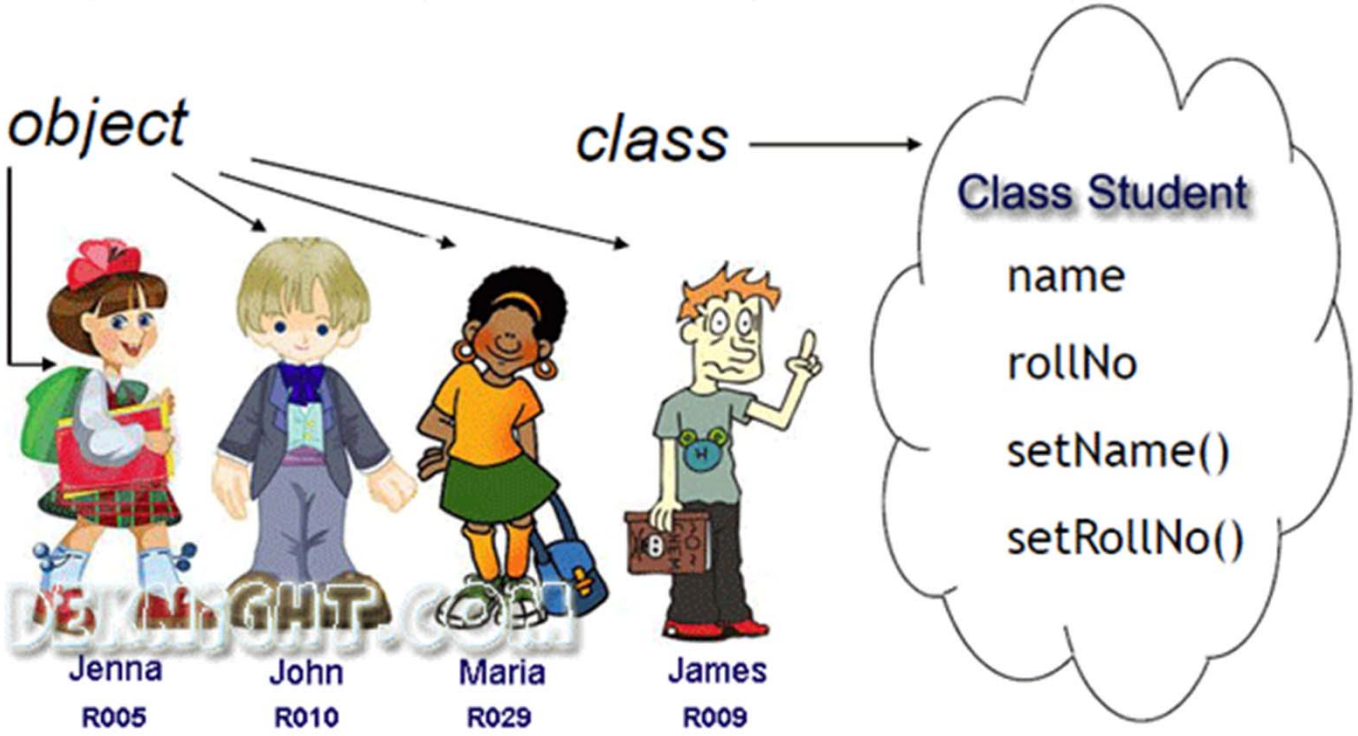
2.2.2. Khai báo lớp trong C#

2.2.3. Tạo đối tượng

2.2.4. Xây dựng các thành phần trong lớp

2.2.1. KHÁI NIỆM

- Lớp là sự trừu tượng hóa dữ liệu, được sử dụng để mô tả một nhóm các đối tượng có chung nhau các thuộc tính và phương thức.
- Lớp là một kiểu dữ liệu trừu tượng được xây dựng trong các ứng dụng và dùng để tạo ra các đối tượng cụ thể trong chương trình.



2.2.2. 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, partial.

2.2.2. KHAI BÁO LỚP TRONG C# (tiếp theo)

- Ví dụ lớp Person:

```
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;
}
```

2.2.3. 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);
```

2.2.4. 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:

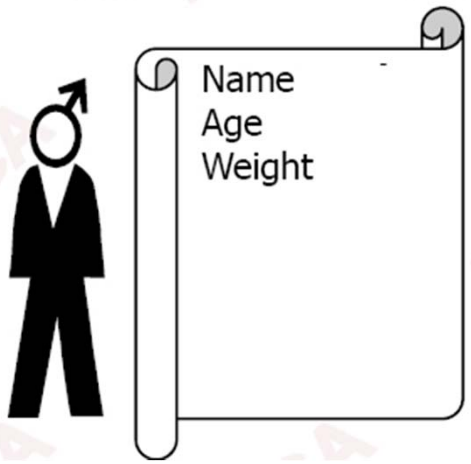
```
<Phạm vi> <Kiểu dữ liệu> <Tên biến> = <Giá trị>
```

- 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.

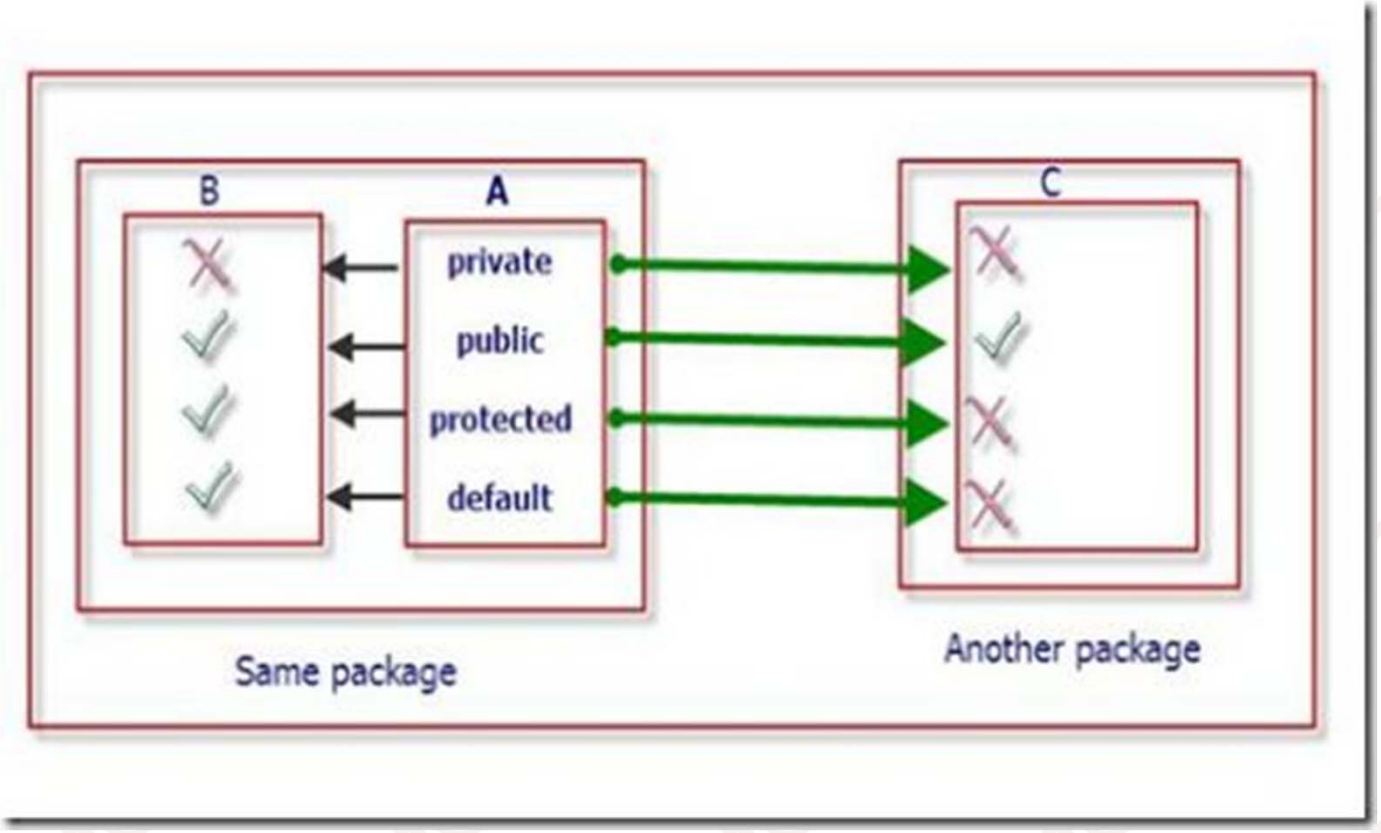
  Ví dụ lớp Person:

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



2.2.4. XÂY DỰNG CÁC THÀNH PHẦN TRONG LỚP (tiếp theo)

Ý nghĩa các từ khóa phạm vi public, private, protected được mô tả như bảng sau:



2.2.4. XÂY DỰNG CÁC THÀNH PHẦN TRONG LỚP (tiếp theo)

Ví dụ: truy cập các trường của lớp Person

```
  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
 }
```

2.2.4. XÂY DỰNG CÁC THÀNH PHẦN TRONG LỚP (tiếp theo)

- 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.

2.2.4. XÂY DỰNG CÁC THÀNH PHẦN TRONG LỚP (tiếp theo)

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;
}
   }
```

2.2.4. XÂY DỰNG CÁC THÀNH PHẦN TRONG LỚP (tiếp theo)

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");
}
}
      }
```

2.2.4. XÂY DỰNG CÁC THÀNH PHẦN TRONG LỚP (tiếp theo)

- 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;
}
}
```

2.2.4. XÂY DỰNG CÁC THÀNH PHẦN TRONG LỚP (tiếp theo)

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;
}
}
```

2.3. TÍNH KẾ THỪA VÀ ĐA HÌNH TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

2.3.1. Tính kế thừa

2.3.2. Lớp trừu tượng

2.3.3. Lớp niêm phong

2.3.4. Tính đa hình

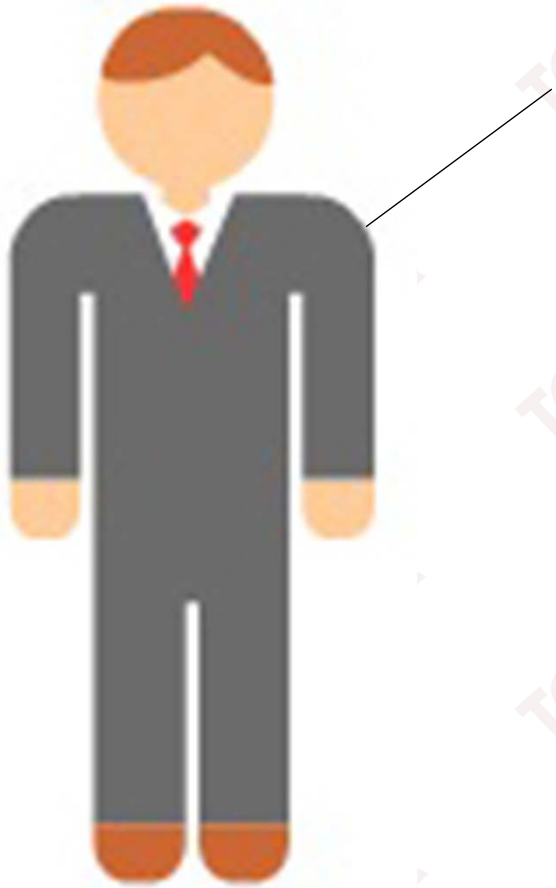
2.3.1. TÍNH KẾ THỪA

- 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,…
}
```

2.3.1. TÍNH KẾ THỪA (tiếp theo)

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.



Class Person

Fields: name
age

Methods:
Display()

2.3.1. TÍNH KẾ THỪA (tiếp theo)

```
  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);
}
  }
```

2.3.1. TÍNH KẾ THỪA (tiếp theo)

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.

Class Staff

Fields: name
age
workingYears
salary

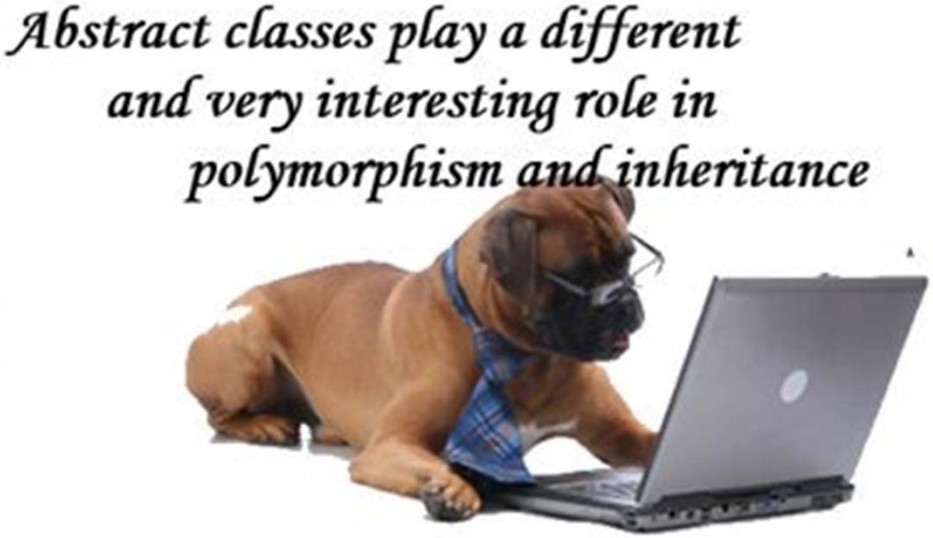
Methods:
Show()

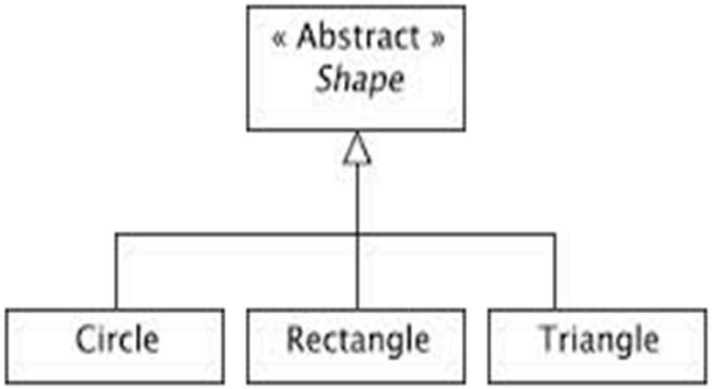
2.3.1. TÍNH KẾ THỪA (tiếp theo)

```
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);
}
}
```

2.3.2. LỚP TRỪU 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.





2.3.2. LỚP TRỪU TƯỢNG (tiếp theo)

- 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 Hinh, sau đó xây dựng 2 lớp kế thừa từ lớp Hinh là lớp HinhTron và HinhChuNhat.

2.3.2. LỚP TRỪU TƯỢNG (tiếp theo)

- Lớp trừu tượng Hinh:

```
abstract class Hinh
{
protected double PI = 3.14159;
abstract public double TinhDienTich();
abstract public double TinhChuVi();
}
```

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

2.3.2. LỚP TRỪU TƯỢNG (tiếp theo)

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

2.3.2. LỚP TRỪU TƯỢNG (tiếp theo)

```
class HinhChuNhat:Hinh
   {
 private double dai, rong;
 public HinhChuNhat(double d, double r)
 {
 dai = d; rong = r;
 }
 public override double TinhDienTich()
 {
 return dai * rong;
 }
 public override double TinhChuVi()
 {
 return dai + rong;
 }
   }
```

2.3.3. LỚP NIÊM PHONG

- Lớp niêm phong (Sealed class): là lớp không cho các lớp khác dẫn xuất từ nó.
- Cú pháp tạo lớp niêm phong trong C#:

```
<Phạm vi> sealed class <Tên lớp>
  {
//Khai báo các thành viên của lớp
  }
```

2.3.4. 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.

2.3.4. TÍNH ĐA HÌNH (POLYMORPHISM) (tiếp theo)

- 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.

2.3.4. TÍNH ĐA HÌNH (POLYMORPHISM) (tiếp theo)

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 = hsl;
}
```

2.3.4. TÍNH ĐA HÌNH (POLYMORPHISM) (tiếp theo)

```
    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);
    }
}
```

2.3.4. TÍNH ĐA HÌNH (POLYMORPHISM) (tiếp theo)

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;
}
```

2.3.4. TÍNH ĐA HÌNH (POLYMORPHISM) (tiếp theo)

```
    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());
    }
}
```

2.3.4. TÍNH ĐA HÌNH (POLYMORPHISM) (tiếp theo)

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
```

TÓM LƯỢC CUỐI BÀI

Trong bài này, chúng ta đã nghiên cứu các nội dung chính sau:

- Tổng quan về lập trình hướng đối tượng trong C#;
- Các khái niệm cơ bản như lớp, đối tượng, phương thức và thuộc tính;
- Cách cài đặt và tạo đối tượng trong C#;
- Một số tính chất cơ bản của lập trình hướng đối tượng như kế thừa, lớp, phương thức trừu tượng, tính đa hình trong lập trình hướng đối tượng.