EASY C#

CLASS

Lương Trần Hy Hiến - hyhien@gmail.com

NỘI DUNG

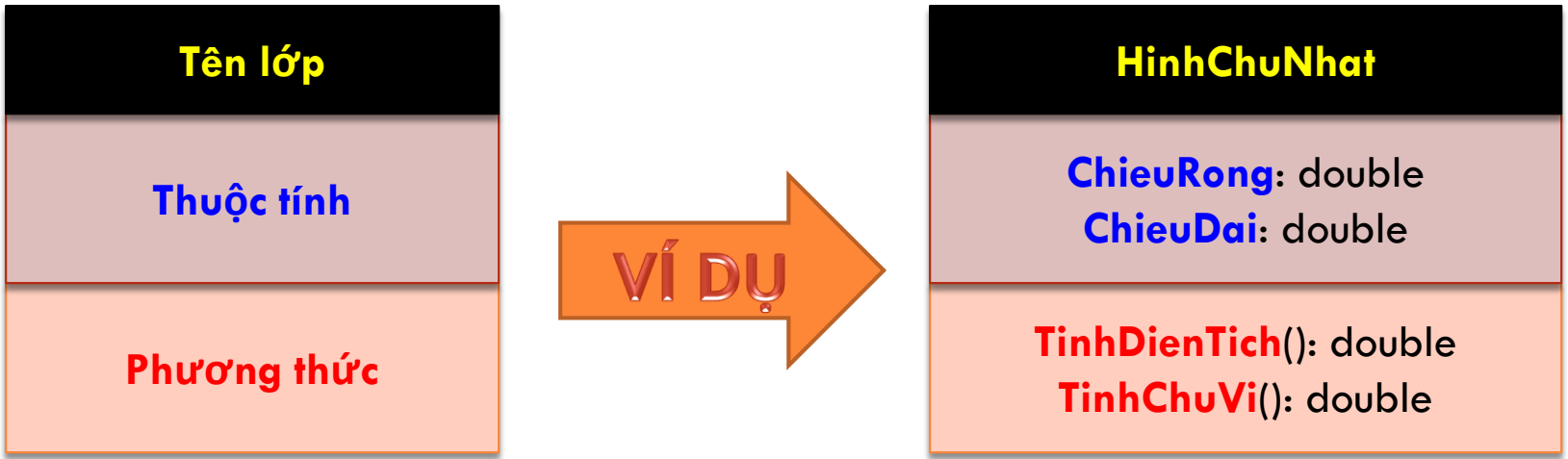
- Lớp và đối tượng (Class and object)
  - Mô hình
  - Định nghĩa
  - Sử dụng
- Phương thức (Method)
- Thuộc tính (Property)
- Kế thừa (inheritance)

Khái niệm đối tượng

- Thực thể
  - Đặc điểm
  - Hành vi
- Ví dụ
  - Máy tính
  - Con người
  - Đơn hàng
  - Phương trình bậc hai
  - …

TÌM HIỂU CLASS

- Class được dùng để mô tả một lớp các thực thể cùng loại.
  - Thuộc tính dùng để mô tả đặc điểm
  - Phương thức được sử dụng để mô tả hành vi của thực thể
- Mô hình UML của class



Class – Lớp

- Khái niệm:
  - Kiểu dữ liệu trong C# được định nghĩa là một lớp
  - Lớp (class) bao gồm:
    - Thuộc tính: *tính chất/đặc điểm của đối tượng*
    - Phương thức: *hành động của đối tượng*
    - *Được đóng gói lại (encapsulation)*
  - Lớp có thểbao gồm: *Fields, Methods, Properties, Constructors, Events*

Object – Đối tượng

- Là một thể hiện (instance) cụ thể của class
- Quan hệ giữa class với object cũng như quan hệ giữa kiểu dữ liệu với biến.

Lập trình Module	Lập trình OOP	Ví dụ
Type	Class	int j;
Variable	Object	SinhVien sv;

ĐỊNH NGHĨA CLASS

```
public class HinhChuNhat
{
    // định nghĩa 2 fields
    private double ChieuDai;
    private double ChieuRong;

    // định nghĩa 2 phương thức
    public double TinhChuVi()
    {
        Double ChuVi= (this.ChieuDai + this.ChieuRong) * 2;
        Return ChuVi;
    }
    public double TinhDienTich()
    {
        Double DienTich= this.ChieuDai * this.ChieuRong;
        return DienTich;
    }
}
```

- Tên lớp
  - HinhChuNhat
- Tên fields:
  - ChieuRong
  - ChieuCao
- Tên phương thức:
  - TinhChuVi()
  - TinhDienTich()

Các **field** dùng để **lưu** dữ liệu
Các **method thao tác** lên các field dừng để thực hiện chức năng

Định nghĩa lớp

```
[thuộc tính truy cập] class <Tên lớp> [:Lớp cơ sở]
{
    //thuộc tính

    //phương thức
}
```

```
class Diem
{
        // Cac thuoc tinh
        private int x; // x viet thuong
        private int y; // y viet thuong
        // Cac phuong thuc
        public double KhoangCach(Diem b)
        {
            double distance;
            distance = Math.Sqrt((x - b.x)*(x - b.x) + (y - b.y)*(y - b.y));
            return distance;
        }
}
```

Thuộc tính truy cập

- public: được dùng ở bất kỳ nơi đâu
- private: được dùng trong lớp
- protected: được dùng trong lớp và lớp con (lớp dẫn xuất – kế thừa)
- internal: dùng trong lớp và bất kỳ lớp nào cùng khối hợp ngữ với lớp này
- protected internal: dùng trong lớp, lớp con và bất kỳ lớp nào cùng khối hợp ngữ với lớp này

SỬ DỤNG LỚP HinhChuNhat

```
// Tạo đối tượng
HinhChuNhat hcn = new HinhChuNhat();

// Gán giá trị cho các thuộc tính
hcn.ChieuDai = 20; //???
hcn.ChieuRong = 10; //???

// Gọi phương thức tính diện tích
Double DienTich = hcn.TinhDienTich();
```

Toán tử **new** đứng trước **Constructor** dùng để tạo đối tượng

VÍ DỤ

MÔ HÌNH

Sản phẩm
Mã sản phẩm: số nguyên Tên sản phẩm: chuỗi Đơn giá: số thực Ngày sản xuất: ngày tháng
Tính thuế nhập khẩu (tỉ lệ: số thực): số thực

CLASS

```
public class SanPham
{
    public int maSanPham;
    public String tenSanPham;
    public double donGia;
    public DateTime ngaySanXuat;

    public double tinhThueNhapKhau(double tiLe)
    {
        return donGia * tiLe;
    }
}
```

THUỘC TÍNH (Properties)

- Thuộc tính được định nghĩa như bí danh của các trường (fields) với mục đích che chắn việc truy xuất dữ liệu trực tiếp lên các trường.
- Định nghĩa thuộc tính

```
public <kiểu> <tên thuộc tính>
{
    get {return <giá trị thuộc tính>;}
    set {<xử lý biến value>;}
}
```

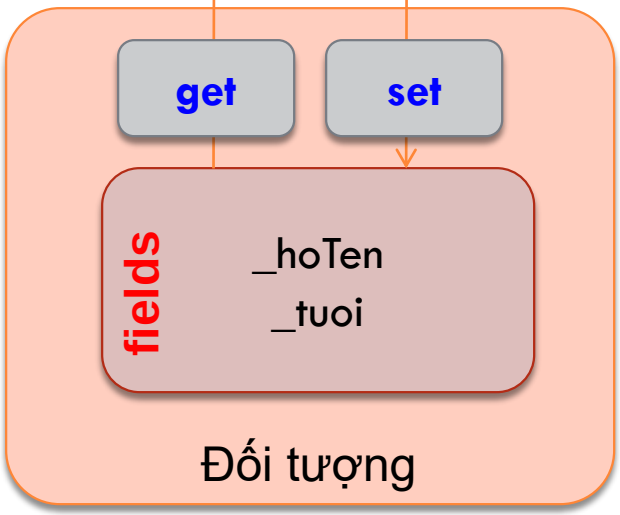
- Chú ý:
  - Mã trong **get** sẽ chạy khi thuộc tính **được truy xuất**
  - Mã trong **set** sẽ chạy khi thuộc tính **được gán giá trị**

THUỘC TÍNH

```
public class SinhVien
{
    private String _hoTen;
    private int _tuoi;

    public String HoTen
    {
        get{return _hoTen;}
        set{_hoTen = value.ToString();}
    }

    public int Tuoi
    {
        get{return _tuoi;}
        set{_tuoi = value;}
    }

    public bool ThanhNien
    {
        get { return Tuoi < 25; }
    }
}
```

```
SinhVien sv = new SinhVien();
// gán giá trị cho thuộc tính
sv.HoTen = "Nguyen Thi Hoa";
// truy xuất giá trị thuộc tính
int age = sv.Tuoi;
```

readonly



KHỞI ĐẦU GIÁ TRỊ THUỘC TÍNH

```
public class SinhVien
{
    private String _hoTen;

    public String HoTen
    {
        get{return _hoTen;}
        set{_hoTen = value.ToString();}
    }

    public int Tuoi { get; set; }

    public bool ThanhNien
    {
        get { return Tuoi < 25; }
    }

    public String Info
    {
        get
        {
            return String.Format("{0}\t{1}\t{2}", HoTen, Tuoi, ThanhNien);
        }
    }
}
```

Thuộc tính tự động

Khởi đầu giá trị cho các thuộc tính **HoTen** và **Tuoi** của class **SinhVien**

```
SinhVien sv = new SinhVien()
{
    HoTen = "Nguyễn Văn Tèo",
    Tuoi = 33
};
```

Giá trị khởi đầu cho thuộc tính Tuoi

VÍ DỤ TỔNG HỢP

```
// Khai báo danh sách và khởi đầu 1 sinh viên
List<SinhVien> dssv = new List<SinhVien>()
{
     new SinhVien()
     {
          HoTen = "Nguyễn Tuyết Mai",
          Tuoi = 33
     }
};
// Tạo một sinh viên và thêm vào danh sách
SinhVien quanh = new SinhVien()
{
     HoTen = "Chu Văn Quành",
     Tuoi = 55
};
dssv.Add(quanh);

// Xuất những sinh viên họ nguyễn
foreach (SinhVien sv in dssv)
{
     if (sv.HoTen.StartsWith("Nguyễn "))
     {
          Console.WriteLine(sv.Info);
     }
}
```

Hàm dựng (Constructor)

- Dùng tạo ra đối tượng (object)
- Tên Constructor trùng tên class, không có kiểu trả về
- Construtor được gọi tự động khi khai báo object

```
class Diem
{
        public Diem()
        {
                x = 0;
                y = 0;
        }
        public Diem(int xx, int yy)
        {
                x = xx;
                y = yy;
        }
}
```

Nạp chồng (overload)

- Các hàm/phương thức có thể trùng tên nhau
- Phân biệt bởi kiểu trả về và danh sách tham số

```
class Diem
{
        public Diem()
        {
            x = 0;
            y = 0;
        }
        public Diem(int x,x int yy)
        {
            x = xx;
            y = yy;
        }
}
```

Static

- Là thành phần độc lập với Object
- Chỉ có một instance duy nhất cho toàn bộ class
- Được gọi từ class:

  <class_name>.<static_component>

- Thành phần static chỉ gọi được thành phần static

Visual Studio

Microsoft® SQL Server® 2012

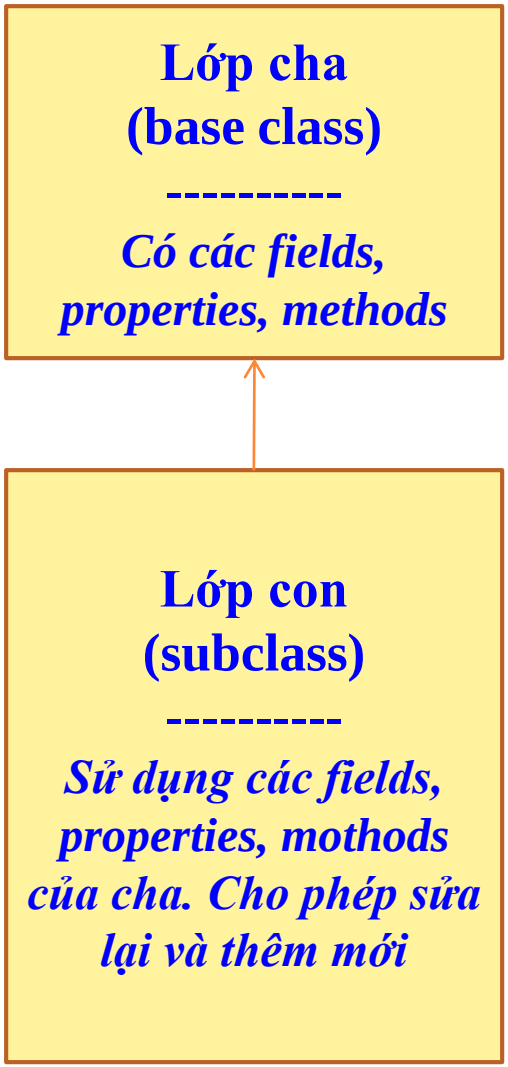
EASY C#

CLASS (TT)

Lương Trần Hy Hiến - hyhien@gmail.com

KẾ THỪA

- Kế thừa để tái sử dụng những gì đã xây dựng trong lớp trước đó (lớp cha).
- Kế thừa để có tổ chức tốt, dễ quản lý dự án phần mềm, tránh rũi ro, giảm chi phí bảo trì.
- Chú ý:
  - Tài sản của lớp cha là Fields, Properties, Methods…
  - Lớp con không thể kế thừa các thành viên khai báo với private
  - Không thể kế thừa constructor (mỗi lớp phải xây dựng các constructor riêng)

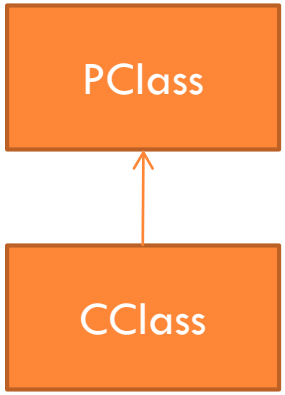


VÍ DỤ KẾ THỪA

```
public class PClass
{
    public String PPro{get; set;}
    public void PMethod(){}
}
```

```
// định nghĩa lớp kế thừa PClass
public class CClass : PClass
{
    public void CMethod(){}
}
```

```
// tạo đối tượng
CClass Obj = new CClass();
// sử dụng field của lớp cha
Obj.PPro = "Hello";
// gọi phương thức của chính lớp con
Obj.CMethod();
// gọi phương thức của lớp cha
Obj.PMethod();
```



Overide và Overload

- Overload:
  - Overload là trường hợp trong mỗi class có nhiều phương thức cùng tên nhưng khác nhau về cú pháp (tham số)
  - Khi gọi phương thức nào, cần truyền đúng tham số của phương thức đó.
- Overide
  - Được sử dụng để viết đè lên phương thức của lớp cha.
  - Phương thức của lớp con phải cùng cú pháp vơi phương thức lớp cha nhứng phương thức lớp cha phải khai báo với từ khóa **virtual** con con phải khai báo với **override**

Ví dụ: Overload và Override

```
public class PClass
{
    public void virtual M1(){}
}
```

```
// tạo đối tượng
PClass Obj = new CClass();
// gọi phương thức của lớp con
Obj.M1();
```

// định nghĩa lớp kế thừa PClass

```
public class CClass : PClass
{
    public void override M1(){}
    // sau đâu là overload
    public void M2(String a){}
    public void M2(int a){}
}
```