







LẬP TRÌNH C# 1

BÀI 4: LỚP VÀ ĐỐI TƯỢNG




- ⊙ Chương trình hướng đối tượng
- ⊙ Phương thức



Phần I: Chương trình hướng đối tượng

-  Khái niệm lớp và đối tượng
-  Khai báo trường, phương thức
-  Sử dụng hàm tạo
-  Đặc tả truy xuất

Phần II: Phương Thức

-  Sử dụng phương thức
-  Tính đóng gói
-  Tham biến và tham trị



- ☐ Tổ chức code hợp lý giúp quá trình gỡ lỗi và bảo trì dễ dàng
- ☐ Lập trình hướng lệnh
- ☐ Lập trình hướng hàm
- ☐ Lập trình hướng cấu trúc
- ☐ Lập trình hướng đối tượng



- ❑ Lập trình hướng đối tượng (Object-oriented programming) là kỹ thuật lập trình được thiết kế để giải quyết các chương trình bằng cách tạo ra các đối tượng mô phỏng các thực thể bên ngoài của vấn đề cần giải quyết.
- ❑ Lập trình hướng đối tượng sẽ định nghĩa các đối tượng bao gồm thuộc tính (data) và phương thức (thao tác với dữ liệu).

- ❑ Biểu diễn đối tượng trong thế giới thực
- ❑ Mỗi đối tượng được đặc trưng bởi các thuộc tính và các hành vi riêng của nó



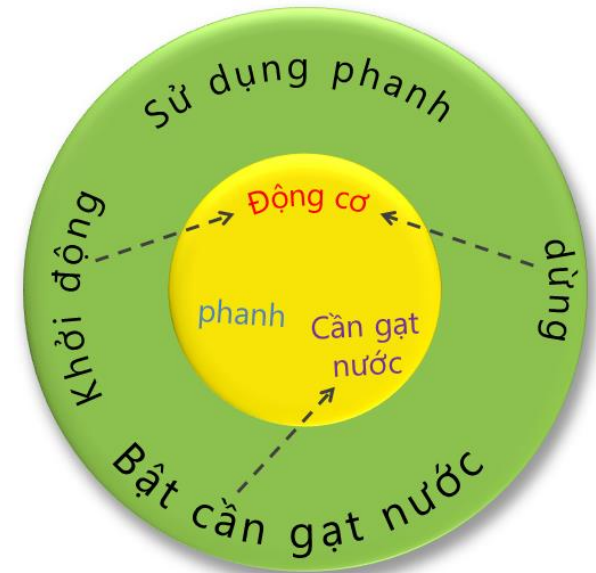
□ Đặc điểm

- Hãng sản xuất
- Model
- Năm
- Màu

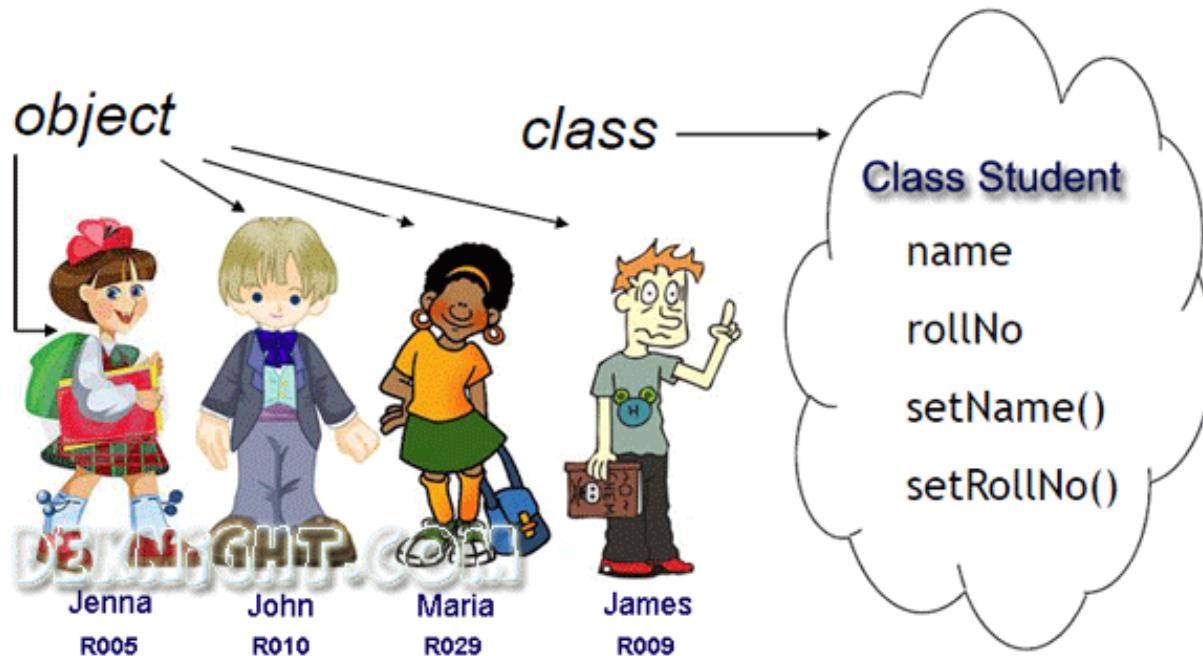


□ Hành vi (Ô tô có thể làm gì?)

- Khởi động
- Dừng
- Phanh
- Bật cần gạt nước



- ❑ Lớp là một khuôn mẫu được sử dụng để mô tả các đối tượng cùng loại.
- ❑ Lớp bao gồm các thuộc tính (trường dữ liệu) và các phương thức (hàm thành viên)



❑ Thuộc tính (field)

- ❖ Hãng sản xuất
- ❖ Model
- ❖ Năm
- ❖ Màu

Danh từ



❑ Phương thức (method)

- ❖ Khởi động()
- ❖ Dừng()
- ❖ Phanh()
- ❖ Bật cần gạt nước()

Động từ



- ❑ Class khai báo khuôn và bản mẫu, Object đại diện cho một thực thể theo bản mẫu đó



class Con Chó



các thực thể của class Con Chó

```
class <<ClassName>>
```

```
{
```

```
<<type>> <<field1>>;
```

Khai báo các trường

```
...
```

```
<<type>> <<fieldN>>;
```

Khai báo các phương thức

```
<<type>> <<method1>>([parameters]) {
```

```
// body of method
```

```
}
```

```
...
```

```
<<type>> <<methodN>>([parameters]) {
```

```
// body of method
```

```
}
```

```
}
```

VÍ DỤ LỚP MÔ TẢ HÌNH TRÒN

```
public class HìnhTron
{
    public double bankinh;
    References
    public double getChuvi()
    {
        return 2 * Math.PI * this.bankinh;
    }
    References
    public double getDientich()
    {
        return Math.PI * Math.Pow(this.bankinh,2);
    }
    References
    public void Xuat()
    {
        |
    }
}
```

- ✓ Lớp HìnhTron
- ✓ Trường bankinh
- ✓ 3 phương thức
 - getChuVi()
 - getDienTich()
 - Xuat()

- ❑ Đoạn mã sau sử dụng lớp HìnhTron để tạo một hình tròn sau đó gán dữ liệu cho trường banKinh và gọi các phương thức print().

```
class Program
{
    //References
    static void Main(string[] args)
    {
        HìnhTron ht = new HìnhTron();
        ht.bankinh = 10;
        ht.getChuvi();
    }
}
```

- ❑ Chú ý:

- ❖ Toán tử **new** được sử dụng để tạo đối tượng
- ❖ Biến ht chứa tham chiếu tới đối tượng
- ❖ Sử dụng dấu chấm (.) để truy xuất các thành viên của lớp (trường và phương thức).



DEMO

Tạo lớp mô tả sinh viên bao gồm họ
tên, điểm và các phương thức nhập,
xuất và xếp loại học lực



- ❑ Phương thức hay còn gọi là hàm thành viên
- ❑ Một phương thức là một nhóm lệnh cùng nhau thực hiện một tác vụ
- ❑ Cú pháp

```
<<kiểu trả về>> <<tên phương thức>> ([danh sách tham số])  
{  
    // thân phương thức  
}
```

```
public class Employee{  
    public String fullname;  
    public double salary;
```

```
    public void input(){...}  
    public void output(){...}
```

Kiểu trả về là **void** nên thân phương thức **không chứa lệnh return giá trị**

```
    public void setInfo(String fullname, double salary) {  
        this.fullname = fullname;  
        this.salary = salary;  
    }
```

Kiểu trả về là **double** nên thân phương thức phải chứa lệnh **return số thực**

```
    public double incomeTax(){  
        if(this.salary < 5000000){  
            return 0;  
        }  
        double tax = (this.salary - 5000000) * 10/100;  
        return tax;  
    }  
}
```


- ❑ **this** được sử dụng để đại diện cho đối tượng hiện tại.
- ❑ **this** được sử dụng trong lớp để tham chiếu tới các thành viên của lớp (field và method)
- ❑ Sử dụng **this.field** để phân biệt field với các biến cục bộ hoặc tham số của phương thức

```
public class MyClass{  
    int field;  
    void method(int field){  
        this.field = field;  
    }  
}
```

Trường

Tham số

- ❑ Hàm tạo là một phương thức đặc biệt được sử dụng để tạo đối tượng.
- ❑ Đặc điểm của hàm tạo
 - ❖ Tên trùng với tên lớp
 - ❖ Không trả về giá trị
- ❑ Ví dụ

Lớp

```
public class ChuNhat{  
    double dai, rong;  
    public ChuNhat(double dai, double rong){  
        this.dai = dai;  
        this.rong = rong;  
    }  
}
```

Đối tượng

```
ChuNhat cn1 = new ChuNhat(20, 15);  
ChuNhat cn2 = new ChuNhat(50, 25);
```

HÀM TẠO (CONSTRUCTOR)

- ❑ Trong một lớp có thể định nghĩa nhiều hàm tạo khác tham số. Mỗi hàm tạo cung cấp 1 cách tạo đối tượng.
- ❑ Nếu không khai báo hàm tạo thì Java tự động cung cấp hàm tạo mặc định (không tham số)

```
public class ChuNhat {  
    public double dai, rong;  
    public ChuNhat(double dai, double rong) {  
        this.dai = dai;  
        this.rong = rong;  
    }  
    public ChuNhat(double canh) {  
        this.dai = canh;  
        this.rong = canh;  
    }  
}
```

```
public static void main(String[] args) {  
    ChuNhat cn = new ChuNhat(20, 15);  
    ChuNhat vu = new ChuNhat(30);  
}
```

SinhVien

+ hoTen: String
+ diemTB: double

+ xepLoai(): String
+ xuất(): void
+ nhập(): void

+ SinhVien()
+ SinhVien(hoTen, diemTB)

DEMO



Xây dựng lớp mô tả sinh viên như mô hình trên.
Trong đó nhập() cho phép nhập họ tên và điểm từ bàn phím; xuất() cho phép xuất họ tên, điểm và học lực ra màn hình; xepLoai() dựa vào điểm để xếp loại học lực
Sử dụng 2 hàm tạo để tạo 2 đối tượng sinh viên



LẬP TRÌNH C# 1

BÀI 4: LỚP VÀ ĐỐI TƯỢNG(P2)

- ❑ Namespace được sử dụng để chia các class và interface thành từng gói khác nhau.
 - ❖ Việc làm này tương tự quản lý file trên ổ đĩa trong đó class (như file) và namespace (như folder)
- ❑ Mục đích sử dụng namespace là phân hoạch không gian các định danh, các kiểu dữ liệu thành những vùng để quản lý hơn, nhằm tránh sự xung đột giữa việc sử dụng các thư viện khác nhau từ các nhà cung cấp
- ❑ Ví dụ: lớp MyClass thuộc gói com.poly

```
Namespace com.poly;  
public class MyClass{...}
```

□ Khai báo một Namespace

```
namespace MyLib
{
    namespace Demo1
    {
        class Example1
        {
            public static void Show1()
            {
                Console.WriteLine("Lop Example1");
            }
        }
    }
    namespace Demo2
    {
        public class Tester
        {
            public static void Main()
            {
                Demo1.Example1.Show1();
                Demo1.Example2.Show2();
            }
        }
    }
}
```

```
namespace NamespaceName
```

```
{
```

```
// nơi chứa đựng tất cả các class
```

```
}
```

- ❑ C# đưa ra từ khóa `using` để khai báo không gian tên cho việc sử dụng các định danh, kiểu dữ liệu định nghĩa thuộc không gian tên trong chương trình
- ❑ Ví dụ: `using System;`
 - ❖ cho phép ta sử dụng `Console.WriteLine()` thay cho `System.Console.WriteLine()`
- ❑ Ví dụ: `using Demo1;`
 - ❖ cho phép ta truy cập `Example1.Show1()` thay cho `Demo1.Example1.Show1();`

❑ Đặc tả truy xuất được sử dụng để định nghĩa khả năng cho phép truy xuất đến các thành viên của lớp. Trong C# có các đặc tả khác nhau:

Độ truy cập (Modifier)	Mô tả
private	Truy cập bị hạn chế trong phạm vi của định nghĩa Class. Đây là loại phạm vi truy cập mặc định nếu không được chính thức chỉ định
protected	Truy cập bị giới hạn trong phạm vi định nghĩa của Class và bất kỳ các class con thừa kế từ class này.
internal	Truy cập bị giới hạn trong phạm vi Assembly của dự án hiện tại.
protected internal	Truy cập bị giới hạn trong phạm vi Assembly hiện tại và trong class định nghĩa hoặc các class con.
public	Không có bất kỳ giới hạn nào khi truy cập vào các thành viên công khai (public)

- ❑ Assembly: là file đã được biên dịch(precompiled), có 2 dạng (.exe và .dll)
- ❑ Assembly có thể chứa một hoặc nhiều namespace

	Cùng Assembly			Khác Assembly	
	Trong class định nghĩa?	Trong class con	Ngoài class định nghĩa, ngoài class con	Trong class con	Ngoài class con
private	Y				
protected	Y	Y		Y	
internal	Y	Y	Y		
protected internal	Y	Y	Y		
public	Y	Y	Y	Y	Y

ĐẶC TẢ TRUY XUẤT

```
namespace AccessModifierTutorial
{
```

```
class Person
```

```
{
    private string Name;
```

private field

```
    public Person(string name)
```

```
    {
        this.Name = name;
    }
```

```
    private void ShowSecret()
```

private method

```
    {
        Console.WriteLine("Secret of " + Name);
    }
```

```
    private static void DoSomething(String job)
```

```
    {
        Console.WriteLine("Do Job: " + job);
    }
```

```
class Diary
```

```
{
    public void Logging()
    {
```

```
        DoSomething("Code CSharp");
```

(private static)

```
        ShowSecret();
```

none-static

```
    }
}
```

```
namespace AccessModifierTutorial
{
```

```
class PersonTest
```

```
{
```

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

```
        Person tom = new Person("Tom");
```

```
        String name = tom.Name; // Error
```

```
        tom.ShowSecret(); // Error
```

```
        Person.DoSomething(); // Error
```

```
    }
```

```
}
```

```
_}
```



Bạn không thể truy cập vào các thành viên **private** từ bên ngoài class định nghĩa ra nó

- ❑ Encapsulation là tính che dấu trong hướng đối tượng.
 - ❖ Nên che dấu các trường dữ liệu
 - ❖ Sử dụng phương thức getter/setter để truy xuất các trường dữ liệu
- ❑ Mục đích của che dấu
 - ❖ Bảo vệ dữ liệu
 - ❖ Tăng cường khả năng mở rộng

- ❑ Xét lớp SinhVien và công khai hoTen và diem như sau

```
public class SinhVien{  
    public String hoTen;  
    public double diem;  
}
```

```
public class MyClass{  
    public static void main(String[] args){  
        SinhVien sv = new SinhVien();  
        sv.hoTen = "Nguyễn Văn Tèo";  
        sv.diem = 20.5;  
    }  
}
```

- ❑ Khi sử dụng người dùng có thể gán dữ liệu cho các trường một cách tùy tiện
- ❑ Điều gì sẽ xảy ra nếu điểm **hợp lệ chỉ từ 0 đến 10**

- ❑ Để che dấu hoàn toàn một trường, sử dụng đặc tả truy xuất private.

private double diem;

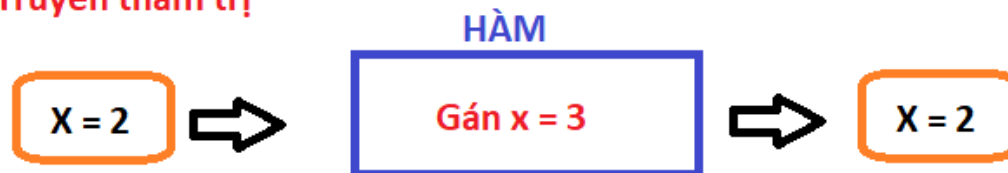
- ❑ Bổ sung các phương thức getter và setter để đọc ghi các trường đã che dấu

```
public void setDiem(double diem){  
    this.diem = diem;  
}  
  
public String getDiem() {  
    return this.diem;  
}
```

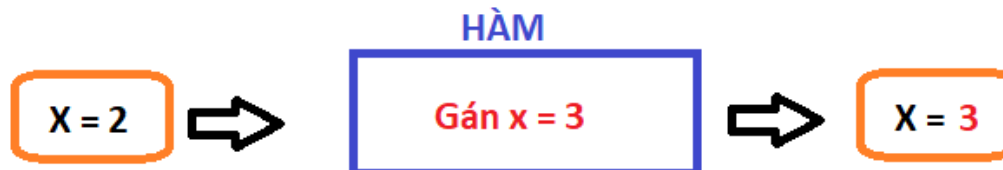
TRUYỀN THAM TRỊ & THAM CHIẾU

- ❖ Tham trị: một bản sao của biến sẽ được tạo ra, sao chép giá trị của biến, truyền biến đã được sao chép này vào hàm, dù có thực hiện bao nhiêu phép tính toán cũng không ảnh hưởng đến biến gốc
- ❖ Tham chiếu: truyền ngay địa chỉ của biến được lưu trên bộ nhớ vào hàm (hay hiểu cách khác là truyền chính biến đó vào hàm) khi thực hiện tính toán thì giá trị biến này thay đổi theo

Truyền tham trị



Truyền tham chiếu



- ❑ Sử dụng từ khóa ref hoặc out
 - ❖ Khi dùng ref: biến phải được khởi tạo trước khi truyền cho phương thức
 - ❖ Khi dùng out: biến không cần khởi tạo trước, bên trong phương thức cần gán giá trị cho biến
- ❑ Khi khai báo và gọi phương thức thì bắt buộc dùng ref hoặc out trước tên biến
- ❑ Không thể truyền vào một hằng vì hằng là giá trị không thay đổi

CÁCH SỬ DỤNG THAM CHIẾU

```
static void Main(string[] args)
{
    // Khai báo và khởi tạo
    int a = 100;
    int b = 99;

    Console.WriteLine( "Before swap: a = {0}, b = {1}", a, b );

    // Truyền 2 biến a, b theo kiểu Tham chiếu, dùng từ khóa [ref]
    Swap( ref a, ref b );
    Console.WriteLine( "After swap: a = {0}, b = {1}", a, b );

    // Chỉ khai báo
    int x;

    // Truyền biến x theo kiểu Tham chiếu, dùng từ khóa [out]
    GetValue( out x );

    Console.WriteLine( "Now, value of x = {0}", x );

    Console.ReadKey();
}

// Method hoán vị 2 số Swap, dùng biến tham chiếu [ref]
public static void Swap(ref int a, ref int b)
{
    int tmp;
    tmp = a;
    a = b;
    b = tmp;
}

// Method GetValue, dùng biến tham chiếu [out]
public static void GetValue(out int x)
{
    x = 5;
}
```


Tổng kết bài học

Phần I: Chương trình hướng đối tượng

 Khái niệm lớp và đối tượng


 Khai báo trường, phương thức

 Sử dụng hàm tạo

 Đặc tả truy xuất

Phần II: Phương Thức

 Sử dụng phương thức

 Tính đóng gói

 Tham biến và tham trị





KẾT THÚC