Hướng đối tượng: OOP => Object Oriented Programming

Lưu ý: 4 tính chất: - Đóng gói, Trừu tượng, kế thừa, đa hình

Đóng gói: Là cách để che dấu những tính chất xử lý bên trong của đối tượng, những đối tượng khác không thể tác động trực tiếp làm thay đổi trạng thái  chỉ có thể tác động thông qua các method public của đối tượng đó. Mình sẽ tạo ra 2 class để thể hiện điều này:

using System;

namespace oop

{

class hinhchunhat

{

private int height;

private int width;

public hinhchunhat(int newHeight, int newWidth) {

height = newHeight;

width = newWidth;

}

public int tinhdientich() {

return height \* width;

}

}

Class HinhTron{

}

Public HinhTron

{

}

}

Hàm khởi tạo chạy khi nào?

Ví dụ: khi khai báo và cấp phát bộ

* Trong cùng 1 lớp (class) có thể gọi toàn bộ các thuộc tính và phương thức trong lớp đó
* Còn đối tượng ngoài class ( lớp) thì chỉ có thể gọi các thuộc tính và phương thức là PUBLIC

**Tính kế thừa (Inheritance):**

Là kỹ thuật cho phép kế thừa lại những tính năng mà một đối tượng khác đã có, giúp tránh việc code lặp dư thừa mà chỉ xử lý công việc tương tự.

* Kế thừa một cấp (Single level Inheritance): Với một class cha và một class con
* using System;
* namespace oop
* {
* class hinhdang
* {
* private void showColor()
* {
* Console.WriteLine("Mau hong");
* }
* public void showType()
* {
* Console.WriteLine("Day la hinh chu nhat");
* }
* }
  + class Program : hinhdang
* {
* static void Main(string[] args)
* {
* Program pg = new Program();
* pg.showType();
* //pg.showColor(); khong the truy cap private method
* Console.ReadLine();
* }
* }
* }
* **Kế thừa: Class được kết thừa thì sẽ dùng hoặc có các thuộc tính và phương thức của class cha( chỉ kế thừa được public và protected)**

**Tính đa hình (Polymorphism ):**

Là một đối tượng thuộc các lớp khác nhau có thể hiểu cùng một thông điệp theo cách khác nhau.

Ví dụ đa hình trong thực tế: Mình có 2 con vật: chó, mèo hai con vật này khi nhận được mệnh lệnh  là **"hãy kêu"** thì chó kêu "gâu gâu",  mèo kêu "meo meo".

Ví dụ trên cả 2 con vật đều hiểu chung một thông điệp **"hãy kêu"** và thực hiện theo cách riêng của chúng.

Trong code để thể hiện tính đa hình có 2 cách:

1. Method Overloading (compile time polymorphism)
2. Method Overriding (run time polymorphism)

* **Method Overloading :** là cách nạp chồng các method có cùng tên nhưng khác tham số

using System;

namespace oop

{

class Program

{

static void Print(object o)

{

Console.WriteLine("Object overload called");

}

static void Print(string a)

{

Console.WriteLine("String overload called");

}

static void Main(string[] args)

{

object o = "hello";

Print(o);

Console.ReadLine();

}

}

}

* **Sửa – quá tải hàm, phương phức, toán với nội dung là tùy chỉnh**