**OPP**

**Trong 1 class:**

+ Fields: trường dữ liệu có trường cụ thể.

+ Properties: thuộc tính trung gian để lấy giá trị vào, ra cho Fields.

+ Constructor: hàm tạo , gán luôn cho các trường bên trong của nó.

Trong C# có thể có nhiều lớp.

+ Static: Từ khóa

* Phương thức static: không cần tạo đối tượng. tên class.method().
* Class static:
* Thuộc tính static: tạo ra một ô nhớ trong Ram, đối tượng khi đối tượng sử dụng thì cùng sử dụng.

+ overload: một phương thức có nhiều cách thể hiện khác nhau. Một lớp có thể có nhiều method cùng tên nhưng khác tham số truyền vào.

**\*Thừa kế: (inheritance)**

+ Lớp con: dùng từ khóa :base để kế thừa class cha. Có hai không gian bộ nhớ của cha và một là của con.

**Lớp ảo , cơ sở (Lớp trừu tượng) :** Abstract class:

* Sinh ra để kế thừa, không thể khởi tạo đối tượng lớp asbtract.

Java: class dẫn xuất : extends class abstract.

**\*Đa hình** :

1) Có nhiều đối tượng cùng kiểu . Chúng ta quản lí trong 1 mảng hoặc trong 1 danh sách.

- Đa hình là nó sẽ quản lí đối tượng ở lớp cha( lớp gốc ), không phải quản lí ở lớp con. Từ lớp này các lớp khác sẽ thừa kế.

2) Sử dụng giao diện chung cho nhiều phương thức khác nhau.

- Từ khóa New sử dụng trong phần kế thừa chỉ là định nghĩa lại phiên bản mới trong lớp dẫn xuất.

+ override: ghi lại hàm của lớp cha khi kế thừa.

* Từ khóa virtual gán vào method cha.
* Từ khóa override gán vào method con.

**\*Đóng gói:**

Private – Protected – Public

* Private: bên trong class
* Protected: trong class và lớp con kế thừa.
* Public: truy cập mọi nơi.

**Quan hệ kết hợp ( Association ) Has – a relationship:**

Has – a : khi mà một class này là một kiểu của class khác.

* Môi quan hệ giữa hai lớp là hai mối quan hệ sử dụng, không có mối quan hệ kế thừa.

Is – a( thừa kế ): class con của class cha.

**Interface:**

**1 )**Một tập một nhóm các hàm trừu tượng, thuộc tính trừu tượng đã được liệt kê sẵn.

+ hỗ trợ đa thừa kế.

+ method là public abstract.

* Chứa thuộc tính. Không chứa kiểu giữ liệu cụ thể.
* Từ khóa is hoặc as để kiểm tra có tồn tại trong cái gì đấy.
* Ép kiểu cha về kiểu con.

**2)** Interface như là một kiểu, tự thân nó gọi và thực hiện được.

**2 Qúa tải toán tử (Operator Overloading)**

Dùng để quá tải một phép toán nào đó trên đối tượng.

Mục đích: - Viết lại chương trình gọn gàng, dễ hiểu hơn, thay vì phải gọi phương thức.

- Khi quá tải toán tử dạng: + , - ,\*, /, % thì toán tử + = , -=, \*= , /= , %= cũng tự động được quá tải.

- Khi quá tải toán tử thì nên quá tải theo cặp đối ngẫu. Chẳng hạn, khi quá tải toán tử == thì quá tải thêm toán tử !=,….

- Khi quá tải toán tử ==, != thì nên phát triển thêm các phương thức Equals(), GetHashCode() để đảm bảo luật “ Hai đối tượng bằng nhau theo toán tử == hoặc phương thức Equals sẽ có cùng mã bắm ”.