BÁO CÁO TASK 1

# Các khái niệm về hướng đối tượng

## Đối tượng (object)

* Đối tượng là duy nhất.
* Là các đối tượng trong thế giới thật: bàn, ghế, xe, …
* Là các khái niệm như: Khoa, phòng ban, …
* Mỗi đối tượng (Ví dụ: ô tô) đều có:
* Các thông tin, trạng thái: VD: màu sắc, tốc độ, năm sx, …
* Các hoạt động: VD: Tăng ga, phanh, giảm tốc, đi đến một địa điểm, …

## Lớp (Class)

* Các đối tượng có đặc tính tương tự nhau được gom chung lại thành lớp đối tượng.

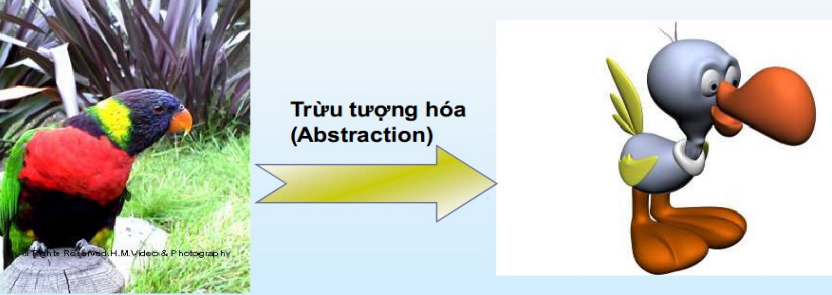
Ví dụ: Người là một lớp đối tượng.

* Lớp định nghĩa các thuộc tính và phương thức chung cho tất cả các đối tượng của cùng một loại nào đó.
* Các thành phần của lớp
* Tên lớp (class name): duy nhất, dùng để phân biệt lớp với lớp khác trong cùng một phạm vi.
* Thành phần dữ liệu, thuộc tính (data members, attributes): là các thành phần cấu tạo nên đối tượng, đặc trưng cho đối tượng.
* Phương thức (methods/functional members): các hoạt động của đối tượng thuộc lớp.

## Các tính chất của hướng đối tượng

### Tính trừu tượng (abstraction)

Là cách nhìn đơn giản hóa về một đối tượng mà trong đó chỉ bao gồm những đặc điểm được quan tâm và bỏ qua những chi tiết không cần thiết.



### Tính đóng gói (encapsulation)

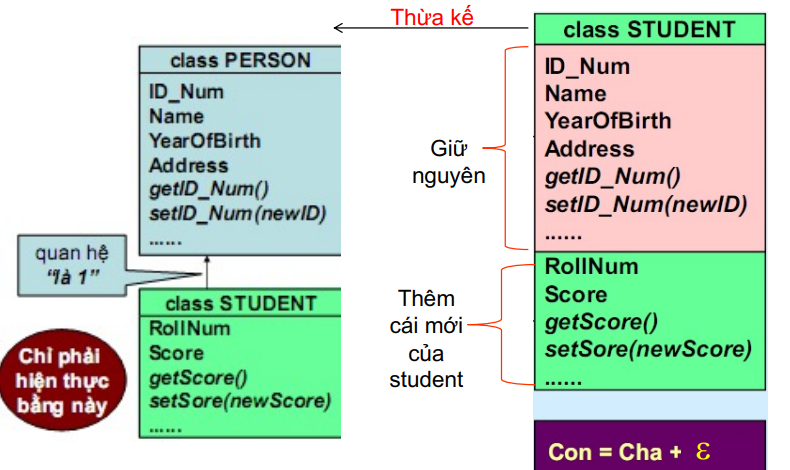
* Là việc che giấu việc thực thi chi tiết của một đối tượng.
* Người dùng không phụ thuộc vào việc sửa đổi sự thực thi bên trong.



### Tính kế thừa (inheritance)

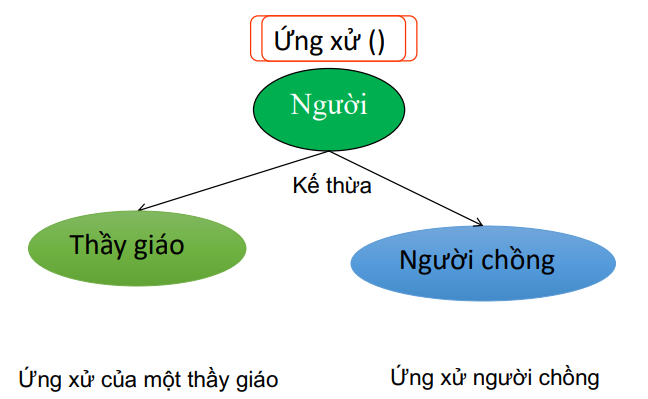
* Các lớp khác nhau nhưng có thể có chung một số đặc điểm.
* Có thể đặc tả lớp đối tượng mới từ những lớp đối tượng đã có. Khi đó, lớp mới mang đặc tính của lớp đã có (di truyền) + đặc tính riêng của nó.

Con = Cha + e (riêng của con mà cha không có)



### Tính đa hình (polymorphism)

* Là khả năng thực hiện khác nhau cho cùng một hành vi.



* Là kỹ thuật cho phép thay đổi nội dung của cùng một hành vi trong hai lớp cha – con.

## Ưu điểm của lập trình hướng đối tượng

* Thông qua nguyên lý kế thừa, có thể loại bỏ được những đoạn chương trình lặp lại trong quá trình mô tả các lớp và mở rộng khả năng sử dụng các lớp đã được xây dựng.
* Những hệ thống hướng đối tượng dễ mở rộng, nâng cấp thành những hệ lớn hơn.
* Có thể quản lý được độ phức tạp của những sản phẩm phần mềm.
* Nguyên lý che giấu thông tin giúp người lập trình tạo ra được những chương trình an toàn không bị thay bởi những đoạn chương trình khác.

# Các nguyên tắc lập trình hướng đối tượng

## Single-Responsibility Principle

Mỗi class chỉ nên đảm nhận một chức năng.

Ví dụ: Có hai class Person và Account. Cả hai đều có trách nhiệm duy nhất để lưu trữ thông tin cụ thể của họ. Nếu chúng ta muốn thay đổi trạng thái của Person thì chúng ta không cần phải thay đổi Account class và ngược lại.

## Open-Closed Principle

Các đối tượng hoặc thực thể nên được mở để mở rộng, nhưng đóng để sửa đổi. Có nghĩa là một lớp nên dễ dàng mở rộng mà không sửa đổi chính lớp đó.

## Liskov Substitution Principle

Là một biến thể của Open-Closed Principle. Thể hiện của lớp con có thể thay thế cho lớp cha mà không thay đổi tính đúng đắn của chương trình.

## Interface Segregation Principle

Nó được áp dụng cho các interfaces như là nguyên tắc trách nhiệm duy nhất giữ cho các lớp. Nội dung nguyên lí: Một clients không bao giờ bị buộc phải thực hiện một interface mà nó không sử dụng hoặc các client không nên bị buộc phải phụ thuộc vào các phương thức mà họ không sử dụng.

## Dependency inversion principle - Nguyên lý đảo ngược phụ thuộc

### Khái niệm

Là nguyên lý cuối cùng trong bộ nguyên lý SOLID. Là quy tắc lập trình mà trong đó modules cấp cao không nên phụ thuộc vào module cấp thấp, mà cả module cấp cao và module cấp thấp nên phụ thuộc vào abstraction. Ngoài ra, nguyên tắc Dependency Inversion còn quy định rằng các class phải giao tiếp với nhau thông qua giao diện (interfaces) chứ không được qua implementation.

### Nội dung nguyên lý

* Các module cấp cao không nên phụ thuộc vào các module cấp thấp. Cả 2 nên phụ thuộc vào abstraction.
* Interface (abstraction) không nên phụ thuộc vào chi tiết, mà ngược lại. (Các class giao tiếp với nhau thông qua interface, không phải thông qua implementation)