PHƯƠNG PHÁP LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Giảng viên: Đỗ Ngọc Như Loan

CÁC PHƯƠNG PHÁP LẬP TRÌNH

- Phương pháp lập trình là các cách tiếp cận giúp cho quá trình cài đặt hiệu quả hơn.
- Một số phương pháp lập trình cổ điển:
  - Lập trình tuyến tính
  - Lập trình cấu trúc/ thủ tục

LẬP TRÌNH TUYẾN TÍNH

- Chương trình là một dãy các lệnh.
- Lập trình là xây dựng các lệnh trong dãy lệnh.
- Không mang tính thiết kế

- **Ưu điểm:** Chương trình đơn giản, dễ hiểu. Lập trình tuyến tính được ứng dụng cho các chương trình đơn giản.
- **Nhược điểm:**
  - Không sử dụng lại được các đoạn mã
  - Mọi dữ liệu trong chương trình là toàn cục. Dữ liệu có thể bị sửa đổi ở bất cứ vị trí nào trong chương trình

```
class Program
{
        public static void main (String[] args) {
                int num1, num2, sum;
                Scanner sc = new Scanner(System.in);
                System.out.println("Nhap so thu nhat: ");
                num1 = sc.nextInt();
                System.out.println("Nhap so thu hai: ");
                num2 = sc.nextInt();
                sum = num 1 + num 2;
                System.out.println("Tong: " + sum);
}
```

LẬP TRÌNH CẤU TRÚC/ THỦ TỤC

- Chương trình chính được chia nhỏ thành các chương trình con (thủ tục, hàm) và mỗi chương trình con thực hiện một công việc xác định. Chương trình sẽ gọi đến chương trình con theo 1 giải thuật hoặc 1 cấu trúc được xác định trong chương trình chính.
- **Ưu điểm**
  - Chương trình sáng sủa, dễ hiểu, dễ theo dõi.
  - Tư duy giải thuật rõ ràng.
- **Nhược điểm**
  - Giải thuật luôn phụ thuộc chặt chẽ vào cấu trúc dữ liệu, do đó, khi thay đổi cấu trúc dữ liệu, phải thay đổi giải thuật.
  - Không hỗ trợ sử dụng lại mã nguồn và không phù hợp với các phần mềm lớn

```
class Program {
        static int num1, num2, sum;
        public static void main (String[] args) {
                nhap();
                tong();
                xuat();
        }
        static void nhap() {
                Scanner sc = new Scanner(System.in);
                System.out.println("Nhap so thu nhat: ");
                num1 = sc.nextInt();
                System.out.println("Nhap so thu hai: ");
                num2 = sc.nextInt();
        }
        static void tong() {   sum = num 1 + num 2;   }
        static void xuat() {  System.out.println("Tong: " + sum);   }
}
```

HẠN CHẾ VÀ CÁCH KHẮC PHỤC

Hạn chế của các phương pháp lập trình cổ điển

- **Không quản lí được sự thay đổi dữ liệu** khi có nhiều chương trình cùng thay đổi một biến chung, đặc biệt là khi ứng dụng ngày càng lớn.
- Giải thuật gắn liền với cấu trúc dữ liệu, nếu thay đổi cấu trúc dữ liệu, sẽ phải thay đổi giải thuật => **viết lại chương trình** từ đầu

Cách khắc phục:

- **Đóng gói dữ liệu** để hạn chế sự truy cập tự do vào dữ liệu
- Cho phép **sử dụng lại** mã nguồn, hạn chế việc phải viết lại mã từ đầu cho các chương trình

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

- Ra đời năm 1960
- Phương pháp lập trình hướng đối tượng (Object Oriented Programming - OOP) lấy đối tượng làm nền tảng để xây dựng chương trình. Dựa trên kiến trúc lớp (class) và đối tượng (object).
- Phân tích bài toán thành các thực thể được gọi là đối tượng và sau đó xây dựng các dữ liệu cùng các hàm xung quanh các đối tượng đó.
- Các ngôn ngữ lập trình hướng đối tượng
  - Java
  - C++
  - C#
  - VB.NET
  - PHP
  - …

ĐỐI TƯỢNG

- **Đối tượng** trong thế giới thực là những thực thể tồn tại và có thể được nhận biết, có thể là hữu hình (một con người) hoặc vô hình (một sự kiện).
- **Đối tượng** trong phần mềm: dùng để biểu diễn các đối tượng trong thế giới thực.

➔ Một đối tượng là 1 thực thể bao gồm thuộc tính và hành động (phương thức)

Ví dụ : Một người A

- Có các **thuộc tính**: tên, tuổi, địa chỉ, màu mắt…
- Có các **hành động**: đi, nói, ăn, ngủ …

ĐỐI TƯỢNG

Thuộc tính + phương thức = **Đối tượng**

- **Thuộc tính** (property): mô tả đối tượng, thể hiện đặc trưng của đối tượng

- **Phương thức** (method): là hành vi của chính đối tượng

LỚP

- **Lớp đối tượng** thể hiện cho một nhóm các đối tượng giống nhau (cùng thuộc tính và hành động)
- Các lớp có thể kế thừa nhau, để tận dụng các thuộc tính và hành động của nhau
- **Ví dụ:**

Lớp **SINHVIEN** có

- Thuộc tính: Họ tên, giới tính, ngày tháng năm sinh, điểm trung bình, đối tượng ưu tiên, ...
- Phương thức: Học bài, làm bài thi, làm bài tập, chơi game ...

Sinh viên Nguyễn Văn A, Lý Thị B là đối tượng thuộc lớp **SINHVIEN**

THIẾT KẾ THEO HƯỚNG ĐỐI TƯỢNG

- Trừu tượng hóa dữ liệu và các hàm/thủ tục liên quan
- Chia hệ thống ra thành các lớp/đối tượng
- Mỗi lớp/đối tượng có các tính năng và hành động chuyên biệt
- Các lớp có thể được sử dụng để tạo ra nhiều đối tượng cụ thể

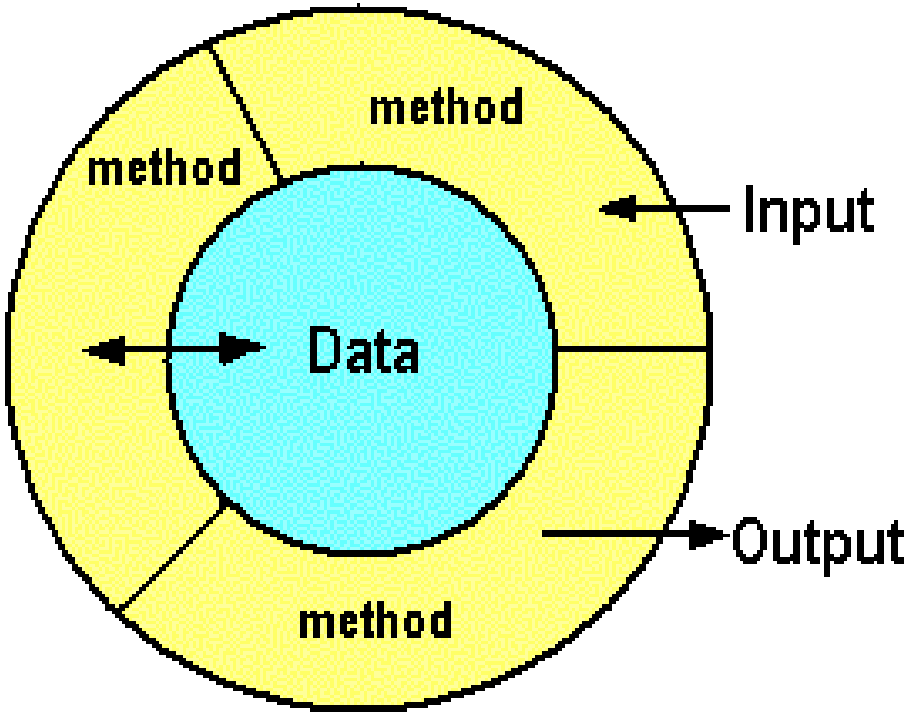
ĐẶC ĐIỂM CỦA OOP

3 đặc điểm chính cơ bản:

- Tính đóng gói (encapsulation)
- Tính kế thừa (inheritance)
- Tính đa hình (polymorphism)

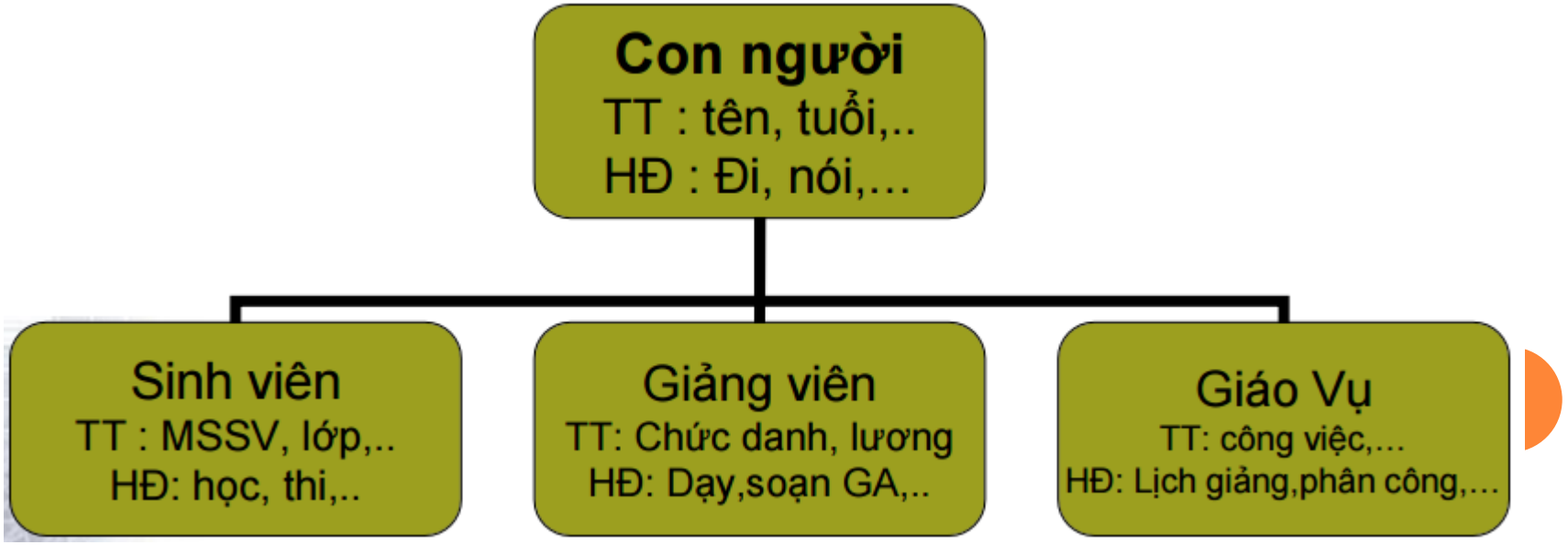
TÍNH ĐÓNG GÓI

- Đây là quan điểm trung tâm trong lập trình hướng đối tượng với việc che dấu dữ liệu thông qua sự đóng gói.
- Sự đóng gói chính là khả năng cất giữ riêng biệt dữ liệu và phương thức tác động lên dữ liệu đó.



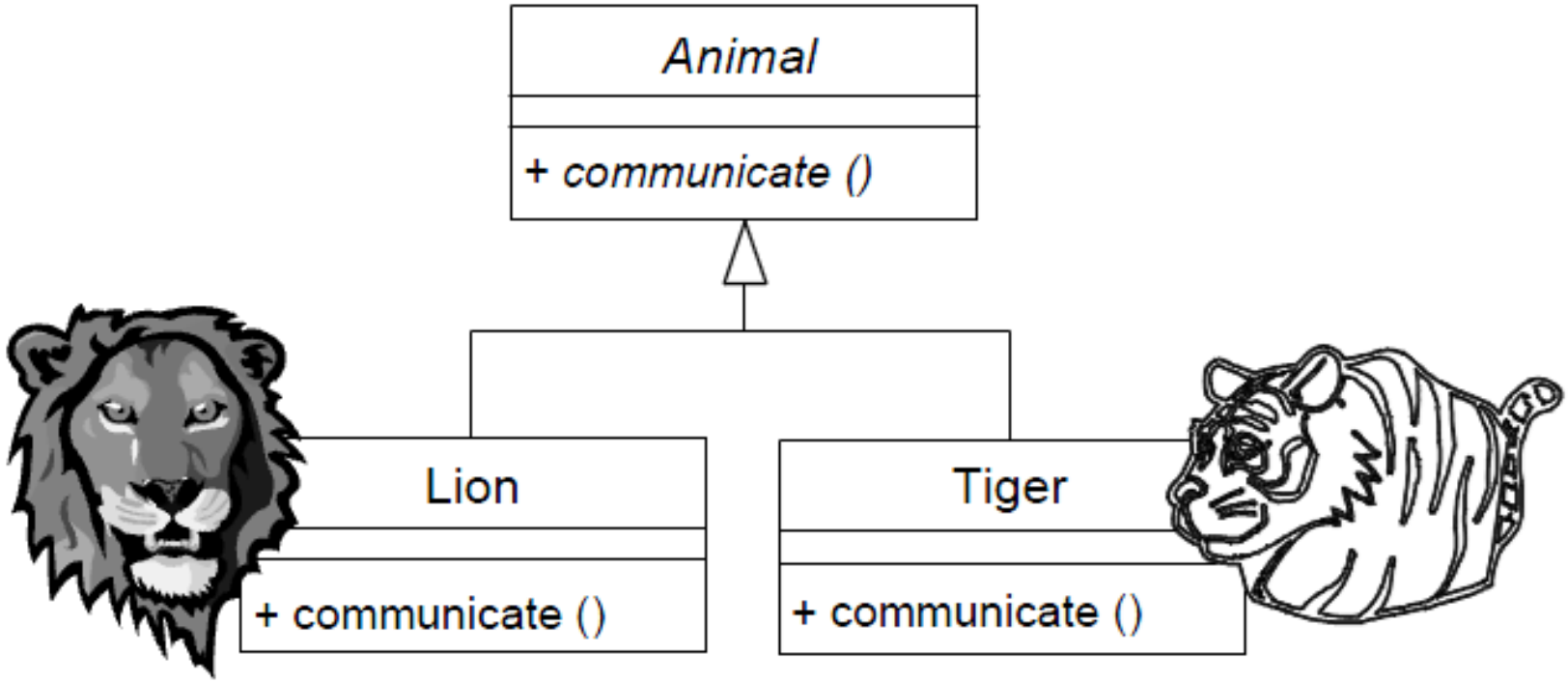
TÍNH KẾ THỪA

- Mục tiêu của lập trình hướng đối tượng là sử dụng lại những gì đã được xây dựng.
- Với tính thừa kế chúng ta không phải mất công xây dựng lại từ đầu những đối tượng mới, chỉ cần bổ sung để có được những đặc tính cần thiết trong đối tượng mới.



TÍNH ĐA HÌNH

- Cho phép gởi cùng một thông điệp đến những đối tượng khác nhau cùng có chung một đặc điểm
- Nếu cùng một hành động (phương thức), ứng dụng cho các đối tượng thuộc các lớp khác nhau thì có thể đưa đến những kết quả khác nhau



Without Polymorphism

```
if animal = "Lion" then
        Lion communicate
else if animal = "Tiger" then
        Tiger communicate
end
```

With Polymorphism

Animal communicate