OBJECT-ORIENTED PROGRAMMING (OOP)

PROCEDURE-ORIENTED PROGRAMMING (POP)

- Tổ chức chương trình thành các **chương trình con**
  - PASCAL: thủ tục & hàm, C: hàm.
- Chương trình hướng cấu trúc = cấu trúc dữ liệu + tập hợp hàm.
- Trừu tượng hóa chức năng (Functional Astraction):
  - Không quan tâm đến cấu trúc hàm;
  - Chỉ cần biết kết quả thực hiện của hàm.

→ Nền tảng của lập trình hướng cấu trúc.

- Tại sao phải thay đổi CTDL:
  - Cấu trúc dữ liệu là mô hình của bài toán cần giải quyết
    - Do thiếu kiến thức về bài toán, về miền ứng dụng, ... không phải lúc nào cũng tạo được CTDL hoàn thiện ngay từ đầu;
    - Tạo ra một cấu trúc dữ liệu hợp lý luôn là vấn đề đau đầu của người lập trình.
  - Bản thân bài toán cũng không bất biến:
    - Cần phải thay đổi cấu trúc dữ liệu để phù hợp với các yêu cầu thay đổi.

- Vấn đề đặt ra khi thay đổi CTDL:
  - Thay đổi cấu trúc:
    - Dẫn đến việc sửa lại mã chương trình (thuật toán) tương ứng và làm chi phí phát triển tăng cao;
    - Không tái sử dụng được các mã xử lý ứng với cấu trúc dữ liệu cũ.
  - Đảm bảo tính đúng đắn của dữ liệu:
    - Một trong những nguyên nhân chính gây ra lỗi phần mềm là gán các dữ liệu không hợp lệ;
    - Cần phải kiểm tra tính đúng đắn của dữ liệu mỗi khi thay đổi giá trị.

- Vấn đề đặt ra khi thay đổi CTDL:
  - Ví dụ: struct Date

```
struct Date
{
    int year, month, day;
};
Date d;
d.day = 32;                    // invalid day
d.day = 31; d.month = 2;       // how to check
d.day = d.day + 1;
```

  - Thay đổi CTDL

```
struct Date
{
    short year;
    short mon_n_day;
};
```

TIẾP CẬN HƯỚNG ĐỐI TƯỢNG

- Xuất phát từ hai hạn chế chính của Lập trình hướng cấu trúc:
  - Không quản lý được sự thay đổi dữ liệu khi có nhiều chương trình cùng thay đổi một biến chung;
  - Không tiết kiệm được tài nguyên: giải thuật gắn liền với CTDL, nếu CTDL thay đổi, sẽ phải thay đổi giải thuật.
- Phương pháp tiếp cận mới: phương pháp lập trình hướng đối tượng. Với hai mục đích chính:
  - Đóng gói, che dấu dữ liệu: (che dấu cấu trúc) để hạn chế sự truy nhập tự do vào dữ liệu. Truy cập dữ liệu thông qua giao diện xác định;
  - Cho phép sử dụng lại mã nguồn, hạn chế việc viết mã lại từ đầu.

TIẾP CẬN HƯỚNG ĐỐI TƯỢNG

- **Đóng gói** được thực hiện theo phương pháp trừu tượng hóa đối tượng từ thấp lên cao:
  - Thu thập các thuộc tính của mỗi đối tượng, gắn các thuộc tính vào đối tượng tương ứng;
  - Nhóm các đối tượng có thuộc tính tương tự nhau thành nhóm, loại bỏ các thuộc tính cá biệt, chỉ giữ lại các thuộc tính chung nhất. Đây gọi là quá trình trừu tượng hóa đối tượng thành lớp;
  - Đóng gói các dữ liệu của đối tượng vào lớp tương ứng. Mỗi thuộc tính của đối tượng trở thành thuộc tính của lớp tương ứng;
  - Việc truy nhập dữ liệu được thực hiện thông qua các phương thức được trang bị cho lớp;
  - Khi có thay đổi trong dữ liệu của đối tượng, chỉ thay đổi các phương thức truy nhập thuộc tính của lớp, mà không cần thay đổi mã nguồn của chương trình sử dụng lớp tương ứng.

TIẾP CẬN HƯỚNG ĐỐI TƯỢNG

- **Tái sử dụng mã nguồn** được thực hiện thông qua cơ chế **kế thừa** trong lập trình hướng đối tượng
  - Các lớp có thể được kế thừa nhau để tận dụng các thuộc tính, các phương thức của nhau;
  - Trong lớp dẫn xuất (lớp được thừa kế) có thể sử dụng lại các phương thức của lớp cơ sở mà không cần cài đặt lại mã nguồn;
  - Khi lớp dẫn xuất định nghĩa lại phương thức cho mình, lớp cơ sở cũng không bị ảnh hưởng và không cần thiết sửa đổi lại mã nguồn.

TIẾP CẬN HƯỚNG ĐỐI TƯỢNG

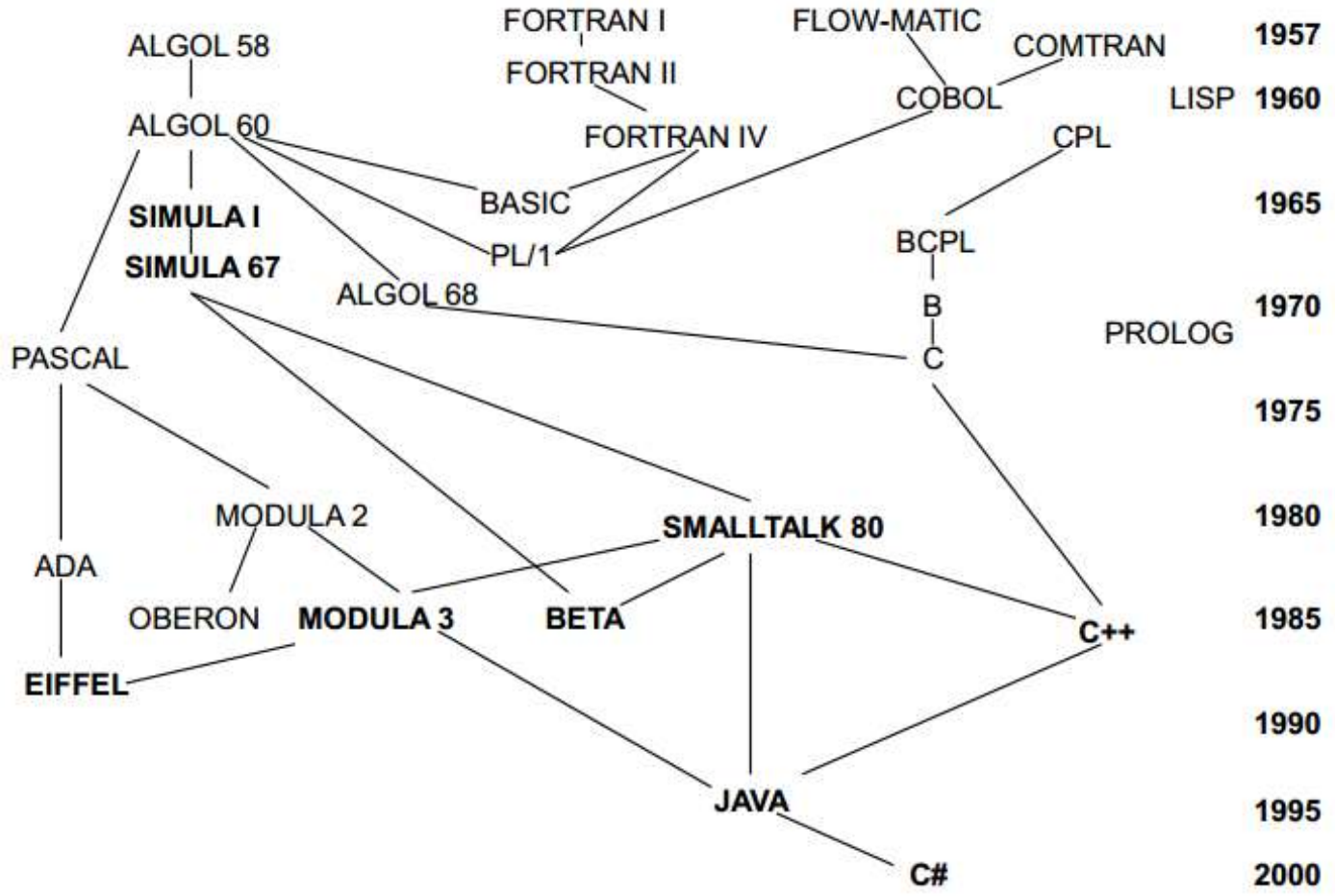
- Ưu điểm:
  - Không có nguy cơ dữ liệu bị thay đổi tự do trong chương trình;
  - Khi thay đổi cấu trúc dữ liệu của một đối tượng, không cần thay đổi mã nguồn của các đối tượng khác, mà chỉ cần thay đổi một số thành phần của đối tượng bị thay đổi;
  - Có thể sử dụng lại mã nguồn, tiết kiệm được tài nguyên;
  - Phù hợp với dự án phần mềm lớn, phức tạp.

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

- Một số hệ thống "hướng đối tượng" thời kỳ đầu không có các lớp, chỉ có các "đối tượng" và các "thông điệp" (v.d. Hypertalk).
- Hiện giờ, đã có sự thống nhất rằng hướng đối tượng là:
  - Lớp (class);
  - Thừa kế (inheritance) và liên kết động (dynamic binding).
- Một số đặc tính của lập trình hướng đối tượng có thể được thực hiện bằng C hoặc các ngôn ngữ lập trình thủ tục khác;
- Điểm khác biệt sự hỗ trợ và ép buộc ba khái niệm trên được cài hẳn vào trong ngôn ngữ;
- Mức độ hướng đối tượng của các ngôn ngữ không giống nhau:
  - Eiffel (tuyệt đối), Java (rất cao), C++ (nửa nọ nửa kia).

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

- Lịch sử ngôn ngữ lập trình hướng đối tượng:



LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

- OOP là phương pháp lập trình:
  - Mô tả chính xác các đối tượng trong thế giới;
  - Lấy đối tượng làm nền tảng xây dựng thuật toán;
  - Thiết kế xoay quanh dữ liệu của hệ thống;
  - Chương trình được chia thành các lớp đối tượng;
  - Dữ liệu được đóng gói, che dấu và bảo vệ;
  - Đối tượng làm việc với nhau qua thông báo;
  - Chương trình được thiết kết theo cách từ dưới lên (bottom-up).

- Hệ thống Hướng Đối Tượng:
  - Gồm tập hợp các đối tượng:
    - Sự đóng gói của 2 thành phần:
      - Dữ liệu (thuộc tính của đối tượng);
      - Các thao tác trên dữ liệu.
  - Các đối tượng có thể kế thừa các đặc tính của đối tượng khác;
  - Hoạt động thông qua sự tương tác giữa các đối tượng nhờ cơ chế truyền thông điệp:
    - Thông báo;
    - Gửi & nhận thông báo.

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

- Hướng thủ tục:
  - Lấy hành động làm trung tâm.
  - Hàm là xương sống.
    - Lặt (Rau) - Ướp (Cá) - Luộc (Rau).
    - Kho (Cá) - Nấu (Cơm).
- Hướng đối tượng:
  - Lấy dữ liệu làm trung tâm.
  - Đối tượng là xương sống.
    - Rau.Lặt - Cá.Ướp - Rau.Luộc
    - Cá.Kho - Cơm.Nấu

Các bước nấu ăn	
Verb	**Object**
Lặt	Rau
Ướp	Cá
Nấu	Cơm
Kho	Cá
Luộc	Rau

Thay đổi tư duy lập trình!!

OBJECT & CLASS

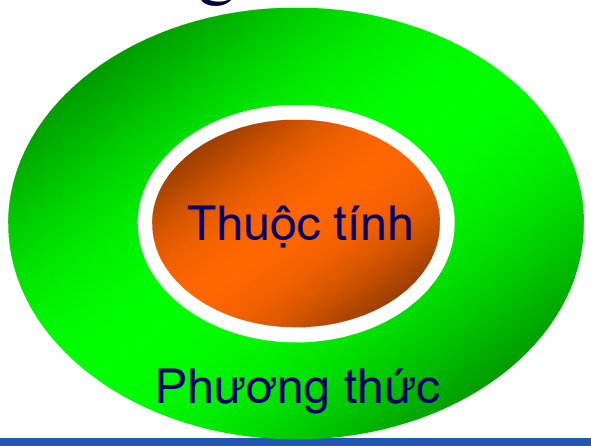
- Object (Đối tượng):
  - Chương trình là "cỗ máy" phức tạp.
  - Cấu thành từ nhiều loại "vật liệu".
  - Vật liệu cơ bản: hàm, cấu trúc.
  - **Đã đủ tạo ra chương trình tốt?**

→Vật liệu mới: **Đối tượng**

→ Viết một chương trình hướng đối tượng nghĩa là đang xây dựng một mô hình của một vài bộ phận trong thế giới thực.

OBJECT & CLASS

- Object (Đối tượng):
  - Đặc trưng:
    - Đóng gói cả dữ liệu và xử lý;
    - Thuộc tính (attribute): dữ liệu của đối tượng;
    - Phương thức (method): xử lý của đối tượng.
  - Cấu trúc:
    - Hộp đen: thuộc tính trong, phương thức ngoài.
    - Bốn nhóm phương thức:
      - Nhóm tạo hủy.
      - Nhóm truy xuất thông tin.
      - Nhóm xử lý nghiệp vụ.
      - Nhóm toán tử.



OBJECT & CLASS

- Object (Đối tượng): là một thực thể đang tồn tại trong hệ thống và được xác định bằng ba yếu tố:
  - *Định danh đối tượng*: xác định duy nhất cho mỗi đối tượng trong hệ thống, nhằm phân biệt các đối tượng với nhau;
  - *Trạng thái của đối tượng*: sự tổ hợp của các giá trị của các thuộc tính mà đối tượng đang có;
  - *Hoạt động của đối tượng*: là các hành động mà đối tượng có khả năng thực hiện được.

	Trạng thái	Hành động
Chiếc xe	Nhãn hiệu: "Ford" Màu sơn: Trắng Giá bán: 5000$	Khởi động Dừng lại Chạy

OBJECT & CLASS

- Object (Đối tượng):
  - Một đối tượng gồm:
    - Định danh;
    - Thuộc tính (dữ liệu);
    - Hành vi (phương thức).
  - Mỗi đối tượng bất kể đang ở trạng thái nào đều có định danh và được đối xử như một thực thể riêng biệt:
    - Mỗi đối tượng có một handle (trong C++ là địa chỉ);
    - Hai đối tượng có thể có giá trị giống nhau nhưng handle khác nhau.

OBJECT & CLASS

- Class (Lớp):
  - Đối tượng là một thực thể cụ thể, tồn tại trong hệ thống;
  - Lớp là một khái niệm trừu tượng, dùng để chỉ một tập hợp các đối tượng có mặt trong hệ thống.
  - Ví dụ:
    - Mỗi chiếc xe có trong cửa hàng là một đối tượng, nhưng khái niệm "xe hơi" là một lớp đối tượng dùng để chỉ tất cả các loại xe có trong cửa hàng.
  - ❖Lưu ý:
    - Lớp là một khái niệm, mang tính trừu tượng, dùng để biểu diễn một tập các đối tượng;
    - Đối tượng là một thể hiện cụ thể của lớp, là một thực thể tồn tại trong hệ thống.

OBJECT & CLASS

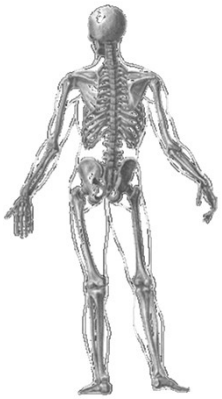
Person1:

- Name: Peter.
- Age: 25.
- Hair Color: Brown.
- Eye Color: Brown.
- Job: Worker.

Person2:

- Name: Thomas.
- Age: 50.
- Hair Color: White.
- Eye Color: Blue.
- Job: Teacher.

Tập hợp đối tượng có cùng thuộc tính và phương thức



Human:

- Name.
- Age.
- Hair Color.
- Eye Color.
- Job.

Bản mô tả đối tượng

Kiểu của đối tượng

OBJECT & CLASS

- Class (Lớp):
  - Một lớp có thể có một trong các khả năng sau:
    - Hoặc chỉ có thuộc tính, không có phương thức;
    - Hoặc chỉ có phương thức, không có thuộc tính;
    - Hoặc có cả thuộc tính, phương thức (phổ biến);
    - Đặc biệt, lớp không có thuộc tính, phương thức nào, gọi là lớp trừu tượng, các lớp này không có đối tượng.
  - Lớp và đối tượng mặc dù có mối liên hệ tương ứng lẫn nhau, nhưng lại khác nhau về bản chất.

Lớp	Đối tượng
Sự trừu tượng hóa của đối tượng	Là thể hiện của lớp
Là một khái niệm trừu tượng, chỉ tồn tại ở dạng khái niệm mô tả đặc tính chung của một số đối tượng	Là một thực thể cụ thể, có thực, tồn tại trong bộ nhớ
Là nguyên mẫu cho các đối tượng, xác định các hành vi và các thuộc tính cần thiết cho một nhóm các đối tượng cụ thể	Tất cả các đối tượng thuộc cùng một lớp có cùng các thuộc tính và hành động

OBJECT & CLASS

- Class (Lớp):
  - Trừu tượng hóa theo chức năng:
    - Mô hình hóa các phương thức của lớp dựa trên hành động của đối tượng.
  - Trừu tượng hóa theo dữ liệu:
    - Mô hình hóa các thuộc tính của lớp dựa trên thuộc tính của các đối tượng tương ứng.
  - Thuộc tính (attribute) là dữ liệu trình bày các đặc điểm về một đối tượng;
  - Phương thức (method): có liên quan tới những thứ mà đối tượng có thể làm. Một phương thức đáp ứng một chức năng tác động lên dữ liệu của đối tượng (thuộc tính).

OBJECT & CLASS

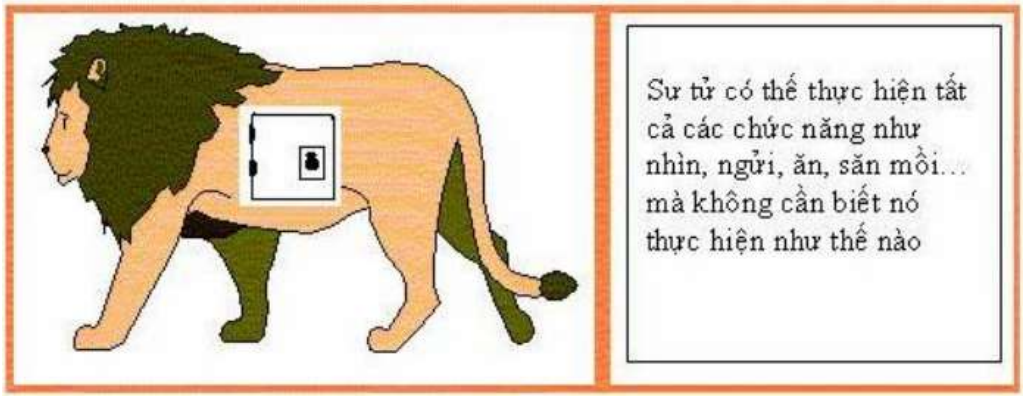
- Thuộc tính (attribute):
  - Là dữ liệu trình bày các đặc điểm về một đối tượng;
  - Bao gồm: Hằng, biến; Tham số nội tại.
  - Kiểu thuộc tính: Kiểu cổ điển; Kiểu do người dùng định nghĩa.
- Phương thức (method):
  - Có liên quan tới những thứ mà đối tượng có thể làm;
  - Một phương thức đáp ứng một chức năng tác động lên dữ liệu của đối tượng (thuộc tính);
  - Hàm nội tại của đối tượng (**hàm thành viên**);
  - Có kiểu trả về.

OBJECT & CLASS

- Thông điệp (message):
  - Là phương tiện để đối tượng này chuyển yêu cầu tới đối tượng khác, bao gồm:
    - Đối tượng nhận thông điệp;
    - Tên của phương thức thực hiện;
    - Các tham số mà phương thức cần.
- Truyền thông điệp:
  - Là cách một đối tượng triệu gọi một hay nhiều phương thức của đối tượng khác để yêu cầu thông tin;
  - Hệ thống yêu cầu đối tượng thực hiện phương thức:
    - Gửi thông báo và tham số cho đối tượng;
    - Kiểm tra tính hợp lệ của thông báo;
    - Gọi thực hiện hàm tương ứng phương thức.

OBJECT & CLASS

- Tính đóng gói (encapsulation):
  - Là tiến trình che giấu việc thực thi chi tiết của một đối tượng;
  - Khái niệm: là cơ chế ràng buộc dữ liệu và các thao tác trên dữ liệu thành thể thống nhất;
  - Đóng gói gồm:
    - Bao gói: người dùng giao tiếp với hệ thống qua giao diện;
    - Che dấu: ngăn chặn các thao tác không được phép từ bên ngoài.
  - Ưu điểm:
    - Quản lý sự thay đổi;
    - Bảo vệ dữ liệu.



OBJECT & CLASS

- Tính đóng gói (encapsulation):
  - Đóng gói → Thuộc tính được lưu trữ hay phương thức được cài đặt như thế nào → được che giấu đi từ các đối tượng khác;
  - Việc che giấu những chi tiết thiết kế và cài đặt từ những đối tượng khác được gọi là **ẩn thông tin**.

OBJECT & CLASS

- Tính đóng gói (encapsulation):
  - Ví dụ: bài toán quản lý nhân viên văn phòng với lớp **Nhân viên**:
  - Cách tính lương nhân viên là khác nhau với mỗi người: Tiền lương = Hệ số lương * lương cơ bản * Tỉ lệ phần trăm
  - Việc gọi phương thức tính tiền lương là giống nhau cho mọi đối tượng Nhân viên;
  - Sự giống nhau về cách sử dụng phương thức cho các đối tượng của cùng một lớp, nhưng cách thực hiện phương thức lại khác nhau với các đối tượng khác nhau gọi là sự đóng gói dữ liệu của lập trình hướng đối tượng.

OBJECT & CLASS

- Tính đóng gói (encapsulation):
  - Cho phép che dấu sự cài đặt chi tiết bên trong:
    - Chỉ cần gọi các phương thức theo một cách thống nhất;
    - Phương thức có thể cài đặt khác nhau cho các trường hợp khác nhau.
  - Cho phép che dấu dữ liệu bên trong đối tượng:
    - Khi sử dụng, không biết thực sự bên trong đối tượng có những gì;
    - Chỉ thấy được những gì đối tượng cho phép truy nhập vào.
  - Cho phép tối đa hạn chế việc sửa lại mã chương trình.

OBJECT & CLASS

- Tính thừa kế (inheritance):
  - Hệ thống hướng đối tượng cho phép các lớp được định nghĩa kế thừa từ các lớp khác;
  - Khái niệm:
    - Khả năng cho phép xây dựng lớp mới được thừa hưởng các thuộc tính của lớp đã có;
    - Các phương thức & thuộc tính được định nghĩa trong một lớp có thể được sử dụng lại bởi lớp khác.
  - Đặc điểm:
    - Lớp nhận được có thể bổ sung các thành phần;
    - Hoặc định nghĩa là các thuộc tính của lớp cha.
  - Các loại thừa kế: Đơn thừa kế & Đa thừa kế.

OBJECT & CLASS

- Tính thừa kế (inheritance):
  - Ví dụ:

Lớp *Nhânviên*	Lớp *Sinhviên*
Thuộc tính: Tên Ngày sinh Giới tính Lương	Thuộc tính: Tên Ngày sinh Giới tính Lớp
Phương thức: Nhập/Xem tên Nhập/Xem ngày sinh Nhập /Xem giới tính Nhập/Xem lương	Phương thức: Nhập/Xem tên Nhập/Xem ngày sinh Nhập /Xem giới tính Nhập/Xem lớp

OBJECT & CLASS

- Tính thừa kế (inheritance):
  - Hai lớp có một số thuộc tính và phương thức chung: tên, ngày sinh, giới tính:
    - Không thể loại bỏ thuộc tính cá biệt để gộp lại thành một lớp;
    - Thuộc tính lương và lớp là cần thiết cho việc quản lý nhân viên, sinh viên.
  - Vấn đề nảy sinh:
    - Lặp lại việc viết mã cho một số phương thức;
    - Phải lặp lại việc sửa mã chương trình nếu có sự thay đổi về kiểu dữ liệu.

OBJECT & CLASS

- Tính thừa kế (inheritance):

Lớp *Người*
- **Thuộc tính:**
  - **Tên**
  - **Ngày sinh**
  - **Giới tính**
- **Phương thức:**
  - **Nhập/Xem tên**
  - **Nhập/Xem ngày sinh**
  - **Nhập /Xem giới tính**

Lớp *Nhânviên* kế thừa từ lớp *Người*
- Thuộc tính:
  - Lương
- Phương thức:
  - Nhập/Xem lương

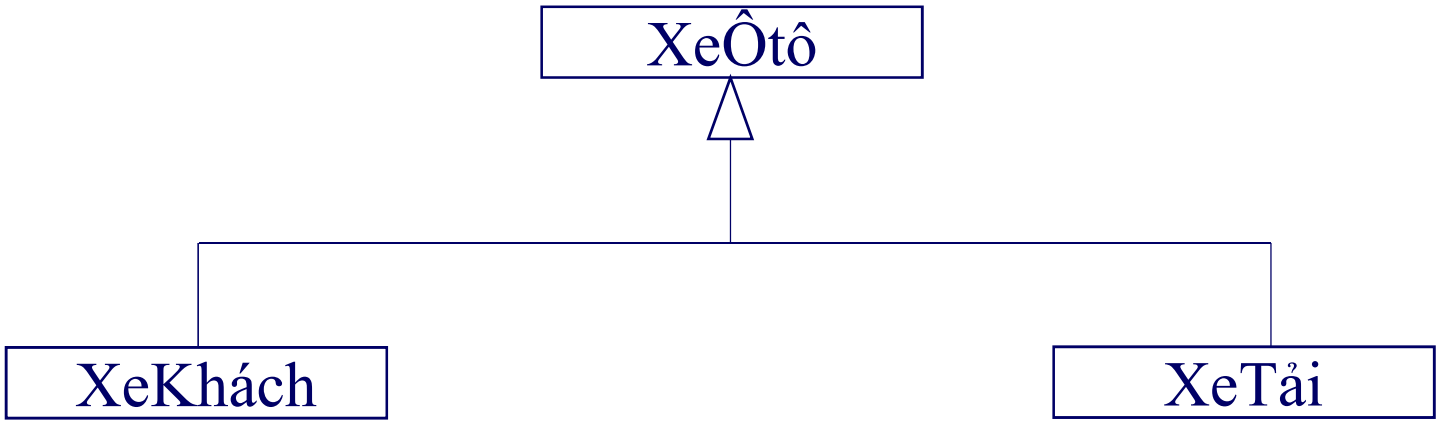
Lớp *Sinhviên* kế thừa từ lớp *Người*
- Thuộc tính:
  - Lớp
- Phương thức:
  - Nhập/Xem lớp

OBJECT & CLASS

- Tính thừa kế (inheritance):
  - Cho phép lớp dẫn xuất có thể sử dụng các thuộc tính và phương thức của lớp cơ sở tương tự như sử dụng thuộc tính và phương thức của mình;
  - Cho phép chỉ cần thay đổi phương thức của lớp cơ sở, có thể sử dụng được ở tất cả các lớp dẫn xuất;
  - Tránh sự cài đặt trùng lặp mã nguồn chương trình;
  - Chỉ cần thay đổi mã nguồn một lần khi thay đổi dữ liệu của các lớp.

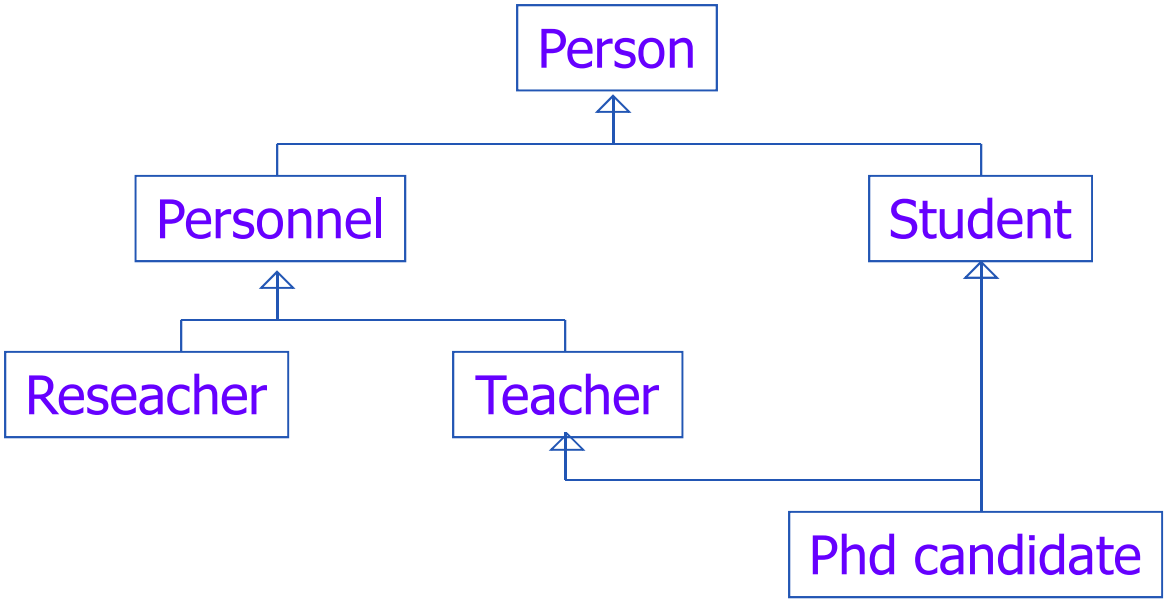
OBJECT & CLASS

- Tính thừa kế (inheritance):
  - Đơn thừa kế: một lớp con chỉ thừa kế từ một lớp cha duy nhất
  - Ví dụ:
    - Lớp trừu tượng hay lớp chung: XeÔtô;
    - Lớp cụ thể hay lớp chuyên biệt: XeKhách, XeTải;
    - Lớp chuyên biệt có thể thay thế lớp chung trong tất cả các ứng dụng.

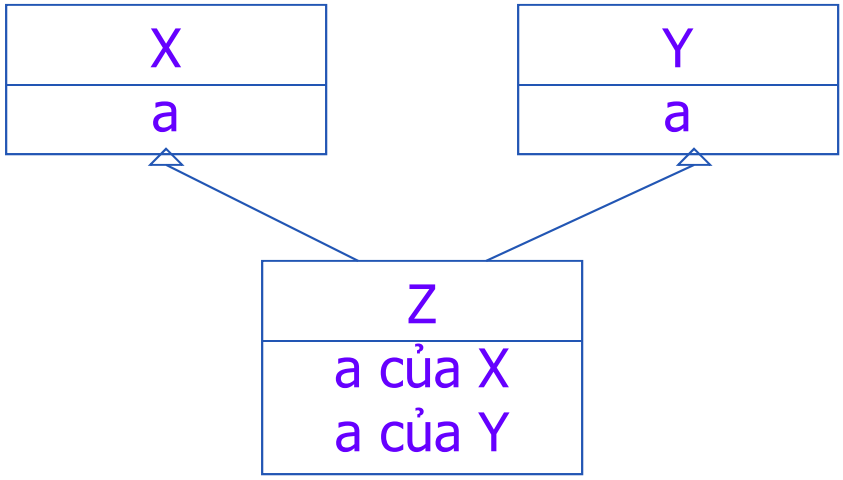


OBJECT & CLASS

- Tính thừa kế (inheritance):
  - Đa thừa kế: một lớp con thừa kế từ nhiều lớp cha khác nhau
  - Ví dụ:



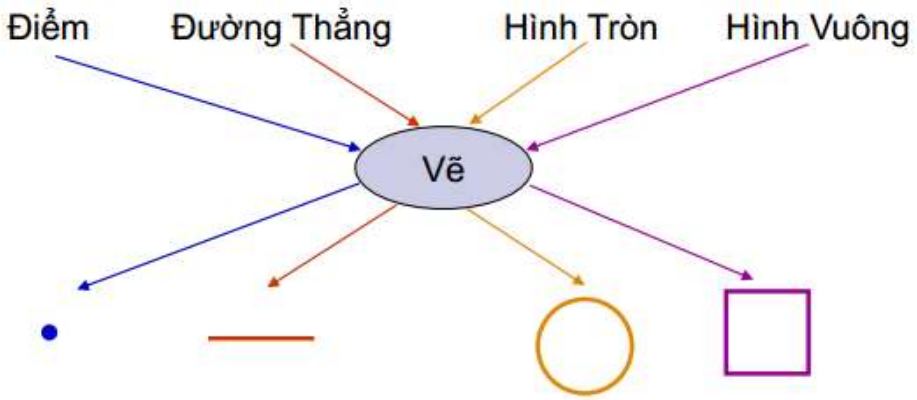
- Tính thừa kế (inheritance):
  - Đa thừa kế:
    - Đụng độ tên các thuộc tính:



    - Đa thừa kế không được chấp nhận bởi một số ngôn ngữ: Java, C#, …

OBJECT & CLASS

- Tính đa hình (polymorphism):
  - Đa hình: "nhiều hình thức", hành động cùng tên có thể được thực hiện khác nhau đối với các đối tượng/các lớp khác nhau.
  - Ngữ cảnh khác → kết quả khác;
  - Khái niệm:
    - Khả năng đưa một phương thức có cùng tên trong các lớp con.
  - Thực hiện bởi:
    - Định nghĩa lại;
    - Nạp chồng.
  - Cơ chế dựa trên sự gán:
    - Kết gán sớm;
    - Kết gán muộn.



OBJECT & CLASS

- Tính đa hình (polymorphism):
  - Gọi phương thức show() từ đối tượng của lớp Người sẽ hiển thị tên, tuổi của người đó;
  - Gọi phương thức show() từ đối tượng của lớp Nhân viên sẽ hiển thị số lương của nhân viên;
  - Gọi phương thức show() từ đối tượng của lớp Sinh viên sẽ biết sinh viên đó học lớp nào.

OBJECT & CLASS

- Tính đa hình (polymorphism):
  - Cho phép các lớp định nghĩa các phương thức trùng nhau: cùng tên, cùng tham số, cùng kiểu trả về:
    - Sự nạp chồng phương thức.
  - Khi gọi các phương thức trùng tên, dựa vào đối tượng đang gọi mà chương trình sẽ thực hiện phương thức của lớp tương ứng:
    - Kết quả sẽ khác nhau.

OBJECT & CLASS

- Tính đa hình (polymorphism):
  - Đa hình hàm - Functional polymorphism
    - Cơ chế cho phép một tên thao tác hoặc thuộc tính có thể được định nghĩa tại nhiều lớp và có thể có nhiều cài đặt khác nhau tại mỗi lớp trong các lớp đó;
    - Ví dụ: lớp Date cài 2 phương thức setDate(), một nhận tham số là một đối tượng Date, phương thức kia nhận 3 tham số day, month, year.
  - Đa hình đối tượng - Object polymorphism
    - Các đối tượng thuộc các lớp khác nhau có khả năng hiểu cùng một thông điệp theo các cách khác nhau;
    - Ví dụ: khi nhận được cùng một thông điệp draw(), các đối tượng Rectangle và Triangle hiểu và thực hiện các thao tác khác nhau.

Thank You !

IT FACULTY