ĐẠI NAM
UNIVERSITY

NGUYÊN LÝ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Thời gian: 4 tiết

NỘI DUNG

- Các kỹ thuật lập trình
- Kỹ thuật hướng đối tượng
- Các khái niệm cơ bản
- Các nguyên lý hướng đối tượng
- Phân tích thiết kế hướng đối tượng

Các kỹ thuật lập trình

Ngôn ngữ lập trình có thể được phân thành 3 loại chính:

- **Ngôn ngữ lập trình phi cấu trúc** (Unstructured Programming Languages): Ngôn ngữ lập trình nguyên thủy nhất có dòng điều khiển tuần tự. Code được lặp lại trong suốt chương trình.
- **Ngôn ngữ lập trình có cấu trúc** (Structured Programming Languages): Có luồng điều khiển không tuần tự. Việc sử dụng các hàm cho phép tái sử dụng code.
- **Lập trình hướng đối tượng** (Object Oriented Programming): Kết hợp dữ liệu và Hành động cùng nhau.

Lập trình phi cấu trúc

- Assembly: ~1951
- Phát triển Phần mềm (software developement) là một hoạt động khá phức tạp.
- Assembly là một ngôn ngữ phổ biến thời điểm đó, nó sử dụng các thao tác cấp thấp như “add”, “sub”, “goto” và thao tác trực tiếp các địa chỉ bộ nhớ.

Lập trình phi cấu trúc

- Việc xây dựng một ứng dụng đơn giản rất chậm và khó.
- Tạo một câu lệnh IF đơn giản, cần một vài dòng code và đối với một vòng lặp, sẽ mất nhiều hơn một vài dòng…
- Kiểu code tuyến tính và việc sử dụng lại code chỉ giới hạn trong phạm vi sao chép, dán trong cùng file hoặc giữa các files.

Lập trình phi cấu trúc

- Lập trình không cấu trúc là mô hình lập trình sớm nhất trong lịch sử có khả năng tạo ra các thuật toán hoàn chỉnh Turing.
- Sử dụng luồng điều khiển phi cấu trúc bằng cách sử dụng các câu lệnh goto hoặc tương đương.
- Bị tuyên bố coi là "có hại" vào năm 1968 bởi nhà khoa học máy tính người Hà Lan Edsger W. Dijkstra, người đã đặt ra thuật ngữ "lập trình có cấu trúc".

Lập trình phi cấu trúc

- Lập trình phi cấu trúc đã bị chỉ trích nặng nề vì tạo ra mã khó đọc.
- Có cả ngôn ngữ lập trình cấp cao và cấp thấp sử dụng lập trình không cấu trúc. Một số ngôn ngữ thường được coi là không có cấu trúc bao gồm JOSS, FOCAL, TELCOMP, hợp ngữ, tệp loạt MS-DOS và các phiên bản đầu tiên của BASIC, Fortran, COBOL và MUMPS .
- Một chương trình bằng ngôn ngữ không có cấu trúc sử dụng các bước nhảy không có cấu trúc đến nhãn hoặc địa chỉ hướng dẫn. Các dòng thường được đánh số hoặc có thể có nhãn: điều này cho phép luồng thực thi nhảy đến bất kỳ dòng nào trong chương trình. Điều này trái ngược với lập trình có cấu trúc sử dụng các cấu trúc có cấu trúc là lựa chọn (if / then / else) và lặp lại (while và for).

Lập trình hướng cấu trúc

- Lập trình có (hướng) cấu trúc hay còn gọi là lập trình hướng thủ tục (Procedure Oriented Programming - POP): là một kỹ thuật lập trình truyền thống, trong đó chương trình được chia thành các hàm (chương trình con)
- Mỗi chương trình con còn có thể được chia ra nhiều chương trình con khác để đơn giản hóa công việc của chúng.
- Một số ngôn ngữ hướng cấu trúc như C, Pascal...

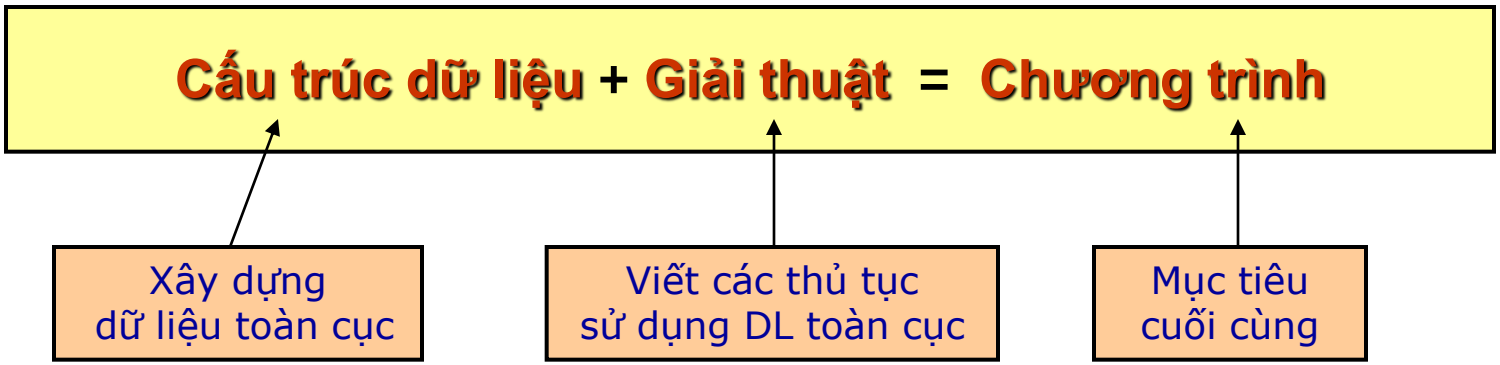
Đặc điểm lập trình hướng cấu trúc

- Tập trung vào công việc cần thực hiện (thuật toán)
- Chương trình lớn được chia thành các chương trình con, mỗi chương trình con có thể gọi tới một hoặc nhiều lần theo thứ tự bất kỳ.
- Phần lớn các hàm sử dụng dữ liệu chung
- Dữ liệu trong hệ thống được chuyển động từ hàm này sang hàm khác.
- Sử dụng cách tiếp cận top-down trong thiết kế chương trình

Ưu nhược điểm LTHCT

- *Ưu điểm:*
  - Tư duy giải thuật rõ ràng.
  - Đơn giản, dễ hiểu.
  - Cung cấp khả năng tái sử dụng cùng một mã tại nhiều nơi khác nhau.
  - Tạo điều kiện trong việc theo dõi dòng chương trình.
- *Nhược điểm:*
  - Trong lập trình hướng cấu trúc ta thường quan tâm đến việc phát triển các hàm mà ít quan tâm tới dữ liệu mà chúng dùng để xử lý công việc.
  - Không hỗ trợ sử dụng lại mã nguồn: mỗi cấu trúc dữ liệu chỉ phù hợp với một số giải thuật, khi thay đổi cấu trúc dữ liệu thì giải thuật phải thay đổi theo

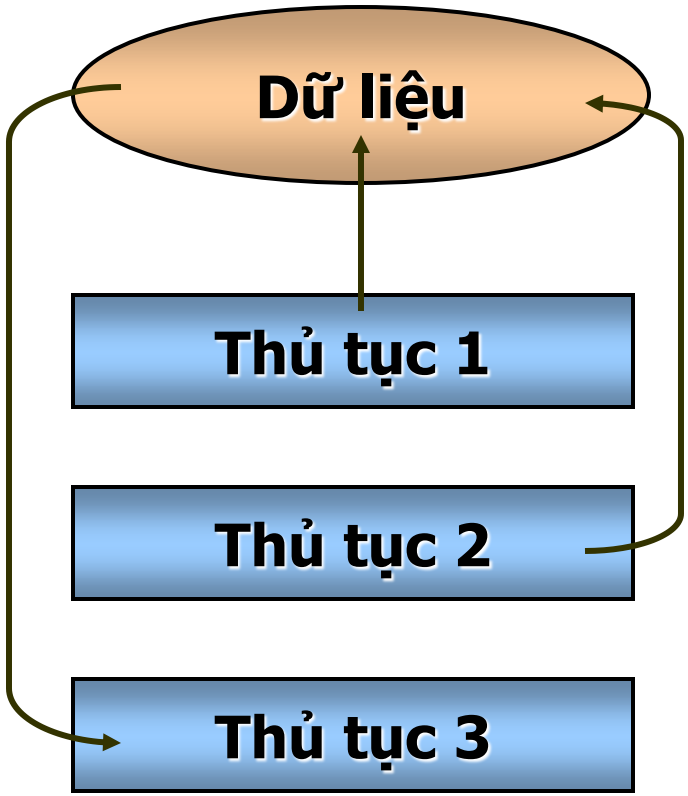
Lập trình cổ điển *(traditional, procedural)*



- Có sự tiến hóa từ:

  Lập trình tuyến tính => lập trình cấu trúc
- Tiếp cận theo hướng thủ tục.
- Chú trọng đến thủ tục hơn là về dữ liệu.

Lập trình cổ điển



- Dữ liệu và thủ tục được xử lý độc lập.
- **Không** quan tâm đến mối liên hệ giữa thủ tục và dữ liệu.
- Khó chỉnh sửa, thêm mới dữ liệu và thủ tục.
- Gặp khó khăn khi sử dụng lại.

Lập trình hướng đối tượng

- Lập trình hướng đối tượng (Object-Oriented Programming, viết tắt: OOP) là một mẫu hình lập trình dựa trên công nghệ đối tượng.
- Đối tượng chứa đựng các dữ liệu, trên các trường, thường được gọi là các thuộc tính.
- Mã nguồn được tổ chức thành các phương thức.
- Phương thức giúp cho đối tượng có thể truy xuất và hiệu chỉnh các trường dữ liệu của đối tượng khác, mà đối tượng hiện tại có tương tác (đối tượng được hỗ trợ các phương thức “this” hoặc “self”).

Hướng cấu trúc vs. Hướng ĐT

- Hướng cấu trúc:
  - data structures + algorithms = Program
  - (cấu trúc dữ liệu + giải thuật = Chương trình)
- Hướng đối tượng:
  - objects + messages = Program
  - (đối tượng + thông điệp = Chương trình)

Lập trình hướng đối tượng

Lập trình hướng đối tượng, với các mức độ hiển thị, phương thức (thông báo), đối tượng, lớp và sau này là các gói, giống như làm tăng tính đóng gói và mô-đun. Các đặc điểm:

- **Visibility levels** (Mức độ phạm vi): cho phép chúng ta điều khiển những gì code có thể truy cập vào một tập dữ liệu cụ thể.
- **Classes** (lớp): cho phép định nghĩa hay mô hình hóa các domain concepts.

Lập trình hướng đối tượng

- **Objects** (đối tượng): cho phép chúng ta có các instances khác nhau trong cùng domain concepts.
- **Packages** (gói): cho phép nhóm các lớp lại với nhau để thể hiện một domain hay khái niệm hàm và làm việc cùng nhau trên cùng một tác vụ.
- **Methods** (phương thức): đại diện cho các thủ tục và chức năng, nhưng về mặt khái niệm nên được xem như là các thông báo (hoặc tốt hơn là các lệnh) có thể được cấp cho một loại đối tượng cụ thể.

Lập trình hướng đối tượng

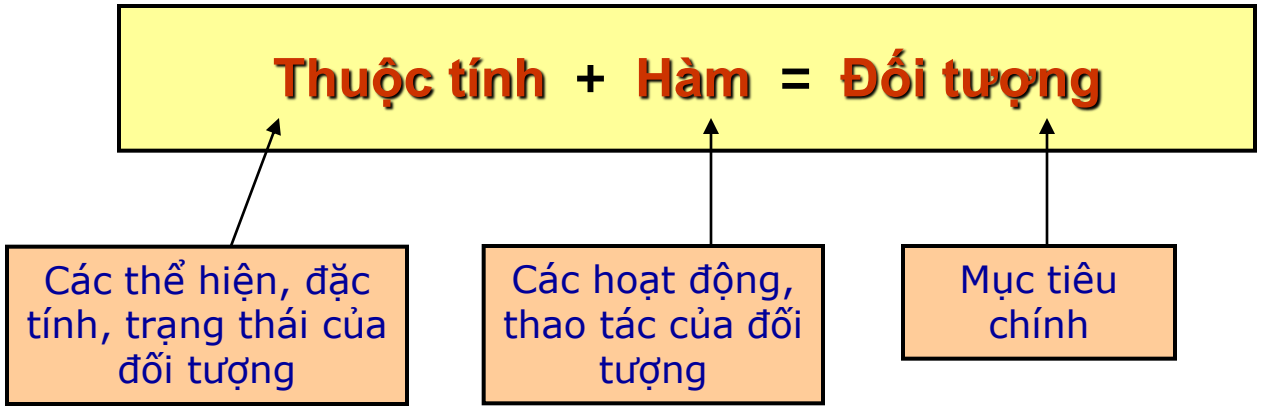
- Đối tượng chứa đựng các dữ liệu, trên các trường, thường được gọi là các thuộc tính; và mã nguồn, được tổ chức thành các phương thức.
- Phương thức giúp cho đối tượng có thể truy xuất và hiệu chỉnh các trường dữ liệu của đối tượng khác, mà đối tượng hiện tại có tương tác.

Đặc điểm lập trình hướng đối tượng

- Lập trình OOP giúp tăng năng suất và đơn giản hóa công việc xây dựng phần mềm, bảo trì phần mềm, cho phép lập trình viên tập trung vào các đối tượng giống như trong thực tế.
- Trong lập trình hướng đối tượng, chương trình máy tính được thiết kế bằng cách tách nó ra khỏi phạm vi các đối tượng tương tác với nhau.
- Ngôn ngữ lập trình hướng đối tượng khá đa dạng, phần lớn là các ngôn ngữ lập trình theo lớp, nghĩa là các đối tượng trong các ngôn ngữ này được xem như thực thể của một lớp, được dùng để định nghĩa một kiểu dữ liệu.

Thế nào là lập trình hướng đối tượng?

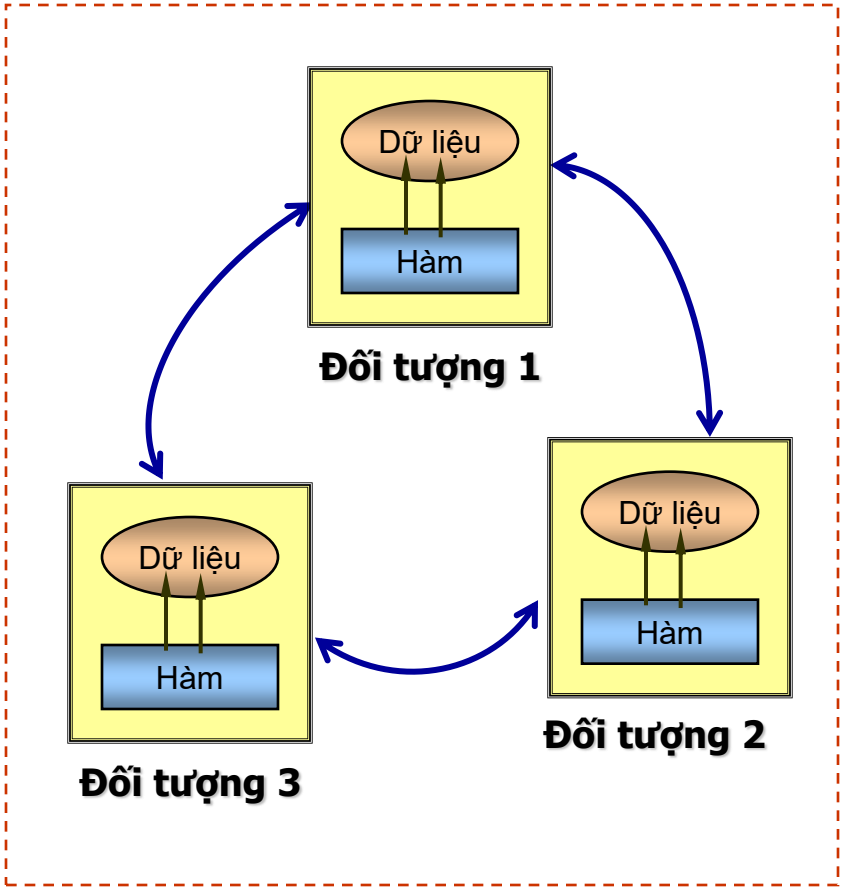
- **Lập trình hướng đối tượng**
  (Object Oriented Programming - OOP)



- Phân tích bài toán thành nhiều đối tượng.
- Quan tâm đến dữ liệu hơn thủ tục.
- Dữ liệu được bảo vệ (che giấu) và không thể truy xuất từ bên ngoài đối tượng.

Thế nào là lập trình hướng đối tượng?

- **Lập trình hướng đối tượng**



- Dữ liệu và thủ tục được xử lý chặc chẽ với nhau.
- Quan tâm đến mối liên hệ giữa thủ tục và dữ liệu.
- Dễ chỉnh sửa, thêm mới dữ liệu và hàm trong 1 đối tượng.
- Mục tiêu là tăng cường khả năng sử dụng lại.

Một số cơ chế trong lập trình HĐT

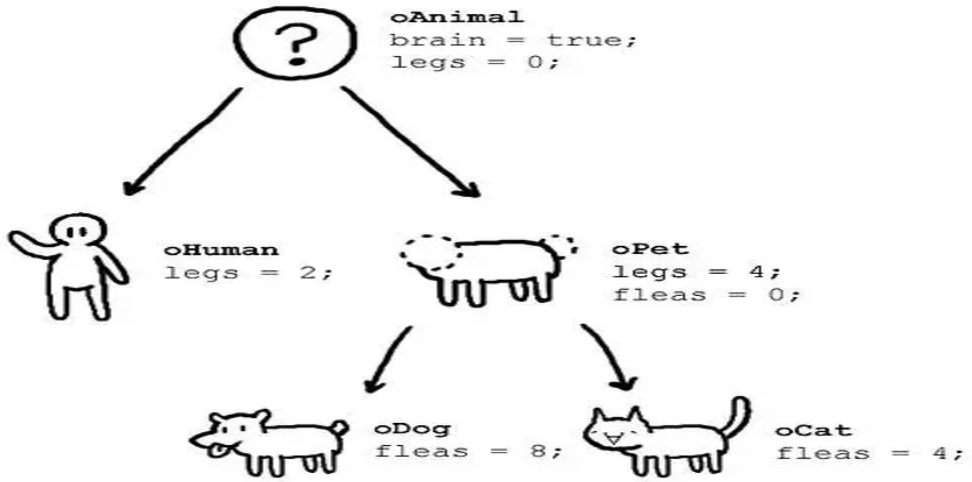
Tính chất LT hướng đối tượng

Lập trình hướng đối tượng là một phương pháp lập trình có 4 tính chất chính sau:

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

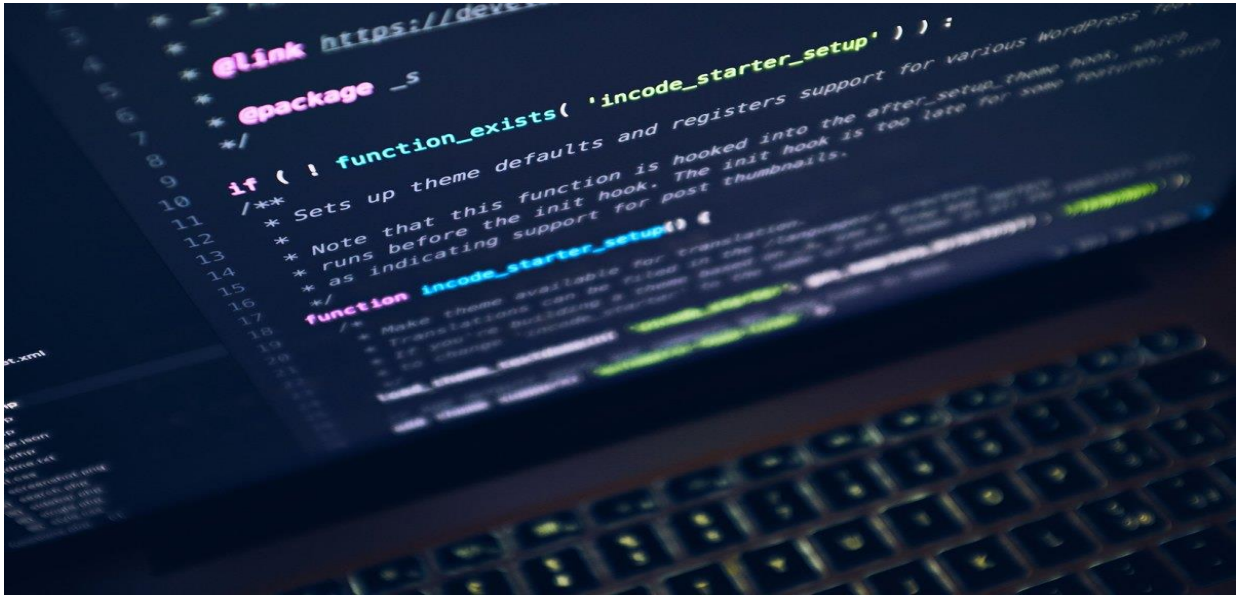
Tính trừu tượng (abstraction)

- Tính trừu tượng (abstraction) là khả năng của chương trình bỏ qua hay không chú ý đến một số khía cạnh của thông tin mà nó đang trực tiếp làm việc lên, nghĩa là nó có khả năng tập trung vào những cốt lõi cần thiết.
- Mỗi đối tượng phục vụ có thể hoàn tất các công việc một cách nội bộ, báo cáo, thay đổi trạng thái của nó và liên lạc với các đối tượng khác mà không cần cho biết làm cách nào đối tượng tiến hành được các thao tác.



Tính trừu tượng (abstraction)

- Tính trừu tượng còn thể hiện qua việc một đối tượng ban đầu có thể có một số đặc điểm chung cho nhiều đối tượng khác như là sự mở rộng của nó nhưng bản thân đối tượng ban đầu này có thể không có các biện pháp thi hành. Tính trừu tượng này thường được xác định trong khái niệm gọi là lớp trừu tượng hay lớp cơ sở trừu tượng.



Tính trừu tượng (abstraction)

- Tác động trên các dữ liệu cũng tương tự như sự trừu tượng hóa theo chức năng. Khi có trừu tượng hóa dữ liệu, các cấu trúc dữ liệu và các phần tử có thể được sử dụng mà không cần bận tâm đến các chi tiết cụ thể.
- Chẳng hạn như các số dấu chấm động đã được trừu tượng hóa trong tất cả các ngôn ngữ lập trình, Chúng ta không cần quan tâm cách biểu diễn nhị phân chính xác nào cho số dấu chấm động khi gán một giá trị, cũng không cần biết tính bất thường của phép nhân nhị phân khi nhân các giá trị dấu chấm động. Điều quan trọng là các số dấu chấm động hoạt động đúng đắn và hiểu được.
- Sự trừu tượng hóa dữ liệu giúp chúng ta không phải bận tâm về các chi tiết không cần thiết. Nếu lập trình viên phải hiểu biết về tất cả các khía cạnh của vấn đề, ở mọi lúc và về tất cả các hàm của chương trình thì chỉ ít hàm mới được viết ra.

Tính trừu tượng (abstraction)

Phụ thuộc vào góc nhìn

Registration

- Vehicle Identification Number
- License plate
- Current Owner
- Tax due, date

Owner

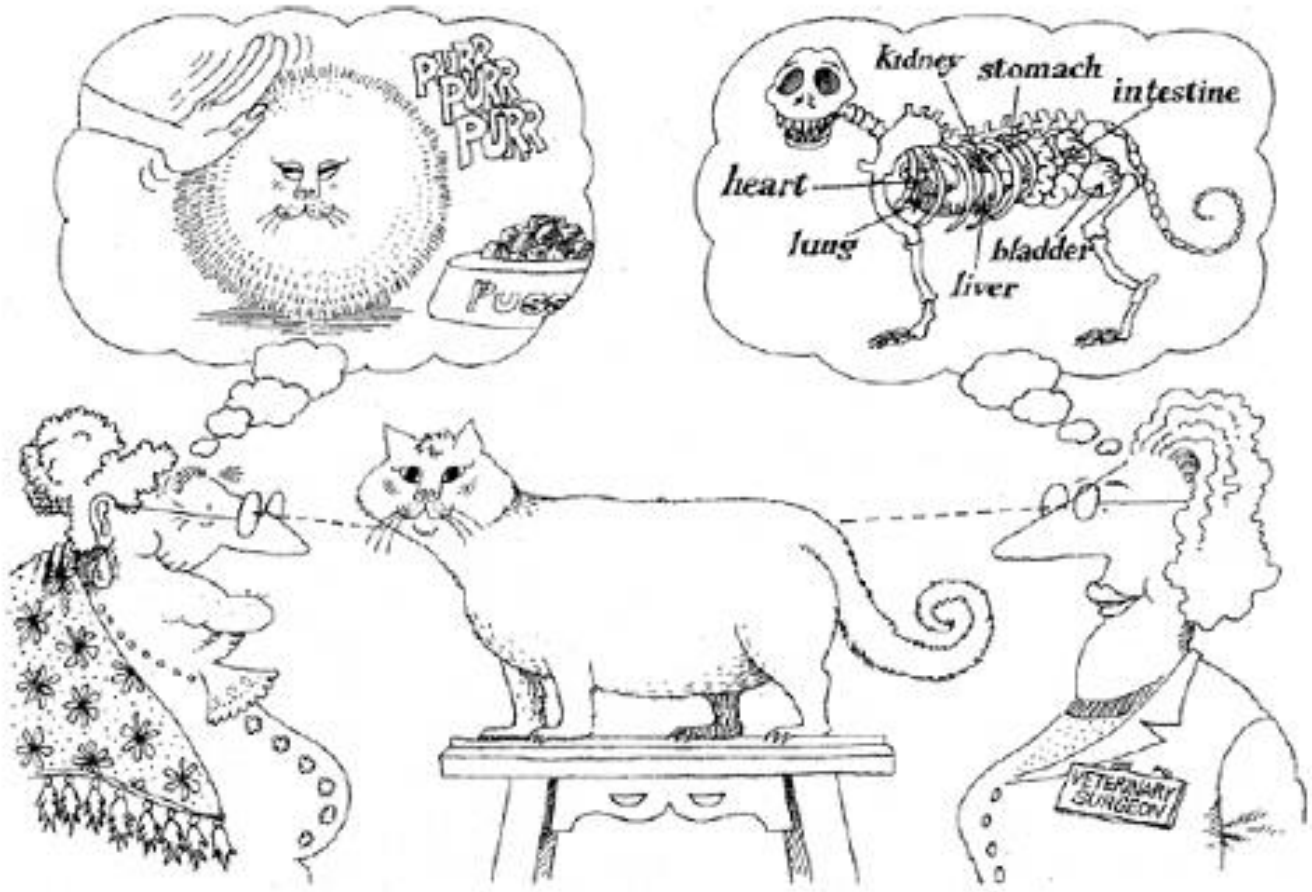
- Car description
- Service history
- Petrol mileage history

Garage

- License plate
- Work description
- Billing info
- Owner

Thế nào là lập trình hướng đối tượng?

- **Sự trừu tượng hóa** *(abstraction)*



Sự trừu tượng hóa *(abstraction)*

- Là bước tiến hóa tiếp theo từ lập trình cấu trúc.
- Chỉ quan tâm đến những đặc điểm cần thiết (phớt lờ đi những chi tiết) tùy vào ngữ cảnh:

VD: Phân tích thông tin của 1 Sinh viên:

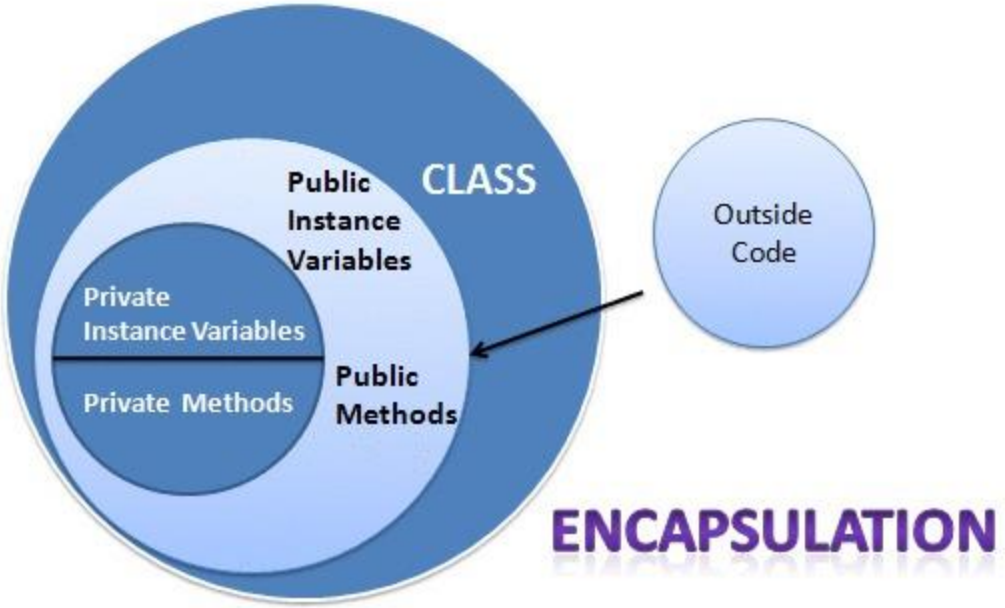
- Trong ngữ cảnh trường học:
- Trong ngữ cảnh cuộc thi hoa hậu?

SinhViên
– Mã số sinh viên – Họ và tên – Năm sinh – Lớp – Ngành – Điểm trung bình – …

- Chú ý phân biệt:
  - Giao diện và cài đặt
  - Cái gì và thế nào
  - Phân tích và thiết kế

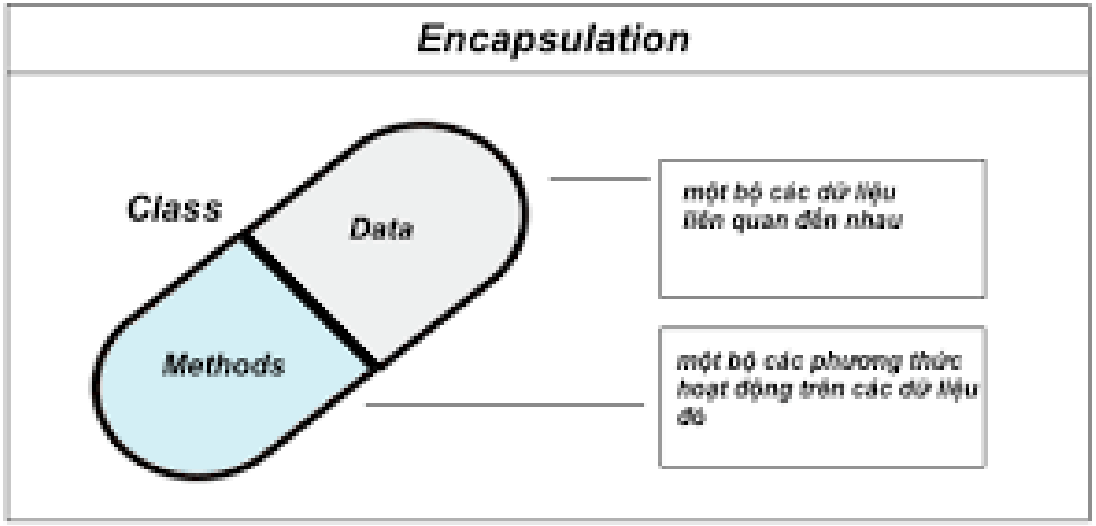
Sự đóng gói (encapsulation)

Tính đóng gói (encapsulation) và che giấu thông tin (information hiding): Tính chất này không cho phép người sử dụng các đối tượng thay đổi trạng thái nội tại của một đối tượng. Chỉ có các phương thức nội tại của đối tượng cho phép thay đổi trạng thái của nó. Việc cho phép môi trường bên ngoài tác động lên các dữ liệu nội tại của một đối tượng theo cách nào là hoàn toàn tùy thuộc vào người viết mã. Đây là tính chất đảm bảo sự toàn vẹn của đối tượng.



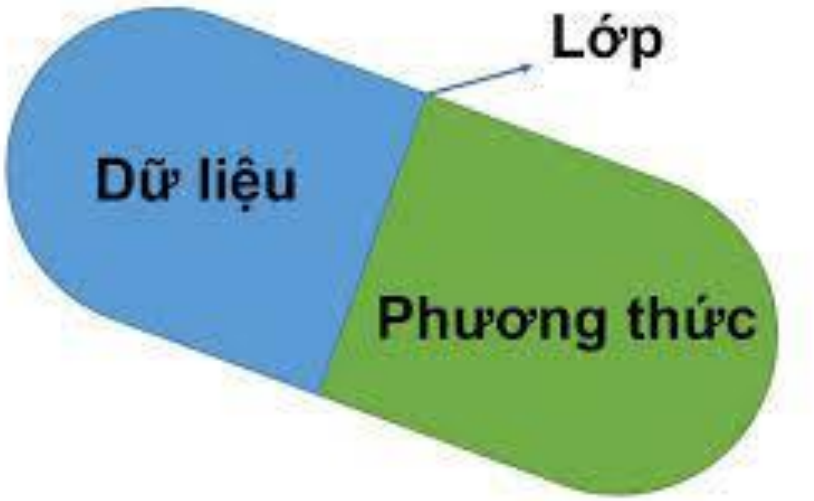
Sự đóng gói (encapsulation)

Tất cả các thông tin của một hệ thống định hướng đối tượng được lưu trữ bên trong đối tượng của nó và chỉ có thể hành động khi các đối tượng đó được ra lệnh thực hiện các thao tác. Như vật, sự đóng gói không chỉ đơn thuần là sự gom chung dữ liệu và chương trình vào trong một khối, chúng còn được hiểu theo nghĩa là sự đồng nhất giữa dữ liệu và các thao tác tác động lên dữ liệu đó.



Sự đóng gói (encapsulation)

- Sự đóng gói (Encapsulation) cung cấp 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 và Bảo vệ dữ liệu.



Các đặc điểm của OOP

- **Tính đóng gói (encapsulation)**
  - Là đặc điểm chủ yếu của OOP.
  - Che giấu việc thực thi chi tiết của 1 đối tượng.
  - Ngăn sự tác động từ đối tượng khác đến dữ liệu.

Ẩn thông tin là 1 tính chất của bao gói

Tính đóng gói

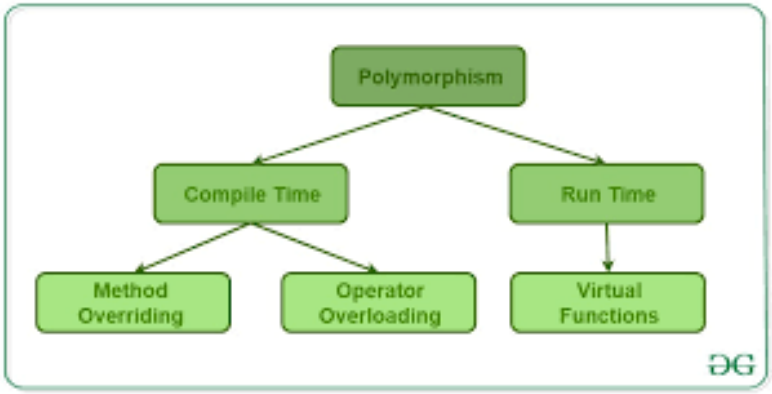
- Thể hiện sự kết hợp chặc chẽ giữa thuộc tính và phương thức.
- Định nghĩa lớp: là việc đóng gói nhiều thành phần lại với nhau.
- Xác định và giới hạn các đường truy cập đến lớp.
- Các thuộc tính truy cập:
  - Dùng chung *(public)*: có quyền được truy xuất từ bất kỳ vị trí nào.
  - Dùng riêng *(private):* chỉ truy xuất được từ bên trong lớp (trong các phương thức của lớp đó).
  - Được bảo vệ *(protected):* chỉ truy xuất được từ bên trong lớp và các lớp kế thừa từ nó.

Tính đa hình (polymorphism)

- Từ “Polymorphism” được tạo bắt nguồn từ 2 từ của Hy Lạp đó là “Poly” và “morph”. Ý ngĩa của từ poly là nhiều (many) và morph là dạng “forms”. Nó có ngĩa là có khả năng thể hiện ở nhiều dạng trong cùng 1 phương thức (have many forms).
- Đa hình thể hiện thông qua việc gửi các thông điệp (message). Việc gửi các thông điệp này có thể so sánh như việc gọi các hàm bên trong của một đối tượng. Các phương thức dùng trả lời cho một thông điệp sẽ tùy theo đối tượng mà thông điệp đó được gửi tới sẽ có phản ứng khác nhau.

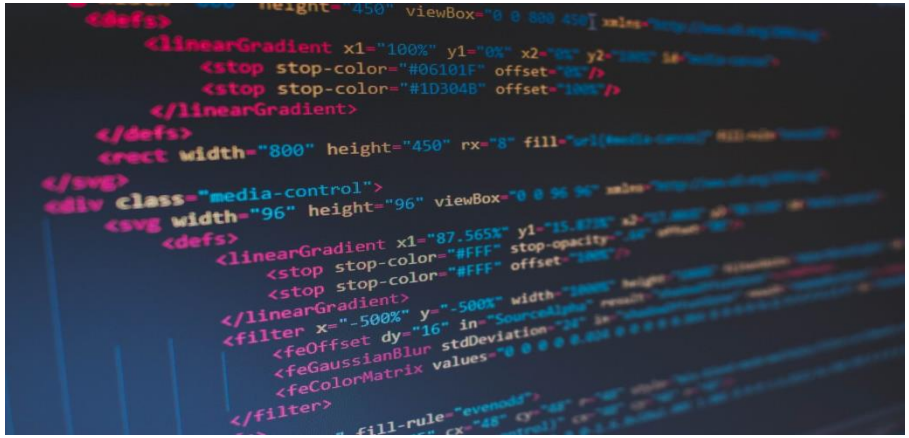
Tính đa hình (polymorphism)

- Người lập trình có thể định nghĩa một đặc tính (chẳng hạn thông qua tên của các phương thức) cho một loạt các đối tượng gần nhau nhưng khi thi hành thì dùng cùng một tên gọi mà sự thi hành của mỗi đối tượng sẽ tự động xảy ra tương ứng theo đặc tính của từng đối tượng mà không bị nhầm lẫn.
- Ví dụ khi định nghĩa hai đối tượng “hinh_vuong” và “hinh_tron” thì có một phương thức chung là “chu_vi”. Khi gọi phương thức này thì nếu đối tượng là “hinh_vuong” nó sẽ tính theo công thức khác với khi đối tượng là “hinh_tron”.



Vì sao cần có tính đa hình?

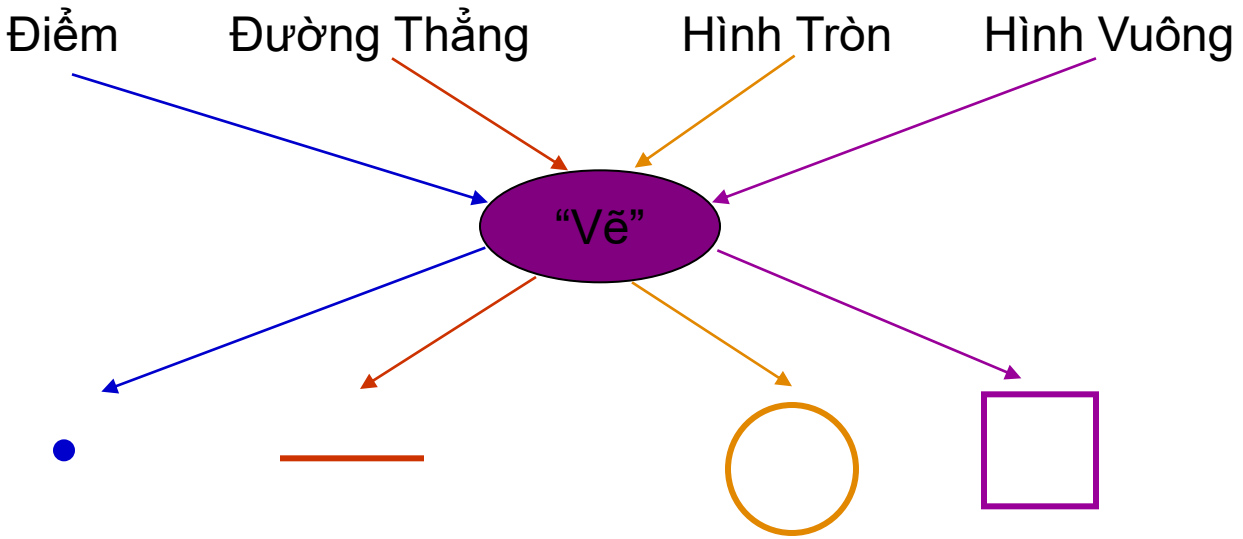
- Lập trình viên tận dụng được tính đa hình thì sẽ mang lại nhiều lợi ích trong quá trình phát triển phần mềm.
- Lập trình viên không phải viết lại mã hoặc lớp đã có sẵn. Sau khi một đoạn mã hoặc lớp được khởi tạo thành công, ta có thể tái sử dụng chúng nhờ vào tính đa hình.
- Lập trình viên có thể dùng một tên duy nhất để lưu trữ biến của nhiều kiểu dữ liệu khác nhau (float, double, long, int,…).
- Lập trình viên có thể phát triển thêm tính trừu tượng từ những đoạn mã đơn giản.



Tính đa hình *(polymorphism)*

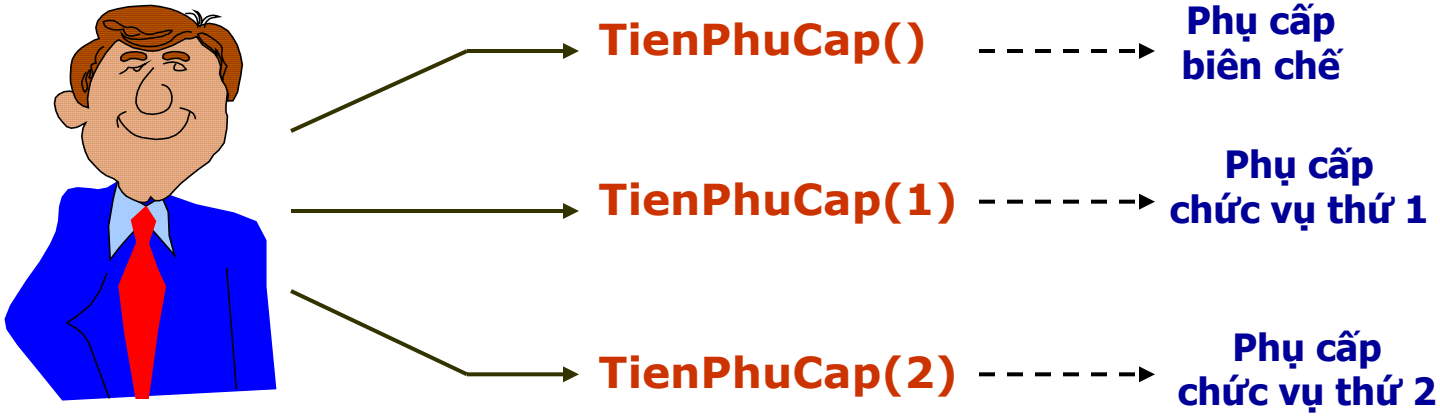
– Các đối tượng khác nhau khi nhận chung 1 yêu cầu vẫn có thể có những ứng xử khác nhau.

– Kỹ thuật sử dụng: hàm ảo



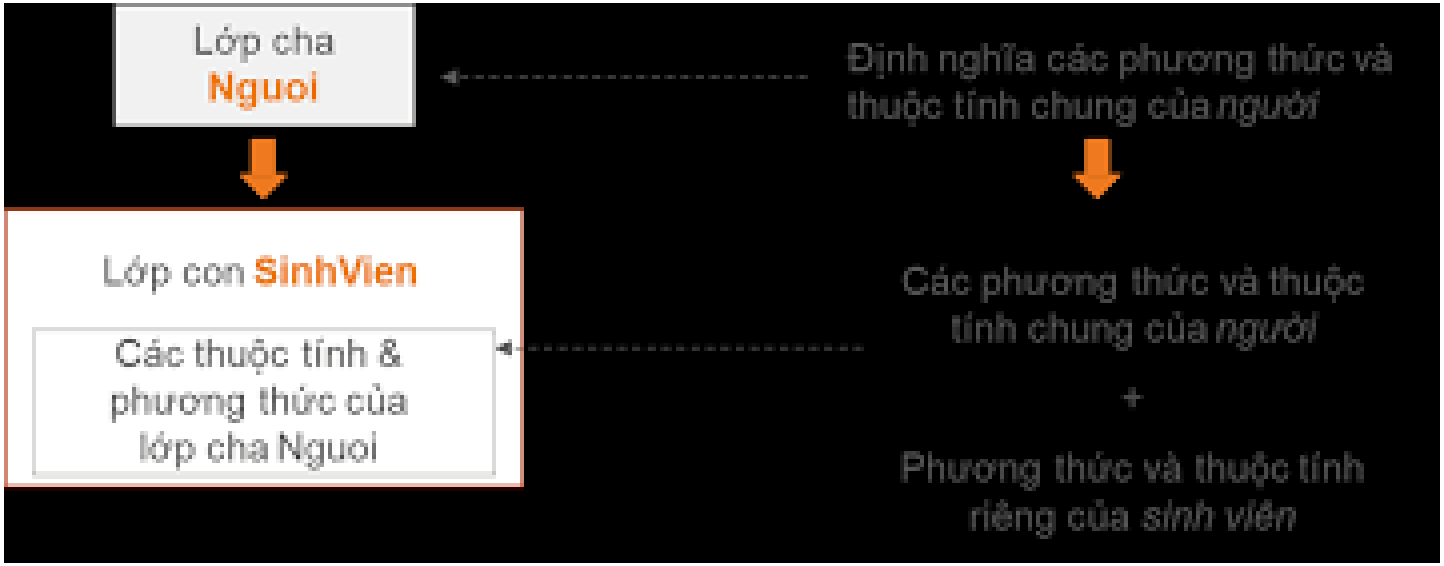
Tính đa hình

- Các hành động cùng tên thực thi trên cùng 1 đối tượng vẫn có thể thực hiện khác nhau (có kết quả khác nhau) tùy thuộc vào ngữ cảnh.
- Kỹ thuật sử dụng: tái định nghĩa hàm
  - Hàm trùng tên
  - Khác tham số (số lượng, thứ tự, kiểu)



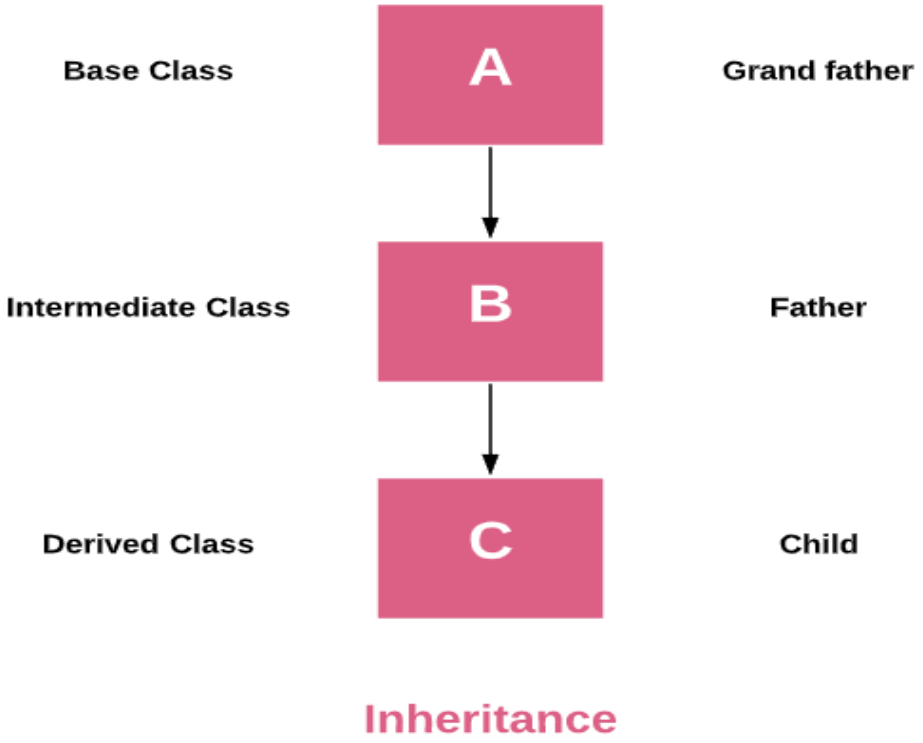
Tính kế thừa (inheritance)

- Tính kế thừa cho phép một đối tượng có thể có sẵn các đặc tính mà đối tượng khác đã có thông qua kế thừa. Điều này cho phép các đối tượng chia sẻ hay mở rộng các đặc tính sẵn có mà không phải tiến hành định nghĩa lại.



Tính kế thừa (inheritance)

- Trong lập trình hướng đối tượng, kế thừa là việc tái sử dụng lại một số thuộc tính, phương thức đã có sẵn từ lớp cơ sở. Là một đặc điểm của ngôn ngữ dùng để biểu diễn mối quan hệ đặc biệt hóa - tổng quát hóa giữa các lớp.



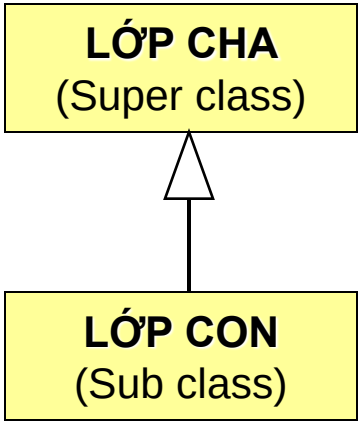
Tính kế thừa *(inheritance)*

- Nhóm các đặc tính chung lại với nhau.
- Sử dụng lại (kế thừa) các đặc trưng chung (thuộc tính + phương thức) của 1 lớp cho trước.
- Ích lợi:
  - Chia sẻ và sử dụng lại những lớp đã có:
    - Tận dụng lại các thuộc tính chung.
    - Tận dụng lại các thao tác tương tự.
  - Thiết kế lớp gọn nhẹ, đơn giản hơn.
- Chú ý: tránh thiết kế sai về mặt ý nghĩa.

  *VD: Lớp XeHơi kế thừa từ lớp BánhXe là **SAI**.*

Tính kế thừa

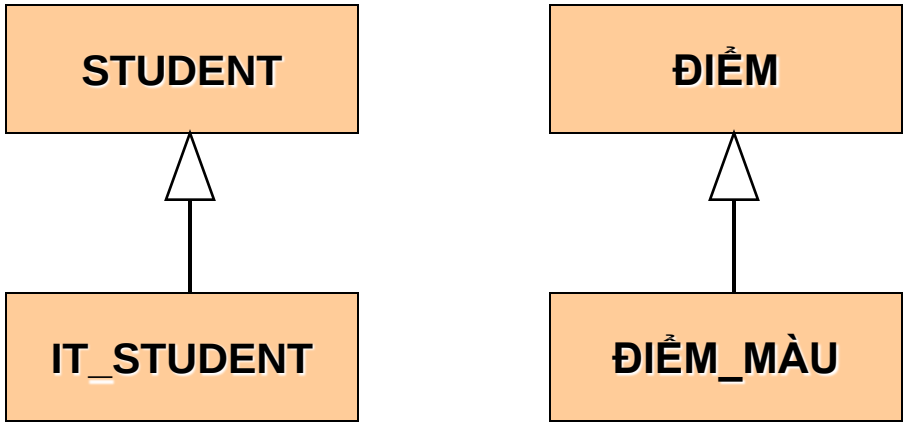
– Sơ đồ kế thừa:



Lớp cơ sở
(Base class)

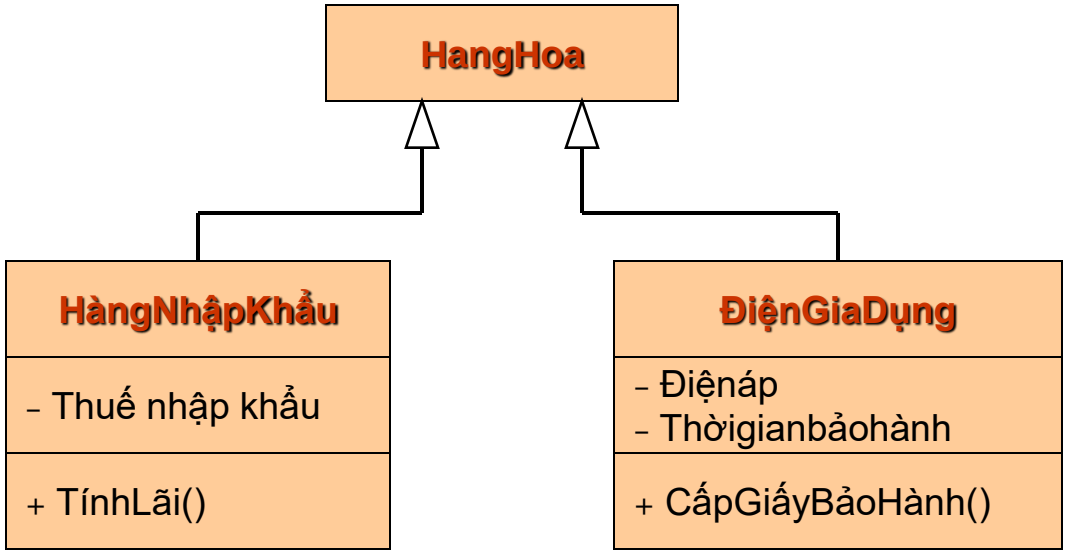
Lớp dẫn xuất
(Derived class)

- Ví dụ:



Tính kế thừa

– Lớp con phải chính là lớp cha, ngoài ra còn phải có thêm những đặc trưng riêng của nó:
  - Thêm hàm hoặc thêm thuộc tính
  - Tái định nghĩa hàm của lớp cha, đổi kiểu thuộc tính
  - Dùng hàm ảo

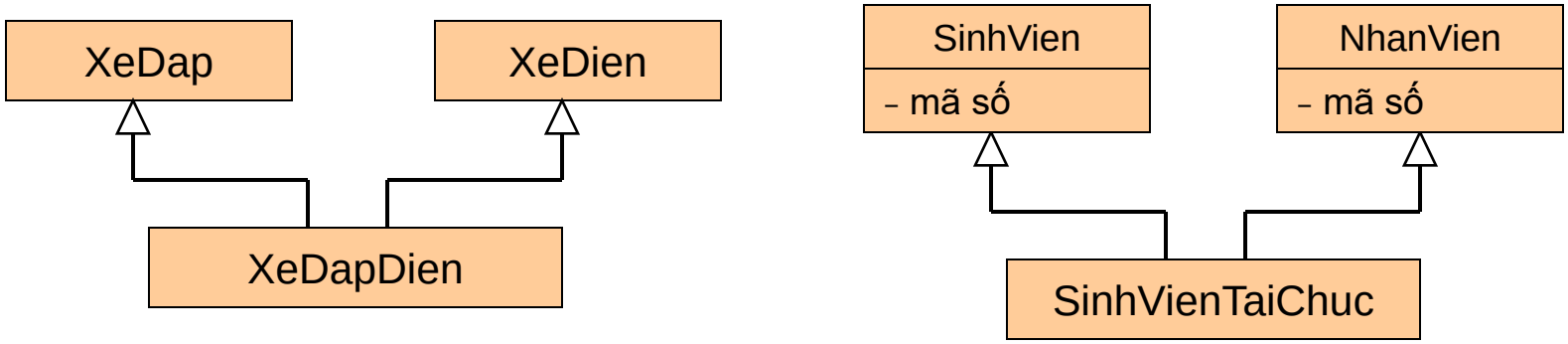


Tính kế thừa

- Lớp con sẽ **có** tất cả các thành phần của lớp cha nhưng **chỉ** được quyền truy xuất thành phần mà lớp cha cho phép.
- Lớp cha không thể truy xuất được thành phần của lớp con.
- Các thành phần của lớp con sẽ che đi các thành phần trùng tên trong lớp cha.
  - Khi truy xuất thành phần trùng tên đó sẽ truy xuất thành phần của lớp con.
  - Muốn truy xuất thành phần của lớp cha, phải chỉ rõ.
    - Trong C++: <Tên lớp cha> :: <Tên hàm>
    - Trong Java: super.<Tên hàm>

Tính kế thừa

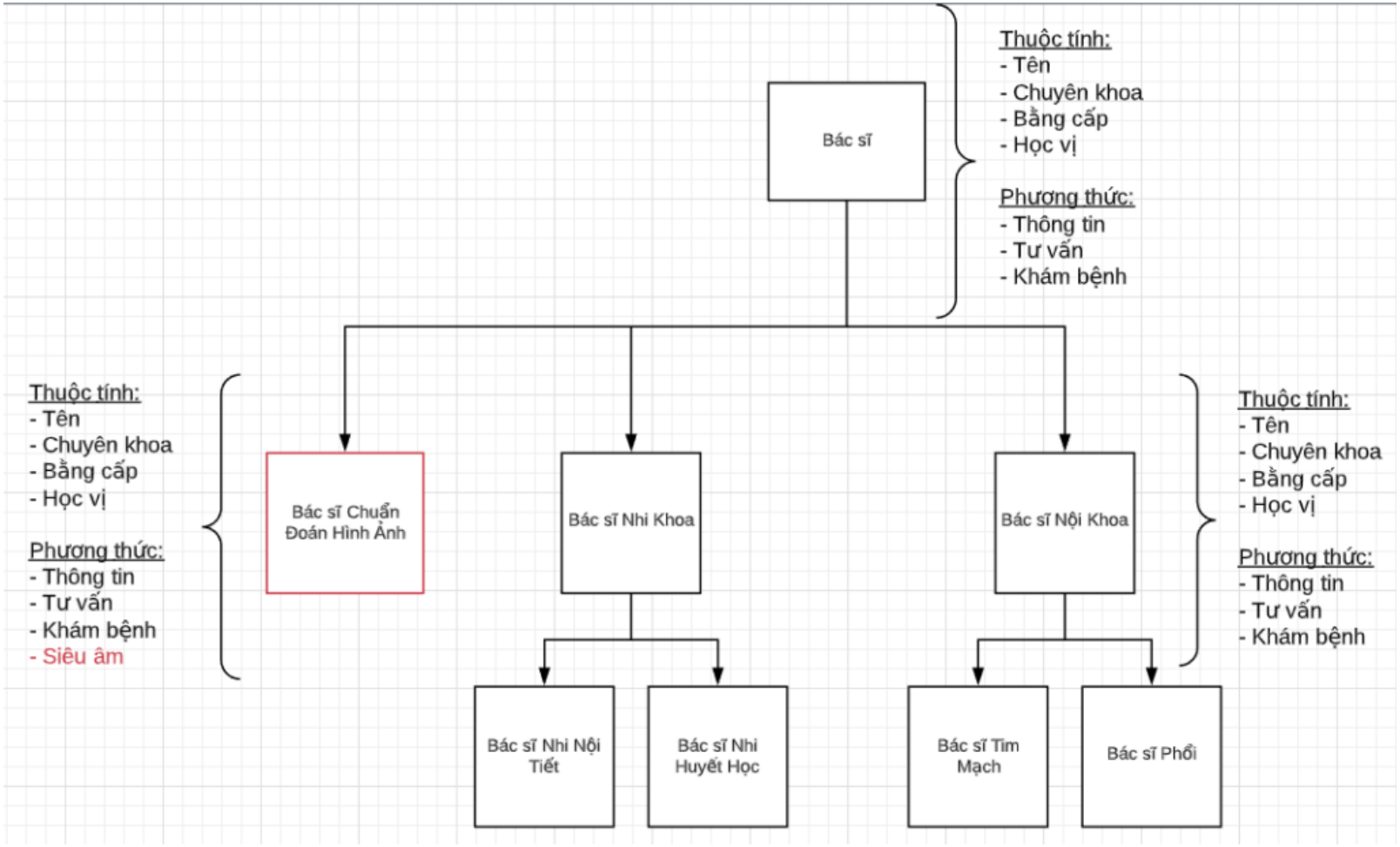
- kế thừa đơn: kế thừa từ 1 lớp
- kế thừa bội: kế thừa từ 2 lớp trở lên.
  - Tận dụng được nhiều thành phần có sẵn.
  - Dễ gây ra xung đột, cạnh tranh.



Hướng giải quyết

- Thiết kế lớp phải đúng ý nghĩa.
- Tránh kế thừa bội khi gây ra xung đột.
- Bỏ kế thừa khi lớp cha có quá ít thành phần

Tính kế thừa (inheritance)



Các ngôn ngữ OOP

Ngôn ngữ lập trình hướng đối tượng

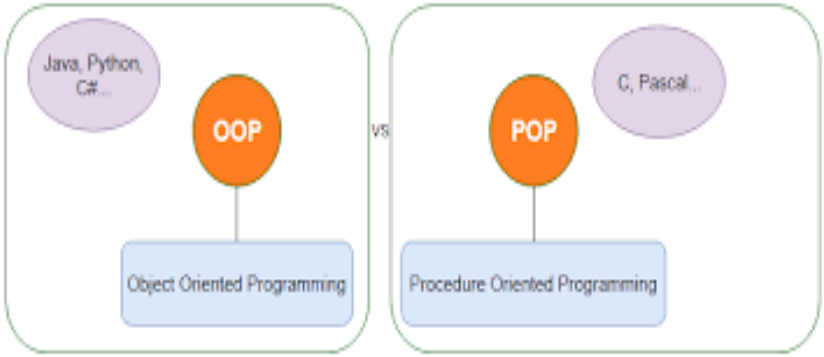
- Đa phần các ngôn ngữ lập trình thông dụng nhất hiện nay (như C++, Delphi, Java, Python etc.) là các ngôn ngữ lập trình đa mẫu hình và đều hỗ trợ lập trình hướng đối tượng ở nhiều mức độ khác nhau, thường được kết hợp với lập trình mệnh lệnh, lập trình thủ tục.
- Các ngôn ngữ lập trình hướng đối tượng đáng chú ý gồm có Java, C++, C#, Python, PHP, Ruby, Perl, Object Pascal, Objective-C, Dart, Swift, Scala, Common Lisp, và Smalltalk…

Ngôn ngữ lập trình hướng đối tượng

- C#
- C++
- Common Lisp Object System
- Eiffel
- Fortran 2003
- Java
- Objective-C
- OCaml
- Object Pascal
- Pascal

Ngôn ngữ lập trình hướng đối tượng

- Perl
- PHP
- Python
- Ruby
- Simula
- Sleep
- Smalltalk
- Ada
- Visual FoxPro
- Actionscript
- Visual Basic
- Swift



Nhận xét về OOP

Ưu thế của lập trình OOP

- Dễ dàng quản lý code khi có sự thay đổi chương trình.
- Dễ mở rộng dự án phần mềm.
- Tiết kiệm được tài nguyên đáng kể cho hệ thống.
- Có tính bảo mật cao.
- Có tính tái sử dụng cao.
- Có khả năng lập biểu đồ cho các đối tượng.

Ưu thế của lập trình OOP

- Cho phép phân loại các đối tượng thành các lớp khác nhau.
- Nguyên lý kế thừa: tránh lặp, tái sử dụng.
- Nguyên lý đóng gói – che dấu thông tin: chương trình an toàn không bị thay đổi bởi những đoạn chương trình khác
- Dễ mở rộng, nâng cấp
- Mô phỏng thế giới thực tốt hơn.

Ưu thế của lập trình OOP

Nâng cao hiệu năng phát triển phần mềm

- **Tính mô-đun:** nó tách biệt các nhiệm vụ trong quá trình phát triển phần mềm dựa trên những đối tượng cụ thể. Mỗi đối tượng sẽ có một nhiệm vụ khác nhau.
- **Tính mở rộng**: các đối tượng có thể được mở rộng dễ dàng, bao gồm mở rộng thuộc tính và các hành vi mới.
- **Tính tái sử dụng:** các đối tượng cũng có thể được sử dụng lại trong một ứng dụng hoặc nhiều ứng dụng khác nhau.

Ưu thế của lập trình OOP

- **Nâng cao khả năng bảo trì phần mềm:** Chính vì những lý do nêu trên, phần mềm được lập trình theo hướng đối tượng cũng dễ bảo trì hơn. Vì thiết kế là mô-đun, nên việc thay đổi một phần của chương trình sẽ không ảnh hưởng đến những phần còn lại. Điều này rất có lợi trong trường hợp dự án của chúng ta có quy mô lớn, đòi hỏi nhiều thay đổi.
- **Phần mềm phát triển nhanh hơn:** Tính tái sử dụng của lập trình hướng đối tượng cho phép phát triển phần mềm nhanh hơn. Các phần mềm được lập trình theo hướng đối tượng thường có thư viện đối tượng phong phú, các đoạn mã được tối ưu hóa và có thể tái sử dụng ở các dự án khác trong tương lai.

Ưu thế của lập trình OOP

- **Giảm thiểu chi phí phát triển:** Việc tái sử dụng phần mềm cũng làm giảm thiểu chi phí phát triển cho nhà sản xuất. Thông thường, phần lớn công sức chỉ tập trung vào việc phân tích đối tượng và thiết kế phần mềm. Do đó, tổng chi phí phát triển phần mềm cũng giảm đi đáng kể.

Ưu thế của lập trình OOP

- **Chất lượng phần mềm cao hơn:** Thực tế, chất lượng của phần mềm phụ thuộc vào nhiều yếu tố khác nhau. Chẳng hạn, kinh nghiệm và trình độ của nhóm phát triển cũng sẽ ảnh hưởng đến sản phẩm đầu ra. Tuy nhiên, phương pháp này có xu hướng tạo ra những phần mềm chất lượng cao hơn.

Nhược điểm của lập trình OOP

- Lập trình hướng đối tượng là một phương pháp đòi hỏi khá nhiều tư duy. Do đó, nó có thể không dễ dàng với một số người, đặc biệt là những người mới vào nghề. Các lập trình viên cần một khoảng thời gian để học và tập làm quen với nó.
- Phương pháp này phức tạp vì phần mềm phải dựa trên sự tương tác của đối tượng. Do đó, lập trình viên cần phải hiểu bản chất của những khái niệm cơ bản như: lớp, đối tượng, phương thức, thuộc tính.
- Đồng thời, ta cũng cần nắm được bốn tính chất cơ bản của lập trình hướng đối tượng. Đó là: Tính trừu tượng (Abstraction), Tính đóng gói (Encapsulation), Tính kế thừa (Inheritance) và Tính đa hình (Polymorphism).

Nhược điểm của lập trình OOP

- **Chương trình chậm và có kích thước lớn hơn:** Phần mềm được lập trình theo hướng đối tượng thường chậm hơn các phần mềm dựa trên thủ tục. Lý do là vì các phần mềm này thường yêu cầu nhiều câu lệnh hơn để thực thi. Lập trình viên phải viết ra nhiều dòng mã để đảm bảo những thuộc tính của phương pháp này. Do đó, kích thước cũng chương trình cũng lớn hơn.

Nhược điểm của lập trình OOP

Phương pháp lập trình hướng đối tượng không phù hợp với mọi loại vấn đề:

- Mỗi phương pháp khác nhau sẽ phù hợp với một vấn đề khác nhau. Lập trình hướng đối tượng cũng vậy.
- Thực tế, có những vấn đề mặc định sẽ được giải quyết tốt hơn nếu lập trình viên sử dụng phương pháp lập trình chức năng (Functional Programming), lập trình logic, hoặc lập trình thủ tục. Nếu ta áp dụng lập trình hướng đối tượng, có thể sẽ không đem lại hiệu quả tốt.
- Nhìn chung, lập trình hướng đối tượng là một phương pháp hay nhưng không hoàn hảo. Khi lập trình, chúng ta có thể cân nhắc những ưu, nhược điểm của phương pháp lập trình hướng đối tượng để có thể vận dụng tốt trong công việc.

Sự khác nhau giữa POP và OOP

- **Đối tượng hướng tới**
  - Sự khác nhau dễ thấy nhất là trong OOP những code được viết trong chương trình sẽ được thể hiện giống như các đối tượng. Các đối tượng cũng giống như những đối tượng trong thực tế như sinh viên, cuốn sách,… Còn POP hướng tới đối tượng là thủ tục.
- **Phương pháp thực hiện**
  - Trong POP chương trình sẽ chia ra nhiều hàm để giải quyết một bài toán cụ thể. Vì vậy POP phù hợp với nhiều bài toán nhỏ, có luồng dữ liệu rõ ràng, cần phải tư duy giải thuật rõ ràng và người lập trình có khả năng tự quản lý được mọi truy cập đến các dữ liệu của chương trình.

Sự khác nhau giữa POP và OOP

- Trong OOP chương trình được định nghĩa bởi các lớp …
- Các lớp này chứa thuộc tính (biến thành viên) và phương thức (hàm thành viên). Các lớp (Class) này sẽ tạo ra các đối tượng (instance).
- Các instance sau khi tạo ra sẽ có các thuộc tính và phương thức được định nghĩa trong lớp (Class) và quản lí nó.
- OOP được sinh ra để giải quyết các bài toán lớn, phức tạp.

Sự khác nhau giữa POP và OOP

- **Khả năng truy cập, bảo mật**
  - OOP chia ra các đặc tính: Private, Public, Protected, Default còn POP thì không. Nhờ việc giới hạn truy cập dữ liệu mà bảo mật trong OOP cao hơn.
- **Điều khiển dữ liệu**
  - Với OOP thì dữ liệu và hàm của một đối tượng giống như một thành phần riêng biệt và bị hạn chế truy cập bởi các đối tượng khác. Với POP, dữ liệu có thể truy cập một cách tự do giữa các hàm. Vì vậy bảo mật của OOP sẽ cao hơn POP.

Sự khác nhau giữa POP và OOP

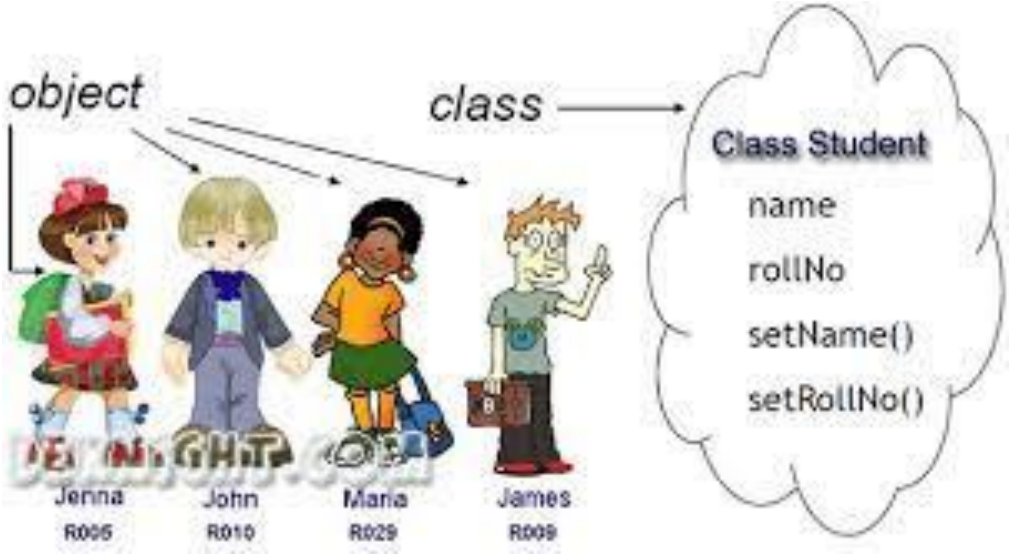
- **Quản lý dữ liệu**
  - Lập trình hướng đối tượng sẽ làm cho việc viết chương trình trở nên trực quan và đơn giản. Do đó nên sẽ mất ít công sức hơn, làm được nhiều hơn, dễ bảo trì và phát triển hơn.
  - Trong OOP dữ liệu có thể thêm mới một cách dễ dàng từ các đối tượng trong khi với POP thì rất khó.
- **Các ngôn ngữ thường sử dụng**
  - Trong khi POP sử dụng các ngôn ngữ cơ bản như c, pascal thì OOP sử dụng các ngôn ngữ thông dụng hiện tại như ruby, c++, java...

Sự khác nhau giữa POP và OOP

- **Trọng tâm chính**
  - Trọng tâm chính của **POP** là về cách thức thực hiện nhiệm vụ của hệ thống, nó tuân theo biểu đồ dòng chảy để hoàn thành nhiệm vụ.
  - Trọng tâm chính của **OOP** là bảo mật dữ liệu vì chỉ các đối tượng của một lớp mới được phép truy cập các thuộc tính hoặc chức năng của một lớp.
- **Tính kế thừa, đa hình**
  - OOP hỗ trợ khái niệm **Overloading /polymorphism**, nghĩa là sử dụng cùng tên hàm để thực hiện các chức năng khác nhau. Chúng ta có thể **Overload** các hàm, hàm tạo và toán tử trong OOP.

Sự khác nhau giữa POP và OOP

- Không có khái niệm kế thừa trong POP trong khi đó OOP hỗ trợ kế thừa cho phép sử dụng thuộc tính và chức năng của lớp khác bằng cách kế thừa nó.
- Ngoài ra thì POP dễ học, dễ hiểu hơn OOP



ĐẠI NAM
UNIVERSITY

Xin cảm ơn!

TRẦN QÚY NAM

Giảng viên Khoa CNTT
namtq@dainam.edu.vn