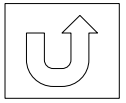
Chương 02. Phương pháp lập trình hướng đối tượng

I. Lập trình cấu trúc và lập trình hướng đối tượng

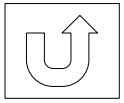
II. Các khái niệm cơ bản trong lập trình hướng đối tượng

III. Các ngôn ngữ lập trình hướng đối tượng



I. Lập trình cấu trúc và lập trình hướng đối tượng

1. Lập trình cấu trúc
2. Lập trình hướng đối tượng

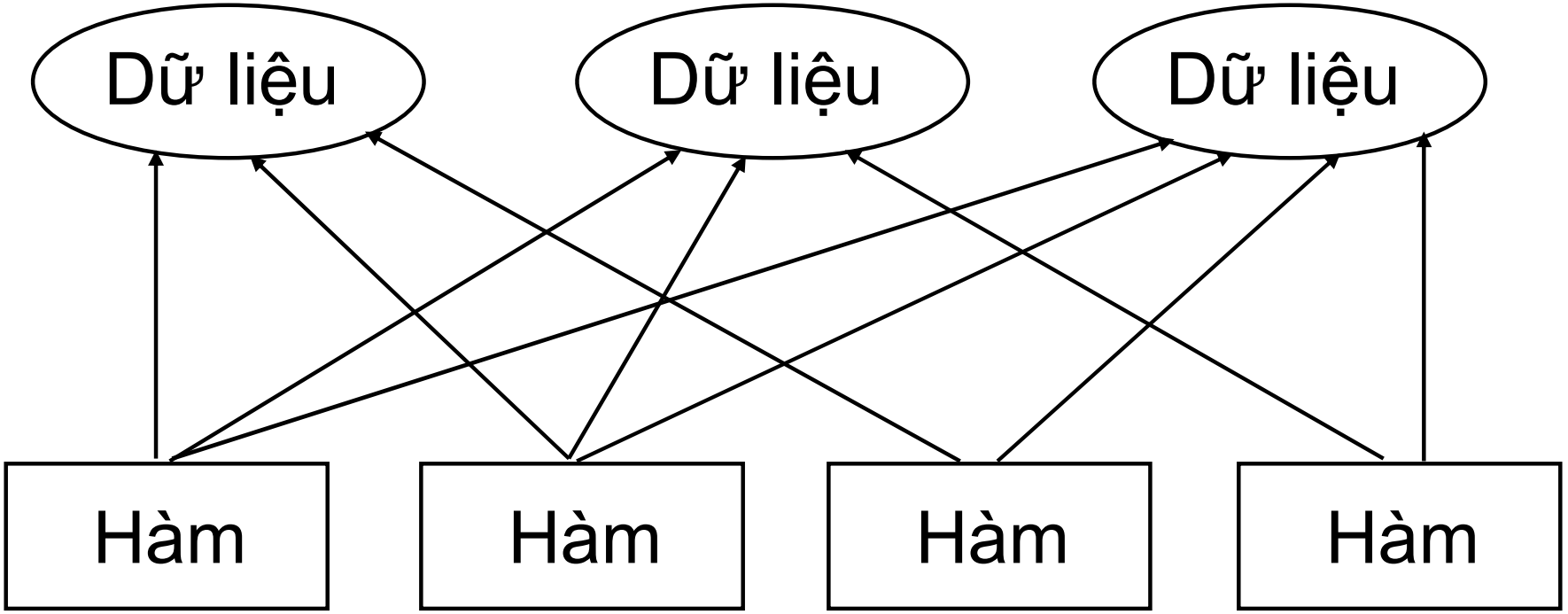


1. Lập trình cấu trúc

- Tư tưởng chính của lập trình cấu trúc (structural programming) là chia chương trình thành các chương trình con (trong C++ gọi là hàm) và các module. Mỗi hàm thực hiện một nhiệm vụ xác định nào đó, còn mỗi module bao gồm một số hàm liên quan.
- Khi các chương trình ngày càng lớn và phức tạp thì lập trình cấu trúc bắt đầu bộc lộ những điểm yếu. Và cho dù các chương trình lớn này có được cài đặt tốt đến mấy thì nó vẫn quá phức tạp.

1. Lập trình cấu trúc (tiếp)

✧ Mô hình lập trình cấu trúc như sau:

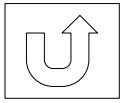


1. Lập trình cấu trúc (tiếp)

- Lý do chính làm cho phương pháp lập trình cấu trúc tự bộc lộ những điểm yếu là dữ liệu của chương trình không được coi trọng. Các dữ liệu quan trọng của chương trình được lưu trữ trong các biến toàn cục, nó cho phép mọi hàm có thể truy nhập. Mà các hàm lại được viết bởi nhiều người lập trình khác nhau nên nguy cơ hỏng, mất dữ liệu là rất lớn.
- Hơn nữa, vì nhiều hàm truy nhập cùng một dữ liệu nên khi dữ liệu thay đổi thì các hàm này cũng phải thay đổi theo. Việc tìm các hàm cần thay đổi đã khó nhưng việc thay đổi các hàm này sao cho đúng còn khó hơn.

1. Lập trình cấu trúc (tiếp)

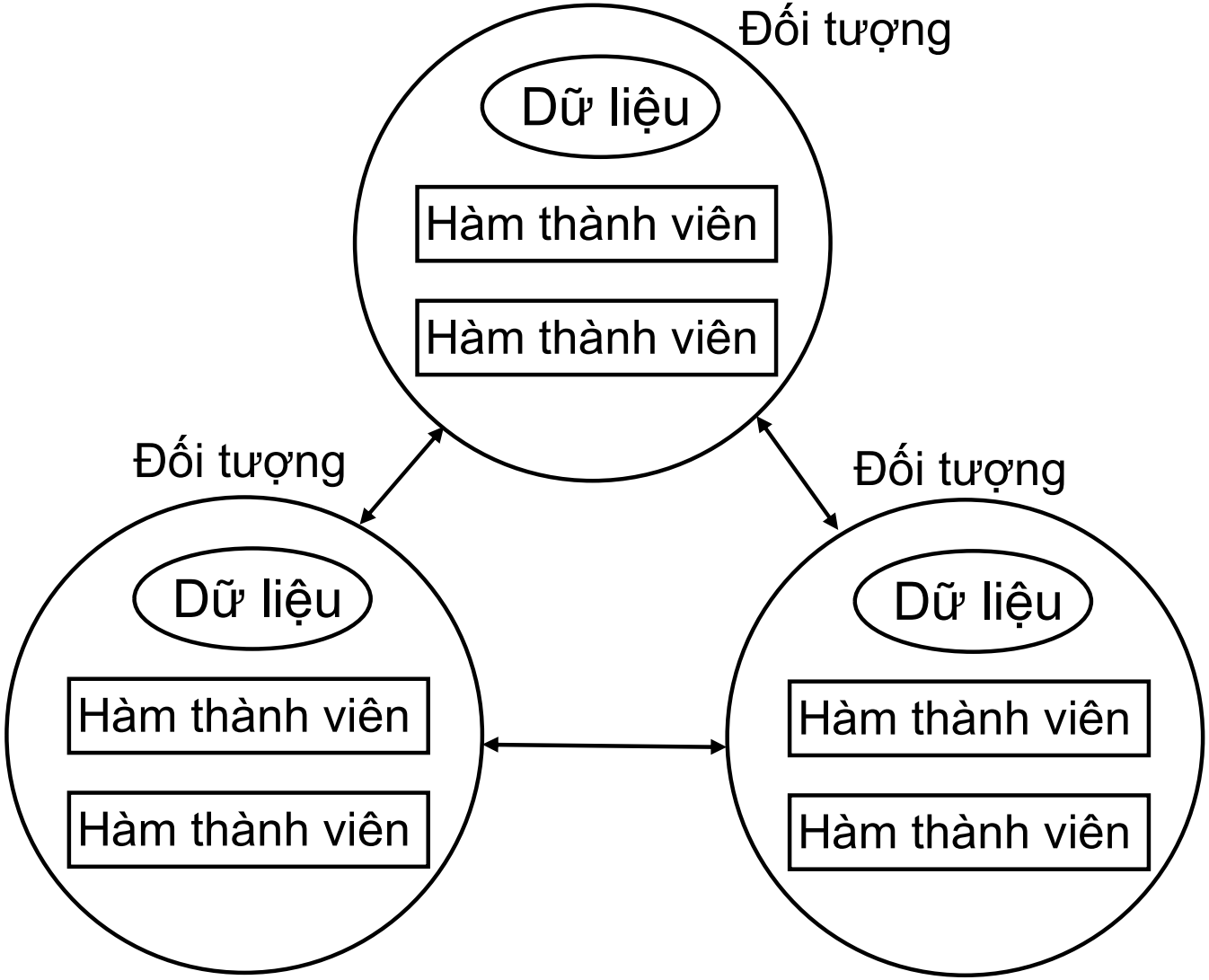
- Lập trình cấu trúc thường khó thiết kế chương trình bởi vì các thành phần chính của chương trình cấu trúc (là hàm và cấu trúc dữ liệu) không mô phỏng được thế giới thực. Ví dụ: giả sử ta cần viết mã để tạo giao diện đồ họa với người sử dụng như menu, cửa sổ, nút bấm,… Nếu lập trình cấu trúc thì câu hỏi đặt ra là dùng cấu trúc dữ liệu nào? Các hàm cần làm gì?



2. Lập trình hướng đối tượng

- Ý tưởng chính của lập trình hướng đối tượng (object oriented programming, OOP) là chia chương trình thành các đối tượng. Đối tượng là thực thể chương trình kết hợp cả dữ liệu và các hàm thao tác trên dữ liệu đó.
- Cách duy nhất để truy nhập dữ liệu của một đối tượng là thông qua các hàm của đối tượng đó (trong C++, các hàm của đối tượng được gọi là các hàm thành viên). Nếu ta muốn đọc dữ liệu trong một đối tượng thì ta phải gọi một hàm thành viên của đối tượng đó. Hàm thành viên này sẽ đọc dữ liệu và trả về giá trị cho ta. Ta không thể truy nhập trực tiếp dữ liệu của đối tượng.

2. Lập trình hướng đối tượng (tiếp)

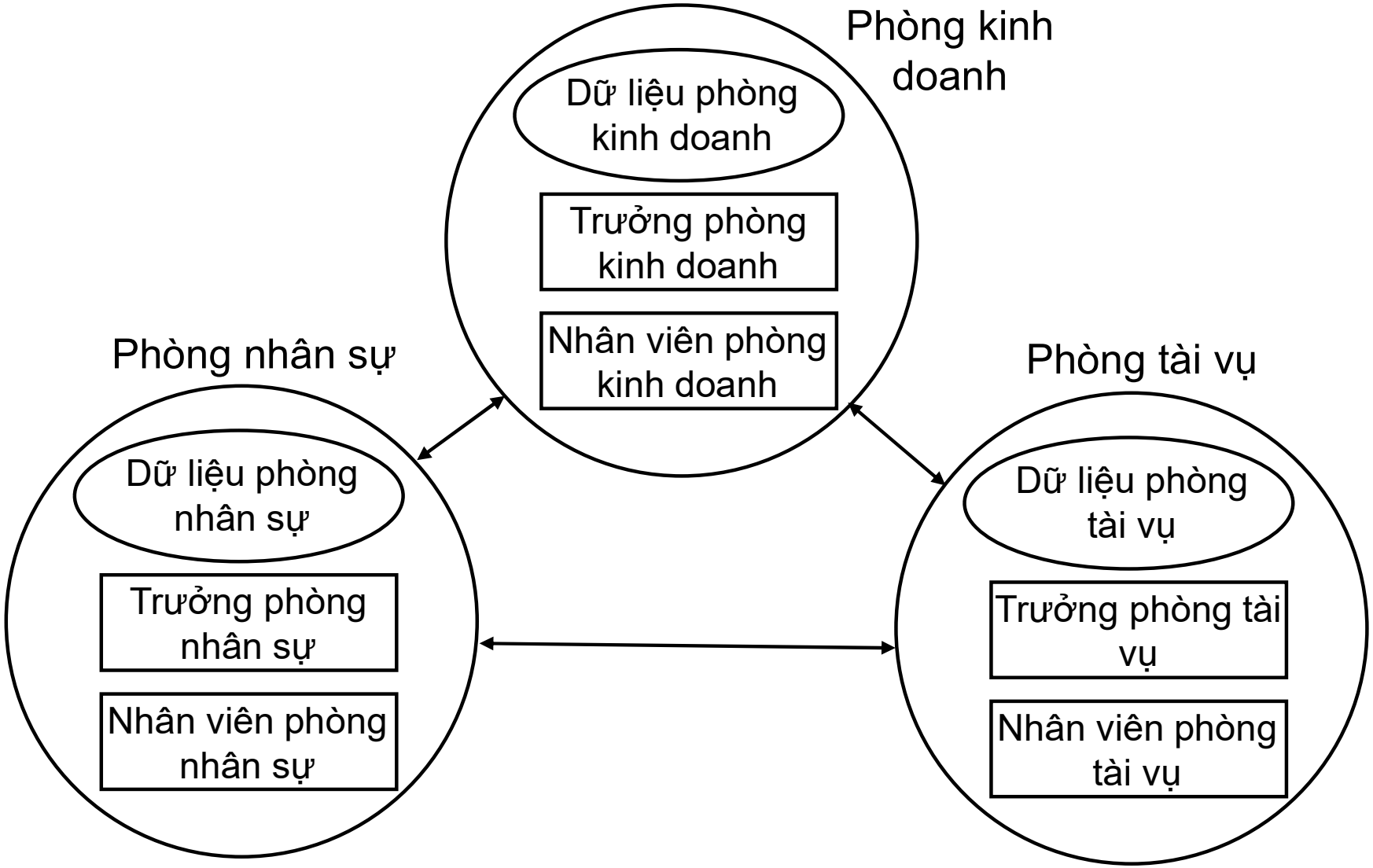


Mô hình lập trình hướng đối tượng

2. Lập trình hướng đối tượng (tiếp)

- ✧ Trong lập trình hướng đối tượng dữ liệu được ẩn đi để tránh những thay đổi vô tình làm hỏng dữ liệu. Dữ liệu và hàm tác động lên nó được đóng gói trong một thực thể chương trình.
- ✧ Nếu chúng ta muốn thay đổi dữ liệu trong một đối tượng thì chúng ta phải biết chính xác hàm nào tương tác với nó; tức là các hàm thành viên trong đối tượng đó. Không có hàm nào có thể truy nhập dữ liệu. Điều này giúp đơn giản hoá việc viết, gỡ rối, và bảo trì chương trình.

2. Lập trình hướng đối tượng (tiếp)



Mô hình công ty kinh doanh

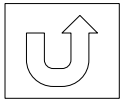
2. Lập trình hướng đối tượng (tiếp)

- Tóm lại, lập trình hướng đối tượng là tìm cách chia chương trình thành các đối tượng.
- Học lập trình hướng đối tượng là học cách đóng gói dữ liệu và hàm thành đối tượng.
- Hướng đối tượng là phải xem thiết kế chương trình như thế nào chứ không đi vào chi tiết từng lệnh. Cụ thể là các chương trình hướng đối tượng phải được tổ chức xung quanh các đối tượng.

2. Lập trình hướng đối tượng (tiếp)

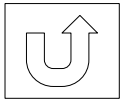
Người ta đã tổng hợp các đặc tính của LTHDT:

1. Tất cả đều là đối tượng.
2. Chương trình hướng đối tượng có thể coi là một tập hợp các đối tượng tương tác với nhau
3. Mỗi đối tượng trong chương trình có các dữ liệu độc lập của mình và chiếm bộ nhớ riêng của mình.
4. Mỗi đối tượng đều có dạng đặc trưng của lớp các đối tượng đó.
5. Tất cả các đối tượng thuộc về cùng một lớp đều có các hành vi giống nhau.



II. Các nội dung của lập trình hướng đối tượng

1. Đối tượng (object)
2. Lớp (class)
3. Sự kế thừa (inheritance)
4. Sự sử dụng lại (Reusability)
5. Sự đa hình và chồng hàm (polymorphism and overloading)
6. Che giấu dữ liệu
7. Truyền thông báo



1. Đối tượng (object)

- ✧ Như ta đã biết, đối tượng là một thành phần chương trình chứa cả dữ liệu và các hàm thao tác trên dữ liệu đó.
- ✧ Trong lập trình hướng đối tượng chúng ta không đi tìm cách chia chương trình thành các hàm mà đi tìm cách chia chương trình thành các đối tượng. Việc chia chương trình thành các đối tượng làm cho việc thiết kế chương trình trở nên dễ dàng hơn vì các đối tượng trong chương trình rất gần gũi với các đối tượng trong thực tế.

1. Đối tượng (tiếp)

Ví dụ về một số đối tượng trong thực tế có thể trở thành đối tượng trong chương trình.

- ✧ Các đối tượng vật lý:
  - Các thang máy trong chương trình điều khiển tháng máy
  - Các máy bay trong chương trình điều hành bay
  - Các xe ô tô trong chương trình mô phỏng luồng giao thông.
- ✧ Các phần tử trong môi trường người sử dụng máy tính:
  - Các cửa sổ
  - Các menu
  - Các đối tượng đồ họa (như hình chữ nhật, hình tròn, hình tam giác,…)
  - Chuột, bàn phím, các ổ đĩa, máy in

1. Đối tượng (tiếp)

Ví dụ về một số đối tượng trong thực tế có thể trở thành đối tượng trong chương trình.

- ✧ Các cấu trúc dữ liệu:
  - Ngăn xếp
  - Hàng đợi
  - Danh sách liên kết
  - Cây nhị phân
- ✧ Nhân sự:
  - Nhân viên
  - Sinh viên
  - Khách hàng

1. Đối tượng (tiếp)

Ví dụ về một số đối tượng trong thực tế có thể trở thành đối tượng trong chương trình.

- Các tệp dữ liệu:
  - Một file nhân sự
  - Một từ điển
- Các kiểu dữ liệu của người sử dụng:
  - Thời gian
  - Các số phức
  - Các điểm trong mặt phẳng

1. Đối tượng (tiếp)

Ví dụ về một số đối tượng trong thực tế có thể trở thành đối tượng trong chương trình.

- Các thành phần trong trò chơi:
  - Các viên bi trong trò chơi Line
  - Các quân cờ trong trò chơi cờ tướng, cờ vua
  - …

1. Đối tượng (tiếp)

- *Một câu hỏi đặt ra là khi các đối tượng thực tế trở thành các đối tượng trong chương trình thì cái gì là dữ liệu, cái gì là hàm thành viên của đối tượng?*
  - Các đối tượng trong thực tế thường có trạng thái và khả năng. Trạng thái là các tính chất của đối tượng mà có thể thay đổi. Khả năng là những gì mà đối tượng có thể làm.
  - Khi trở thành đối tượng trong chương trình thì dữ liệu sẽ lưu trạng thái còn các hàm thành viên sẽ đáp ứng với các khả năng của đối tượng.

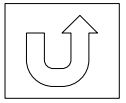
1. Đối tượng (tiếp)

✧ Ví dụ một đối tượng thang máy thì dữ liệu có thể là:

- Tầng hiện tại
- Số lượng hành khách
- Các nút ấn

Các hàm thành viên có thể là:

- DiXuong()
- DiLen()
- MoCua()
- DongCua()
- LayTTin()
- TinhTangSeToi()

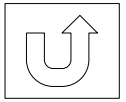


2. Lớp (class)

- Một lớp đối tượng (gọi tắt là lớp) là một mô tả về một nhóm đối tượng tương tự nhau. Nó xác định dữ liệu gì và các hàm nào sẽ có trong các đối tượng của lớp đó.
- Khái niệm lớp trong lập trình hướng đối tượng giống khái niệm lớp trong sinh học. Ví dụ: cá chép, cá trôi, cá mè đều thuộc lớp cá.
- Nếu so sánh với các kiểu dữ liệu thì lớp giống như kiểu dữ liệu, còn đối tượng giống như các biến của kiểu đó.

2. Lớp (tiếp)

- Một đối tượng được gọi là một thể hiện (instance) của một lớp bởi vì đối tượng là một trường hợp cụ thể của mô tả lớp. Vì dữ liệu trong các đối tượng của cùng một lớp là khác nhau nên dữ liệu của lớp cũng được gọi là dữ liệu thực thể (instance data).



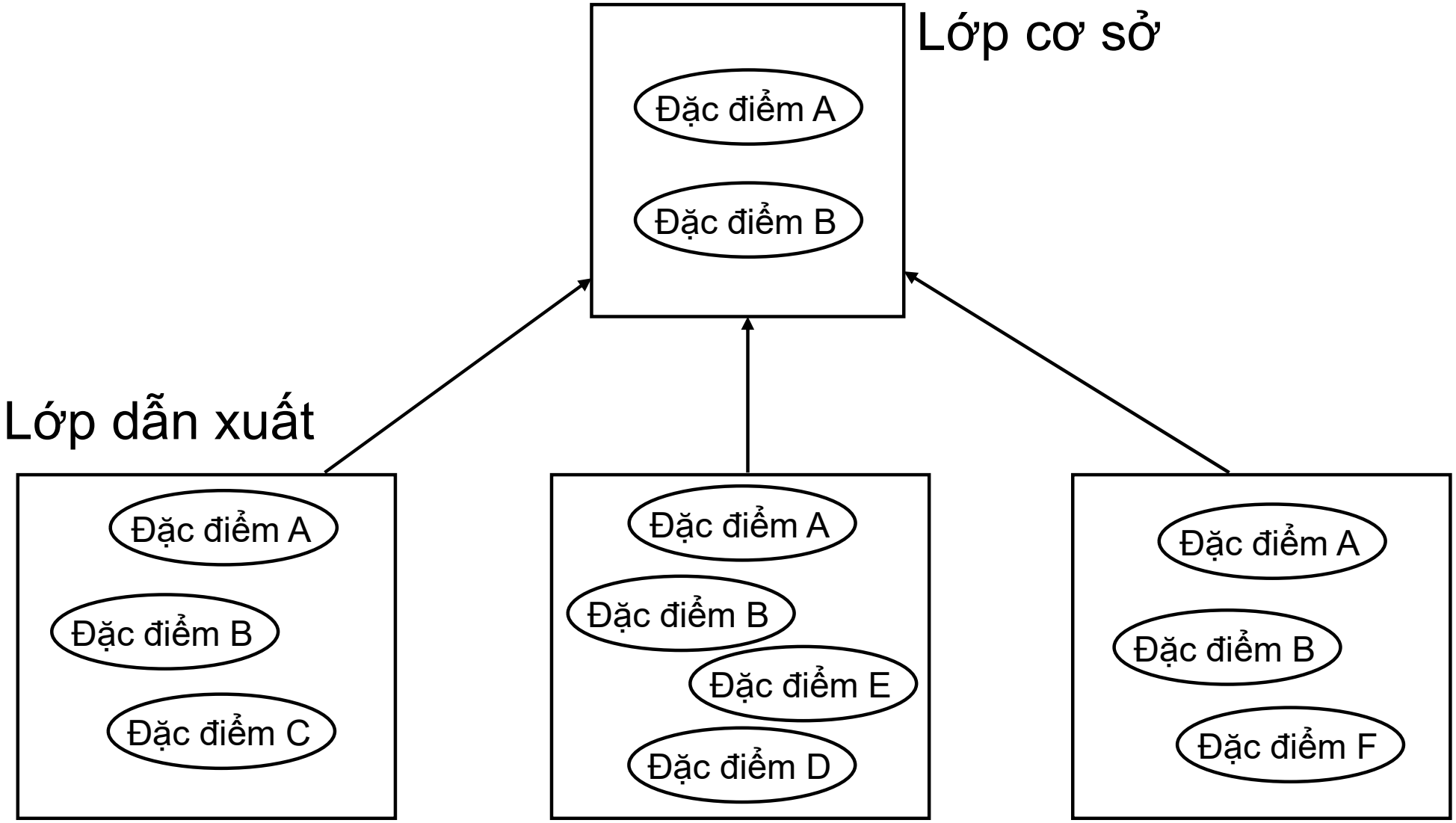
3. Sự kế thừa (inheritance)

- ✧ Trong cuộc sống ta thấy các lớp lại được chia thành các lớp con. Chẳng hạn như lớp động vật được chia thành cá, chim, động vật có vú...; lớp xe cộ được chia thành xe con, xe buýt, xe tải, xe máy,...
- ✧ Nguyên tắc phân chia thành các lớp con là các lớp con đều có các đặc điểm giống với lớp mà nó tách ra. Ví dụ: xe con, xe tải, xe buýt và xe máy, tất cả đều có tay lái, động cơ, được dùng để vận chuyển người và hàng hoá. Đây là các đặc điểm của xe cộ. Ngoài các đặc điểm này, mỗi lớp con còn có các đặc điểm của riêng nó: Các xe buýt có chỗ ngồi cho nhiều người, xe tải có thùng xe để chở hàng hoá.

3. Sự kế thừa (tiếp)

- Trong lập trình hướng đối tượng một lớp có thể làm cơ sở cho một hoặc nhiều lớp con khác nhau. Một lớp như vậy gọi là lớp cơ sở. Các lớp mà được định nghĩa là có các đặc điểm của lớp cơ sở nhưng thêm vào các đặc điểm mới của riêng nó gọi là các lớp dẫn xuất. Như vậy, các lớp dẫn xuất kế thừa những đặc điểm của lớp cơ sở.

3. Kế thừa (tiếp)

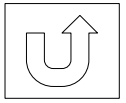


4. Sự sử dụng lại (Reusability)

- Khi một lớp đã được viết hoàn hảo thì có thể bán cho những người lập trình khác để sử dụng trong các chương trình của riêng họ. Việc này gọi là sự sử dụng lại.
- Việc sử dụng lại tương tự như việc sử dụng một thư viện hàm trong lập trình cấu trúc. Tuy nhiên, trong lập trình hướng đối tượng, nhờ có sự kế thừa mà ý tưởng sử dụng lại được mở rộng rất nhiều. Người lập trình có thể lấy một lớp đã có, thêm các đặc điểm và khả năng cho nó mà không cần thay đổi gì. Để làm được điều này người lập trình chỉ đơn giản tạo ra một lớp dẫn xuất kế thừa toàn bộ các đặc điểm của lớp đã có và còn có thể thêm vào các đặc điểm mới.

4. Sự sử dụng lại (Tiếp)

- Ví dụ: giả sử ta đã viết (hoặc mua) một lớp tạo hệ thống menu. Lớp menu này rất tốt và ta không muốn thay đổi nó, nhưng ta lại muốn làm cho một số mục menu nhấp nháy. Để làm được điều này ta chỉ đơn giản tạo ra một lớp dẫn xuất kế thừa tất cả các khả năng của lớp đã có nhưng có thêm các mục menu nhấp nháy.
- Việc có thể sử dụng lại các phần mềm đã có là một lợi ích chính của lập trình hướng đối tượng.

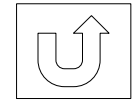


5. Sự đa hình (polymorphism)

- Trong lập trình hướng đối tượng ta có thể sử dụng các hàm và các toán tử theo nhiều cách khác nhau tùy thuộc vào những gì mà chúng tác động. Đây gọi là sự đa hình.
- Sự đa hình có thể thực hiện theo hai cách:
  - Đa hình tại thời điểm biên dịch thông qua việc chồng hàm và chồng toán tử.
  - Đa hình tại thời điểm chạy chương trình thông qua việc sử dụng hàm ảo.

5. Sự đa hình và chồng hàm (tiếp)

- Chồng hàm cho phép có nhiều hàm trùng tên nhưng có đối số khác nhau. Chồng toán tử cho phép sử dụng các toán tử đã có (chẳng hạn như +, -) tác động trên các kiểu dữ liệu mới do người lập trình định nghĩa. Khi các hàm hay các toán tử này được gọi thì trình biên dịch biết cách chọn hàm, toán tử nào để thực hiện.
- Trường hợp lớp dẫn xuất và lớp cơ sở có hàm thành viên giống hệt nhau thì khi gọi hàm thành viên này trình biên dịch sẽ không xác định được là gọi hàm nào, hàm trong lớp cơ sở hay lớp dẫn xuất. Chỉ đến khi chạy chương trình mới biết được hàm nào được gọi dựa vào kiểu đối tượng gọi hàm đó.



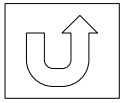
6. Đóng gói thông tin (Encapsulation)

- Trong LTHDT người ta phân biệt hai công việc: thứ nhất là công việc tạo ra các lớp đối tượng (class creators), thứ hai là công việc sử dụng các lớp đối tượng này.
- Khi tạo lớp, người tạo lớp sẽ xác định những gì cho phép người sử dụng lớp truy nhập, phần còn lại được che giấu và không cho người sử dụng lớp quyền truy nhập.

6. Đóng gói thông tin (Encapsulation)

- Khả năng che giấu dữ liệu cho phép những người tạo lớp có thể thay đổi hay định nghĩa lại lớp mà vẫn chắc chắn rằng không ảnh hưởng tới chương trình của những người sử dụng lớp này.
- C++ sử dụng các từ khóa sau để xác định khả năng truy nhập các thông tin dữ liệu từ bên ngoài lớp:

  public, private, và protected.

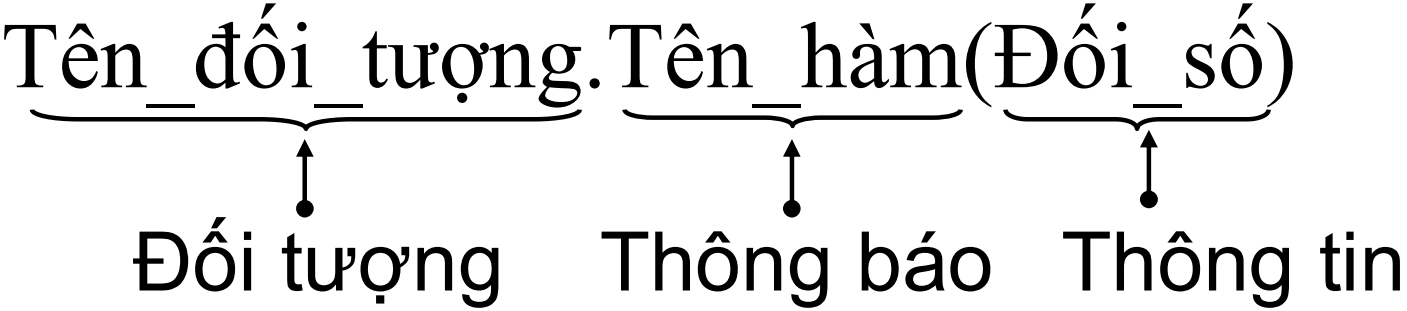


7. Truyền thông điệp

- ✧ Chương trình hướng đối tượng bao gồm một tập các đối tượng và mối quan hệ giữa các đối tượng với nhau.
- ✧ Các đối tượng gửi và nhận thông tin với nhau giống như con người trao đổi với nhau. Chính nguyên lý trao đổi thông tin bằng cách truyền thông điệp giúp chúng ta dễ dàng xây dựng hệ thống mô tả được đầy đủ, trung thực hệ thống trong thực tế. Truyền thông điệp cho một đối tượng tức là báo cho nó phải thực hiện một việc gì đó. Cách đáp ứng của đối tượng được mô tả qua các hàm thành viên của đối tượng.

7. Truyền thông điệp (tiếp)

- Thông điệp truyền đi phải chỉ ra được tên đối tượng nhận thông điệp, tên hàm cần thực hiện và thông tin truyền đi. Cấu trúc một thông điệp như sau:



III. Các ngôn ngữ lập trình hướng đối tượng

- Tư tưởng lập trình hướng đối tượng có thể cài đặt trong nhiều ngôn ngữ lập trình khác nhau như C, Pascal. Tuy nhiên, nếu sử dụng những ngôn ngữ không phải là ngôn ngữ hướng đối tượng thì sẽ gặp rất nhiều khó khăn, nhất là với những chương trình lớn và phức tạp. Những ngôn ngữ được thiết kế để hỗ trợ cho việc mô tả, cài đặt các khái niệm của phương pháp lập trình hướng đối tượng gọi chung là ngôn ngữ hướng đối tượng.

III. Các ngôn ngữ lập trình hướng đối tượng (tiếp)

- Các ngôn ngữ lập trình được gọi là ngôn ngữ hướng đối tượng phải có các đặc điểm sau:
  - Bao gói thông tin: có thể đưa dữ liệu và các hàm thao tác trên dữ liệu đó vào một cấu trúc (gọi là lớp)
  - Cơ chế che giấu dữ liệu
  - Tự động tạo lập và xóa bỏ các đối tượng
  - Sự kế thừa
  - Sự đa hình (chồng hàm, chồng toán tử và liên kết động)
- C++, Smalltalk, Object Pascal, Java, C#, … là các ngôn ngữ lập trình hướng đối tượng.