Học phần:

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG (OBJECT-ORIENTED PROGRAMMING)

Thông tin về học phần:

Số tín chỉ: 3

1. Điểm quá trình: 5 đ
2. Điểm thi cuối kỳ: 5 đ

Giảng viên: Trương Công Tuấn

tctuan@husc.edu.vn

1. Đánh giá quá trình học tập: 50%

Ký hiệu	Hình thức đánh giá	Phương pháp đánh giá	Trọng số
CA-1.1	Điểm danh theo từng buổi học, sinh viên làm bài tập ở nhà và nộp online	Điểm danh	15%
CA-1.2	Làm bài kiểm tra trên máy tính lần 1	Kiểm tra trên máy tính	15%
CA-1.3	Làm bài kiểm tra trên máy tính lần 2	Kiểm tra trên máy tính	20%
Tổng cộng			50%

2. Đánh giá cuối kỳ: 50%

Ký hiệu	Bài đánh giá	Phương pháp đánh giá	Trọng số
CA-2.1	Làm bài trên máy tính	Làm bài trên máy tính	50%
Tổng cộng			50%

TÀI LIỆU THAM KHẢO

1. Trương Công Tuấn. Giáo trình Lập trình hướng đối tượng. NXB Đại học Huế, 2019.
2. Phạm Văn Ât, C++ và Lập trình hướng đối tượng, NXB Khoa học và Kỹ thuật, 1999.
3. Nguyễn Thanh Thủy, Lập trình hướng đối tượng với C++, NXB Khoa học và Kỹ thuật, 1999.
4. Nguyễn Thanh Thủy, Bài tập Lập trình hướng đối tượng với C++, NXB Khoa học và Kỹ thuật, 1999.
5. Truong Cong Tuan – Tran Thanh Luong. Object-Oriented Programming, Hue University Publishing House, 2019.

MỞ ĐẦU

- Lập trình hướng đối tượng là phương pháp lập trình mới trên bước đường tiến hóa của việc lập trình máy tính, nhằm giúp chương trình trở nên linh hoạt, tin cậy và dễ phát triển.
- Tư tưởng lập trình hướng đối tượng được áp dụng cho hầu hết các ngôn ngữ lập trình như C++, Python, Java, Visual C, C#,...
- Việc nghiên cứu phương pháp lập trình mới này là thật sự cần thiết đối với những người làm Tin học.

CHƯƠNG 1
CÁC KHÁI NIỆM CƠ SỞ CỦA LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

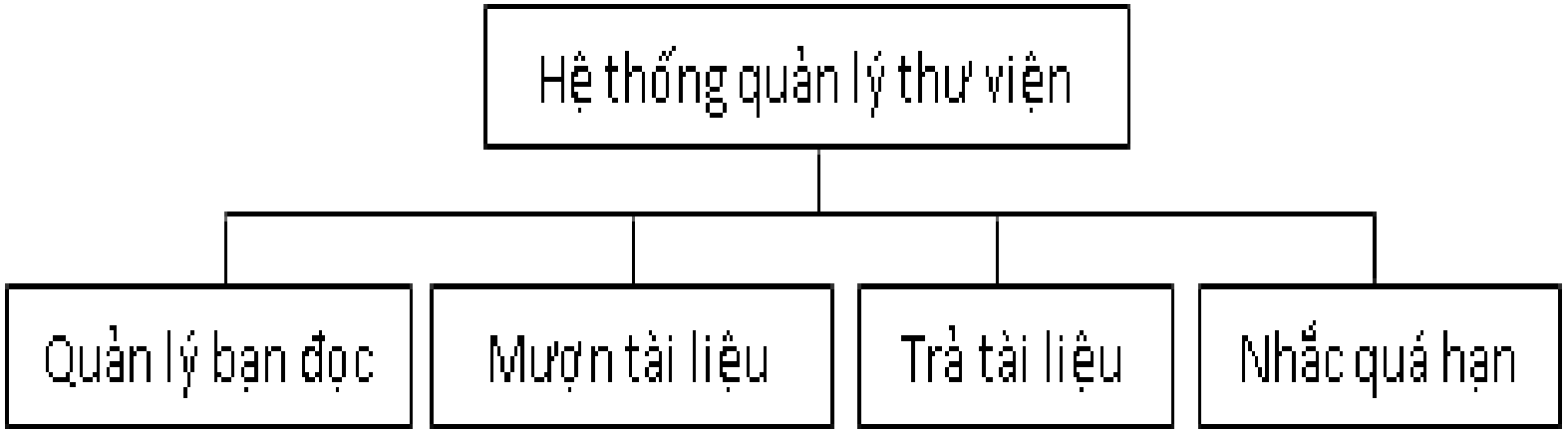
1. *Thảo luận về các cách tiếp cận trong lập trình*
2. *Các khái niệm cơ sở của phương pháp hướng đối tượng*
3. *Các bước cần thiết để thiết kế chương trình theo hướng đối tượng*
4. *Các ưu điểm của lập trình hướng đối tượng*
5. *Các ngôn ngữ hướng đối tượng*
6. *Một số ứng dụng của lập trình hướng đối tượng*

1.1. Các cách tiếp cận trong lập trình

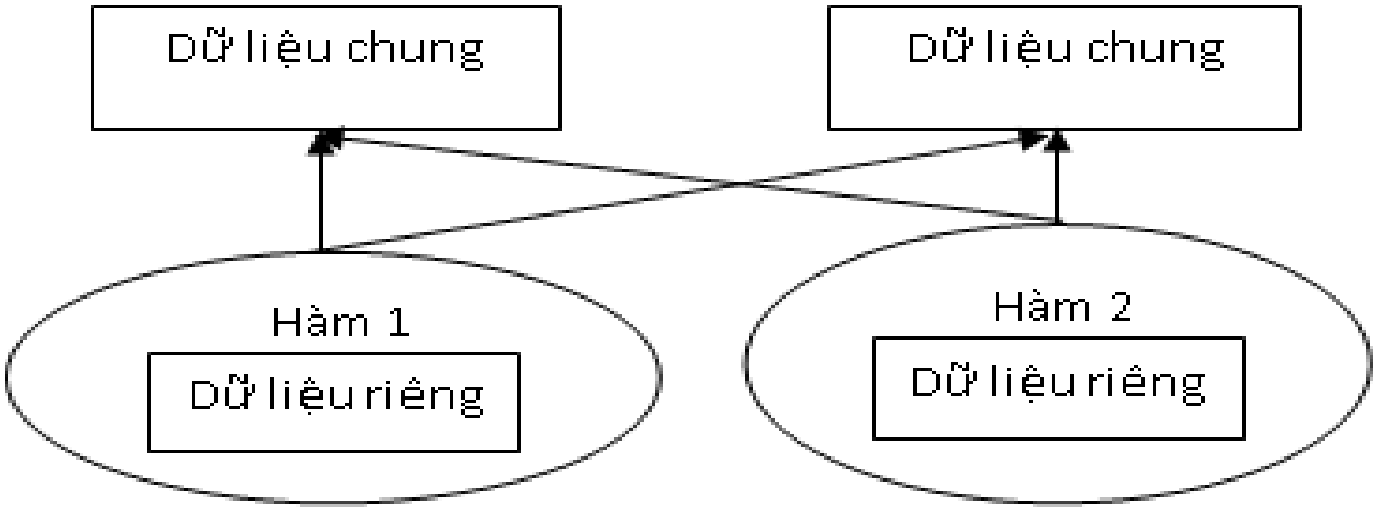
1.1.1. Tiếp cận hướng thủ tục

- Cách tiếp cận lập trình truyền thống là lập trình hướng thủ tục (LTHTT). Theo cách tiếp cận này thì một hệ thống phần mềm được xem như là dãy các công việc cần thực hiện như đọc dữ liệu, tính toán, xử lý, lập báo cáo và in ấn kết quả v.v... Mỗi công việc đó sẽ được thực hiện bởi một số hàm nhất định.
- Như vậy trọng tâm của cách tiếp cận này là các chức năng. Một hệ thống được phân tích dựa trên các chức năng sẽ được chia thành các hệ thống con và tạo ra cấu trúc phân cấp các chức năng.
- Các ngôn ngữ lập trình bậc cao như PASCAL, C,..., là những ngôn ngữ lập trình hướng thủ tục.

- Chẳng hạn, hệ thống quản lý thư viện có thể phân chia từ trên xuống (top-down) như hình sau:



- Chương trình được xây dựng theo tiếp cận HTT có ưu điểm là có cấu trúc rõ ràng, chương trình sẽ bao gồm nhiều hàm/thủ tục nhằm thực hiện nhiều chức năng khác nhau và mỗi khi muốn trao đổi dữ liệu được với nhau thì nhất thiết phải sử dụng dữ liệu chung hoặc liên kết với nhau bằng cách truyền tham biến.
- Mỗi hàm/thủ tục không những chỉ thao tác, xử lý trên những biến dữ liệu cục bộ mà còn phải sử dụng các biến chung, thường đó là các biến toàn cục.



Những nhược điểm chính của LTHTT là:

- Chương trình khó kiểm soát và khó khăn trong việc bổ sung, nâng cấp chương trình.
- Mô hình được xây dựng theo cách tiếp cận hướng thủ tục không mô tả được đầy đủ, trung thực hệ thống trong thực tế.
- Phương pháp LTHTT đặt trọng tâm vào **hàm** là hướng tới hoạt động sẽ không thực sự tương ứng với các thực thể trong hệ thống của thế giới thực.

1.1.2. Tiếp cận hướng đối tượng

- Lập trình hướng đối tượng là phương pháp lập trình lấy **đối tượng làm nền tảng để xây dựng thuật giải, xây dựng chương trình.**
- Đối tượng được xây dựng trên cơ sở **gắn cấu trúc dữ liệu với các phương thức** sẽ thể hiện được đúng cách mà chúng ta suy nghĩ, bao quát về thế giới thực.

- Điểm căn bản của phương pháp LTHĐT là thiết kế chương trình xoay quanh dữ liệu của hệ thống. Nghĩa là các thao tác xử lý của hệ thống được gắn liền với dữ liệu và như vậy khi có sự thay đổi của cấu trúc dữ liệu thì chỉ ảnh hưởng đến một số ít các phương thức xử lý liên quan.
- LTHĐT không cho phép dữ liệu chuyển động tự do trong hệ thống. **Dữ liệu được gắn chặt với từng phương thức thành các vùng riêng** mà các phương thức đó tác động lên và nó được bảo vệ để cấm việc truy nhập tùy tiện từ bên ngoài.

Tóm lại LTHĐT có những đặc tính chủ yếu:

1. Tập trung vào dữ liệu thay cho các phương thức.
2. Chương trình được chia thành các lớp đối tượng.
3. Các cấu trúc dữ liệu được thiết kế sao cho đặc tả được các đối tượng.
4. Các phương thức xác định trên các vùng dữ liệu của đối tượng được gắn với nhau trên cấu trúc dữ liệu đó.
5. Dữ liệu được bao gói, che dấu và không cho phép các hàm bên ngoài truy nhập tự do.
6. Các đối tượng trao đổi với nhau thông qua các phương thức.
7. Dữ liệu và các phương thức mới có thể dễ dàng bổ sung vào đối tượng nào đó khi cần thiết.
8. Chương trình được thiết kế theo kiểu dưới-lên (bottom-up).

1.2. Các khái niệm cơ bản của lập trình hướng đối tượng

1.2.1. Đối tượng

Trong thế giới thực, khái niệm đối tượng được hiểu như là một thực thể, nó có thể là người, vật, đồ vật,... Mỗi đối tượng đặc trưng bởi các thuộc tính và hành vi riêng của nó. Hình sau minh họa 2 đối tượng:

Họ tên: ABC
Tuổi: 28
Cân nặng: 65 kg

Hành vi:
- Đi bộ
- Nói
- Ngủ

Hiệu: Honda
Màu: Đỏ
Năm: 2019

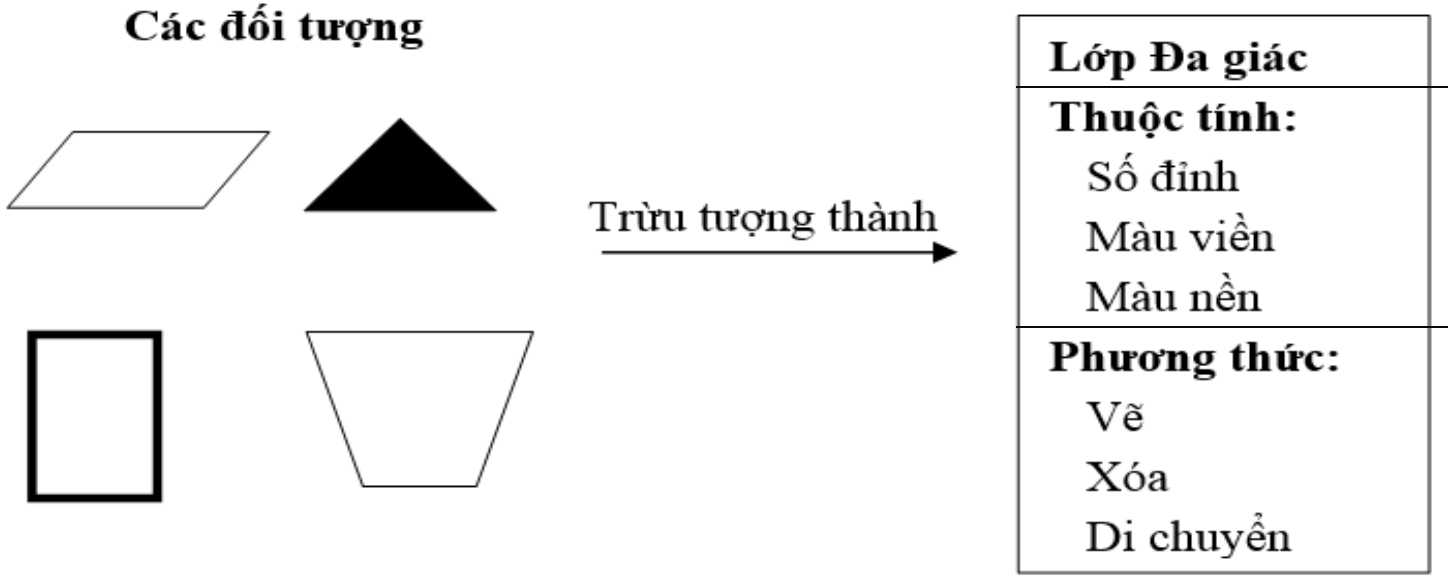
Hành vi:
- Khởi động
- Di chuyển
- Dừng

1.2.2. Lớp

Lớp là một khái niệm mới trong LTHĐT so với kỹ thuật LTHTT. Nó là một khuôn mẫu mô tả các đối tượng có cấu trúc dữ liệu và phương thức giống nhau.

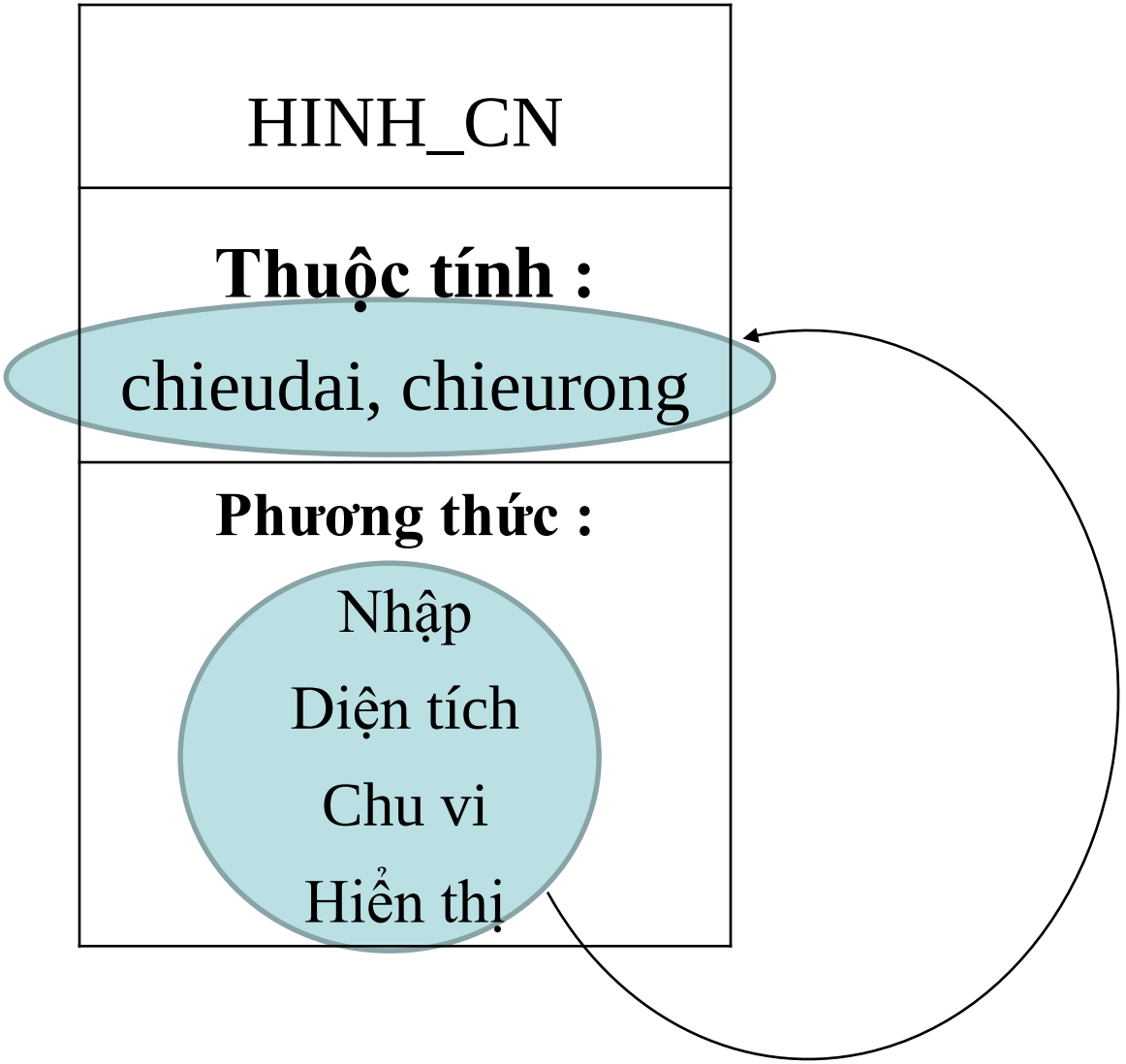
Mỗi đối tượng là một thể hiện cụ thể của lớp.

Hình 1.5 sau minh họa lớp đa giác để mô tả các đối tượng là đa giác:



Hình 1.5. Minh họa lớp Đa giác

Ví dụ lớp mô tả các đối tượng là hình chữ nhật



- Trong LTHĐT, lớp là một kiểu dữ liệu được định nghĩa bởi người sử dụng và các đối tượng là các biến có kiểu của lớp.
- Các hàm được định nghĩa trong một lớp được gọi là các phương thức (method) và chúng là các hàm duy nhất có thể xử lý dữ liệu của các đối tượng của lớp đó.
- **Mỗi lớp bao gồm: danh sách các thuộc tính (attribute) và danh sách các phương thức (method) để xử lý các thuộc tính đó.**
- Lớp là khái niệm tĩnh, có thể nhận biết ngay từ văn bản chương trình, ngược lại đối tượng là khái niệm động, nó được xác định trong bộ nhớ của máy tính.

1.2.3. Trừu tượng hóa dữ liệu và bao gói thông tin

- ***Trừu tượng hóa*** **là cách biểu diễn những đặc tính chính và bỏ qua những chi tiết vụn vặt hoặc những giải thích**. Ví dụ ta có thể định nghĩa một lớp để mô tả các đối tượng trong không gian hình học bao gồm các thuộc tính trừu tượng như là kích thước, hình dáng, màu sắc và các phương thức xác định trên các thuộc tính này.
- **Việc đóng gói dữ liệu và các phương thức vào một đơn vị cấu trúc lớp được xem như một nguyên tắc *bao gói thông tin*.**
- Dữ liệu được tổ chức sao cho thế giới bên ngoài (các đối tượng ở lớp khác) không truy nhập vào, mà chỉ cho phép các phương thức trong cùng lớp hoặc trong những lớp có quan hệ kế thừa với nhau mới được quyền truy nhập.

- Nguyên lý bao gói thông tin như thế ta thường hay gặp trong thực tế, chẳng hạn, như thiết kế viên thuốc, ta chỉ biết nó chữa bệnh nào đó và một số thành phần chính, còn cụ thể bên trong nó có chứa gì thì hoàn toàn không biết.
- Nguyên tắc bao gói dữ liệu để ngăn cấm sự truy nhập trực tiếp trong lập trình được gọi là sự che giấu thông tin.

1.2.4. Truyền thông báo

- Các đối tượng gửi và nhận thông tin với nhau giống như con người trao đổi với nhau. ***Truyền thông báo*** cho một đối tượng là yêu cầu đối tượng thực hiện một việc gì đó. Cách ứng xử của đối tượng được mô tả bên trong lớp thông qua các phương thức.
- Trong chương trình, **thông báo gửi đến cho một đối tượng chính là yêu cầu thực hiện một công việc cụ thể**, nghĩa là sử dụng những hàm tương ứng để xử lý dữ liệu đã được khai báo trong đối tượng đó.
- Vì vậy, trong thông báo phải chỉ ra được hàm cần thực hiện trong đối tượng nhận thông báo. Thông báo truyền đi cũng phải xác định tên đối tượng và thông tin truyền đi.

- Ví dụ, lớp CANBO có thể hiện là đối tượng cụ thể **ob** được đại diện bởi Hoten, nhận được thông báo cần tính lương thông qua phương thức Tinhluong đã được xác định trong lớp CANBO. Thông báo đó sẽ được xử lý như sau:

ob.Tinhluong(Hoten)

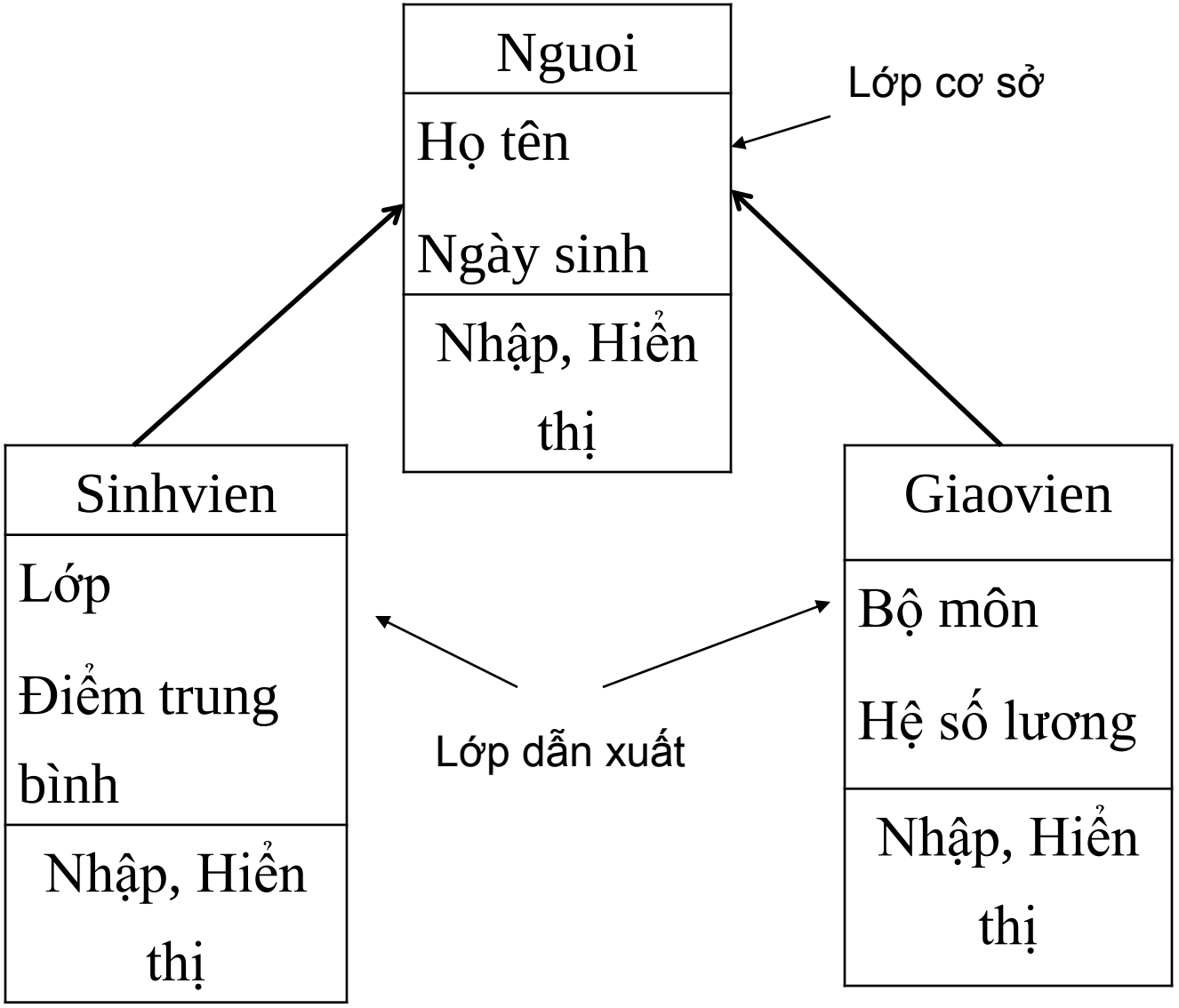
1.2.5. Sự quá tải (Tương ứng bội) - Overloading

- **Sự quá tải** là khả năng của một khái niệm (chẳng hạn các phép toán) có thể sử dụng với nhiều chức năng khác nhau.
- Ví dụ, phép + có thể biểu diễn cho phép "cộng" các số nguyên (int), số thực (float), số phức (complex) hoặc xâu ký tự (string) v.v... Hành vi của phép toán tương ứng bội phụ thuộc vào kiểu dữ liệu mà nó sử dụng để xử lý.
- **Sự quá tải** đóng vai quan trọng trong việc tạo ra các đối tượng có cấu trúc bên trong khác nhau nhưng cùng dùng chung một giao diện bên ngoài (chẳng hạn tên gọi).

1.2.6. Kế thừa

- *Kế thừa* là quá trình mà các đối tượng của lớp này được quyền sử dụng một số thuộc tính và phương thức của các đối tượng của lớp khác.
- Kế thừa cho phép ta định nghĩa một lớp mới trên cơ sở các lớp đã tồn tại. Lớp mới này được gọi là **lớp cơ sở (lớp cha)**, lớp con ngoài những thành phần được kế thừa, sẽ có thêm những thuộc tính và các hàm mới (Lớp kế thừa từ lớp cơ sở được gọi là **lớp dẫn xuất (lớp con))**.
- Nguyên lý kế thừa hỗ trợ cho việc tạo ra cấu trúc phân cấp các lớp.

Ví dụ


Nguoi
Họ tên
Ngày sinh
Nhập, Hiển thị
Lớp cơ sở
Sinhvien
Lớp
Điểm trung bình
Nhập, Hiển thị
Giaovien
Bộ môn
Hệ số lương
Nhập, Hiển thị
Lớp dẫn xuất

1.2.7. Tính đa hình

Tính đa hình là khả năng để cho một thông báo có thể thay đổi cách thực hiện của nó theo lớp cụ thể của đối tượng nhận thông báo.

Khi một lớp dẫn xuất được tạo ra, nó có thể thay đổi cách thực hiện các phương thức nào đó mà nó thừa hưởng từ lớp cơ sở của nó.

Một thông báo khi được gửi đến một đối tượng của lớp cơ sở, sẽ dùng phương thức đã định nghĩa cho nó trong lớp cơ sở. Nếu một lớp dẫn xuất định nghĩa lại một phương thức thừa hưởng từ lớp cơ sở của nó thì một thông báo có cùng tên với phương thức này, khi được gửi tới một đối tượng của lớp dẫn xuất sẽ gọi phương thức đã định nghĩa cho lớp dẫn xuất.

- Như vậy, đa hình là khả năng cho phép gửi cùng một thông báo đến những đối tượng khác nhau có cùng chung một đặc điểm. Nói cách khác thông báo được gửi đi không cần biết đối tượng nhận thuộc lớp nào, chỉ biết rằng tập hợp các đối tượng nhận có chung một tính chất nào đó.
- Chẳng hạn, thông báo “vẽ hình” được gửi đến cả hai đối tượng hình hộp và hình tròn. Trong hai đối tượng này đều có chung phương thức vẽ hình, tuy nhiên tùy theo thời điểm mà đối tượng nhận thông báo, hình tương ứng sẽ được vẽ lên.

- Trong các ngôn ngữ lập trình hướng đối tượng, tính đa hình thể hiện qua khả năng cho phép mô tả những phương thức có tên giống nhau trong các lớp khác nhau.
- Đặc điểm này giúp người lập trình không phải viết những cấu trúc điều khiển rườm rà trong chương trình, các khả năng khác nhau của thông báo chỉ thực sự đòi hỏi khi chương trình thực hiện.
- Tính đa hình dựa trên sự nối kết, khi các phương thức kiểu đa hình được sử dụng thì trình biên dịch chưa thể xác định phương thức nào sẽ được gọi.
- Phương thức cụ thể được gọi sẽ tùy thuộc vào việc đối tượng nhận thông báo lúc đó là thuộc lớp nào, do đó phương thức được gọi chỉ xác định được vào lúc **chương trình chạy**. Điều này gọi là **sự liên kết muộn** (late binding) hay liên kết lúc chạy (runtime binding) vì nó xảy ra khi chương trình đang thực hiện.

1.3. Các bước cần thiết để thiết kế chương trình theo hướng đối tượng

Chương trình theo hướng đối tượng bao gồm một tập các đối tượng và mối quan hệ giữa các đối tượng với nhau. Vì vậy, lập trình trong ngôn ngữ hướng đối tượng bao gồm các bước sau:

1. Xác định các dạng đối tượng (lớp) của bài tóan.
2. Tìm kiếm các đặc tính chung (dữ liệu chung) trong các dạng đối tượng này, những gì chúng cùng nhau chia xẻ.
3. Xác định lớp cơ sở dựa trên các đặc tính chung của các dạng đối tượng.
4. Từ lớp cơ sở, xây dựng các lớp dẫn xuất chứa các thành phần, những đặc tính không chung còn lại của các dạng đối tượng. Ngoài ra, ta còn đưa ra các lớp có quan hệ với các lớp cơ sở và lớp dẫn xuất.

1.4. Các ưu điểm của lập trình hướng đối tượng

Cách tiếp cận hướng đối tượng giải quyết được nhiều vấn đề tồn tại trong quá trình phát triển phần mềm và tạo ra được những sản phẩm phần mềm có chất lượng cao. Những ưu điểm chính của LTHĐT là:

1. 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.
2. Chương trình được xây dựng từ những đơn thể (đối tượng) trao đổi với nhau nên việc thiết kế và lập trình sẽ được thực hiện **theo quy trình nhất định** chứ không phải dựa vào kinh nghiệm và kỹ thuật như trước. Điều này đảm bảo rút ngắn được thời gian xây dựng hệ thống và tăng năng suất lao động.

3. 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.
4. Có thể xây dựng được ánh xạ các đối tượng của bài toán vào đối tượng của chương trình.
5. Cách tiếp cận thiết kế đặt trọng tâm vào đối tượng, giúp chúng ta xây dựng được mô hình chi tiết và gần với dạng cài đặt hơn.
6. Những hệ thống hướng đối tượng dễ mở rộng, nâng cấp thành những hệ lớn hơn.
7. Kỹ thuật truyền thông báo trong việc trao đổi thông tin giữa các đối tượng giúp cho việc mô tả giao diện với các hệ thống bên ngoài trở nên đơn giản hơn.
8. Có thể quản lý được độ phức tạp của những sản phẩm phần mềm.

1.5. Các ngôn ngữ hướng đối tượng:

C++, Java, Python, MS Visual C++, C#,…

Câu hỏi Chương 1

1. Sinh viên trả lời các Câu hỏi cuối chương 1 , Trang 16 trong Giáo trình Lập trình Hướng đối tượng.
2. Cho biết kết quả thực thi chương trình sau (có giải thích):

```
#include <stdio.h>

int a = 2, b = 3; //biến toàn cục
int f1()
{ int k;
  ++a;
  k = a + b;
  return k;
}
int f2()
{ int k;
  ++b;
  k = a + b;
  return k;
}
int  main()
{
  printf("\n f1 = %d", f1());  //f1
  printf("\n f2 = %d", f2());  //f2
  //
  printf("\n a = %d, b = %d ",a,b); //gia tri a, b bi thay doi khi goi cac ham f1, f2
  return 0;
}
```