Bài 2: Tổng quan về Lập trình hướng đối tượng

Trường ĐH Công nghệ, ĐHQGHN

Đã học

- Vấn đề của lập trình cấu trúc để cần sự tồn tại của lập trình hướng đối tượng?
  - Dữ liệu và mã xử lý tách rời nhau
- Lập trình hướng đối tượng
  - Che giấu dữ liệu (đóng gói)
  - Truy cập thông qua phương thức (giao diện)
  - Lớp và đối tượng
  - Hệ thống hướng đối tượng: nhiều đối tượng tương tác với nhau qua thông điệp
- Java: lập trình 1 lần, chạy khắp nơi
  - Java Virtual Machine

Nội dung

- Các phương pháp lập trình
- Lập trình thủ tục & Lập trình HĐT
- Các khái niệm cơ bản về Lập trình HĐT

Tài liệu tham khảo

- *Giáo trình Lập trình HĐT*, Chương 1, 3

Các phương pháp lập trình

- Lập trình phi cấu trúc (imperative / how to do)
- Lập trình thủ tục (structured / procedural, imperative)
- Lập trình logic; Lập trình hàm (declarative / what)
- Lập trình hướng đối tượng (imperative)

Lập trình phi cấu trúc

- Là phương pháp xuất hiện đầu tiên
  - các ngôn ngữ như Assembly, Basic
  - sử dụng các biến tổng thể
  - lạm dụng lệnh GOTO
- Các nhược điểm
  - khó hiểu, khó bảo trì, hầu như không thể sử dụng lại
  - chất lượng kém
  - chi phí cao
  - không thể phát triển các ứng dụng lớn

Lập trình phi cấu trúc - Ví dụ

```
10    k =1
20    gosub 100
30    if y > 120 goto 60
40    k = k+1
50    goto 20
60    print k, y
70    stop
100   y = 3*k*k + 7*k-3
110   return
```

Lập trình thủ tục (Algol, Pascal, C, ...)

- Cấu trúc lặp: for, while, do-while, repeat, ...
- Chương trình là chuỗi các hàm/thủ tục
- Tập trung vào biểu diễn thuật toán (làm như thế nào)
- Ưu điểm
  - Chương trình: cục bộ hóa, dễ hiểu, dễ bảo trì hơn
  - Dễ dàng tạo ra các thư viện phần mềm

```
int func(int j){
   return (3*j*j + 7*j-3);
}

int main(){
   int k = 1
   while (func(k) < 120)
      k++;
   printf("%d\t%d\n", k, func(k));
   return(0);
}
```

Lập trình thủ tục (2)

- Nhược điểm
  - Phân tách dữ liệu (bị động) và các xử lý (chủ động)
  - Khó đảm bảo nhất quán dữ liệu và các ràng buộc
  - Khó bảo trì mã nguồn

```
struct Date{
   int day;
   int month;
   int year;
};

void setDate(Date& date, int newDay, int newMonth, int newYear){
   date.day = newDay;
   ...
}
...
```

Bộ giá trị **newDay,newMonth,newYear** có thể dẫn đến ngày không hợp lệ!

Lập trình logic - Ví dụ

```
grandparent(X,Z) :- parent(X,Y), parent(X,Y).
parent(X,Y):- father(X,Y).
parent(X,Y):- mother(X,Y).

father(john,lily).
mother(kathy,lily).
mother(lily,bill).
father(ken,karen).
```

?-parent(lily,bill)

?-grandparent(Q,bill)

Lập trình hàm - Ví dụ

Tìm ước chung lớn nhất (Python)

```
def ucln(x, y):
    if y == 0:
        return x
    else:
        return ucln(y, x % y)
```

```
ucln(1452,407)
~> ucln(407,231)
  ~> ucln(231,176)
    ~> ucln(176,55)
      ~> ucln(55,11)
        ~> ucln(11,0)
          ~> 11
        ~> 11
      ~> 11
    ~> 11
  ~> 11
~> 11
```

Khó cho việc phân tích, kiểm thử và song song hóa chương trình (!?)

Lập trình hướng đối tượng (HĐT)

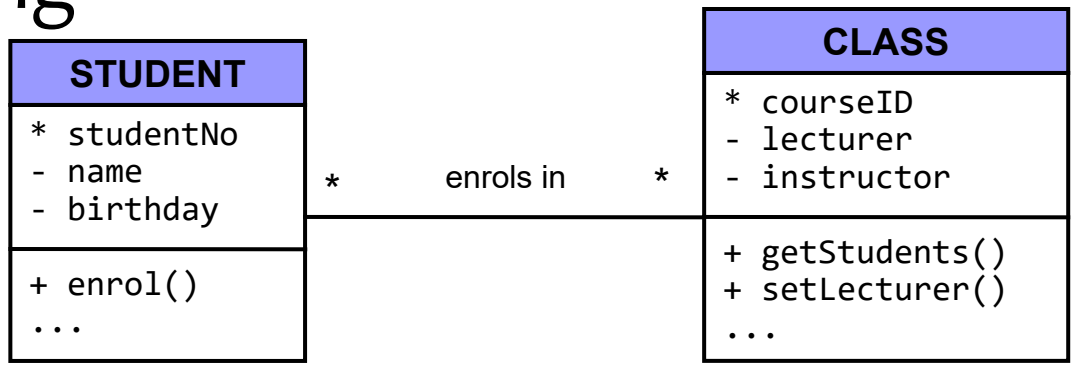
Khắc phục các hạn chế của Lập trình thủ tục

- Đóng gói dữ liệu và xử lý (che dấu dữ liệu; truy cập qua giao diện)
- Đảm bảo nhất quán dữ liệu và các ràng buộc
- Dễ bảo trì hơn

```
class Date{
public:
   void setDate(int newDay, int newMonth, int newYear);
   int getDay() { return day; } ...
private:
   int day;
   int month;
   int year;
};

void Date::setDate(int newDay, int newMonth, int newYear){
   //check validity of newDay, newMonth, newYear
   ...
   //set new values
   ...
}
```

Lập trình HĐT (OOP) là gì?

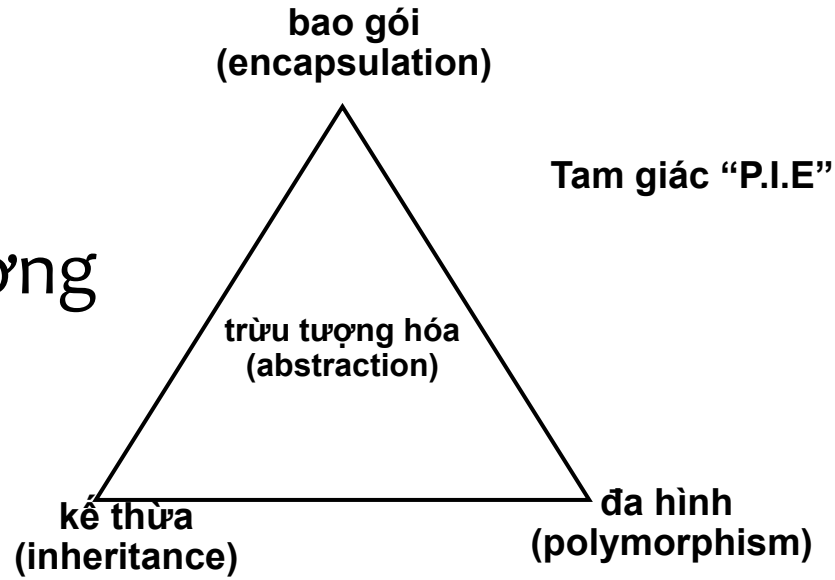
- Lập trình hướng đối tượng
  - Tái hiện / ánh xạ vấn đề trong thế giới thực vào các chương trình



  - Xây dựng cơ chế tổ chức các "thực thể" (đối tượng) có thể làm gì đó hoặc yêu cầu các đối tượng khác thực hiện
  - Tạo ra "kiểu" (lớp) của các đối tượng để cho phép tạo ra chúng mà không phải định nghĩa lại thuộc tính và hành vi cho từng đối tượng

Khái niệm cơ bản về HĐT

- Trừu tượng hóa
- Các đối tượng & lớp
  - Trạng thái và hành vi đối tượng
  - Định danh đối tượng
  - Các thông điệp
- Bao gói
  - Che dấu thông tin (cài đặt)
- Kế thừa
- Đa hình



Trừu tượng hóa

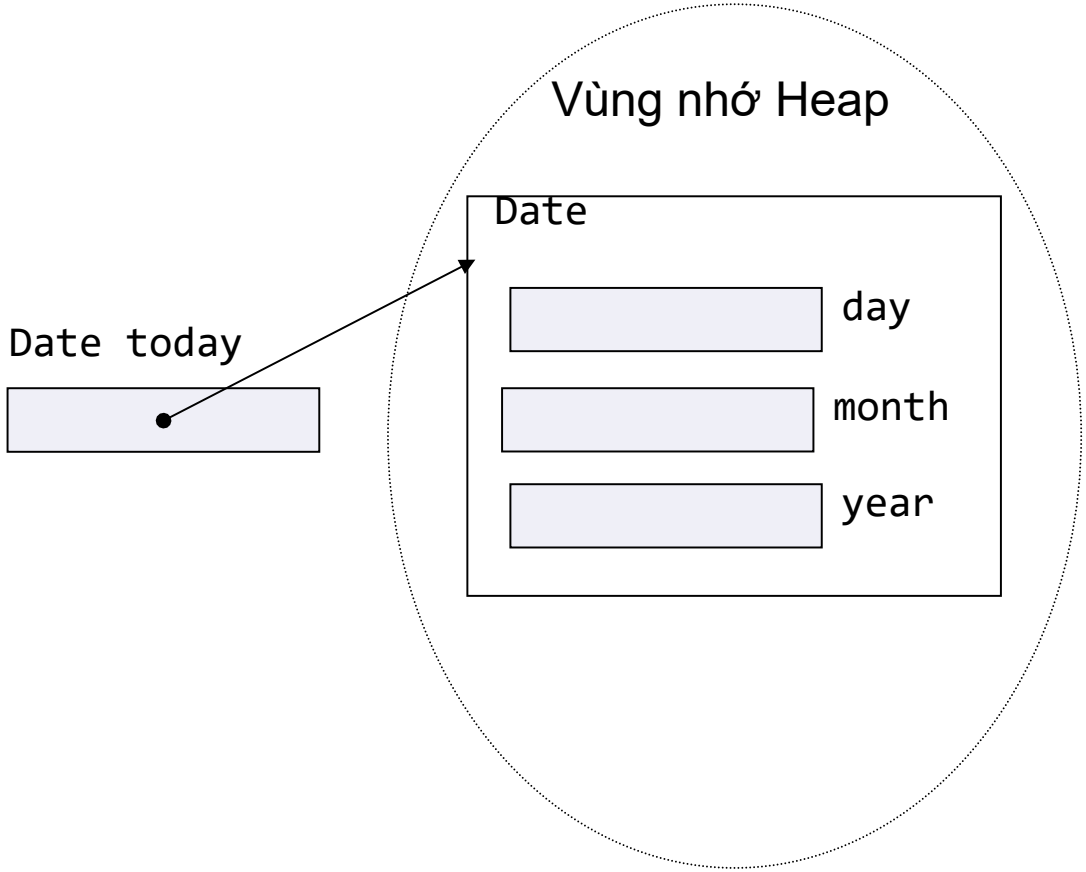
- Đặc điểm của ngôn ngữ hướng đối tượng (khởi đầu bởi Smalltalk)
  - Mọi thứ đều là đối tượng (everything is an object)
  - Hoạt động của chương trình là quá trình "nói chuyện" giữa các đối tượng bằng cách "gửi thông điệp"
  - Mỗi đối tượng có vùng nhớ riêng và được hình thành từ các đối tượng khác
  - Mỗi đối tượng có một kiểu (type)
  - Các đối tượng cùng kiểu có thể nhận cùng một thông điệp

Các đối tượng

- Bao gồm ba thành phần
  - Trạng thái
    - Được xác định bởi dữ liệu bên trong (giá trị thuộc tính)
  - Hành vi
    - Tương ứng với các phương thức
  - Định danh
    - Các đối tượng ở bất kỳ trạng thái nào luôn có định danh duy nhất
      - Các đối tượng khác nhau có thể cùng trạng thái (cùng giá trị thuộc tính) nhưng khác nhau về định danh
      - Đối tượng được thao tác thông qua tham chiếu (handle) (con trỏ trong C++; tham chiếu đối tượng trong Java)

```
class Date{
public:
   void setDate(…);
   int getDay();
   ...
private:
   int day;
   int month;
   int year;
};

Date today = new Date();

today.setDate(2, 9, 2020);
```

Đối tượng và Tham chiếu đối tượng



Các thông điệp (messages)

today.setDate(2, 9, 2020);

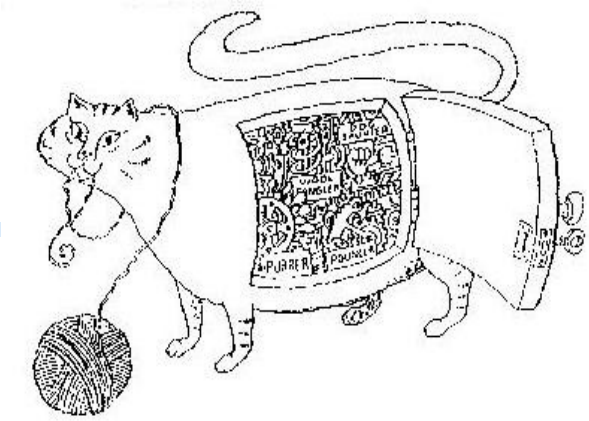
- Là cách thức để đối tượng A yêu cầu đối tượng B thực hiện phương thức của B
- Một thông điệp bao gồm
  - Tham chiếu đối tượng đích - nhận thông điệp (today)
  - Tên của phương thức cần thực hiện (setDate)
  - Các thông tin cần thiết khác - các tham số (2, 9, 2020)
- Thông điệp được cài đặt ở dạng lời gọi hàm với tham số ngầm là đối tượng nhận thông điệp (method invocation)
- Khái niệm thông điệp tạo nên sự khác biệt cho lập trình HĐT
  - Dữ liệu trở thành chủ động
  - Với các lược đồ lập trình khác
    - Các hành động được thực hiện qua cơ chế truyền điều khiển (gọi hàm/thủ tục)
    - Dữ liệu là bị động; các thủ tục thao tác trên dữ liệu

Các lớp đối tượng

```
class Date{
public:
   void setDate(…);
   ...
private:
   int day;
   ...
};

Date today = new Date();

today.setDate(2, 9, 2020);
```

- Lớp (Class)
  - Khuôn để tạo các đối tượng (các thể hiện)
  - Các đối tượng của một lớp có cùng cấu trúc và hành vi được mô tả bởi lớp đó
- Quan hệ "Kiểu dữ liệu – Biến"
  - Các lớp được thiết kế và cài đặt (kiểu dữ liệu)
  - Các đối tượng được tạo khi thực thi chương trình

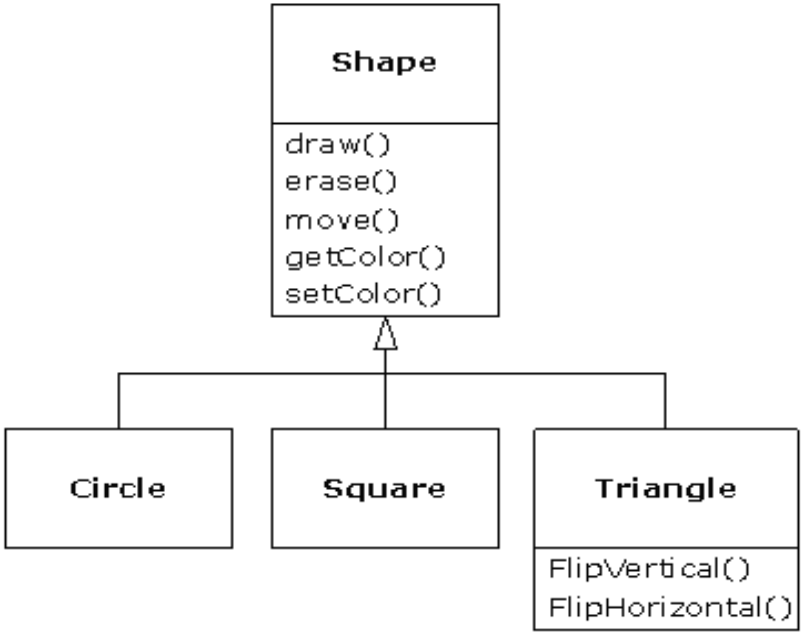
Bao gói / Che dấu thông tin



- Bao gói (encapsulation): để nhóm các thứ liên quan thành một tổng thể có định danh
  - Hàm/Thủ tục bao gói các câu lệnh
  - Đối tượng bao gói dữ liệu và thủ tục
- Che dấu thông tin: để che dấu các thông tin và chi tiết cài đặt bên trong đối với bên ngoài
  - Để các "clients" (các lập trình viên khác) xem các đối tượng như các hộp đen
  - public, private, và protected
  - Giao diện (interface) & Cài đặt (implementation)

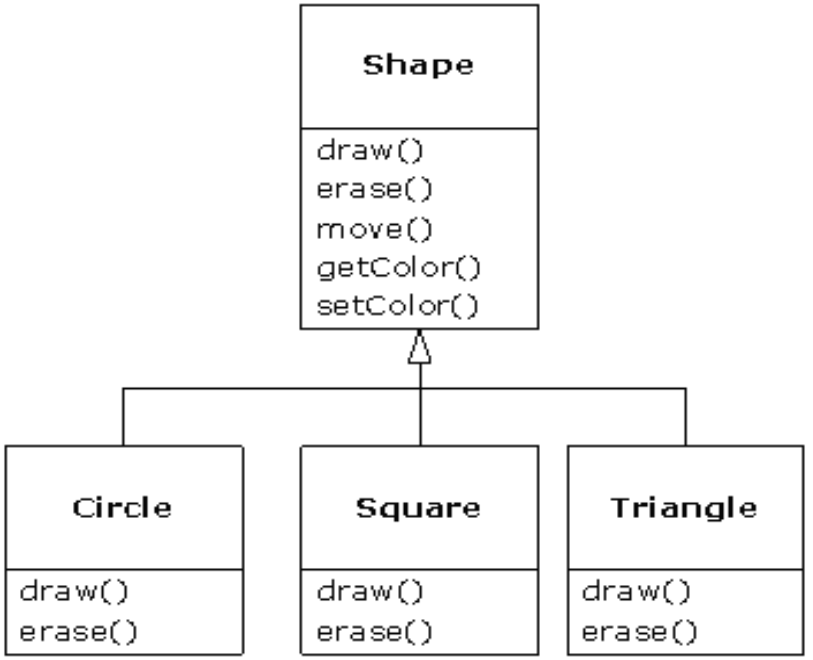
```
class Date{
public:
   void setDate(…);
   int getDay()
...
private:
   int day;
   int month;
   int year;
};

void Date::setDate{
   //check validity
   ...
   //set new values
   ...
}
```

Kế thừa



- Sử dụng lại các giao diện và cài đặt
- Các mối quan hệ "is-a"
- Cơ chế cho phép các đối tượng lớp dẫn xuất (derived class) mang đặc điểm (thuộc tính) và hành vi (phương thức) của lớp cơ sở (base class)
- Các dịch vụ tổng quát (generic) có thể được xác định và thiết kế trước khi đặc biệt hóa chúng

Đa hình



- Đa hình:
  - Khả năng được xem xét ở các dạng thức (forms / shapes) khác nhau
  - Có thể tồn tại ở nhiều dạng
- Đa hình đối tượng:
  - Đối tượng của các lớp dẫn xuất khác nhau có thể được xem như thể hiện của cùng một lớp chung (lớp cơ sở)
  - Các đối tượng của các lớp khác nhau có thể hiểu cùng một thông điệp theo các cách khác nhau
    - Ví dụ: khi nhận thông điệp draw(), các đối tượng **Rectangle** và **Triangle** hiểu và thực thi thông điệp theo cách khác nhau

Lịch sử của Lập trình HĐT

- Các ngôn ngữ lập trình HĐT là không mới
  - Simula (1967) - ngôn ngữ HĐT đầu tiên, hỗ trợ lớp, kế thừa, kết gắn động (các hàm ảo/trừu tưởng)
  - Smalltalk (1970s): giới thiệu về HĐT ("object-oriented programming")
- Ở giai đoạn đầu, ngôn ngữ HĐT chậm hơn các ngôn ngữ truyền thống
  - Trở nên phổ biến khi tốc độ máy tính tăng nhanh (kể từ giai đoạn xuất hiện các máy Pentium)

HĐT trong các ngôn ngữ lập trình

- Các hệ thống HĐT giai đoạn đầu không có các lớp
  - Chỉ các "đối tượng" và "thông điệp" (chẳng hạn Hypertalk)
- Gần đây, được hợp nhất và bổ sung thêm các khái niệm
  - Lớp
  - Kế thừa và Kết gắn động
- Một vài đặc tính HĐT có thể được cài đặt trong các ngôn ngữ lập trình thủ tục như C
- Các ngôn ngữ HĐT: Các khái niệm lập trình HĐT được nhúng vào và phải được tuân thủ
- Cấp độ HĐT ở các ngôn ngữ HĐT có thể khác nhau
  - Eiffel (thuần khiết), Java (cao), C++ (hỗn hợp)

Thanks for your attention!