# TỔNG QUAN VỀ
# LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG


**Khoa Công nghệ phần mềm**

# Nội dung

1. Các phương pháp lập trình
2. Một số khái niệm cơ bản trong OOP
3. Các đặc điểm quan trọng của OOP
4. Phân tích, thiết kế và lập trình hướng đối tượng

# 1. Các phương pháp lập trình

- Lập trình không có cấu trúc
- Lập trình có cấu trúc
- Lập trình hướng đối tượng

# 1.1 Lập trình không có cấu trúc

- Là phương pháp xuất hiện đầu tiên:
  - Các ngôn ngữ như Assembly, Basic.
  - Sử dụng các biến toàn cục.
  - Lạm dụng lệnh GOTO.
- **Nhược điểm:**
  - ?

# 1.1 Lập trình không có cấu trúc (tt)

❖ **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
```

# 1.2 Lập trình có cấu trúc

- Tổ chức phân chia chương trình thành các chương trình con (hàm/thủ tục).

  -> Vì vậy còn được gọi là **"Lập trình thủ tục".**
- Mỗi chương trình con đảm nhận xử lý một công việc nhỏ hay một nhóm công việc trong toàn bộ hệ thống.
- Việc trao đổi dữ liệu giữa các chương trình con được thực hiện thông qua các đối số của hàm và các biến toàn cục.

# 1.2 Lập trình có cấu trúc (tt)

- Sử dụng các lệnh có cấu trúc:

  for, do, while, if then else...

- Các ngôn ngữ như Pascal, C.
- **Ưu điểm:**
  - Dễ hiểu, dễ bảo trì hơn vì chương trình được module hóa.
  - Dễ dàng tạo ra các thư viện phần mềm.

# 1.2 Lập trình có cấu trúc (tt)

❖ **Ví dụ:**

```
struct Date {
    int year, mon, day;
};
void print_date(Date d) {
    printf("%d / %d / %d\n", d.day, d.mon, d.year);
}
```

# 1.2 Lập trình có cấu trúc (tt)

- **Nhược điểm:**
  - Chương trình dùng để xử lý dữ liệu nhưng lại tách rời dữ liệu.
  - Chương trình = Cấu trúc dữ liệu + Giải thuật
  - Không tự động khởi tạo hay giải phóng dữ liệu động.
  - Không mô tả được đầy đủ, trung thực hệ thống trong thực tế.

# 1.3 Lập trình hướng đối tượng

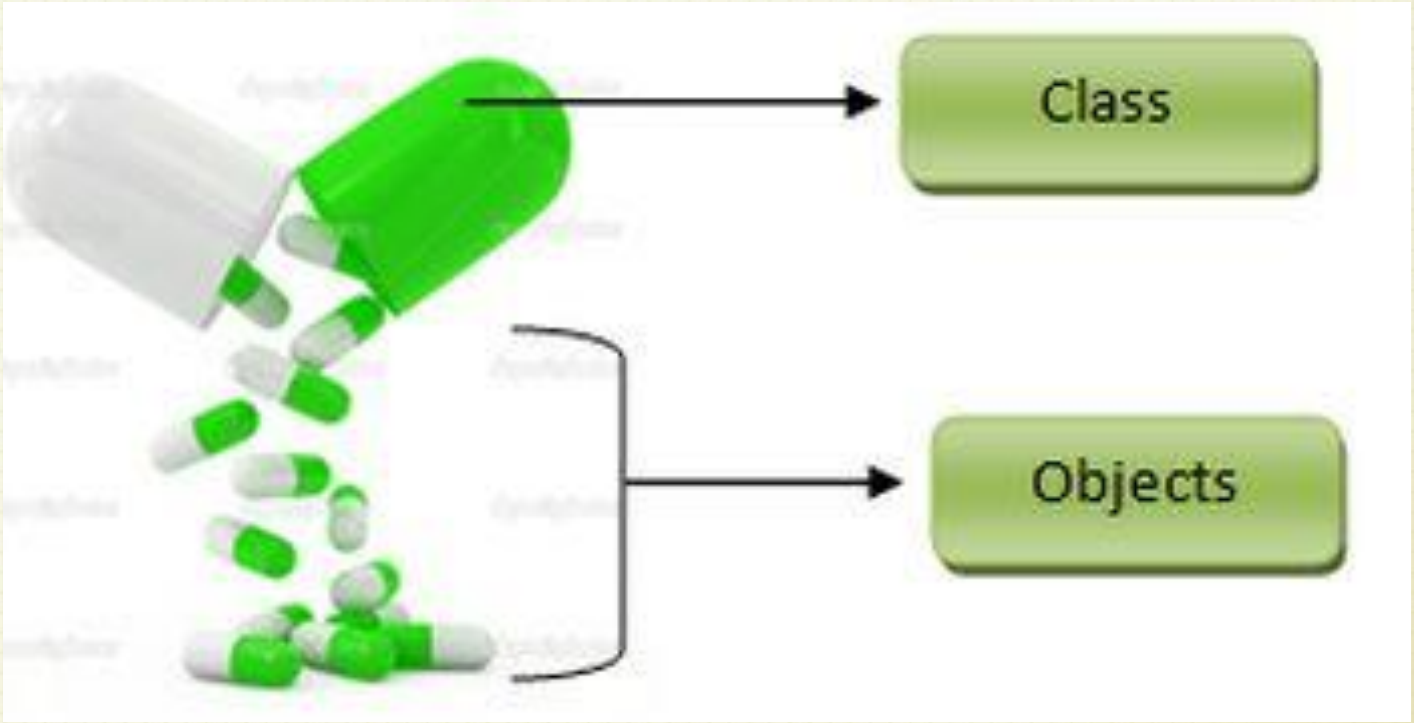
- Được xây dựng trên nền tảng của Lập trình có cấu trúc (hướng chức năng) và sự trừu tượng hóa dữ liệu.
- Một chương trình hướng đối tượng được thiết kế xoay quanh dữ liệu mà chúng ta có thể làm việc trên đó, hơn là theo bản thân chức năng của chương trình.

# 1.3 Lập trình hướng đối tượng (tt)

- Lập trình hướng đối tượng là phương pháp lập trình lấy đối tượng làm nền tảng để xây dựng chương trình.
- Một chương trình hướng đối tượng sẽ bao gồm tập các đối tượng có quan hệ với nhau.

# 2. Một số khái niệm cơ bản trong OOP

- Đối tượng (object)
- Lớp (class)



# 2. Một số khái niệm cơ bản trong OOP (tt)

## ❖ Đối tượng (object):

- Trong thế giới thực, đối tượng được hiểu như là một thực thể: người, vật, …
- Mỗi đối tượng sẽ tồn tại trong một hệ thống và có ý nghĩa nhất định trong hệ thống.
- Đối tượng giúp biểu diễn tốt hơn thế giới thực trên máy tính.
- Mỗi đối tượng bao gồm 2 thành phần:

  Thuộc tính + Thao tác (hành động)

# 2. Một số khái niệm cơ bản trong OOP (tt)

- **Ví dụ:** đối tượng Người
  - Các thuộc tính: *tên, tuổi, địa chỉ, màu mắt, …*
  - Các hành động: *đi, nói, thở, …*

**Một đối tượng là 1 thực thể bao gồm thuộc tính và hành động**

# 2. Một số khái niệm cơ bản trong OOP (tt)

## ❖ Lớp (class):

- Các đối tượng có các đặc tính tương tự nhau được gom chung thành lớp đối tượng.
- Một lớp đối tượng đặc trưng bằng các thuộc tính và các hành động (hành vi, thao tác) của các đối tượng thuộc lớp.
- Một đối tượng cụ thể thuộc một lớp được gọi là một thể hiện (instance) của lớp đó.

# 2. Một số khái niệm cơ bản trong OOP (tt)

- **Lớp (tt):**
  - Thuộc tính (Attribute): Một thành phần của đối tượng, có giá trị nhất định cho mỗi đối tượng tại mỗi thời điểm trong hệ thống.
  - Thao tác (Operation): Thể hiện hành vi của một đối tượng tác động qua lại với các đối tượng khác hoặc với chính nó.

# 2. Một số khái niệm cơ bản trong OOP (tt)

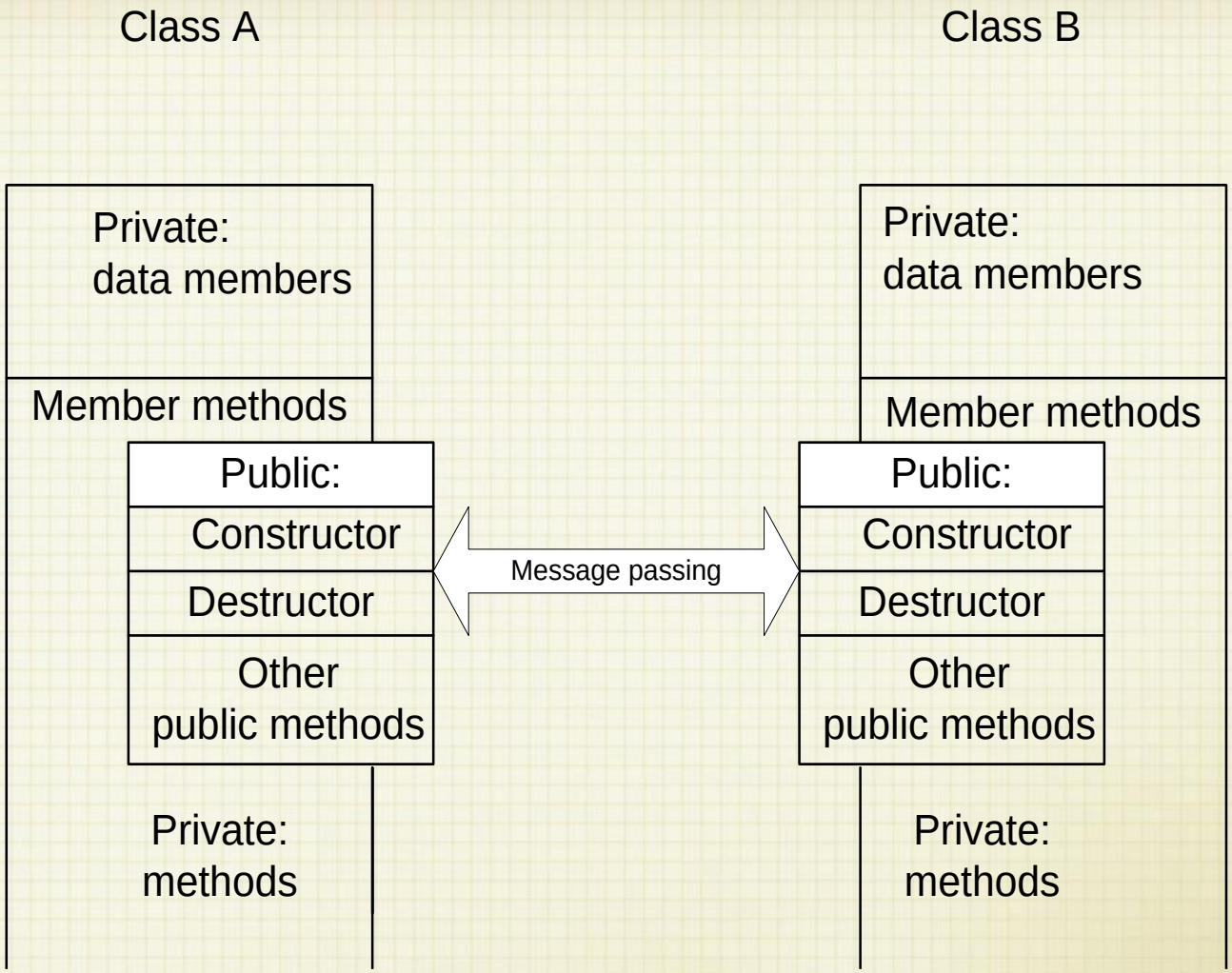
- Mỗi thao tác trên một lớp đối tượng cụ thể tương ứng với một cài đặt cụ thể.

  Một cài đặt như vậy được gọi là một phương thức (method).

- Cùng một phương thức có thể được áp dụng cho nhiều lớp đối tượng khác nhau.

  Một thao tác như vậy được gọi là có tính đa hình (polymorphism).

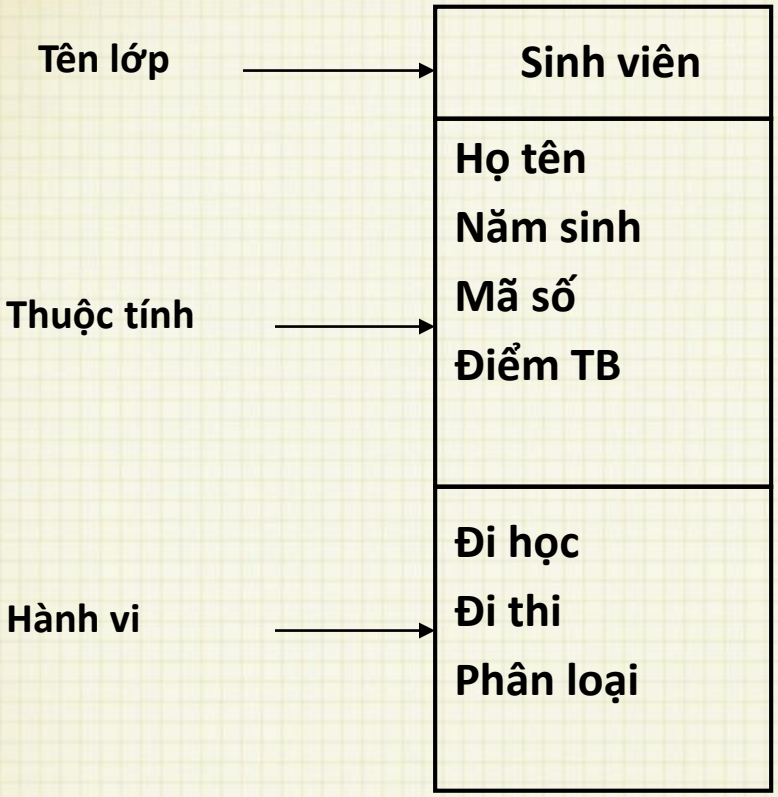
# Interacting Objects



# Sơ đồ đối tượng

- Dùng để mô tả các lớp đối tượng.
- Bao gồm (sơ đồ lớp + sơ đồ thể hiện).
- Một **Lớp đối tượng** được mô tả bằng một hình chữ nhật gồm 3 phần:
  - Phần đầu chỉ **Tên lớp;**
  - Phần 2 mô tả **Các thuộc tính;**
  - Phần 3 mô tả **Các thao tác** của các đối tượng thuộc lớp.

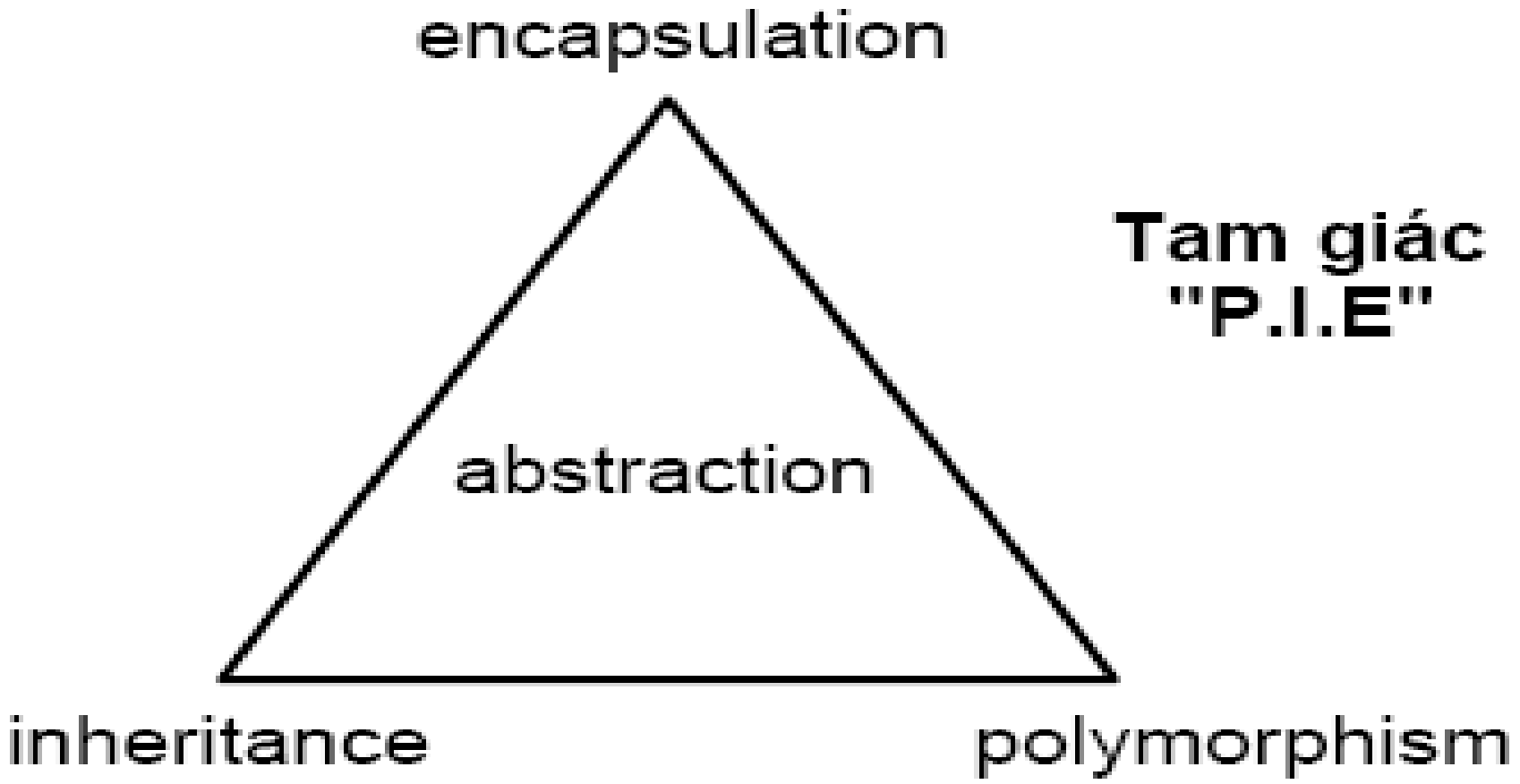
# Sơ đồ lớp và sơ đồ thể hiện




Sơ đồ lớp

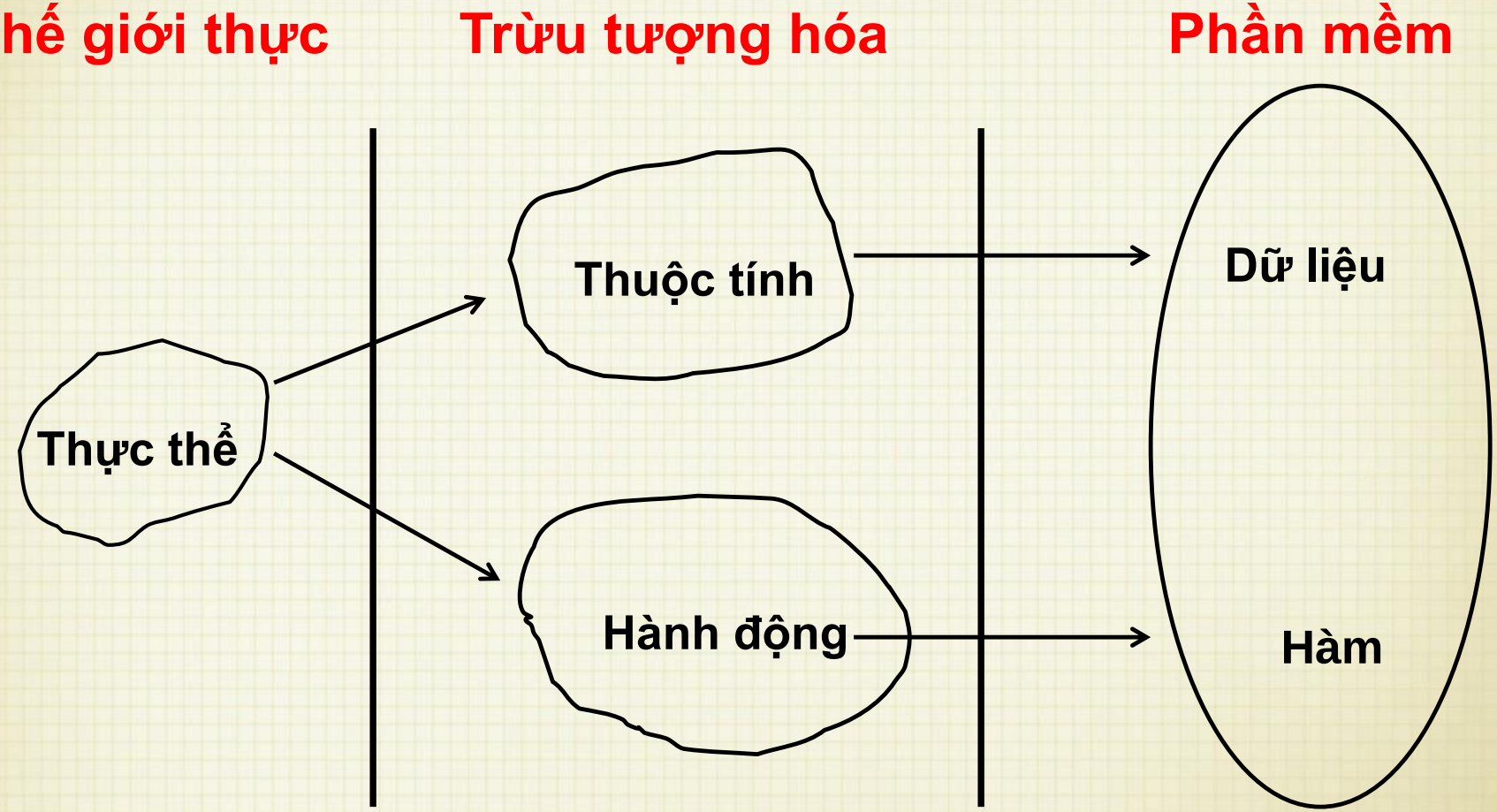

Sơ đồ thể hiện

# 3. Các đặc điểm quan trọng của OOP

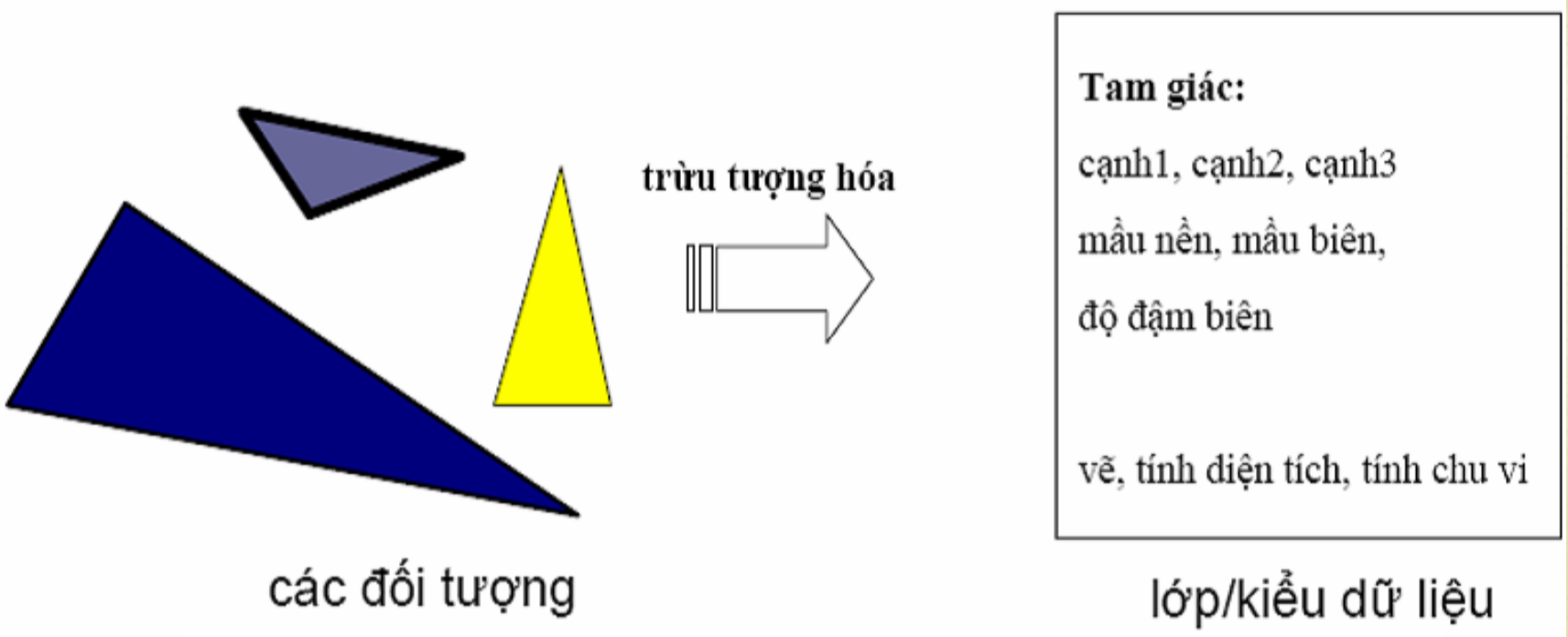



**Tam giác "P.I.E"**

# Trừu tượng hóa (Abstraction)



# Trừu tượng hóa (Abstraction)




Trừu tượng hóa là cách nhìn khái quát hóa về một tập các đối tượng có chung các đặc điểm được quan tâm (bỏ qua những chi tiết không cần thiết).

# Đóng gói (Encapsulation)

❖ **Đóng gói:**

Nhóm những gì có liên quan với nhau vào làm một để sau này có thể dùng một cái tên để gọi đến.

- Các hàm/ thủ tục đóng gói các câu lệnh.
- Các đối tượng đóng gói dữ liệu của chúng và các thủ tục có liên quan.

# Đóng gói (Encapsulation)

# Đóng gói (Encapsulation)

- **Che dấu thông tin:**

  Đóng gói để che một số thông tin và chi tiết cài đặt nội bộ để bên ngoài không nhìn thấy.

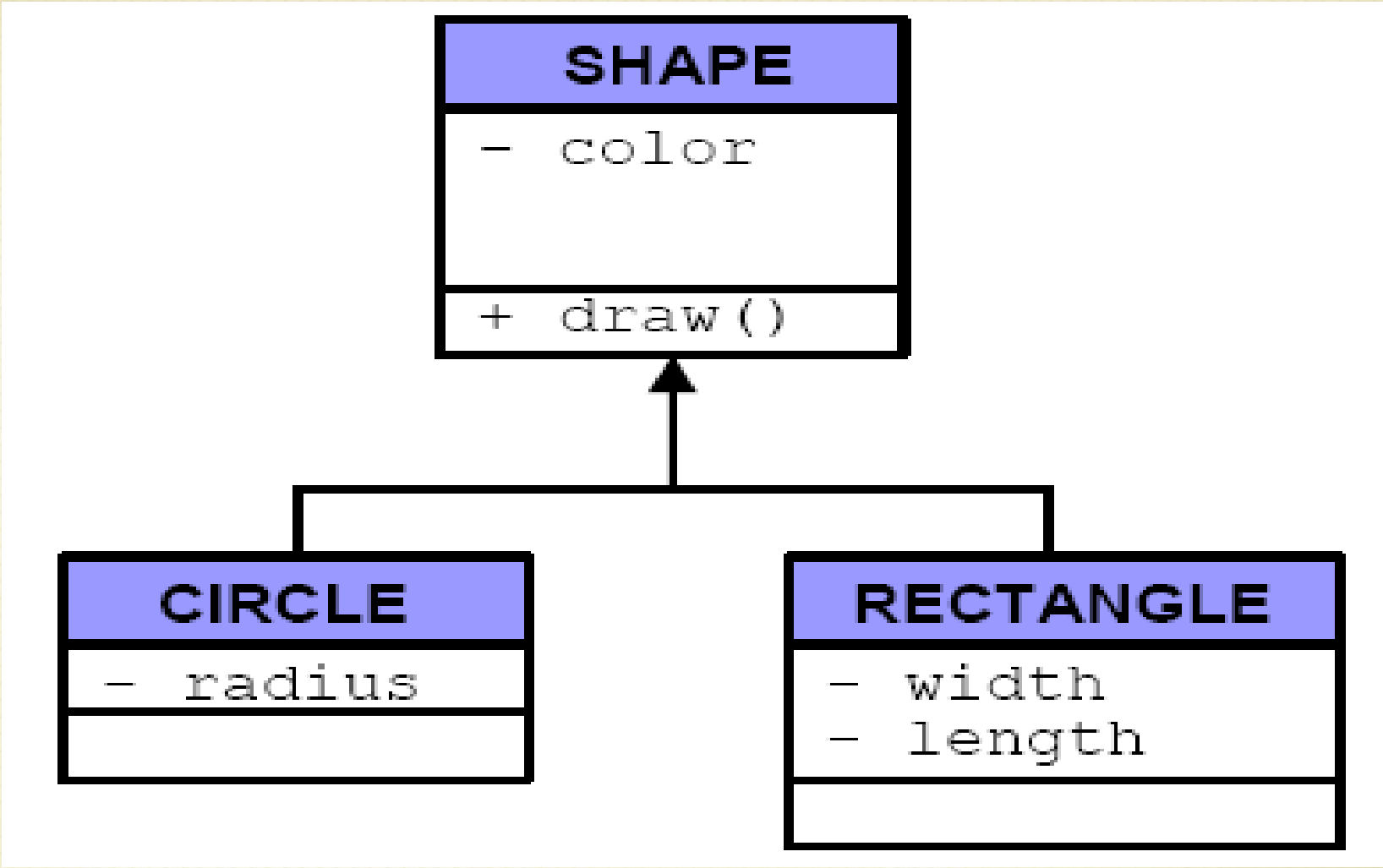
  - Che giấu những gì mà mình cần giữ bí mật.
  - Che giấu những gì mà người dùng không cần.

# Đóng gói (Encapsulation)

# Kế thừa (Inheritance)

- Là cơ chế cho phép một lớp D có được các thuộc tính và thao tác của lớp C (như thể các thuộc tính và thao tác đó đã được định nghĩa tại lớp D).
- Cho phép cài đặt các quan hệ sau giữa các đối tượng:
  - Đặc biệt hóa (“là”)
  - Khái quát hóa

# Kế thừa (Inheritance)



# Đa hình (Polymorphism)

Là cơ chế cho phép tên một thao tác hoặc thuộc tính có thể được định nghĩa tại nhiều lớp và có thể có những cài đặt khác nhau tại các lớp đó.

# Đa hình (Polymorphism)


# 4. Phân tích, thiết kế và lập trình hướng đối tượng

**Mục tiêu của việc thiết kế phần mềm:**

- Tính tái sử dụng (reusability):

  Thiết kế các thành phần có thể được sử dụng trong nhiều phần mềm khác nhau.

- Tính mở rộng (extensibility):

  Hỗ trợ các plug-in/add-n.

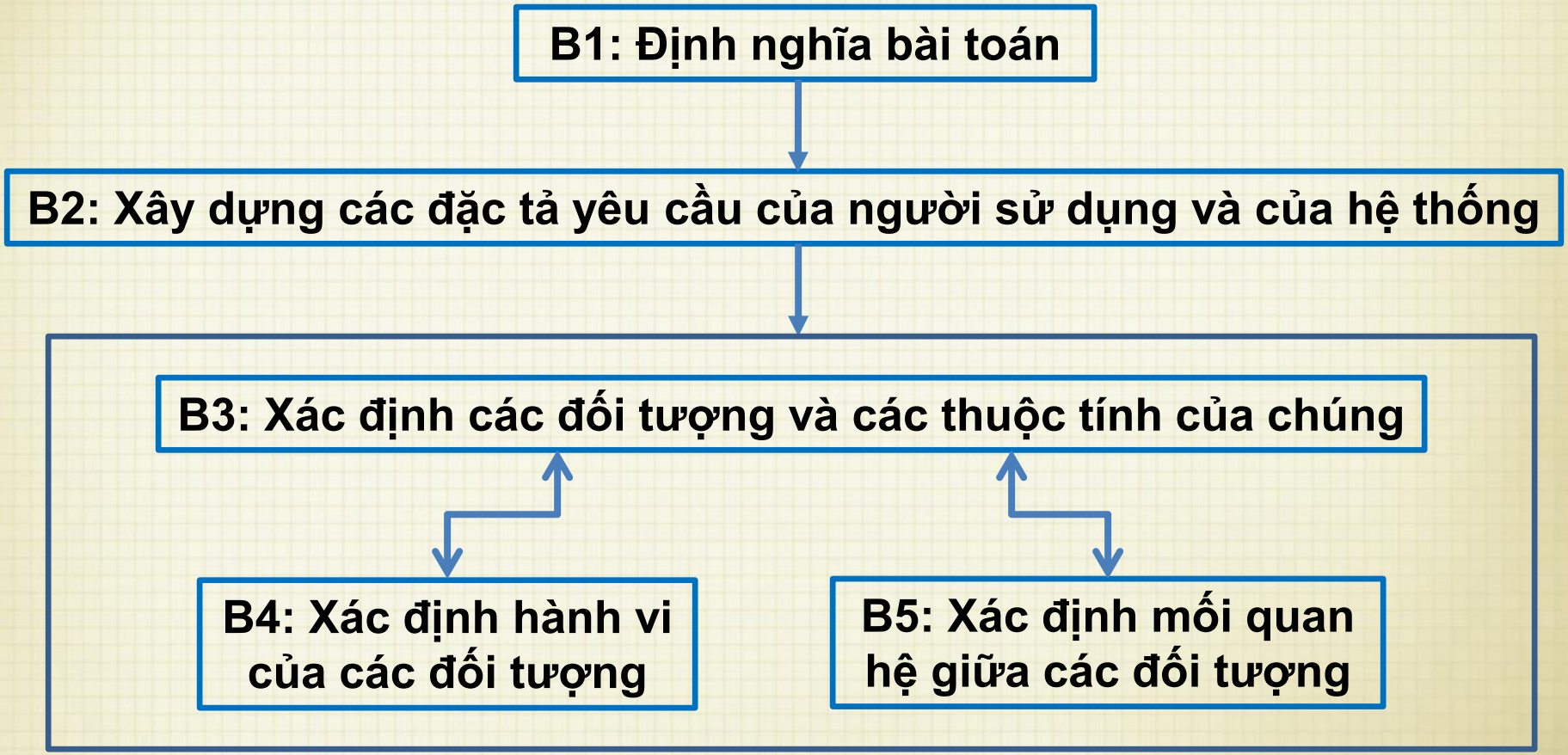
- Tính mềm dẻo (flexibility):
  - Dễ dàng thay đổi khi thêm mới dữ liệu hay tính năng.
  - Các thay đổi không làm ảnh hưởng nhiều đến toàn bộ hệ thống.

# 4. Phân tích, thiết kế và lập trình hướng đối tượng (tt)

- Phân tích hướng đối tượng
  (Object Oriented Analysis - OOA )
- Thiết kế hướng đối tượng
  (Object Oriented Design - OOD)
- Lập trình hướng đối tượng
  (Object Oriented Programming - OOP)

# 4.1 Phân tích hướng đối tượng

Phân tích hướng đối tượng bao gồm các bước sau:



# 4.1 Phân tích hướng đối tượng (tt)

**B1:** Tìm hiểu bài toán để xác định chính xác bài toán cần giải quyết.

+ Tìm hiểu hệ thống cũ;

+ Lấy yêu cầu từ nhà đầu tư và người sử dụng;

+ Làm rõ hơn các yêu cầu của bài toán và định nghĩa lại theo quan điểm của người phát triển.

**B2:** Dựa trên những yêu cầu đã xác định để đưa ra các đặc tả chi tiết, phục vụ cho việc xây dựng và kiểm tra hệ thống sau này.

# 4.1 Phân tích hướng đối tượng (tt)

**B3:** Các đối tượng sẽ được xác định thông qua các thực thể trong thế giới thực và được trừu tượng hóa thành các đối tượng trừu tượng.

Để xác định danh sách các đối tượng của bài toán có thể sử dụng:

**+ Sơ đồ dòng dữ liệu:** dữ liệu và kho dữ liệu trong sơ đồ dòng dữ liệu có thể được xem như là các đối tượng.

**+ Phân tích văn bản:** các đối tượng thường được mô tả bằng các danh từ.

# 4.1 Phân tích hướng đối tượng (tt)

**B4:** Để mô tả đầy đủ và chính xác các đối tượng cần xác định các hàm mô tả hành vi của chúng.

**+ Sơ đồ dòng dữ liệu:** các xử lý trong sơ đồ.

**+ Phân tích văn bản:** các động từ chỉ hành động và so sánh.

**B5:** Xác định mối quan hệ giữa các đối tượng dựa trên sự trao đổi thông tin giữa chúng.

Trong một hệ thống, mỗi thực thể phải có quan hệ ít nhất với một thực thể khác.

# 4.2 Thiết kế hướng đối tượng

**Sử dụng cách tiếp cận Bottom – Up**, bao gồm các bước:

**B1:** Xác định các đối tượng trong không gian lời giải dựa trên các đối tượng đã xác định được trong không gian bài toán.

**B2:** Xây dựng các đặc tả cho các đối tượng, các lớp và mối quan hệ giữa chúng.

**+ Quan hệ kế thừa:** một lớp có thể sử dụng lại một số thuộc tính, hàm của một hay nhiều lớp đã định nghĩa trước.

**+ Quan hệ thành phần:** đối tượng của lớp này cũng là phần tử của lớp khác.

**+ Quan hệ về sử dụng:** khả năng sử dụng của một lớp để đọc, xử lý các đối tượng của những lớp khác.

# 4.2 Thiết kế hướng đối tượng (tt)

**B3:** Xây dựng cấu trúc phân cấp các lớp.

- Theo nguyên lý tổng quát hóa.
- Dựa vào mối quan hệ giữa các lớp để xây dựng cấu trúc phân cấp trên nguyên tắc sử dụng lại tối đa các thuộc tính và hàm của những lớp đã được thiết kế trước.

# 4.2 Thiết kế hướng đối tượng (tt)

**B4:** Thiết kế các lớp: bổ sung thêm những thuộc tính và hàm cần thiết cho các lớp đối tượng.

- **Hàm quản lý lớp:** một đối tượng được tạo lập/hủy bỏ như thế nào?
- **Hàm thực hiện cài đặt lớp:** những phép toán nào được thực hiện trên dữ liệu kiểu lớp?
- **Hàm truy nhập vào lớp:** làm thế nào để nhận được thông tin về các biến nội bộ của một lớp?
- **Hàm xử lý lỗi:** làm thế nào xử lý được các lỗi xuất hiện khi thao tác với các đối tượng của lớp?

# 4.2 Thiết kế hướng đối tượng (tt)

**B5:** Thiết kế các hàm thành phần của lớp.

- Sử dụng kỹ thuật phân rã chức năng **Top – Down.**
- Sử dụng kỹ thuật thiết kế có cấu trúc để tạo ra cấu trúc phân cấp về chức năng cho những hàm phức tạp.
- Kết quả của thiết kế có cấu trúc cho một hàm là một cấu trúc có mội lối vào và một lối ra được tổ hợp từ ba cấu trúc cơ bản là tuần tự, tuyển chọn và vòng lặp.

# 4.2 Thiết kế hướng đối tượng (tt)

**B6:** Thiết kế chương trình chính với các nhiệm vụ:

- Nhập dữ liệu từ người sử dụng;
- Tạo ra các đối tượng theo định nghĩa các lớp;
- Tổ chức thực hiện trao đổi thông tin giữa các đối tượng;
- Lưu trữ kết quả xử lý hoặc hiện lên màn hình, máy in, thiết bị ngoại vi theo yêu cầu của người sử dụng.

# 4.2 Thiết kế hướng đối tượng (tt)

**Các nguyên tắc thiết kế hướng đối tượng:**

- **S**ingle Responsibility Principle
- **O**pen Closed Principle

  *"Open for extension but Closed for modification"*
- **L**iskov Substitution Principle

  *"Subclasses should be substitutable for their base classes"*

# 4.2 Thiết kế hướng đối tượng (tt)

- **I**nterface Segregation Principle

  *"Many client specific interfaces are better than one general purpose interface"*

- **D**ependency Inversion Principle

  *" Depend upon Abstractions. Do not depend upon concretions"*

=> **Gọi tắc là SOLID**

# 4.3 Lập trình hướng đối tượng

- Tổ chức chương trình thành các Lớp (Lớp bao gồm dữ liệu và các phương thức xử lý dữ liệu).
- Các cấu trúc dữ liệu được thiết kế sao cho đặc tả được các đối tượng.
- Dữ liệu được bao bọc, che giấu và không cho phép các hàm ngoại lai truy nhập tự do.

# 4.3 Lập trình hướng đối tượng (tt)

- Các đối tượng trao đổi với nhau thông qua các hàm.
- Dễ dàng bổ sung dữ liệu và các hàm mới vào đối tượng khi cần thiết.
- Chương trình được thiết kế theo cách tiếp cận **Bottom – Up**.

# Các ưu điểm của OOP

- Tính kế thừa

  => Loại bỏ những đoạn chương trình lặp lại và mở rộng khả năng sử dụng các lớp đã được xây dựng.
- Tính đóng gói, che dấu thông tin

  => Chương trình không bị thay đổi bởi những đoạn chương trình khác.
- Mô phỏng thế giới thực tốt hơn: ánh xạ các đối tượng của bài toán vào đối tượng của chương trình.
- Hệ thống hướng đối tượng dễ mở rộng, nâng cấp thành hệ thống lớn hơn.

# Bài tập

1) Viết chương trình nhập vào 2 phân số, tính tổng, hiệu, tích, thương và xuất kết quả ra màn hình.
2) Viết chương trình cho phép nhập một dãy phân số, tính tổng các phân số và tìm phân số lớn nhất.
3) Viết chương trình nhập vào 2 ma trận. Tính tổng, hiệu, tích hai ma trận đã nhập và in kết quả ra màn hình.
4) Viết chương trình cho phép người dùng:
   - Nhập vào thông tin của n học sinh gồm: mã học sinh (chuỗi), họ tên, giới tính, điểm toán, điểm lý, điểm hóa.
   - Tính điểm trung bình và xuất thông tin chi tiết của các học sinh ra màn hình.

# Q & A