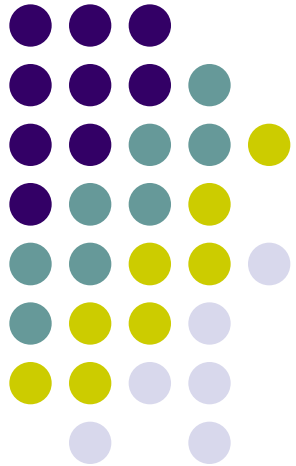
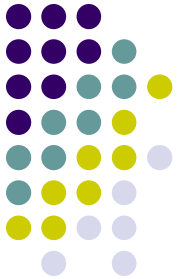
Chương 1

Tổng quan - Lập Trình Hướng Đối Tượng

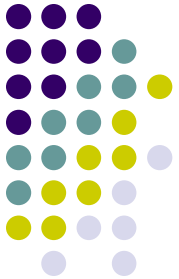


Nội dung



- Các phương pháp lập trình
- Các khái niệm cơ bản LTHĐT
- Ưu điểm của LTHĐT
- Các ngôn ngữ LTHĐT
- Ứng dụng LTHĐT

Các phương pháp lập trình

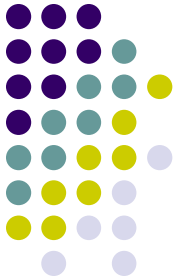


- Lập trình tuyến tính
- Lập trình thủ tục/cấu trúc
- Lập trình module
- Lập trình hướng đối tượng

Lập trình tuyến tính

- Giải quyết các bài toán nhỏ, đơn giản
- Chỉ gồm một chương trình main
- Chương trình gồm 1 chuỗi tuần tự các câu lệnh

Lập trình tuyến tính

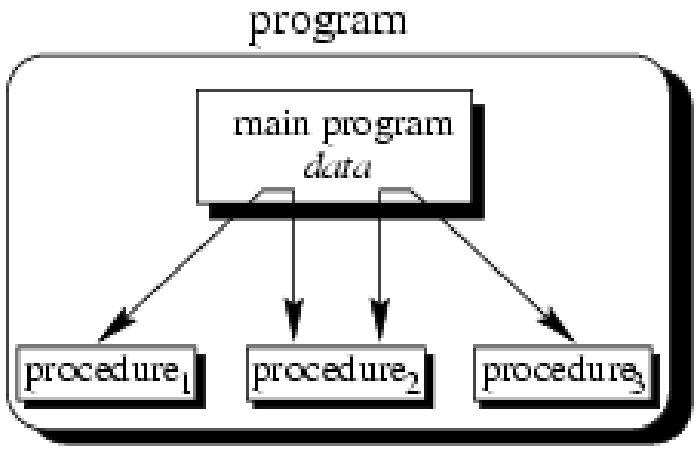


- Nhược điểm
  - Lặp lại những đoạn code giống nhau
  - Tất cả dữ liệu trong chương trình là toàn cục

    → không kiểm soát được phạm vi truy xuất dữ liệu

Lập trình thủ tục/cấu trúc

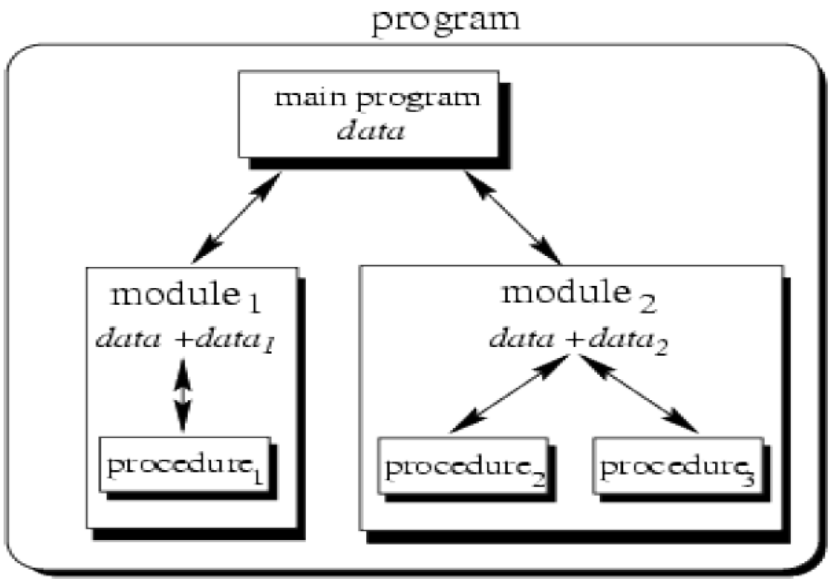
- Chương trình được tổ chức thành những chương trình con



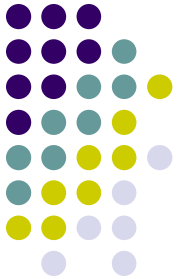
- Chương trình con:
  - Hàm, thủ tục
  - Độc lập nhau, thực hiện một tác vụ cụ thể

Lập trình module

- Các thủ tục/hàm được nhóm với nhau trong 1 module riêng biệt
- Chương trình gồm nhiều phần nhỏ
- Các phần tương tác thông qua việc gọi thủ tục



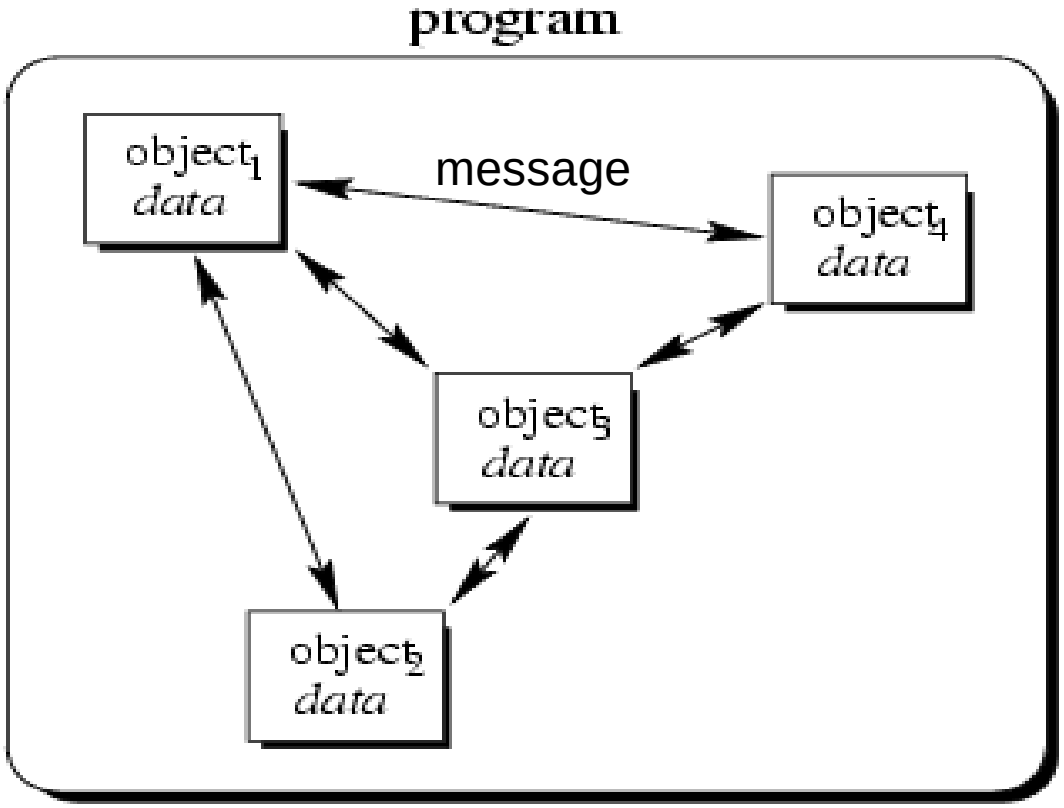
Lập trình thủ tục, module



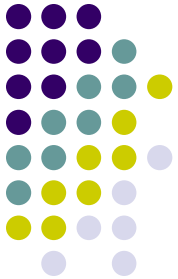
- Nhược điểm:
  - Thay đổi dữ liệu dùng chung → thay đổi tất cả hàm/thủ tục liên quan
  - Chương trình khó kiểm soát
  - Khó mở rộng

Lập trình hướng đối tượng

- Đặt trọng tâm vào đối tượng
- Dữ liệu gắn chặt với các hàm

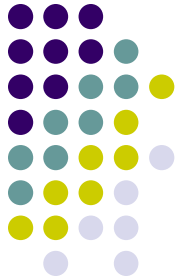


Lập trình hướng đối tượng



- Object-oriented programming(OOP)
- Đặc tính chủ yếu:
  - Chương trình được chia thành các đối tượng
  - Dữ liệu được đóng gói, che dấu, bảo mật
  - Các đối tượng tương tác với nhau qua message
  - Chương trình thiết kế theo hướng bottom-up

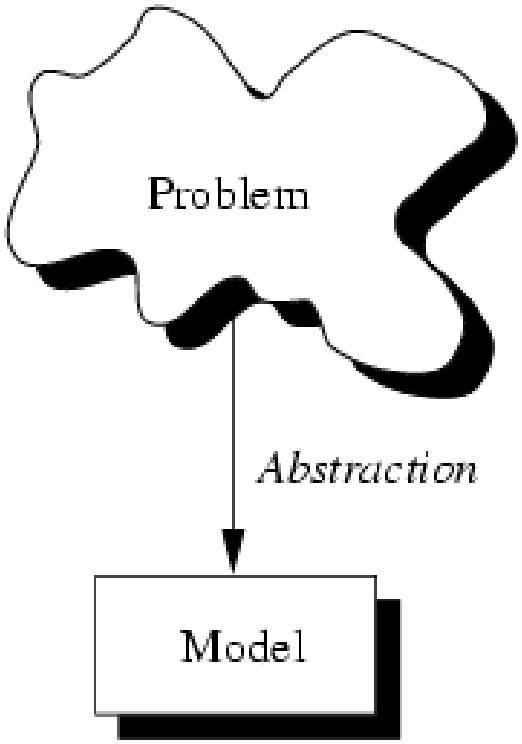
Một số khái niệm



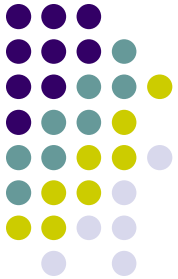
- Trừu tượng hóa
- Kiểu dữ liệu trừu tượng(ADT)
- Đối tượng(object)
- Thuộc tính & phương thức
- Lớp(class)
- Thông điệp(Message)
- Đóng gói, kế thừa, đa hình

Trừu tượng hóa

- Là sự đơn giản hóa, làm cho vấn đề dễ hiểu, sáng sủa hơn.
- Che dấu chi tiết, làm nổi bật cái tổng thể
- Các loại trừu tượng hóa:
  - Trừu tượng hóa dữ liệu
  - Trừu tượng hóa chương trình

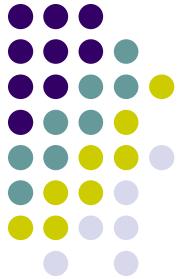


Kiểu dữ liệu trừu tượng (ADT)



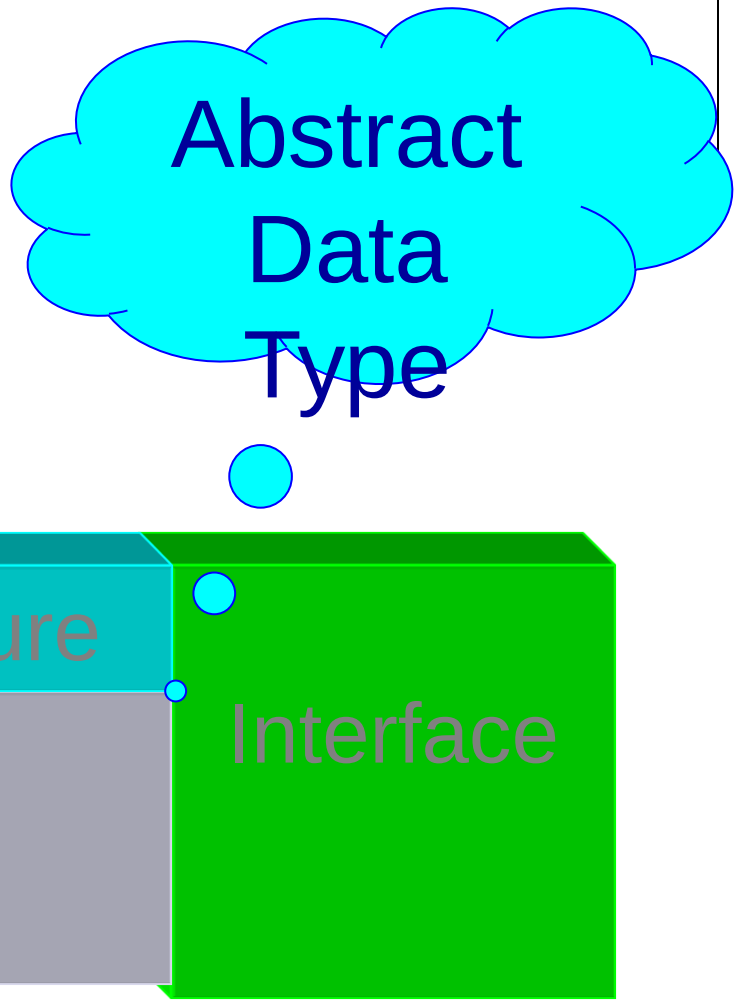
- Data type
  - Dữ liệu
  - Các phép toán trên dữ liệu
- Tách biệt đặc tả các tính chất logic với cài đặt bên trong

Kiểu dữ liệu trừu tượng (ADT)

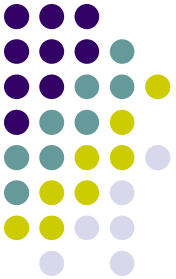


- Abstract Data type (ADT):
  - Kiểu dữ liệu mà biểu diễn bên trong của nó bị che dấu
  - Client không được phép chỉnh sửa dữ liệu trực tiếp mà qua tập các phép toán
  - Các phép toán chỉ được cho phép thông qua interface

ADT


Abstract
Data
Type
Abstract Data Structure
Operations
Interface

Ví dụ ADT



- **Stack ("last in first out" or LIFO).**
  - **Push: thêm phần tử vào stack**
  - **Pop: xóa phần tử khỏi stack**
  - **Initialize: khởi tạo stack**
  - **Empty: kiểm tra stack rỗng**

Ví dụ ADT

- **Stack Interface**

STACK.h

```
void STACKinit(void);
int  STACKisempty(void);
void STACKpush(int);
int  STACKpop(void);
```

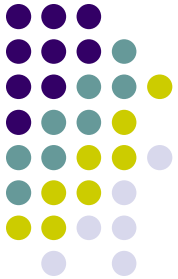
Stack Implementation

stackarray.c

```
#include "STACK.h"
#define MAX_SIZE 1000
static int s[MAX_SIZE];
static int N;

void STACKinit(void) {
  N = 0;
}

int STACKisempty(void) {
  return N == 0;
}

void STACKpush(int item) {
  s[N++] = item;
}

int STACKpop(void) {
  return s[--N];
}
```

big enough?

Stack Client



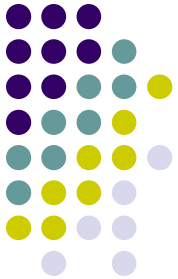
posfix.c

```
#include <stdio.h>
#include <ctype.h>
#include "STACK.h"

int main(void) {
  int c;
  STACKinit();
  while ((c = getchar()) != EOF) {
     if ('+' == c)
        STACKpush(STACKpop() + STACKpop());
     else if ('*' == c)
        STACKpush(STACKpop() * STACKpop());
     else if (isdigit(c))
        STACKpush(c - '0');
  }

  printf("top of stack = %d\n", STACKpop());
  return 0;
}
```

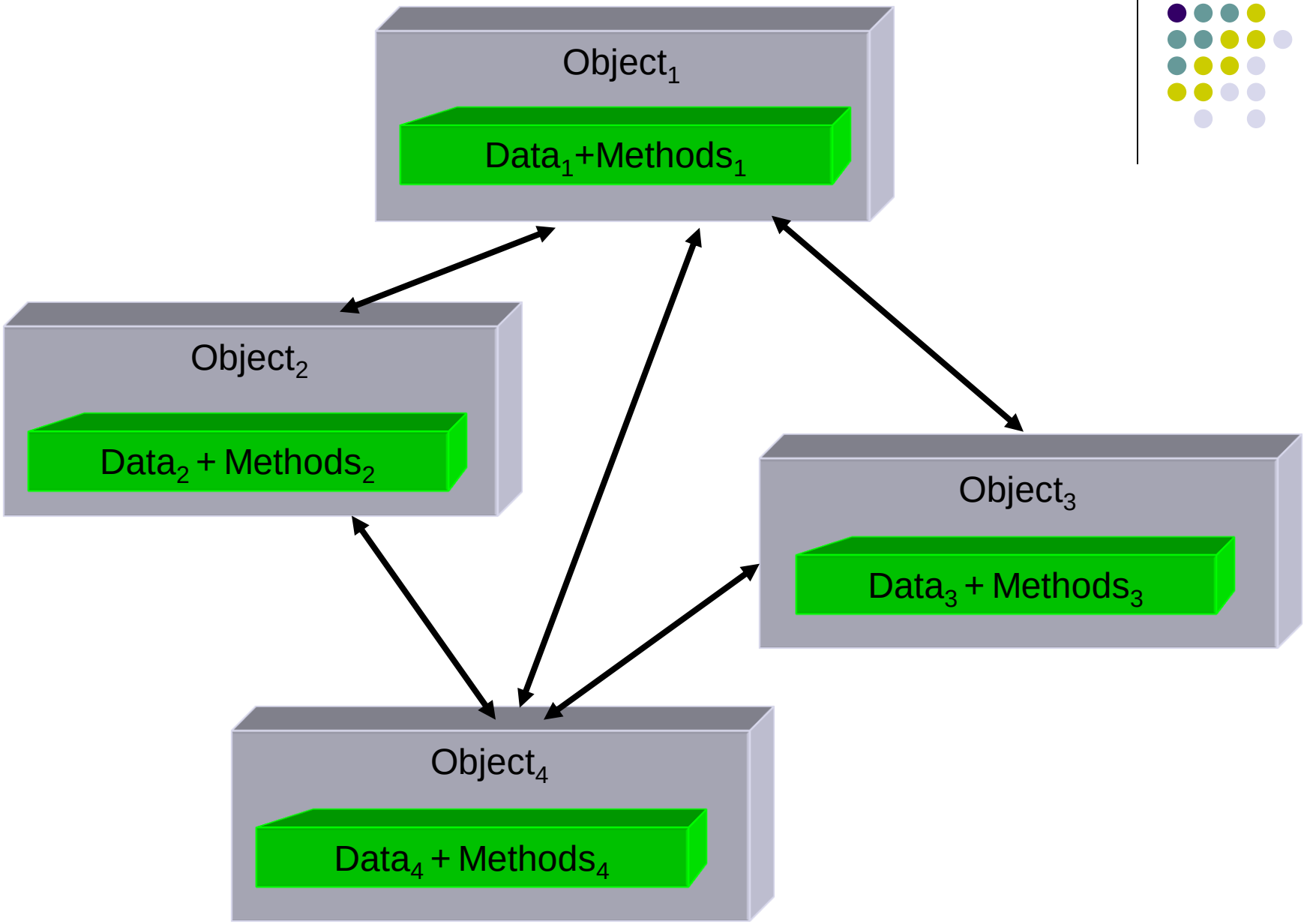
pop 2 elements and push sum

convert char to integer and push

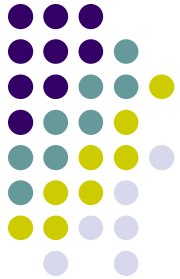
Lập trình hướng đối tượng



- Các đối tượng được dẫn xuất từ các ADT
- Chương trình hoạt động dựa trên cơ chế các đối tượng tương tác bằng cách gửi thông điệp cho nhau

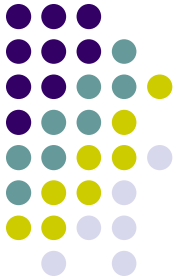

Object1
Data1+Methods1
Object2
Data2 + Methods2
Object3
Data3 + Methods3
Object4
Data4 + Methods4

Đối tượng (object)



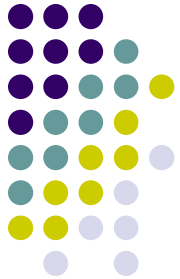
- Là khái niệm trừu tượng phản ánh các thực thể trong thế giới thực
- Object = data(attributes) + methods(behavior)
- Là thể hiện(instance) của 1 lớp
- Các đối tượng là thể hiện của cùng 1 lớp có dữ liệu ở trạng thái khác nhau

Đối tượng (object)



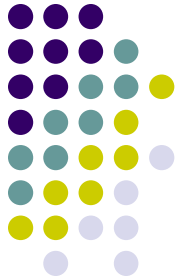
- **Ví dụ**: thiết kế và xây dựng chương trình game hockey
  - Object: Hockey player
  - Attributes: Position, height, weight, salary,…
  - Behaviors: shoot, punch another player,…

Đối tượng (object)



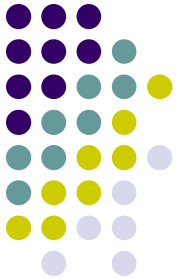
- **Ví dụ**: Viết một chương trình mô phỏng sự phát triển của quần thể virus trong cơ thể con người theo thời gian. Mỗi tế bào virus có thể sinh sản trong một khoảng thời gian. Bệnh nhân có thể trải qua điều trị bằng thuốc để ức chế quá trình sinh sản, và hủy diệt các tế bào virus từ cơ thể của họ. Tuy nhiên, một số tế bào có khả năng kháng thuốc và có thể tồn tại.
  - Objects: ??
  - Attributes: ??
  - Behaviors: ??

Đối tượng (object)



- **Ví dụ**: Viết một chương trình mô phỏng sự phát triển của quần thể virus trong cơ thể con người theo thời gian. Mỗi tế bào virus có thể sinh sản trong một khoảng thời gian. Bệnh nhân có thể trải qua điều trị bằng thuốc để ức chế quá trình sinh sản, và hủy diệt các tế bào virus từ cơ thể của họ. Tuy nhiên, một số tế bào có khả năng kháng thuốc và có thể tồn tại.
  - Objects: ??
  - Attributes: ??
  - Behaviors: ??

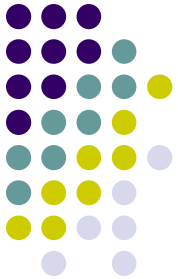
Đối tượng (object)



- Bệnh nhân
  - Attributes
    - Quần thể virus
    - Miễn dịch với virus(%)
  - Behaviors
    - Dùng thuốc

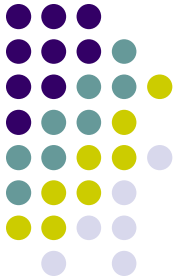
- Virus
  - Attributes
    - Tỉ lệ sinh sản(%)
    - Kháng thuốc(%)
  - Behaviors
    - Sinh sản
    - Tồn tại

Thuộc tính & phương thức



- **Thuộc tính**:
  - Đặc trưng, phân biệt đối tượng
  - Là dữ liệu được lưu trữ trong đối tượng
  - Nên được đặt ở mức cao nhất trong phân cấp kế thừa
  - Hằng, biến,…
  - Thuộc kiểu dữ liệu cơ bản hoặc kiểu do người dùng định nghĩa
- **Phương thức**:
  - Hàm nội tại được ứng dụng cho đối tượng
  - Thao tác trên dữ liệu được lưu trữ trong đối tượng
  - Operations, behaviors, member functions

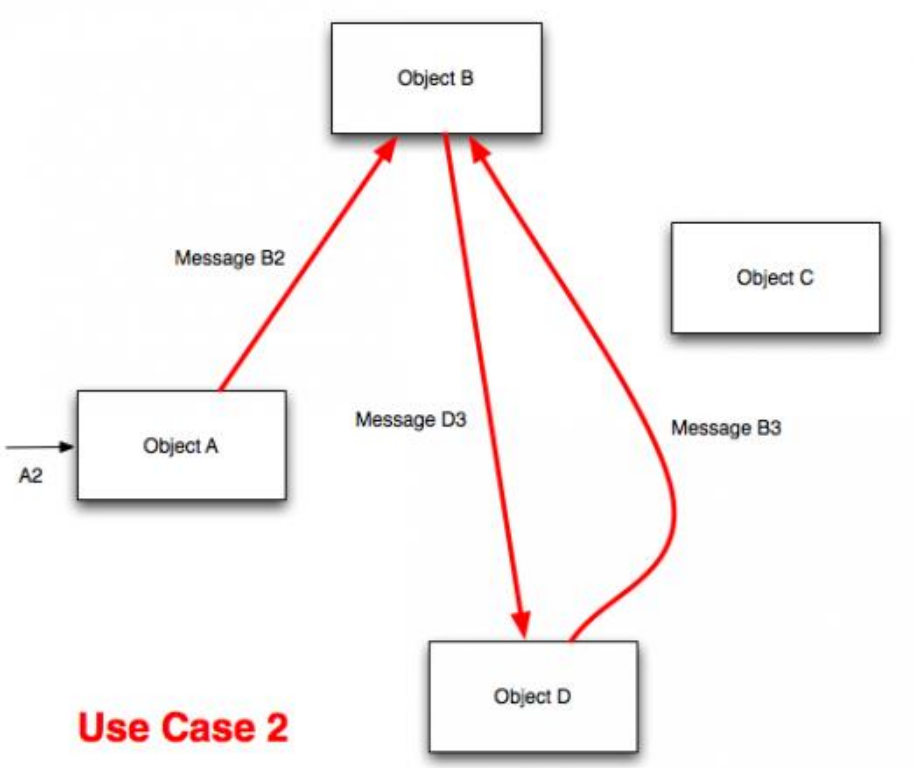
Lớp (class)



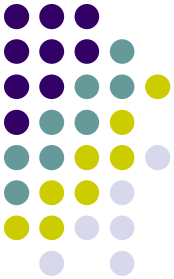
- Là một tập các objects cùng loại
- Là một đại diện của ADT
- Được sử dụng như kiểu dữ liệu do người dùng định nghĩa
- Là bản mẫu để tạo ra thể hiện của object
- Bộ nhớ được cấp phát cho object, không phải cho class

Thông điệp (message)

Các đối tượng tương tác, giao tiếp với nhau sử dụng thông điệp



Thông điệp (message)



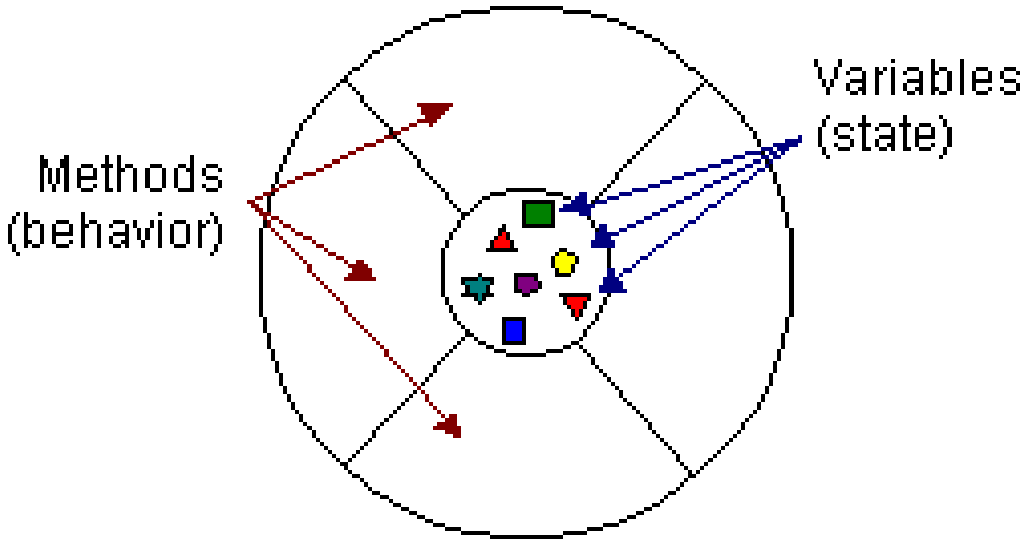
- Message bao gồm
  - Đối tượng đích
  - Tên của phương thức cần thực hiện
  - Các tham số cần thiết

```
class Integer {
attributes:
     int i
methods:
     setValue(int n)
   Integer addValue(Integer j)
}
```

```
Integer i;
i.setValue(1);
```

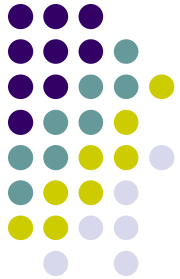
Tính đóng gói (Encapsulation)

- Là việc bao gói các thuộc tính của đối tượng bởi các phương thức



- Thực tế đôi khi cần phải phơi bày một vài thuộc tính và che dấu một vài phương thức

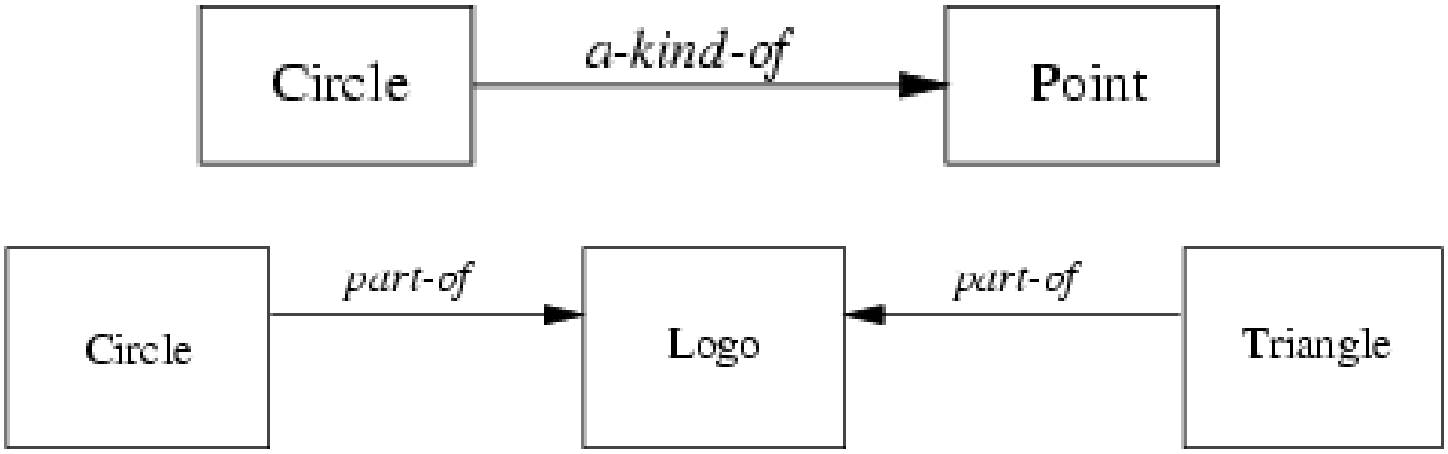
Tính đóng gói (Encapsulation)



- Che dấu thông tin: việc sử dụng đối tượng không cần thiết phải biết chi tiết về dữ liệu và toán tử bên trong. Ngăn chặn các thao tác không được phép từ bên ngoài
- Ưu điểm:
  - Bảo vệ dữ liệu
  - Kiểm soát được sự thay đổi

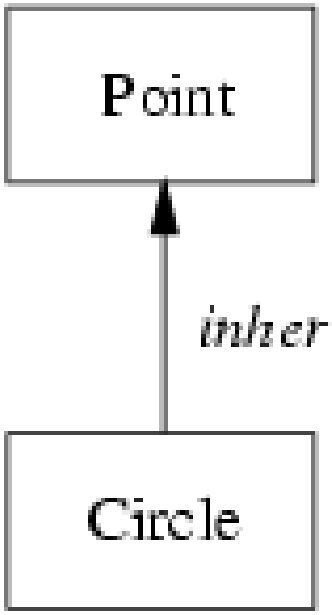
Các mối quan hệ của lớp

Bài toán: Viết chương trình vẽ các đối tượng points, circles, rectangles, triangles,…



Tính kế thừa (Inheritance)

- Lớp thừa kế các thuộc tính và phương thức từ lớp cha
- Cung cấp cơ chế rất mạnh trong việc tổ chức và cấu trúc chương trình

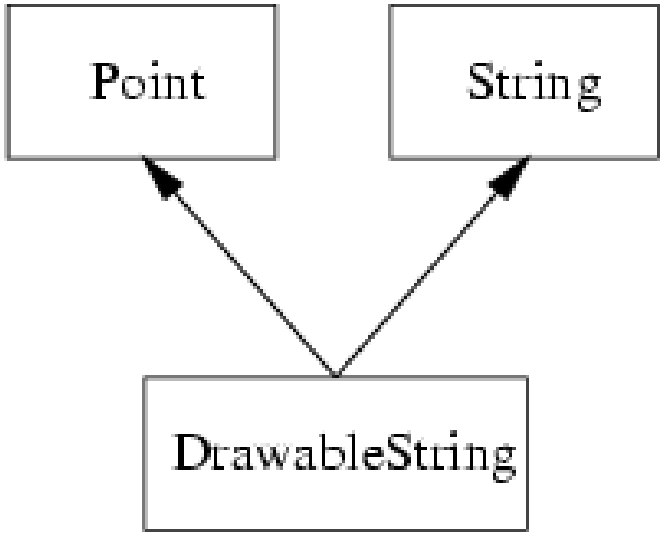


Base class
SuperClass
Parent Class

Derived class
SubClass
Child Class

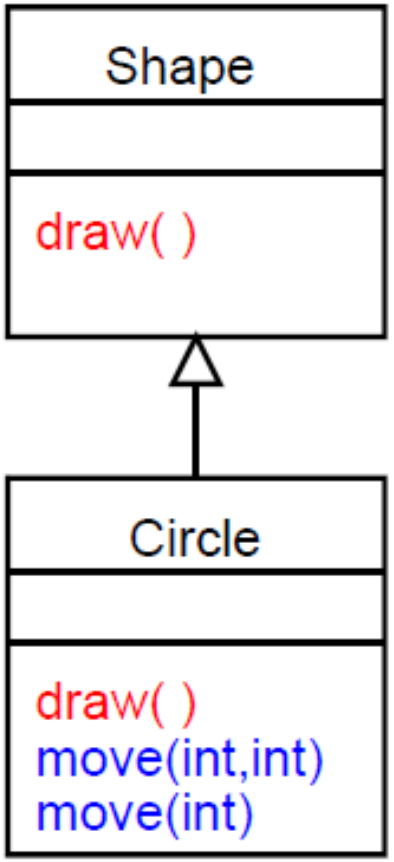
Tính kế thừa (Inheritance)

- Các loại kế thừa:
  - Đơn kế thừa
  - Đa kế thừa
- Lợi ích:
  - Tái sử dụng
  - Có thể cài đặt lớp cha là lớp trừu tượng

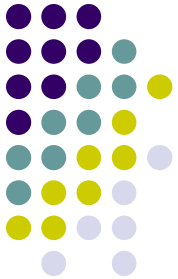


Lớp trừu tượng (Abstract Class)

- Định nghĩa:
  - Class A được gọi là lớp trừu tượng nếu nó chỉ được sử dụng như là SuperClass của lớp khác
  - Các thành phần trong class được chỉ định nhưng chưa được định nghĩa đầy đủ
  - Không được dùng để tạo ra đối tượng

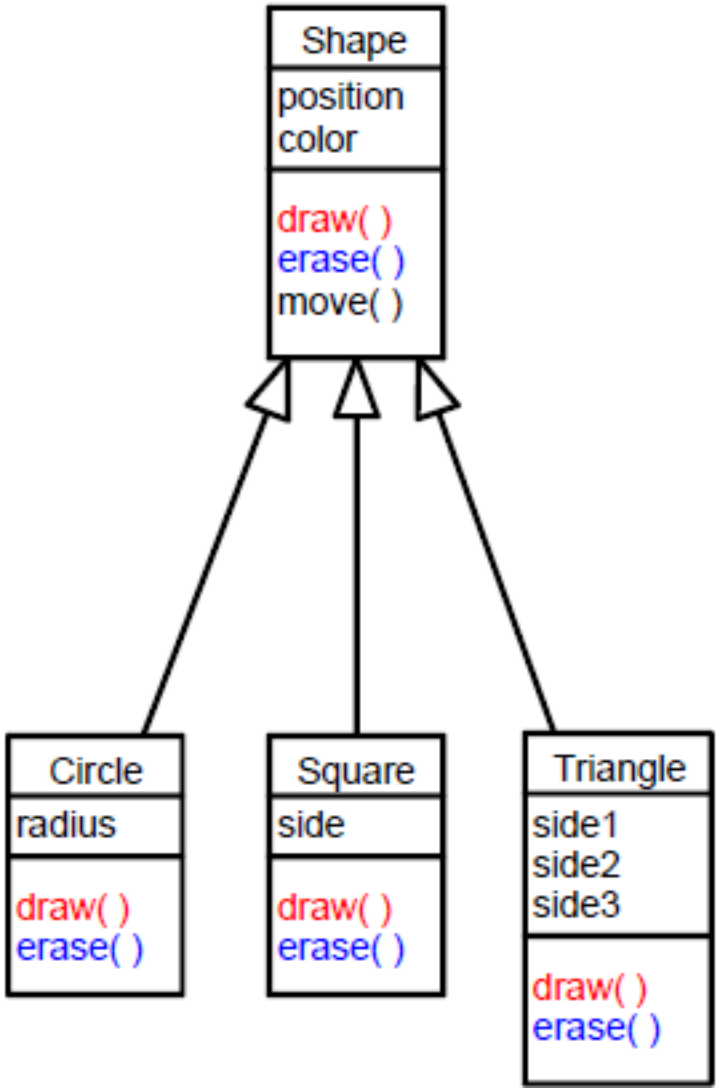


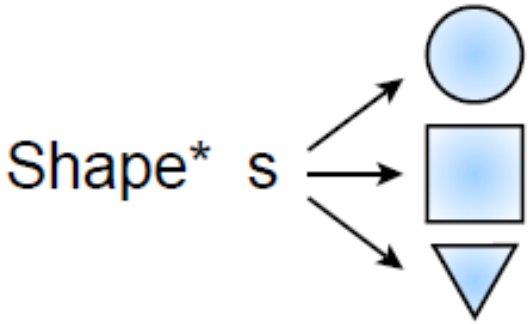
Tính đa hình (Polymorphism)



- "Nhiều hình thức"
- Các phương thức cùng tên thực hiện khác nhau với các đối tượng/lớp khác nhau
- Được thực hiện bởi:
  - Quá tải
  - Nạp chồng

Tính đa hình (Polymorphism)





s->draw( );

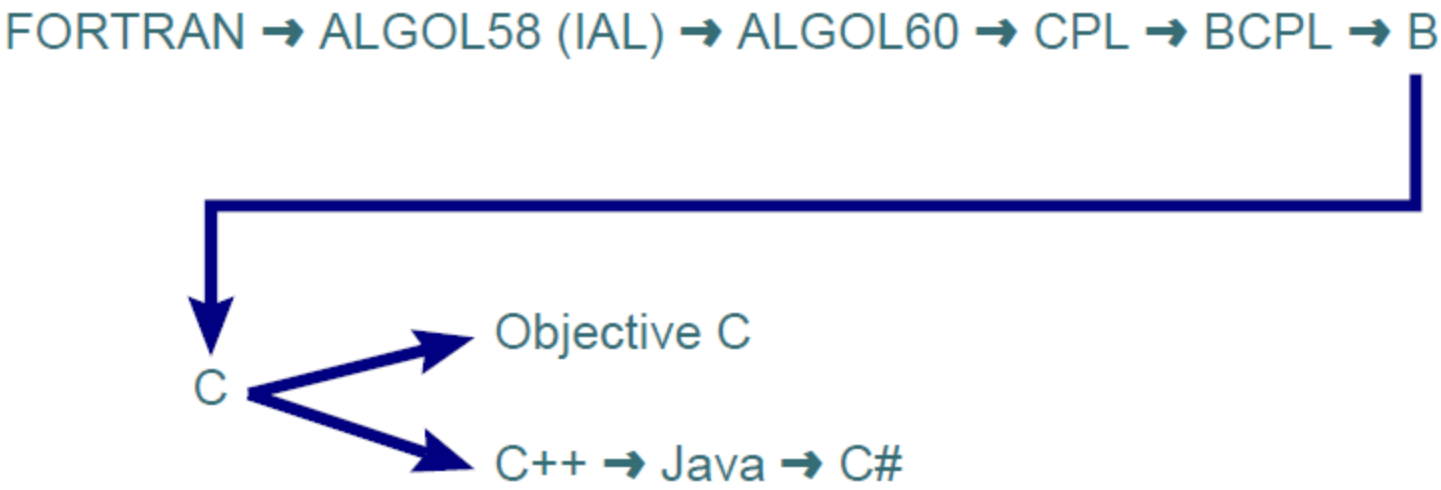
Phương thức draw() nào được gọi ?

Ưu điểm của OOP

- Cho phép phát triển các hệ thống lớn dễ dàng hơn
- Hợp nhất quy trình thiết kế và cài đặt mã
- Chia nhỏ code thành các phần có tính logic, loại bỏ các code lặp
- Hỗ trợ tái sử dụng
  - off-the-shelf code libraries
- Hỗ trợ sự phát triển code theo thời gian:
  - code bên trong của một lớp có thể được viết lại mà không ảnh hưởng đến toàn hệ thống nếu interfaces được đảm bảo.

Các ngôn ngữ OOP

- Ngôn ngữ lập trình



- Ngôn ngữ OOP thuần khiết: Smalltalk, C#, Java,…
- Ngôn ngữ lai hỗ trợ OOP: C++, Objective-C, Object-Pascal, Delphi,…