

Phân tích thiết kế hướng đối tượng

Bài 2: Khái niệm cơ bản về OOP

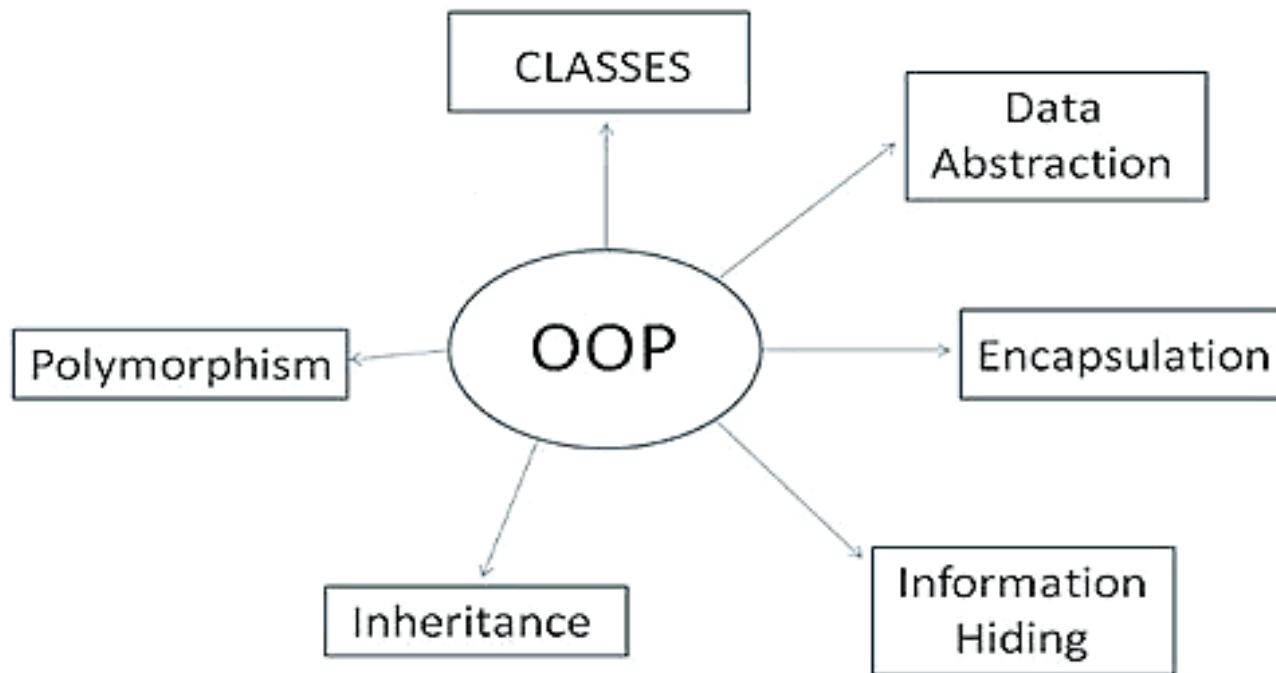
TS. Nguyễn Hiếu Cường

Bộ môn CNPM, Khoa CNTT, Trường ĐH GTVT

Email: cuonggt@gmail.com

Các khái niệm cơ bản trong OOP

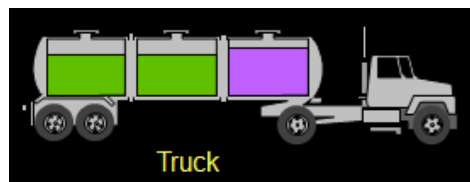
- Khái niệm trung tâm là **đối tượng** (object) và **lớp** (class)



Đối tượng

- Thế giới thực bao gồm các *đối tượng* (object)!

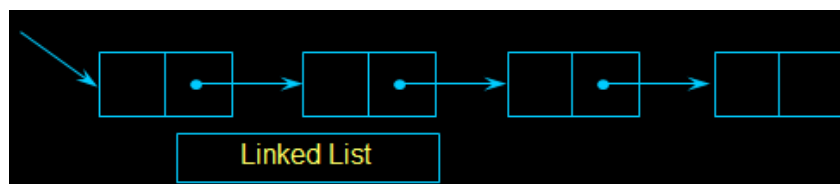
- Đối tượng vật lý



- Đối tượng khái niệm



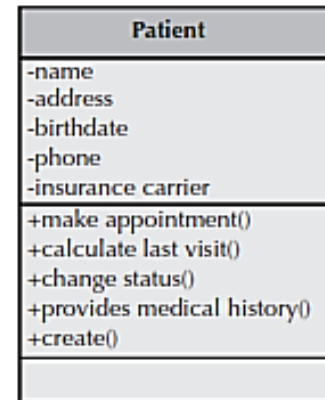
- Đối tượng phần mềm



- Mỗi đối tượng gồm các *thuộc tính* và các *thao tác*.

Lớp

- Lớp (Class)
 - Định nghĩa trừu tượng của các đối tượng có cùng những đặc tính chung
 - Đối tượng (object): thể hiện cụ thể (instance) của một lớp
- Tác dụng của lớp?
 - Trừu tượng hoá dữ liệu
 - Bao gói (dữ liệu + thao tác)
 - Che giấu thông tin
 - ...



Thông điệp và truyền thông điệp

- Thông điệp (message)

- Kích hoạt một hành động của của đối tượng

```
Patient aPatient;
```

```
aPatient.Insert();
```

← Thông điệp Insert()

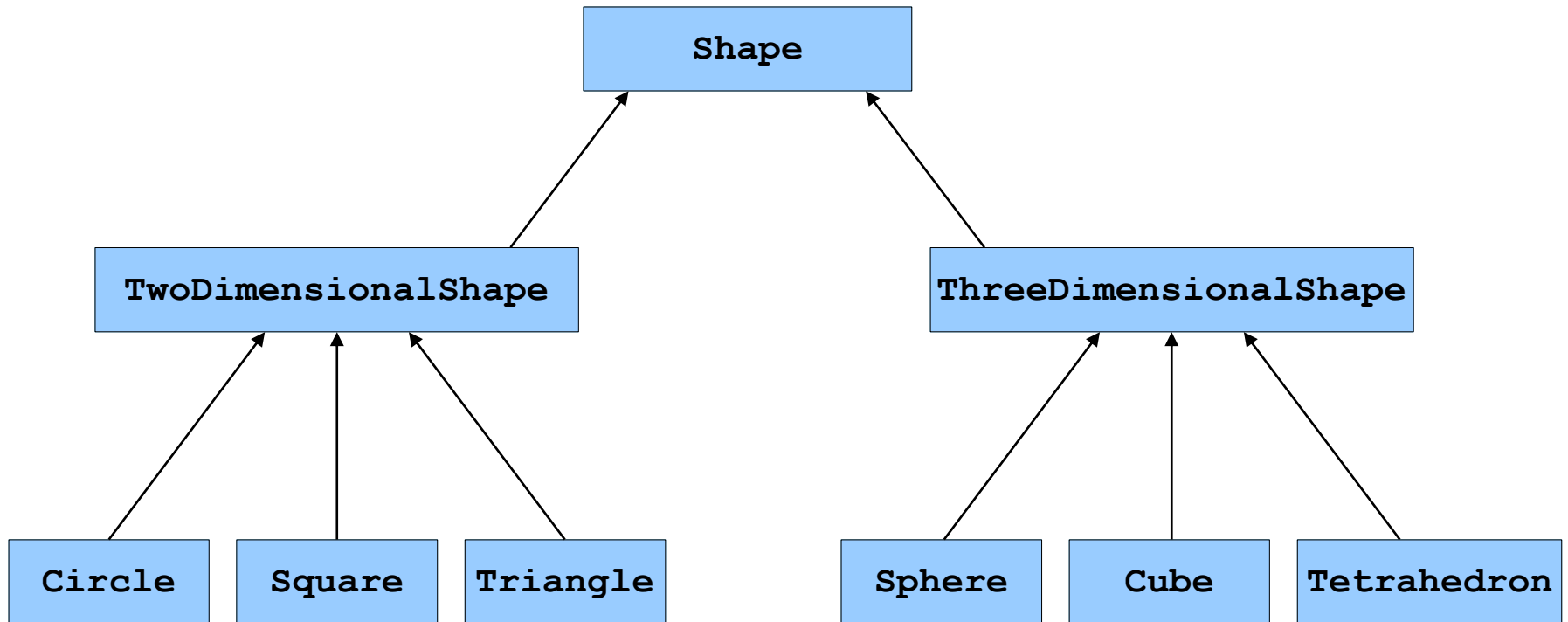
- Phương thức ≠ thông điệp?

- Phương thức (method) là hàm thành phần của lớp (member function)
 - Thông điệp là lời gọi hàm của một đối tượng của lớp đó

Kế thừa

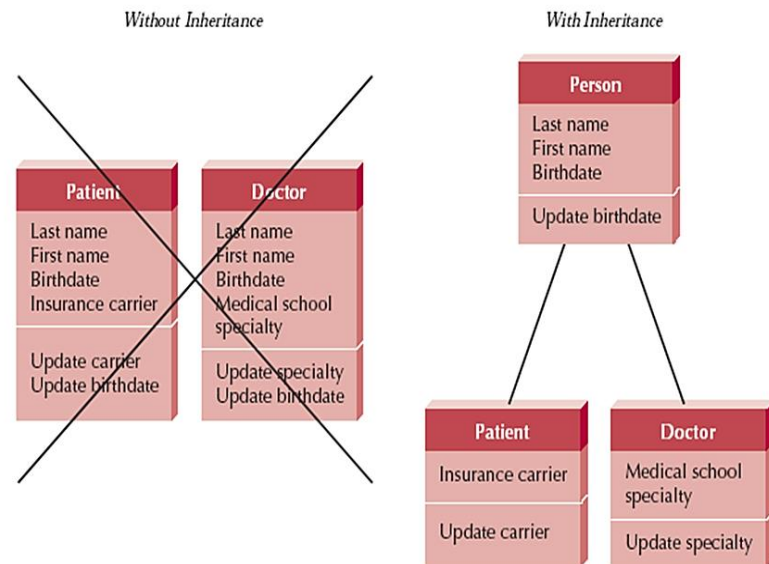
- Mục đích
 - Xây dựng một lớp mới bằng cách kế thừa các lớp đã có
- Một lớp có thể
 - Được thừa kế từ một hoặc nhiều lớp khác
 - Là cơ sở của một hoặc nhiều lớp khác
- Những gì được kế thừa?
 - Dữ liệu
 - Các phương thức

Ví dụ



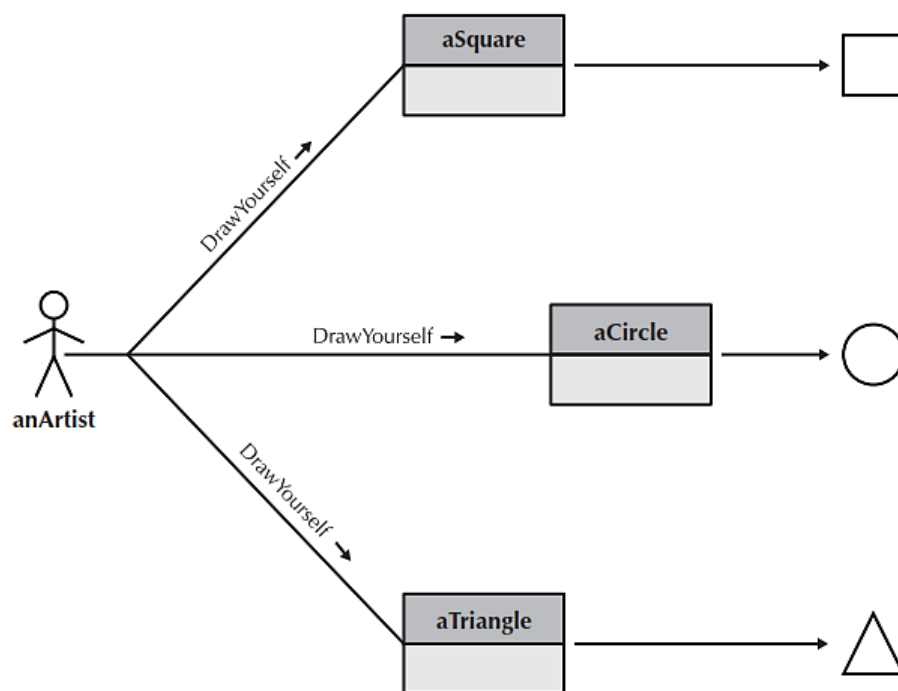
Tác dụng của kế thừa

- Sử dụng lại
- Phát triển chương trình
- Thiết kế hiệu quả hơn
 - ***So sánh 2 thiết kế bên***



Tương ứng bội

- Tương ứng bội hoặc tính đa hình (polymorphism)
 - Một thông điệp có thể được diễn giải theo các cách khác nhau
 - Có thể ẩn các chi tiết cài đặt dưới một giao diện chung



Ví dụ về hàm ảo

```
class Shape {
public:
    virtual void draw()
    {
        cout<<"draw shape \n";
    }
    void paint()
    {
        cout<<"paint shape \n";
    }
};

class Square: public Shape {
public:
    void draw(){cout<<"draw square \n";}
    void paint(){cout<<"paint square";}
};

class Circle: public Shape {
public:
    void draw() {cout<<"draw circle";}
    void paint() {cout<<"paint circle";}
};
```

```
void main()
{
    Shape *ptr;
    Shape p; //OK
    Circle c;
    Square s;
    ptr= &c;
    ptr->draw();    // draw circle
    ptr->paint();   // paint shape
    ptr= &s;
    ptr->draw();    // draw square
    ptr->paint();   // paint shape
}
```

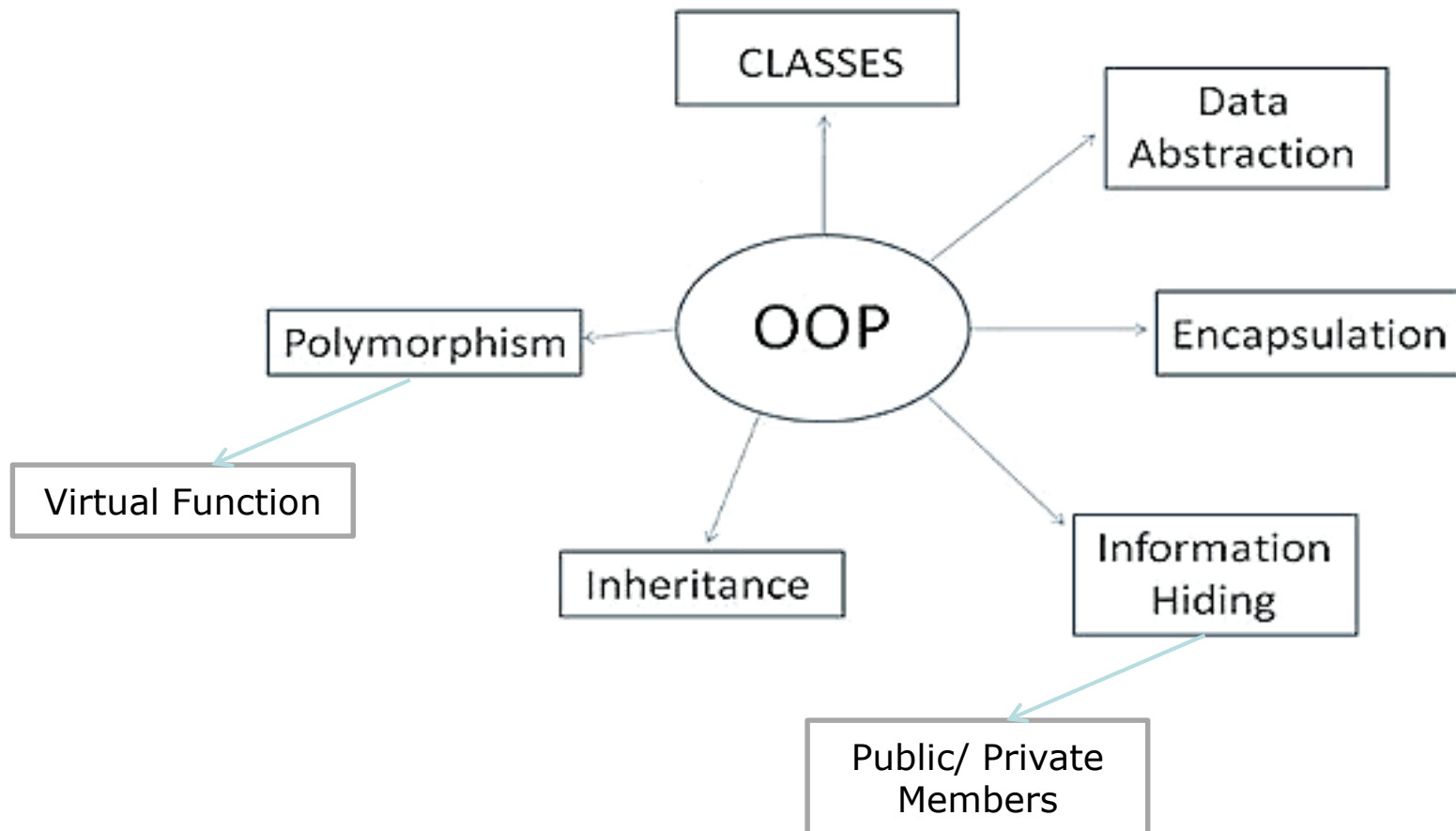
Có gì khác nhau
giữa **draw()** và
paint() ?

Ví dụ về hàm ảo

```
#include <iostream>
using namespace std;
class A {
public:
    A() { cout << "A constructor\n"; }
    void m1() { cout << "A.m1\n"; m2(); }
    virtual void m2() { cout << "A.m2\n"; }
};
class B : public A {
public:
    B() { cout << "B constructor\n"; }
    void m1() { cout << "B.m1\n"; }
    void m2() { cout << "B.m2\n"; }
};
void func(A &a) { a.m1(); }
int main() {
    B b;
    func(b);
}
```

A constructor
B constructor
A.m1
B.m2

Tóm tắt



Question

A model . . . ?

- A. Is not necessary when team members understand their job.
- B. Has to be structural AND behavioral.
- C. Is a simplification of reality.
- D. Is an excuse for building an elaborate plan.

Answer: C

Question

The best models are connected to . . . ?

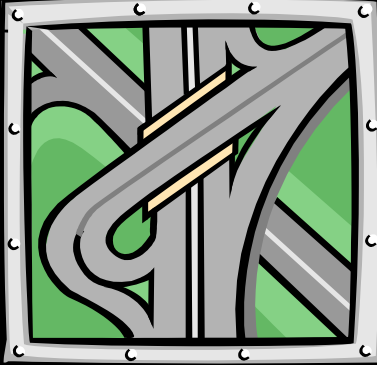
- A. Java-script code
- B. Reality
- C. C ++
- D. Issues that tie it to an object-oriented developer

Answer: B

Question

Which project would be least likely to require a model?

A.



B.



C.



D.



Answer: B

Question

If different properties and functions of a real world entity is grouped or embedded into a single element, what is it called in OOP language?

- A. Inheritance
- B. Polymorphism
- C. Abstraction
- D. Encapsulation

Answer: D

Question

A class . . . ?

- A. Is an encapsulation of an object.
- B. Represents the hierarchy of an object.
- C. Is an instance of an object.
- D. Is an abstract definition of an object.

Answer: D

Question

Object is _____ abstraction

- A. Object
- B. Logical
- C. Real
- D. Hypothetical

Answer: C

Question

The feature by which one object can interact with another object is:

- A. Data transfer
- B. Data Binding
- C. Message Passing
- D. Message reading

Answer: C

Question

Which feature of OOP indicates code reusability?

- A. Encapsulation
- B. Inheritance
- C. Abstraction
- D. Polymorphism

Answer: B

Question

Which among the following best describes the Inheritance?

- A. Copying the code already written
- B. Using the code already written once
- C. Using already defined functions in programming language
- D. Using the data and functions into derived segment

Answer: D

Question

Polymorphism can be described as?

- A. Hiding many different implementations behind one interface
- B. Inheritance
- C. Information placing
- D. Generalization

Answer: A

Question

If same message is passed to objects of several different classes and all of those can respond in a different way, what is this feature called?

- A. Inheritance
- B. Overloading
- C. Polymorphism
- D. Overriding

Answer: C

Question

Which type of function among the following shows polymorphism?

- A. Inline function
- B. Virtual function
- C. Undefined functions
- D. Class member functions

Answer: B