

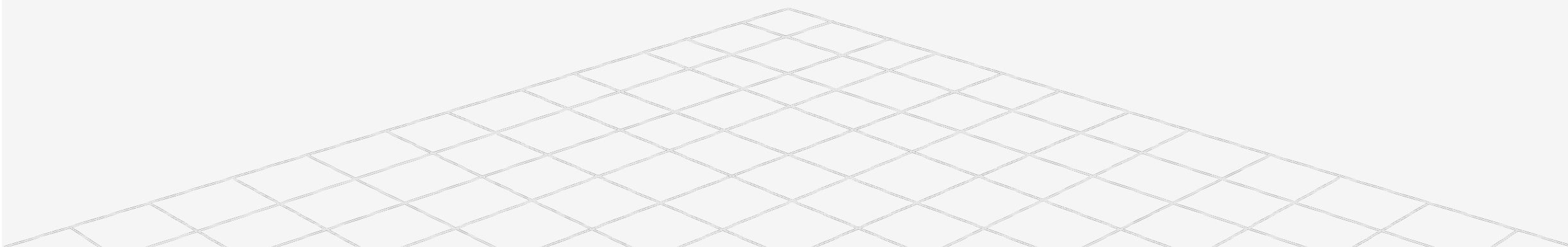
# **PHẦN 2: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG (OBJECT-ORIENTED PROGRAMMING)**

## **KHAI BÁO VÀ SỬ DỤNG LỚP**

# NỘI DUNG CHÍNH

---

- Lịch sử các phương pháp lập trình
- Một số khái niệm trong OOP
- Khai báo và sử dụng lớp



# LỊCH SỬ CÁC PHƯƠNG PHÁP LẬP TRÌNH

---

## ▪ Lập trình tuyến tính:

- Máy tính đầu tiên được lập trình bằng mã nhị phân
- Các ngôn ngữ lập trình bậc cao lần đầu tiên được đưa vào sử dụng
- Các ngôn ngữ lập trình bậc cao được thiết kế để giải các bài toán đơn giản, thường là chỉ mất chục dòng lệnh
- Nhược điểm:
  - Không sử dụng lại được mã các đoạn chương trình đã viết
  - Không có khả năng kiểm soát phạm vi nhìn thấy của dữ liệu
  - Mọi dữ liệu trong chương trình đều là dữ liệu toàn cục

# LỊCH SỬ CÁC PHƯƠNG PHÁP LẬP TRÌNH

---


## ▪ **Lập trình cấu trúc:**

- Cuối những năm 60 – 70, các ngôn ngữ lập trình cấu trúc ra đời.
- Các chương trình được chia nhỏ thành các chương trình con, các chương trình con lại được chia thành các chương trình con nhỏ hơn cho đến khi đơn vị chương trình có thể dễ dàng thực hiện được
- Các chương trình con càng độc lập với nhau càng tốt.
- Mỗi chương trình con có dữ liệu và logic riêng
- Trừu tượng hóa theo chức năng

# LỊCH SỬ CÁC PHƯƠNG PHÁP LẬP TRÌNH

---

## ▪ Lập trình hướng đối tượng:

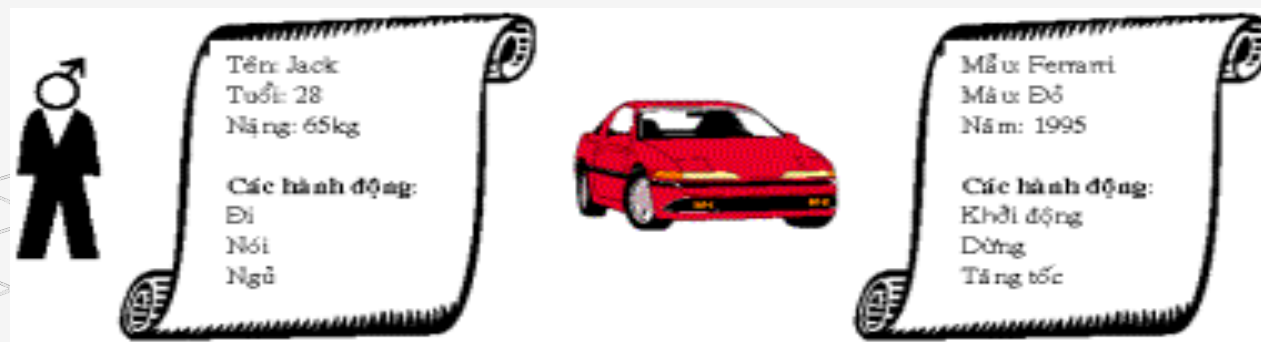
- Dựa trên nền tảng là lập trình cấu trúc và khái niệm trừu tượng hóa dữ liệu.
  - Chương trình hướng đối tượng được thiết kế xoay quanh dữ liệu mà chúng ta có thể làm việc trên đó hơn là bản thân chức năng của chương trình.
  - Liên kết dữ liệu với các thao tác, theo cách suy nghĩ tự nhiên về thế giới xung quanh
  - Mã và dữ liệu được trộn lẫn thành các đối tượng không thể tách rời được.
- 

# MỘT SỐ KHÁI NIỆM TRONG OOP

---

## ▪ Đối tượng (Object)

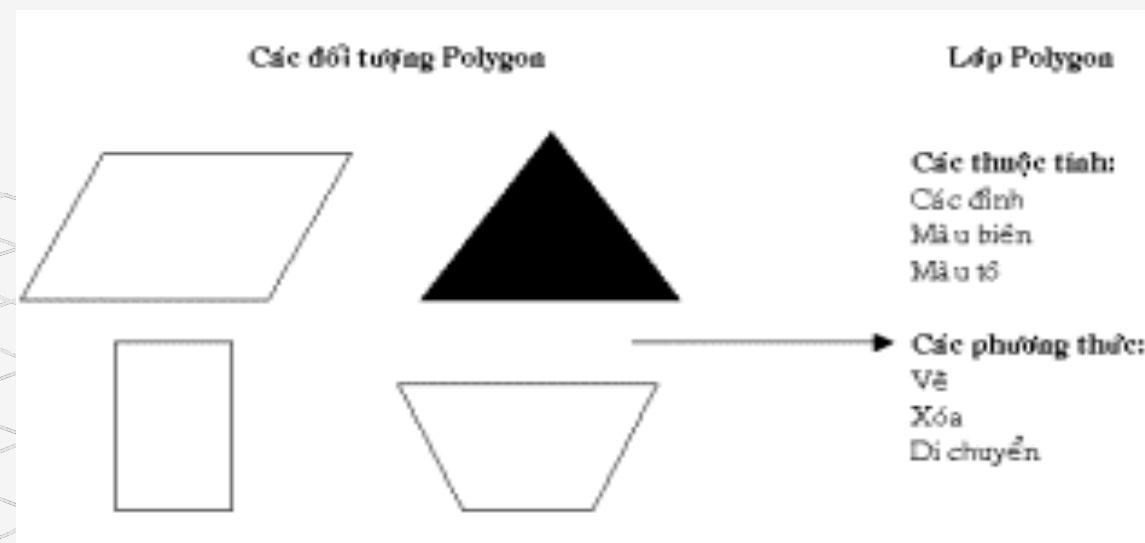
- Là một thực thể trong thế giới thực, ví dụ: một con người cụ thể, một cái ô tô cụ thể,,....
- Mỗi đối tượng có các đặc trưng và thuộc tính mô tả nó là cái gì hoặc nó làm gì.



# MỘT SỐ KHÁI NIỆM TRONG OOP

## ▪ Lớp (Class)

- Các đối tượng có thuộc tính và đặc trưng giống nhau được khái quát hóa thành một lớp.
- Mỗi đối tượng là một thực thể của lớp.
- Ví dụ:

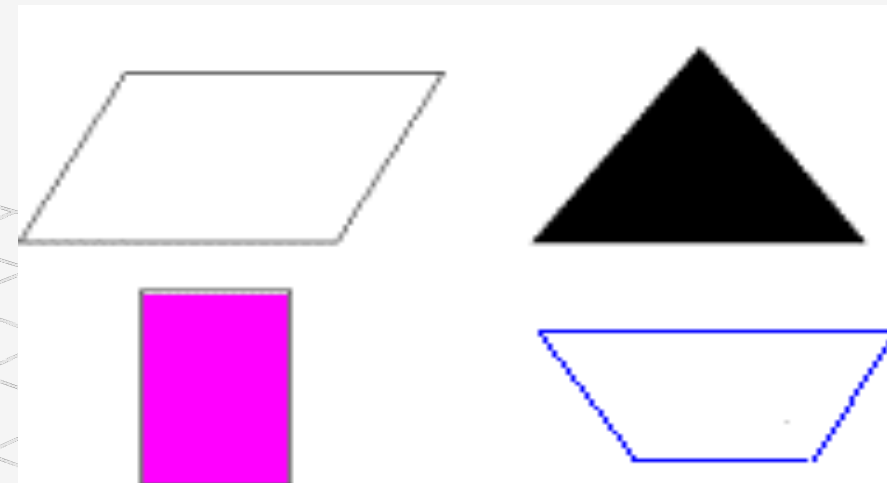


# MỘT SỐ KHÁI NIỆM TRONG OOP

---

## ▪ Thuộc tính (Attribute)

- Mỗi thuộc tính là một đặc trưng chung của nhóm các đối tượng được trừu tượng hóa thành lớp.
- Đây chính là các thành phần dữ liệu của lớp.
- Mỗi đối tượng có thể có các giá trị riêng trên từng thuộc tính nhưng phải cùng nhau tên và kiểu dữ liệu của thuộc tính.



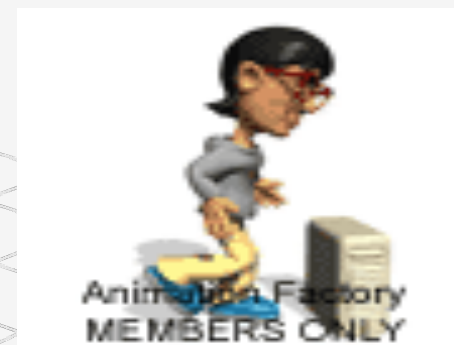


# MỘT SỐ KHÁI NIỆM TRONG OOP

---

## ▪ Phương thức (Method)

- Mỗi hành động của các đối tượng khi được trừu tượng hóa lên gọi là một phương thức trong lớp.
- Đây cũng chính là các hàm thành phần trong lớp.



# MỘT SỐ KHÁI NIỆM TRONG OOP

---

## ▪ Tính đóng kín (Encapsulation)

- Dữ liệu và phương thức của một lớp được gắn liền với lớp đó. Chỉ có các hàm thành phần trong lớp đó mới có thể truy xuất được. Hoặc các hàm ngoài có quan hệ đặc biệt với lớp.
- Đóng kín sẽ thực hiện tốt cơ chế che dấu thông tin.



# MỘT SỐ KHÁI NIỆM TRONG OOP

---

## ▪ Kế thừa (Inheritance)

- Tính kế thừa cho phép sử dụng một lớp đã có để xây dựng lớp mới.
- Lớp cơ sở là lớp được dùng để kế thừa cho các lớp khác
- Lớp kế thừa lớp cơ sở gọi là lớp dẫn suất
- Mỗi lớp dẫn suất dùng chung các thuộc tính và các hàm thành phần với lớp cơ sở. Nó cũng có thể có các hàm thành phần và thuộc tính riêng.

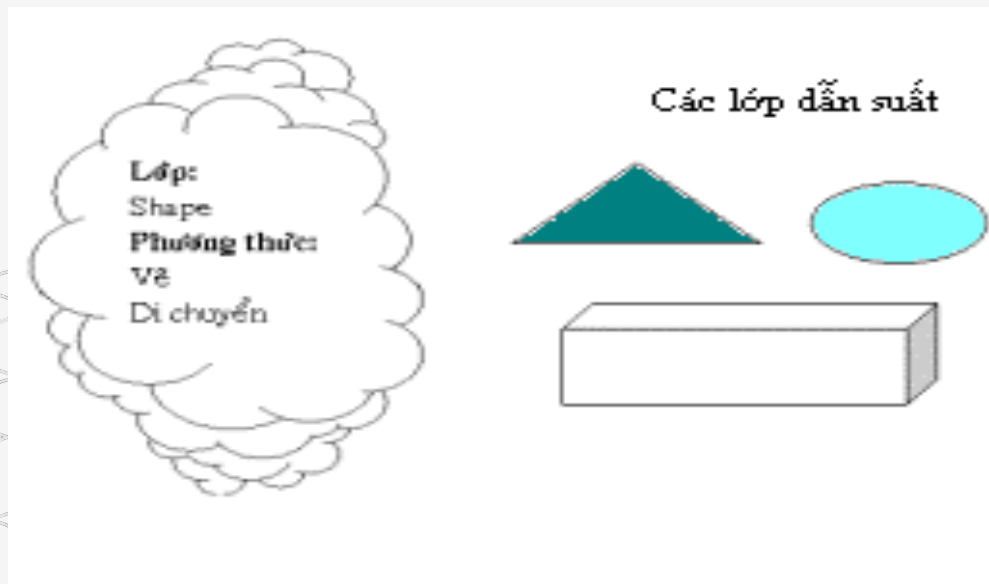


# MỘT SỐ KHÁI NIỆM TRONG OOP

---


## ▪ Đa hình (Polymorphism)

- Tính đa hình cho phép các một một hàm thành phần được kế thừa từ lớp cơ sở nhưng lại được thể hiện khác nhau trên các lớp dẫn suất khác nhau.
- Ví dụ:



# CÁC ƯU ĐIỂM CỦA LẬP TRÌNH OOP

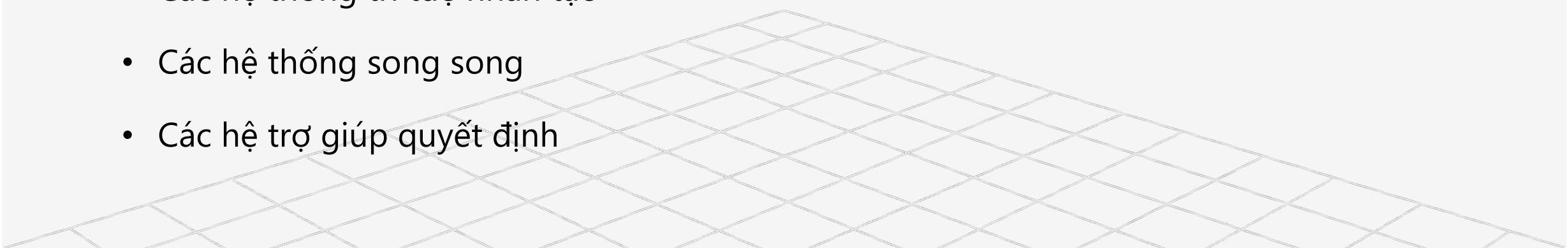
---

- Dựa vào nguyên lý đóng kín của OOP giúp cho người lập trình tạo ra được các chương trình có mức độ an toàn về dữ liệu cao
  - Mô tả một cách gần gũi các mô hình trong thế giới thực
  - Dễ dàng cài đặt các chương trình lớn vì thiết kế xoay quanh dữ liệu và hành vi của đối tượng.
  - Các hệ thống OOP dễ dàng mở rộng.
  - Dựa vào nguyên tắc kế thừa có thể hạn chế được các đoạn trình lặp lại trong các đối tượng đã có.
- 

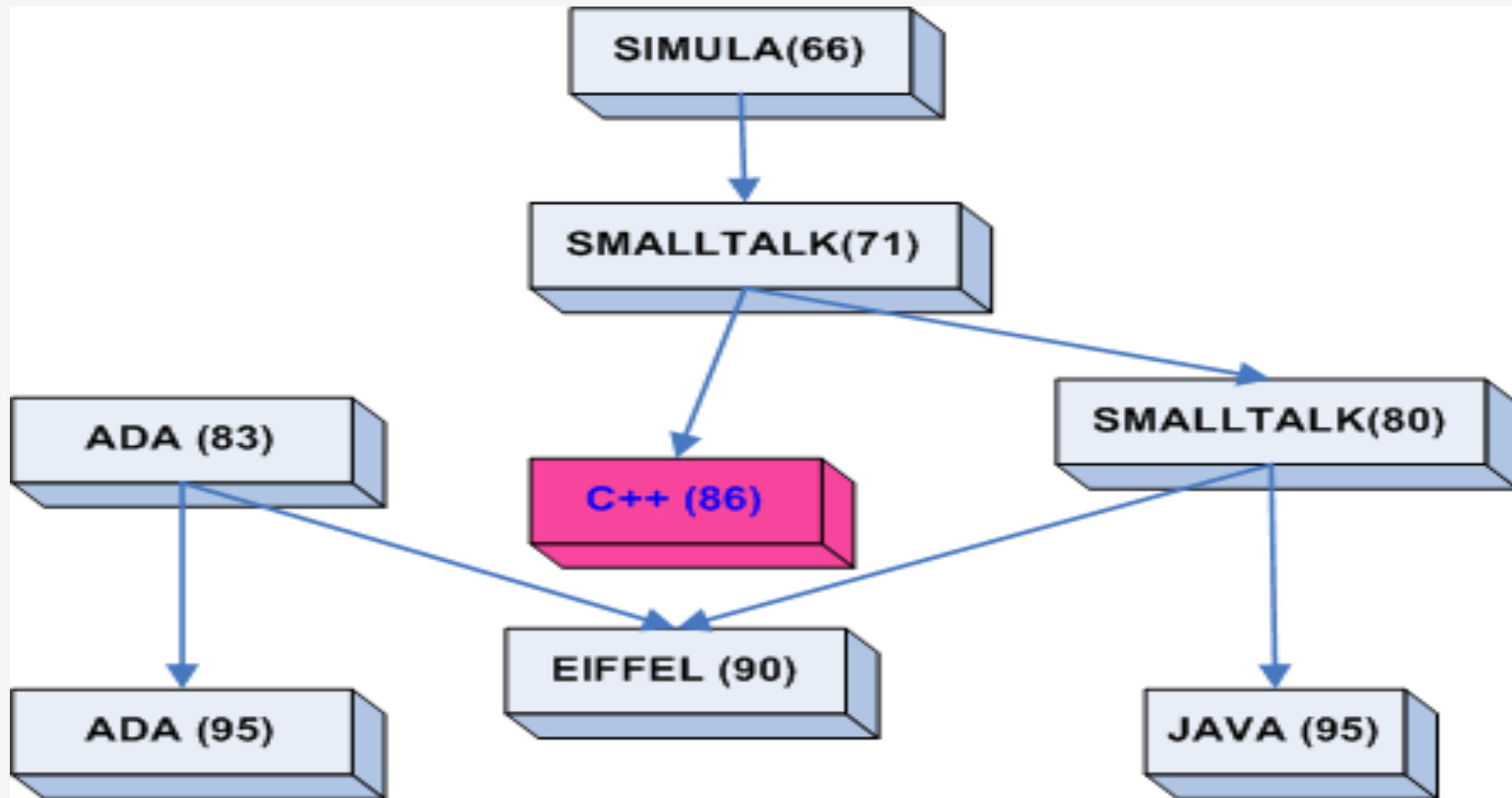
# CÁC ỨNG DỤNG CỦA LẬP TRÌNH OOP

---

## ▪ Các lĩnh vực phù hợp với lập trình OOP:

- Các hệ thống làm việc theo thời gian thực
  - Các hệ thống mô hình hoá hoặc mô phỏng các quá trình
  - Các hệ cơ sở dữ liệu hướng đối tượng
  - Các hệ siêu văn bản và đa phương tiện
  - Các hệ thống trí tuệ nhân tạo
  - Các hệ thống song song
  - Các hệ trợ giúp quyết định
- 

# CÁC NGÔN NGỮ LẬP TRÌNH OOP



# ĐỊNH NGHĨA LỚP VÀ ĐỐI TƯỢNG

---

- Đối tượng (objects) là một thực thể trong thế giới thực.
- Lớp (class) là một kiểu dữ liệu mô tả một tập các đối tượng có cùng tập thuộc tính và hành vi.
- Lớp gần giống với kiểu record của Pascal và kiểu struct của C, nhưng bổ sung thêm các hàm thành phần và các kỹ thuật của lập trình hướng đối tượng như: che dấu thông tin, kế thừa,...



# KHAI BÁO LỚP

- Cú pháp

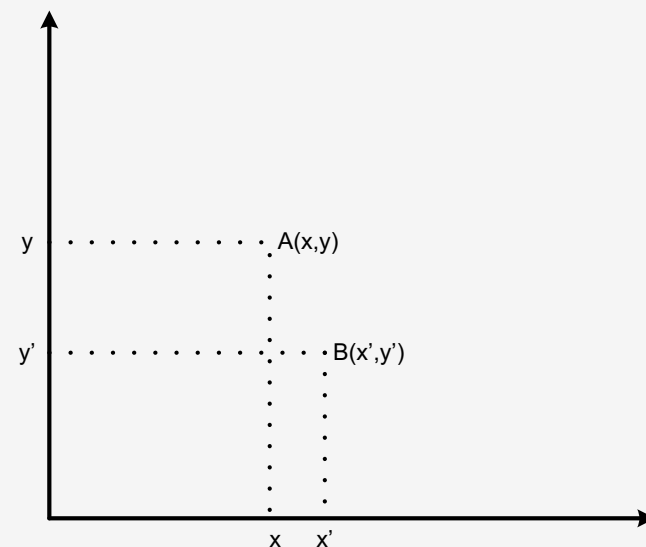
```
class ten_lop {  
    // các thành phần dữ liệu  
    // các hàm thành phần  
};
```

- Ví dụ: Lớp DIEM mô tả các điểm trên mặt phẳng tọa độ.

```
1 class DIEM {  
2     int x;  
3     int y;  
4     void Nhap(int x1, int y1);  
5     float KC();  
6     void Dichuyen(int dx, int dy);  
7     void Hienthi();  
8 };
```

Các thành phần dữ liệu

Các hàm thành phần



# KHAI BÁO LỚP

---

```
1 class SoccerPlayer {    // ten lop
2     private:
3         int number;      // cac thanh phan du lieu
4         string name;
5         int x, y;
6     public:
7         void run();      // cac ham thanh phan
8         void kickBall();
9 }
```

```
1 class Circle {          // Ten lop
2     private:
3         double radius;   // Thanh phan du lieu
4         string color;
5     public:
6         double getRadius(); // Ham thanh phan
7         double getArea();
8 }
```

# TRIỂN KHAI CÁC HÀM THÀNH PHẦN

---

- **Cách 1:** Triển khai ngay trong khai báo lớp.
  - Nội dung của hàm được viết trong phần khai báo lớp.
  - Chỉ áp dụng đối với những hàm đơn giản, không có vòng lặp.

- **Cách 2:** Triển khai ngoài khai báo lớp

- Cú pháp:

```
kiểu_dl_hàm  tên_lớp::tên_hàm(d/s tham số) {  
    //nội dung của hàm  
}
```

- Hầu hết các hàm phức tạp được khai báo ngoài lớp.

# TRIỂN KHAI CÁC HÀM THÀNH PHẦN

- Ví dụ: Triển khai các hàm thành phần của lớp DIEM

```
1 class DIEM {  
2     int x;  
3     int y;  
4     // triển khai trong khai báo lớp  
5     void Nhap(int x1, int y1) {  
6         x = x1;  
7         y = y1;  
8     }  
9     float KC();  
10    void Dichuyen(int dx, int dy);  
11};  
12 // triển khai hàm ngoài khai báo lớp  
13 float DIEM::KC() {  
14     return sqrt(x*x + y*y);  
15 }  
16 void DIEM::Dichuyen(int dx, int dy) {  
17     x += dx; y += dy;  
18 }
```

triển khai trong lớp

triển khai ngoài lớp

# SỬ DỤNG LỚP

---

- Coi lớp như một kiểu dữ liệu trong chương trình, có thể khai báo và sử dụng như các kiểu dữ liệu khác.
- Ví dụ:
  - DIEM A;
  - DIEM B, \*C;

A, B là các biến (đối tượng) của lớp DIEM, mỗi đối tượng này mang đầy đủ các thuộc tính đã khai báo trong lớp DIEM, C là một con trỏ của lớp DIEM

# TRUY XUẤT CÁC THÀNH PHẦN CỦA LỚP

---

- Đối tượng thường
  - Sử dụng toán tử: dấu chấm (.)
  - Cú pháp: Ten\_doi\_tuong.ten\_thanh\_phan;
  - Ví dụ: `A.x = 10;` `A.Dichuyen(2,5);`
- Con trỏ đối tượng
  - Sử dụng toán tử: `->`
  - Cú pháp: `Ten_doi_tuong->ten_thanh_phan;`
  - Ví dụ: `C->x = 10;` `C->Dichuyen(3,4);`

# VÍ DỤ LỚP CRECTANGLE

---

```
1 // classes example
2 #include <iostream>
3 using namespace std;
4
5 class CRectangle {
6     int x, y;
7     public:
8     void set_values (int,int);
9     int area () {return (x*y);}
10 };
11
12 void CRectangle::set_values (int a, int b) {
13     x = a;
14     y = b;
15 }
16
17 int main () {
18     CRectangle rect;
19     rect.set_values (3,4);
20     cout << "area: " << rect.area();
21     return 0;
22 }
```

area: 12

# VÍ DỤ LỚP CRECTANGLE

```
1 // example: one class, two objects
2 #include <iostream>
3 using namespace std;
4
5 class CRectangle {
6     int x, y;
7     public:
8     void set_values (int,int);
9     int area () {return (x*y);}
10 };
11
12 void CRectangle::set_values (int a, int b) {
13     x = a;
14     y = b;
15 }
16
17 int main () {
18     CRectangle rect, rectb;
19     rect.set_values (3,4);
20     rectb.set_values (5,6);
21     cout << "rect area: " << rect.area() << endl;
22     cout << "rectb area: " << rectb.area() << endl;
23     return 0;
24 }
```

```
rect area: 12
rectb area: 30
```



## VÍ DỤ 5-1

---

- Xây dựng lớp DIEM hoàn chỉnh, với các dữ kiện như sau:
  - Thành phần dữ liệu:  $x, y$
  - Hàm thành phần:
    - Nhập dữ liệu điểm
    - Tính khoảng cách từ gốc tọa độ đến điểm
    - Di chuyển điểm một khoảng  $dx, dy$
    - Hiển thị điểm
- Sử dụng lớp DIEM trong hàm `main()` của chương trình.

# CON TRỎ THIS

- Trong mỗi lớp C++ cung cấp một con trỏ đặc biệt trỏ đến chính đối tượng hiện thời.
- Ví dụ:

```
1 class Diem{
2     int x, y, c;
3     Diem(int x, int y) {
4         this->x = x;
5         this->y = y;
6     }
7 };
```

```
9 class Circle {
10     private:
11         double radius;           // thanh phan du lieu co ten "radius"
12         .....
13     public:
14     void setRadius(double radius) { // tham so cua chuong trinh con cung co ten "radius"
15         this->radius = radius;
16         // "this->radius" chi den thanh phan du lieu cua doi tuong
17         // "radius" chi den tham so radius cua chuong trinh con
18     }
19     .....
20 }
```

# CÁC MỨC CHE DẤU THÔNG TIN

---

- C++ cho phép định nghĩa 3 mức độ che dấu thông tin trong lớp.
  - **public**: cho phép truy xuất vào các thành phần của lớp từ mọi nơi trong chương trình.
  - **protected**: chỉ cho phép các hàm thành phần và hàm bạn của lớp đó truy xuất.
  - **private**: chỉ cho phép các hàm thành phần của chính lớp đó truy xuất.
- Khai báo có phân quyền truy xuất (trang bên).

# CÁC MỨC CHE DẤU THÔNG TIN

```
15 class Circle {  
16     private:  
17         double radius;  
18         string color;  
19     public:  
20         double getRadius() const;  
21         void setRadius(double radius);  
22         string getColor() const;  
23         void setColor(string color);  
24         double getArea() const;  
25 };
```

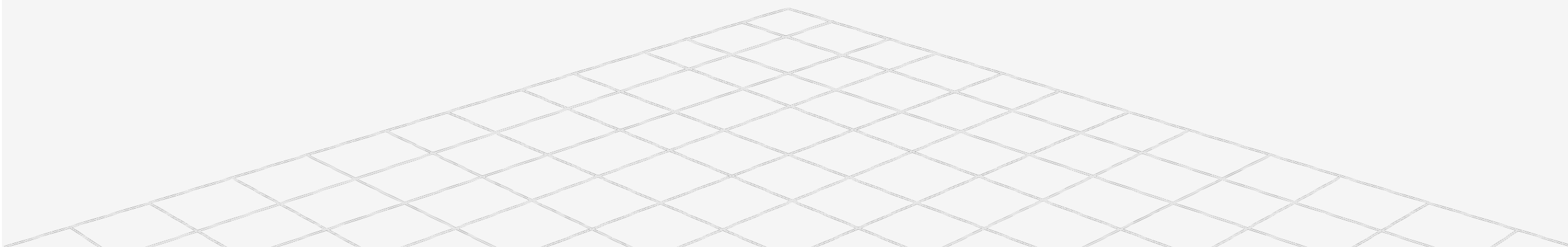
- Nếu không khai báo nhãn phân quyền thì C++ ngầm định là **private**.
- Các nhãn phân quyền tác động đến các khai báo phía dưới nó, cho đến khi gặp một nhãn khác.

```
1 class DIEM {  
2     protected:  
3         int x;  
4         int y;  
5     public:  
6         void Nhap(int x1, int y1);  
7         float KC();  
8     private:  
9         void Dichuyen(int dx, int dy);  
10        void Hienthi();  
11 };
```

## VÍ DỤ 5-2

---

- Xây dựng và thử nghiệm lớp DIEM với sự phân quyền truy xuất cho các thành phần của nó.
- Thử nghiệm việc gọi các thành phần của lớp để kiểm chứng các mức độ che dấu thông tin



# BÀI TẬP

---

## **Bài 5-1:** Xây dựng lớp HOCSINH với các thành phần như sau

- Dữ liệu
  - Họ tên; Điểm toán; Điểm lý; Điểm hoá; Điểm trung bình
- Hàm thành phần
  - Nhập dữ liệu cho học sinh gồm họ tên và điểm ba môn (Toán, Lý, Hóa)
  - Tính điểm Trung bình cho học sinh
  - Phân loại học sinh (Giỏi, Khá, TB, Yếu) dựa trên điểm trung bình
  - Hiển thị thông tin học sinh

# BÀI TẬP

---

**Bài 5-2:** Hãy xây dựng lớp **Time** gồm các thành phần dữ liệu và các hàm thành phần như hình dưới đây. Chú ý hàm `nextSecond()` sẽ cộng thêm 1 giây vào thời gian hiện tại theo quy luật cộng của thời gian; hàm `print()` hiển thị thời gian dưới dạng: hh:mm:ss.

Time
-hour:int = 0 -minute:int = 0 -second:int = 0
+Time(h:int, m:int, s:int) +getHour():int +getMinute():int +getSecond():int +setHour(h:int):void +setMinute(m:int):void +setSecond(s:int):void +setTime(h:int, m:int, s:int) +print():void +nextSecond():void