Phương pháp Lập trình Hướng đối tượng

Kế thừa Lớp đối tượng

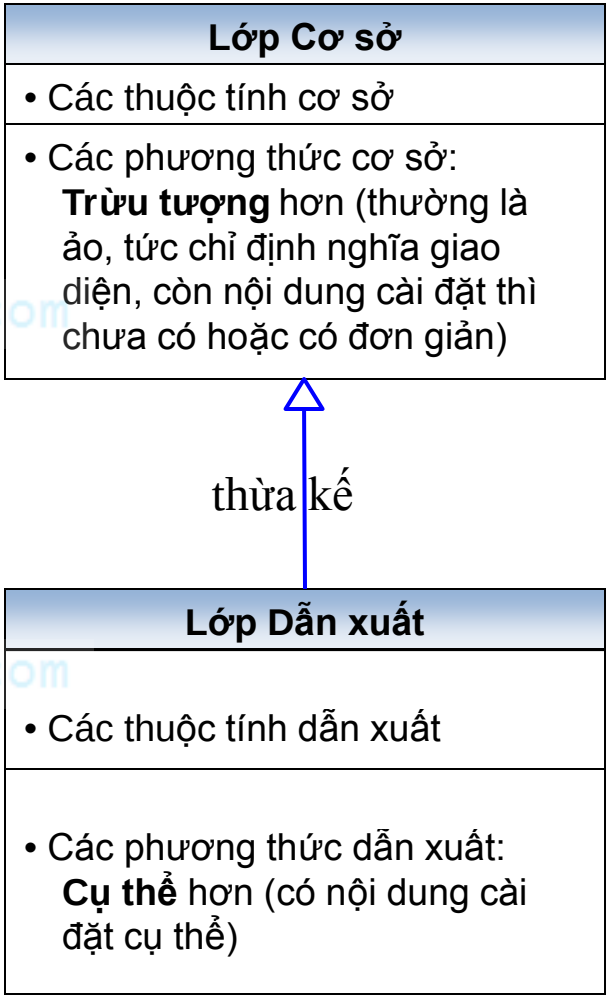
"Con hơn cha là nhà có phúc!"

"Inheritance = IS-A + HAS-A + Protected"

GV: Lê Xuân Định

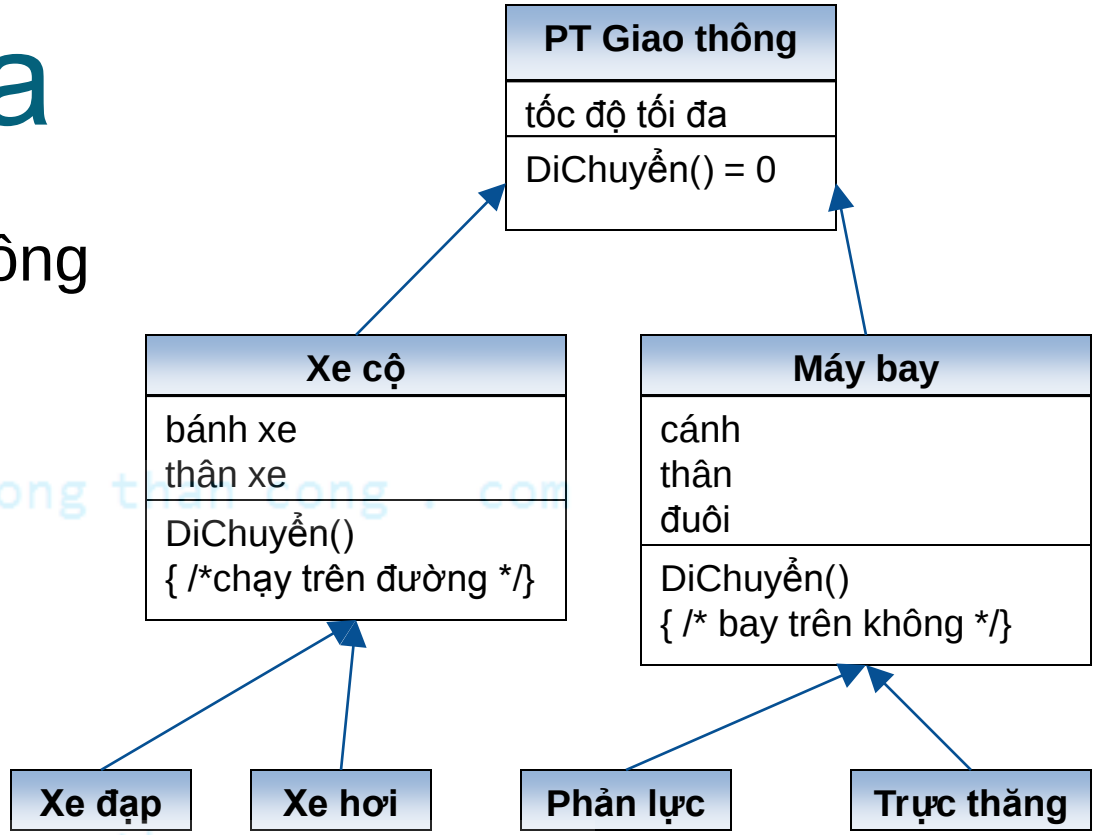
Lớp Cơ sở – Lớp Dẫn xuất

- Phương tiện Giao thông
  - Thuộc tính: Tốc độ tối đa
  - Phương thức: Di chuyển
  - Xe cộ
    - Thuộc tính: Tốc độ tối đa, **bánh, thân**
    - Phương thức: Di chuyển (**chạy trên đường**)
    - Xe đạp
      - Thuộc tính: Tốc độ tối đa, bánh, thân, **xích, líp, ...**
      - Phương thức: Di chuyển
    - Xe hơi
      - Thuộc tính: Tốc độ tối đa, bánh, thân, **xăng, động cơ, ...**
      - Phương thức: Di chuyển, **đổ xăng**



Cây kế thừa

- Phương tiện Giao thông
  - Xe cộ
    - Xe máy
    - Xe đạp
    - Xe hơi
    - Xe tải
  - Tàu thuyền
    - Tàu cánh ngầm
    - Thuyền buồm
  - Máy bay
    - Phản lực
    - Trực thăng

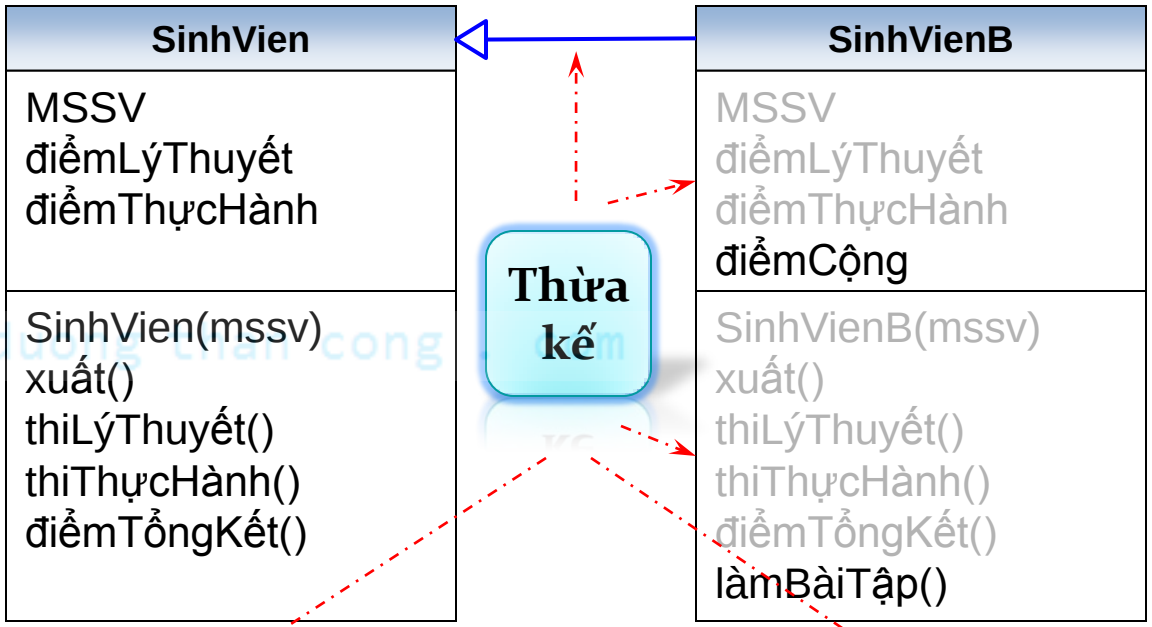


Kế thừa (chiều thuận)

- Bài toán 1
  - Đã ***có sẵn*** lớp SinhVien
    - Thuộc tính: ***MSSV***, điểm lý thuyết (***đLT***), điểm thực hành (***đTH***)
    - Phương thức:
      - **Khởi tạo, xuất**, **thi LT**, **thi TH**
      - **Tính điểm Tổng kết** (***đTK***): Trả về đTK = (6*đLT + 4*đTH)/10
  - Cần định nghĩa lớp SinhVienB
    - Thuộc tính: ***MSSV***, điểm lý thuyết (***đLT***), điểm thực hành (***đTH***), điểm cộng (***đCộng***)
    - Phương thức:
      - **Khởi tạo, xuất**, **thi LT**, **thi TH**, **làm bài tập (lấy điểm cộng)**
      - **Tính điểm Tổng kết** (***đTK***): Trả về đTK = (6*đLT + 4*đTH)/10 **+ đCộng**

Kế thừa – Giao diện

- Lớp ***dẫn xuất***
  - **Thừa kế** mọi thành phần(*) của lớp cơ sở;
  -
  -




```
class SinhVien
{
 protected:
   int mssv;
   float dLT, dTH;
 public:
   SinhVien(int mssv);
   ...
};
```

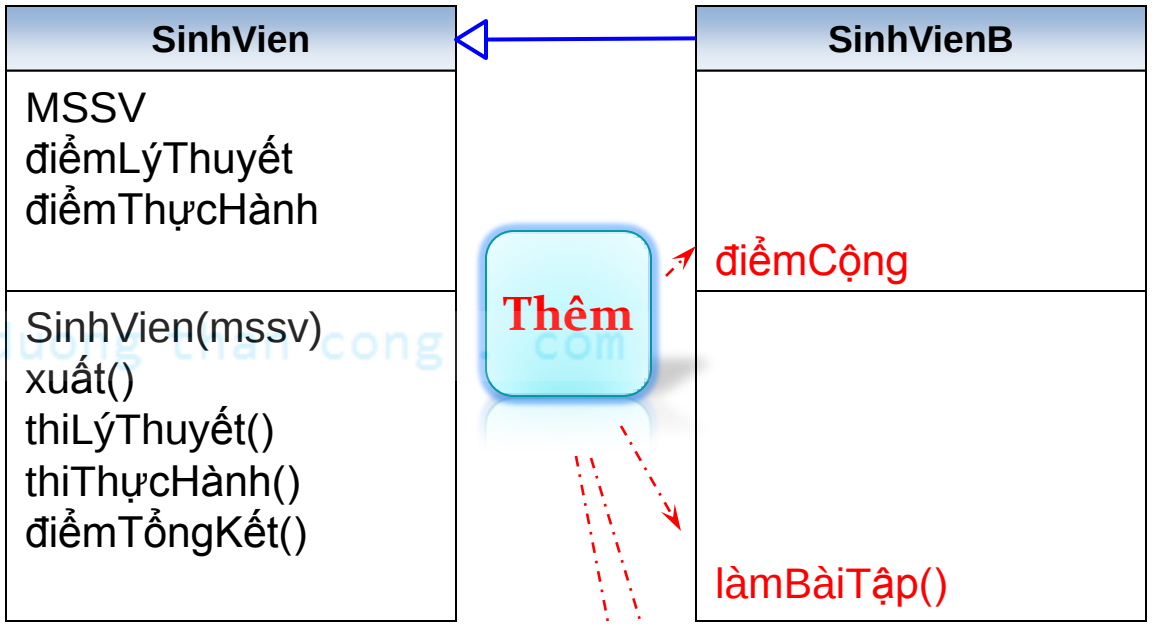
```
class SinhVienB
{



};
```

† (*) Nhưng lớp dẫn xuất chỉ truy cập được tới các thành phần trong phạm vi protected & public của lớp cơ sở.

†

Kế thừa – Giao diện

- Lớp ***dẫn xuất***
  - **Thừa kế** mọi thành phần(*) của lớp cơ sở;
  - **Thêm** một số thành phần riêng của lớp dẫn xuất;
  -




```
class SinhVien
{protected: ...
 public:
  SinhVien(int mssv);
  void xuat();
  void thiLT();
  void thiTH();
  float diemTK();
};
```

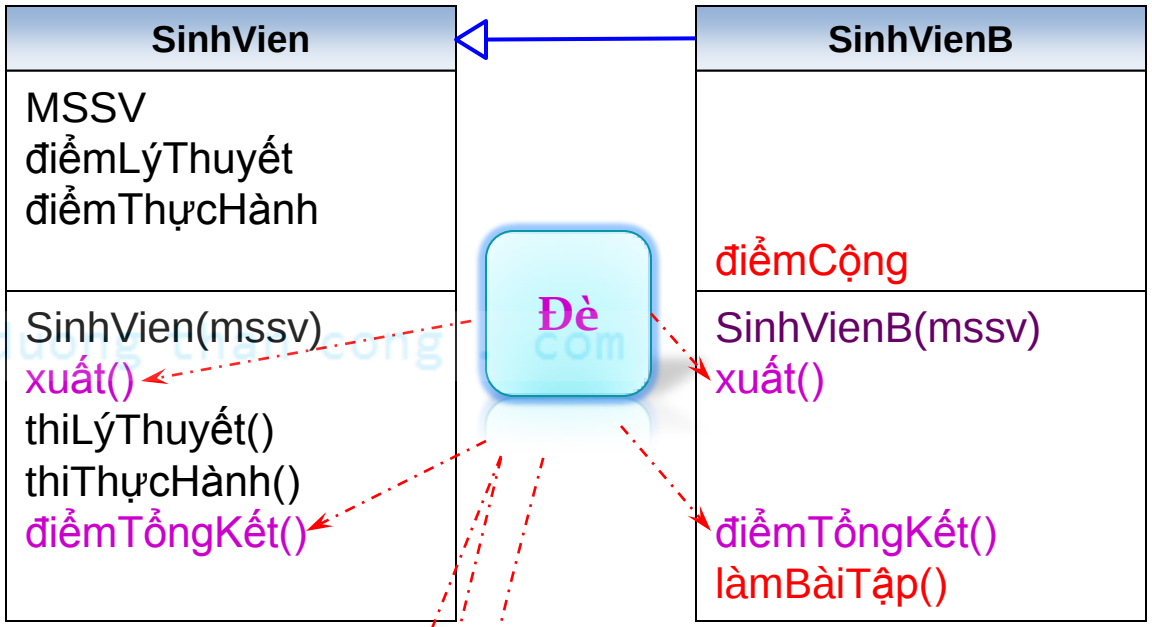
```
class SinhVienB:public SinhVien
{private:
  float dCong;
 public:
  void lamBT();


};
```

† (*) Nhưng lớp dẫn xuất chỉ truy cập được tới các thành phần trong phạm vi protected & public của lớp cơ sở.

†

Kế thừa – Giao diện

- Lớp ***dẫn xuất***
  - **Thừa kế** mọi thành phần(*) của lớp cơ sở;
  - **Thêm** một số thành phần riêng của lớp dẫn xuất;
  - **Cài đặt lại** (đè lên) một số phương thức của lớp cơ sở.




```
class SinhVien
{protected: ...
 public:
  SinhVien(int mssv);
  virtual void xuat();
  void thiLT();
  void thiTH();
  virtual float diemTK();
};
```

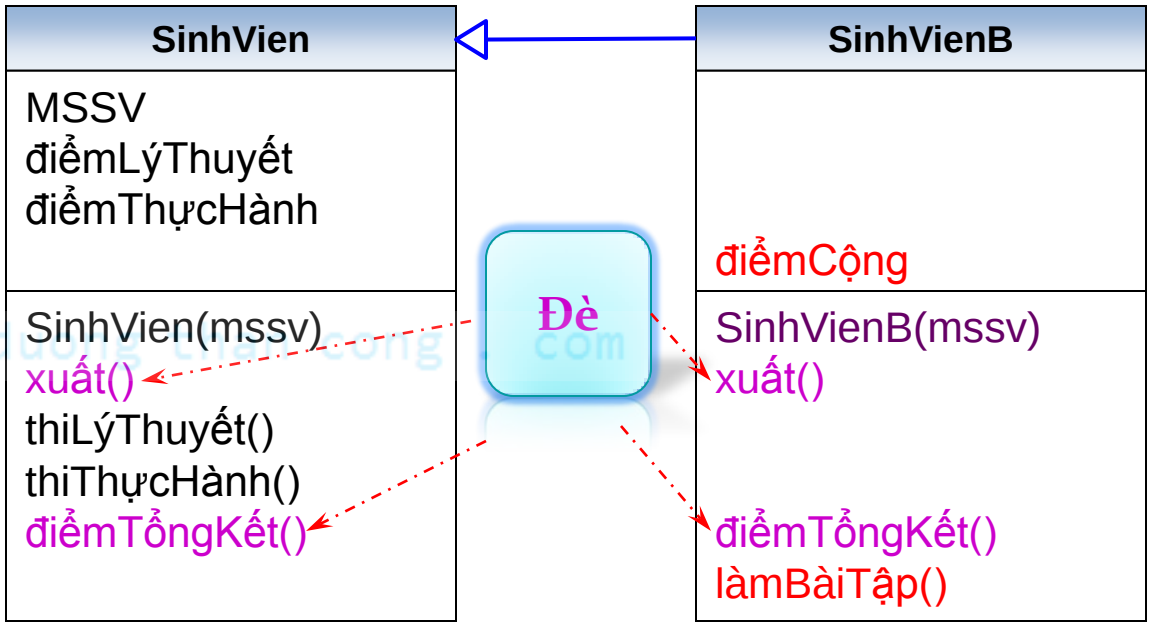
```
class SinhVienB:public SinhVien
{private:
  float dCong;
 public:
  void lamBT();
  SinhVienB(int mssv);
  void xuat();
  float diemTK();
};
```

† (*) Nhưng lớp dẫn xuất chỉ truy cập được tới các thành phần trong phạm vi protected & public của lớp cơ sở.

† Cài đặt đè: override

Kế thừa – Cài đặt đè

- Điều kiện để cài đặt đè
  - Lớp cơ sở phải cho phép **đè** bằng khai báo ***virtual***;
  - Lớp dẫn xuất phải khai báo lại đúng như nguyên mẫu hàm ở lớp cơ sở.
- Nếu một phương thức không virtual thì
  - PThức của lớp dẫn xuất độc lập với phương thức của lớp cơ sở!




```
class SinhVien
{protected: ...
 public:
   SinhVien(int mssv);
   virtual void xuat();
   void thiLT();
   void thiTH();
   virtual float diemTK();
};
```

```
class SinhVienB:public SinhVien
{private:
   float dCong;
 public:
   void lamBT();
   SinhVienB(int mssv);
   void xuat();
   float diemTK();
};
```

Kế thừa – Cài đặt đè

- Một số phương thức của lớp dẫn xuất
  - Có ***cùng giao diện*** (nguyên mẫu hàm) với lớp cơ sở, nhưng
  - **Khác phần cài đặt** (thường cụ thể, chi tiết hơn).


```
class SinhVien
{
protected:
  int mssv;
  float dLT, dTH;
public:
  ...
  virtual void xuat();
    // Xuất tất cả các thuộc tính
};
```

mssv, dLT, dTH

```
class SinhVienB:public SinhVien
{
private:
  float dCong;
public:
  ...
  void xuat();
    // Xuất tất cả các thuộc tính
};
```

mssv, dLT, dTH, dCong

Kế thừa – Cài đặt đè

- Một số phương thức của lớp dẫn xuất
  - Có ***cùng giao diện*** (nguyên mẫu hàm) với lớp cơ sở, nhưng
  - **Khác phần cài đặt** (thường cụ thể, chi tiết hơn).

➔ Lựa chọn 1: Cài đặt lại hoàn toàn.

```
void SinhVien::xuat()
{
 cout<<"MSSV: "<<this->mssv<<endl;
  cout<<"diem LT: "<<this->dLT<<endl;
  cout<<"diem TH: "<<this->dTH<<endl;
}
```

```
void SinhVienB::xuat()
{
                    ???
  cout<<"diem cong: "
      <<this->dCong<<endl;
}
```

Kế thừa – Cài đặt đè

- Một số phương thức của lớp dẫn xuất
  - Có ***cùng giao diện*** (nguyên mẫu hàm) với lớp cơ sở, nhưng
  - **Khác phần cài đặt** (thường cụ thể, chi tiết hơn).

➔ Lựa chọn 1: Cài đặt lại hoàn toàn.

➔ Lựa chọn 2: **Tái sử dụng** phần cài đặt của lớp cơ sở (nếu có)

```
void SinhVien::xuat()
{
  cout<<"MSSV: "<<this->mssv<<endl;
  cout<<"diem GK: "<<this->dGK<<endl;
  cout<<"diem CK: "<<this->dCK<<endl;
}
```

```
void SinhVienB::xuat()
{

  this->SinhVien::xuat();
  cout<<"diem cong: "
      <<this->dCong<<endl;
}
```

Kế thừa – Constructor

- Constructor của lớp dẫn xuất
  - ***Không tự động kế thừa* từ lớp cơ sở** (kể cả default constructor)
    - Buộc **phải gọi lại constructor của lớp cơ sở** nếu muốn kế thừa!
  - ♣ Nhưng... "*sinh cha rồi mới sinh con*"

    ➔ C++: Thay lời gọi phương thức bằng cú pháp "kế thừa constructor"

    ➔ Java: Buộc phải gọi constructor của lớp cơ sở trước tiên.

```
SinhVien::SinhVien(int mssv)
{
  this->mssv = mssv;
  this->dLT = this->dTH = -1;
}
```

```
SinhVienB::SinhVienB(int mssv)
:SinhVien(mssv)
{
  this->dCong = 0;
}
```

Đây là lời ***gọi hàm***,
không phải là ~~khai báo nguyên mẫu hàm~~!!!
Mang ý nghĩa "giống như" viết trong thân hàm:
`this->SinhVien(mssv);`

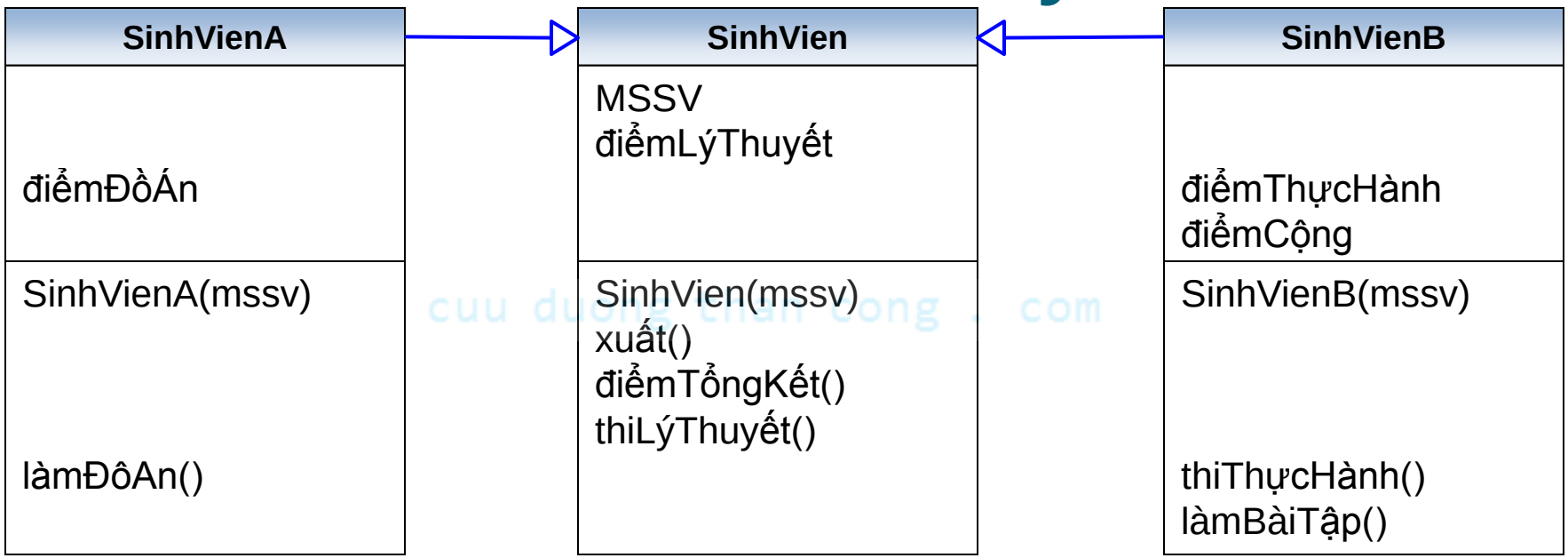
Bài tập: Ứng dụng Kế thừa

- Lấy lớp Xe trong bài ứng dụng "Xe - Xăng" làm lớp cơ sở để định nghĩa lớp dẫn xuất sau:
  - XeTải
    - **Độ cao (mét)**, **tải trọng (tấn)** có thể được chỉ định bởi nhà sản xuất nhưng không đổi đối với mỗi xe.
    - **Thực tải (tấn)** là lượng hàng mà xe đang chở, chỉ được thay đổi thông qua phương thức ***lên/xuống hàng***, và không được vượt quá tải trọng của xe.
    - 1/(**Độ hao xăng**)= (tải trọng – thực tải + 10)/(tải trọng) km/lít
    - **Số bánh**: người sử dụng có thể biết đang có mấy bánh và *tuỳ tiện* ***thay đổi số bánh*** với điều kiện là số chẵn ≥ 4 bánh và ≥ 2*[(thực tải)/10]. ([x] = ceil(x) trong thư viện math)

Kế thừa (chiều nghịch)

- Bài toán 2: Cần định nghĩa 2 lớp gần giống nhau
  - SinhVienA
    - Thuộc tính: ***MSSV***, điểm lý thuyết (***đLT***), điểm đồ án (***đĐA***)
    - Phương thức:
      - **Khởi tạo, xuất, thi LT, làm đồ án**
      - **Tính điểm Tổng kết** (***đTK***): Trả về đTK = (đLT + đĐA)/2
  - SinhVienB
    - Thuộc tính: ***MSSV***, điểm lý thuyết (***đLT***), điểm thực hành (***đTH***), điểm cộng (***đCộng***)
    - Phương thức:
      - **Khởi tạo, xuất, thi LT, thi TH, làm bài tập (lấy điểm cộng)**
      - **Tính điểm Tổng kết** (***đTK***): Trả về đTK = (6*đLT + 4*đTH)/10 + đCộng

Kế thừa – Thiết kế Cây kế thừa

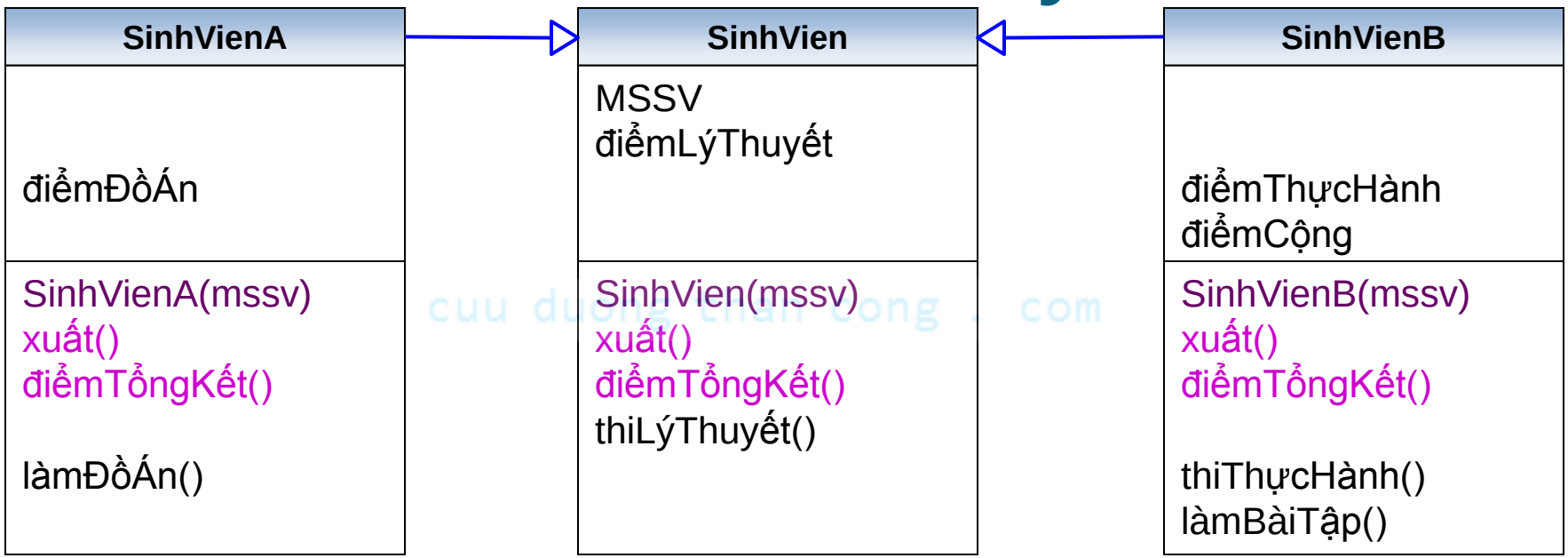


- Định nghĩa lớp ***cơ sở***
  - Có hết những **thành phần chung** (về ***giao diện***) của các lớp cần định nghĩa.
  -

    1.

    2.

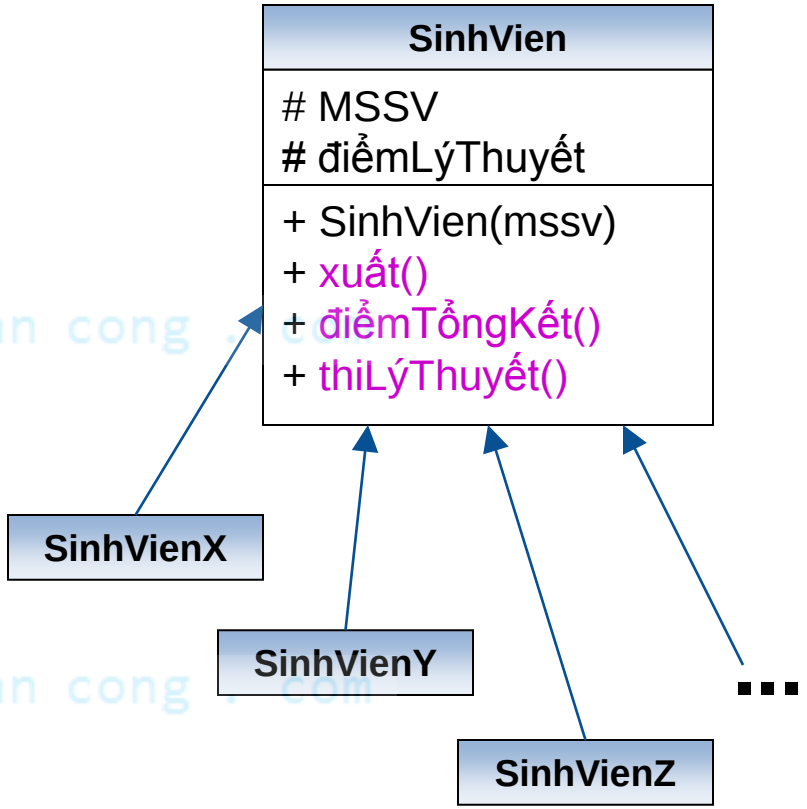
Kế thừa – Thiết kế Cây kế thừa



- Định nghĩa lớp ***cơ sở***
  - Có hết những **thành phần chung** (về ***giao diện***) của các lớp cần định nghĩa.
  - Những phương thức giống nhau về giao diện nhưng khác nhau phần cài đặt giữa các lớp thì có 2 lựa chọn:
    1. Lớp cơ sở cài đặt rỗng (hoặc thuần ảo, tức không có phần cài đặt)
       Các lớp dẫn xuất tự cài đặt hoàn toàn những phương thức đó.
    2. Lớp cơ sở cài đặt phần xử lý chung (nếu có) (vẫn ảo, nhưng không thuần ảo)
       Các lớp dẫn xuất **tái sử dụng** phần xử lý chung, và cài đặt thêm phần xử lý riêng.

Thiết kế Lớp Hướng Kế thừa

- Khi thiết kế một lớp, nên nghĩ đến những lớp dẫn xuất của nó (trong tương lai).
  - Những thuộc tính mà lớp dẫn xuất có thể dùng: Cung cấp p.thức get/set qua giao diện **protected**.
  - Những phương thức mà lớp dẫn xuất có thể thay đổi: ***virtual***.



Bài tập 3: ƯD Cây kế thừa

- Hãy viết chương trình nuôi thú (chó & mèo) sao cho ***tính tái sử dụng*** được cao nhất:
  - Mỗi con Chó và Mèo đều có ***tên*** và ***trọng lượng*** của nó.
  - Mỗi tháng ta đều phải ***mua thức ăn*** để nuôi chúng với số tiền
    - Chó: ***Trọng lượng*** × 15000₫, và
    - Mèo: ***Trọng lượng*** × 12000₫ – ***số chuột*** ×1000₫
      - Ở đây "***số chuột***" là tổng số chuột bắt được trong tháng. Và mỗi tháng (sau khi mua thức ăn xong) thì số chuột được reset lại thành 0.
  - Ngoài ra, chúng còn có các hành động khác:
    - ***Kêu***: In tiếng kêu ra màn hình.
    - Mèo ***bắt chuột***: tăng số chuột bắt được lên 1 lượng xác định.

Bài tập 4: Lớp quản lý

- Hãy thiết kế lớp **Lớp học** để quản lý các **Sinh viên** trong một lớp học.
  - Thuộc tính: Mảng các đối tượng Sinh viên (và số SV trong lớp)
  - Phương thức:
    - Khởi tạo mặc định: Lớp trống (không có SV nào).
    - Thêm một SV vào danh sách lớp.
    - Xuất bảng điểm của cả lớp. (Mỗi SV xuất các thuộc tính & điểm tổng kết)
    - Tìm SV giỏi nhất (có điểm tổng kết lớn nhất).