**Lab 12**

**Hàm ảo, liên kết động**

# Lập trình hướng đối tượng

|  |  |
| --- | --- |
| **Mục tiêu** | 1. Hiểu về hàm ảo và hàm thuần ảo 2. Hiểu về liên kết động 3. Hiểu tại sao hàm hủy đối tượng phải là hàm ảo |

**Bộ môn Công nghệ phần mềm – Khoa Công nghệ thông tin**



# Hướng dẫn khởi đầu 1 – Hàm ảo (virtual)

**Mô tả bài tập**

Mục tiêu hàm ảo là cung cấp **cài đặt mặc định** cho một hành vi.

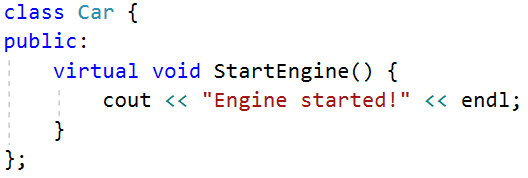
Nếu lớp con không cài thì sẽ dùng cách hành xử lớp cha.

Nếu có cài thì sẽ dùng cách hành xử lớp con chứ không dùng hành xử mặc định.

**Hướng dẫn cài đặt bước 1**

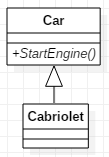
Xét lớp Xe hơi (**Car**) có cài đặt mặc định của hàm **StartEngine** như sau

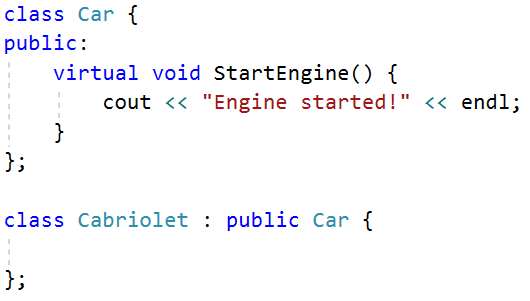




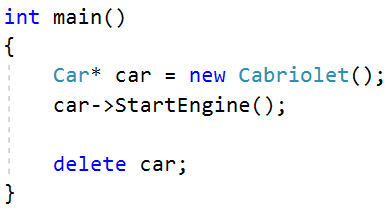
**Bước 2 – Mở rộng hệ thống – Sử dụng hành xử mặc định**

Sau này ta mở rộng hệ thống, có lớp Xe hơi mui trần (**Cariolet**) kế thừa từ lớp **Car**





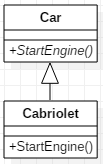
Lúc này nếu ta tạo ra một xe hơi mui trần và khởi động động cơ

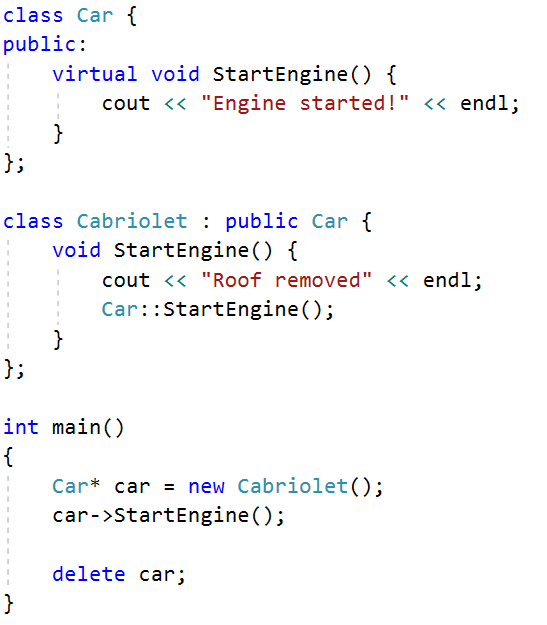


Thì ta sẽ thấy kết quả:

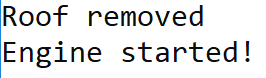


**Bước 3 – Nâng cấp hệ thống – Cài đặt cách hành xử riêng**





Chạy lên để thấy kết quả như sau



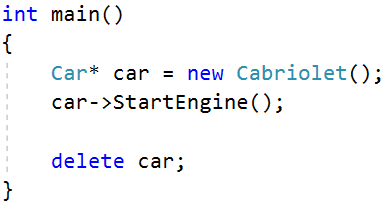
Chú ý bên trong hàm StartEngine của lớp con đã gọi hàm StartEngine của lớp cha, và bởi vì cả cha và con đều có hàm StartEngine cùng tên nên để trình biên dịch biết gọi hàm này là từ cha hay con, ta sẽ thêm vào tên lớp cha ở phía trước!

Hệ quả là ta có: **Car**::**StartEngine**();

Người mới bắt đầu sẽ dễ bị nhầm lẫn vì cách viết này giống như gọi hàm tĩnh của lớp, tuy nhiên ý nghĩa hai việc này khác nhau nhé!

Ngoài ra có một việc cực kì quan trọng.

Cần coi lại hàm main:



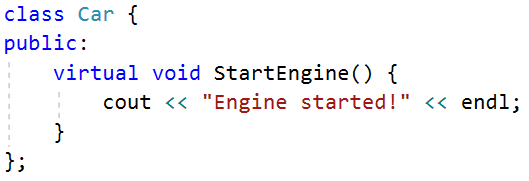
Biến car có kiểu là con trỏ lớp cha – Car\*, lại giữ địa chỉ đến vùng nhớ có kiểu của lớp con – Cabriolet\*.

Câu hỏi rất quan trọng là hàm car->StartEngine() gọi hàm ở lớp con hay lớp cha?

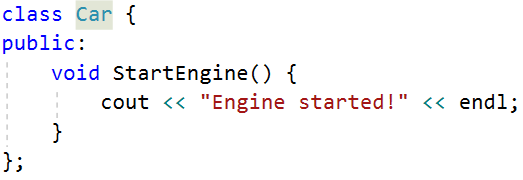
Dựa trên kết quả bạn thấy thì bạn kết luận được ngay là từ đâu?

Tất nhiên là hàm **StartEngine** được gọi là của lớp **con** - **Cabriolet**.

Thử sửa một chút khai báo của lớp cha – **Car**, hàm **StartEngine** lúc ban đầu:



Từ chỗ đang có từ khóa virtual, ta thử bỏ đi thành



Chạy lại và ta thấy kết quả kì lạ:



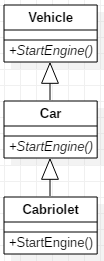
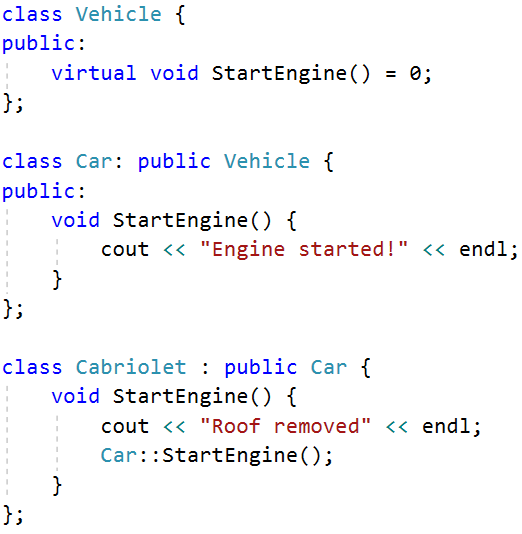
Kết quả này cho thấy hàm **StartEngine** được gọi lúc này là của **lớp cha**!!!

Tóm lại từ khóa virtual tạo ra liên kết động cho con trỏ. Tùy thuộc con trỏ đang trỏ đến đối tượng cụ thể nào mà hành động tương ứng của lớp sẽ được gọi.

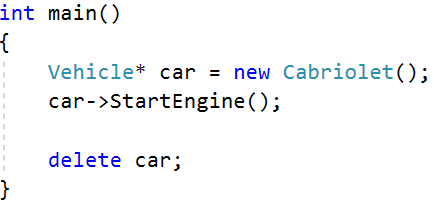
# Hướng dẫn khởi đầu 2 – Hàm thuần ảo (pure virtual)

Hàm thuần ảo là hàm ảo nhưng chỉ có phần khai báo mà không có cài đặt. Việc cài đặt cụ thể hoàn toàn phó thác cho lớp kế thừa.

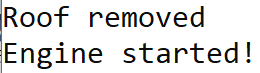
Cải tiến thiết kế về xe hơi ở trên ta sẽ có một lớp mô tả thông tin về phương tiện di chuyển tổng quát (**Vehicle**) như sau:



Hàm main như sau:



Kết quả:

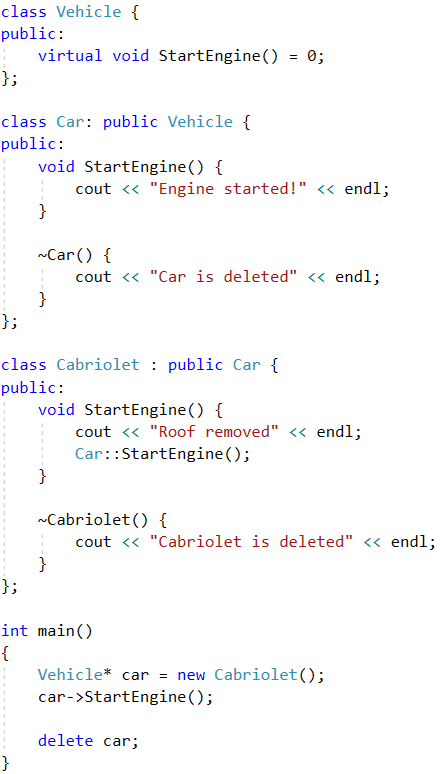


Một lần nữa chú ý từ khóa virtual của lớp cha giúp cho con trỏ car gọi thực thi hàm đúng của lớp con.

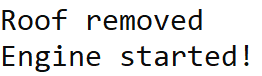
# Hướng dẫn khởi đầu 3 – Hàm hủy ảo

Nếu hiểu được từ khóa virtual giúp con trỏ lớp cha gọi hàm ứng với kiểu dữ liệu mà nó đang giữ thì hàm hủy ảo cũng tương tự, nhờ có từ khóa virtual mà con trỏ lớp cha gọi hàm hủy của lớp con mà nó đang trỏ tới.

Xét ví dụ như mô tả bên dưới (sử dụng mã nguồn ở hướng dẫn trước, chỉ thêm hàm hủy)

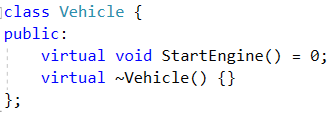


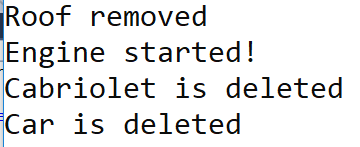
Chạy chương trình lên và ta … không thấy hàm hủy nào được gọi! Lý do là gì?



Lí do là con trỏ car đang có kiểu **Vehicle**\*, mặc dù nó đang trỏ đến đối tượng kiểu **Cabriolet**\*. Tuy nhiên khi gọi hàm delete car thì hàm hủy của lớp Vehicle sẽ được gọi. Mà ta không cài đặt hàm hủy thì trình biên dịch sẽ gọi hàm hủy mặc định (vốn không làm gì cả - hàm rỗng).

Vậy chỉ cần sửa lại hàm hủy của lớp **Vehicle** thành hàm hủy ảo như sau:



Chạy lại mã nguồn sau khi thay đổi để thấy như sau: 

# Bài tập vận dụng

**Yêu cầu**

1. Thực hiện định nghĩa lớp theo thiết kế cho trước vào tập tin .h.
2. Thực hiện cài đặt lớp trong tập tin .cpp cho lớp tương ứng.
3. Viết các đoạn mã nguồn theo yêu cầu như bên dưới

**Danh sách các lớp cần cải tiến cụ thể**

1. Viết lại bài tập **nông trại** ở tuần trước sử dụng hàm thuần ảo.

**Gợi ý:** áp dụng hàm thuần ảo trong các hàm **Kêu**, **Cho sữa** và **Sinh con**.

+ Tất cả các con vật có thể có chung lớp cha là gì?

2. Viết bài tập vẽ hình: **đường thẳng**, **hình chữ nhật**, hình **ê líp** sử dụng hàm thuần ảo.

Tự phát sinh n hình thuộc về 3 loại hình trên và ngẫu nhiên giá trị các thuộc tính luôn.

Lần lượt lặp qua từng hình, vẽ hình tương ứng. Để đơn giản hàm Draw chỉ cần cout là đang vẽ hình gì.

+ Tất cả các hình trên có thể kế thừa từ lớp cha nào?

+ Ve so do lop

# Hướng dẫn nộp bài

**Trước khi nộp cần chú ý:**

- Lấy tập tin exe được biên dịch sẵn trong thư mục Debug, copy nó ra thư mục Release bên ngoài mã nguồn.

- Xóa hết tất cả các tập tin trung gian trong quá trình biên dịch bằng cách chọn **Build** > **Clean**.

- Chú ý thư mục ẩn **.vs** rất nặng. Cần hiển thị file ẩn mới thấy và xóa nó đi được.

Nếu bạn muốn biết cách làm đúng thì cần tự tìm cách build ở chế độ Release và copy file exe kết quả ra bên ngoài để nộp mới đúng. Tuy nhiên nếu chưa hiểu ý nghĩa thì cứ lấy đại file exe có sẵn đi nộp cũng được (hiện tại đang trong thư mục Debug ứng với chế độ biên dịch Debug)

**Tổ chức bài nộp**

+ Thư mục **Source**: chứa mã nguồn đã được clean

+ Thư mục **Release**: chứa tập tin thực thi đã được biên dịch từ mã nguồn

+ Tập tin **readme.txt**: chứa thông tin sinh viên, gồm MSSV và họ tên. Ghi chú kèm các thông tin giáo viên cần chú ý khi chấm bài.

Để nộp bài, nén tất cả lại và đặt tên với định dạng **MSSV.zip** hoặc **MSSV.rar** và nộp.

**Nếu làm đúng các bước trên file này sẽ có kích thước < 100 KB!**

(Tuy nhiên cũng đừng quá lo lắng nếu nó khác con số trên, miễn < 12 MB để nộp được trên moodle là okie nhé!)

**Ngoài lề**: Để đảm bảo sau này nhìn vào file nén còn biết ngay nó làm gì, ta nên thêm vào một số thông tin theo sau MSSV. Ví dụ: 0712221-Lab12-VirtualDynamicBinding.zip. Tuy nhiên việc này là KHÔNG bắt buộc nhé.

**-- HẾT --**