<https://xuanthulab.net/su-dung-giao-dien-idisposable-va-tu-khoa-using-trong-c-sharp.html>

## Giao diện IDisposable

Trong thư viện .NET đưa ra một giao diện interface có tên là IDisposable (System.IDisposable). Giao diện này chỉ có định nghĩa một phương thức:

public void Dispose ();

Các lớp triển khai giao diện này chỉ việc định nghĩa nội dung phương thức này, mục đich code viết trong phương thức này là các thao tác để giải phóng các tài nguyên chiếm giữ - khi đối tượng bị hủy

**Tại sao cần tự giải phóng tài nguyên**

Như đã biết, trong .NET hầu hết các loại tài nguyên là được quản lý bởi CLR của hệ thống .NET, nên các tài nguyên, đối tượng không còn tham chiếu đến nó sẽ tự động được CLR thu hồi (GC). Đó là những tài nguyên quản lý được bởi .NET CLR

Tuy nhiên, vẫn có những loại tài nguyên mà CLR .NET không quản lý được như:

* Mở file - stream
* Các kết nối mạng, kết nối đến CSDL ...
* Những vùng bộ nhớ không quản lý được, các font chữ ...

Với những loại tài nguyên này, .NET không biết tự giải phóng nó thế nào, nên bạn phải có một cơ chế chủ động làm việc này khi không còn dùng đến nữa.

Bạn cần viết code giải phóng tài nguyên thích hợp ở [phương thức hủy (Finalize)](https://xuanthulab.net/mot-so-chu-de-lap-trinh-lop-c-nang-cao.html#destructor) và có thể là triển khai giao diện IDisposable để sử dụng với câu lệnh using



Bạn thấy, phương thức Dispose() sẽ tự động chạy khi hết using

Đây chính là cách để chủ động giải phóng tài nguyên không quản lý được bởi .NET, khi tài nguyên đó không còn dùng đến nữa. Thư viện .NET có hàng trăm lớp triển khai IDisposable bạn có thể sử dụng với using như: Font, Brush, Stream, TextReader, WebResponse, Socket, DbConnection ...

Khi sử dụng đối tượng triển khai IDisposable mà không dùng using, bạn phải chủ động gọi thủ công Dispose khi tài nguyên không dùng đến.

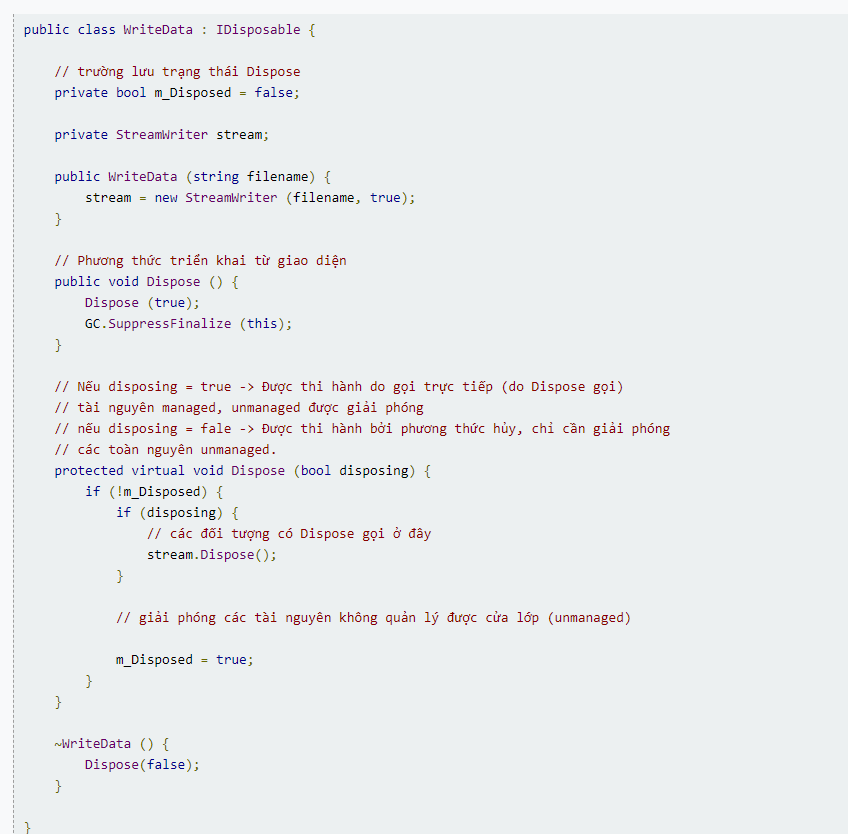
## Triển khai IDisposable cùng với hàm Hủy

Hàm hủy - phương thức hủy (còn gọi là Finalize) như trình bày ở mục [phương thức hủy (Finalize)](https://xuanthulab.net/mot-so-chu-de-lap-trinh-lop-c-nang-cao.html#destructor), nó là phương thức tự động chạy khi đối tượng không còn tham chiếu - và cũng dùng nó để viết code giải phóng tài nguyên

Vấn đề xảy ra khi một lớp vừa có hàm hủy vừa có Dispose() - tức triển khai IDisposable là: thao tác giải phóng tài nguyên có thể thực hiện hai lần - một lần khi ra khỏi using - một lần đối tượng mất tham chiếu. Hoặc khi đối tượng chủ động gọi Dispose nhiều lần. Điều này có thể dẫn đến lỗi.

Cách giải quyết là cần có biến lưu lại trạng thái cho biết Dispose đã được thi hành hay chưa.

Ví dụ, một lớp WriteData có triển khai IDisposable có thể cần giải phóng một số tài nguyên nó chiếm giữ không quản lý được bởi .NET. WriteData cũng có sử dụng đối tượng lớp StreamWriter, đối tượng này cũng triển khai IDisposable, nên nó cần gọi thủ công Dispose() của nó khi không còn dùng đến nữa.



Mẫu trên bạn có thể áp dụng cho bất ký lớp nào muốn triển khai IDisposable, lúc đó bạn sử dụng using hay không using nó đều hoạt động đúng logic

Sử dụng với using, hết lệnh Dispose sẽ thi hành và mọi tài nguyên được giải phóng

using (WriteData writeData = new WriteData("filename.txt")) {

// do something

}

Nếu không dùng using, thì chủ động gọi Dispose, tài nguyên cũng giải phóng đúng yêu cầu.

WriteData writeData = new WriteData("filename.txt");

//do something

writeData.Dispose();