Logo

Description automatically generatedA picture containing text

Description automatically generatedShape, square

Description automatically generated

**GVHD: Ths. Nguyễn Thị Thanh**

**Nhóm sinh viên thực việc: Nhóm 22**

12 Lê Hải Đăng 20110243

70 Lương Xuân Thắng 20110724

Mã lớp: OOSD330879\_22\_2\_02

**Tp. Hồ Chí Minh, tháng 5 năm 2023**

**ĐỀ TÀI   
COMPOSITE**

**BÁO CÁO**

**THIẾT KẾ PHẦN MỀM HƯỚNG ĐỐI TƯỢNG**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**🕯✡🕮🕮✡🕯**

**MỤC LỤC**

[1. Giới thiệu (Thắng) 1](#_Toc462)

[1.1. Định nghĩa 1](#_Toc12537)

[1.2. Công dụng 1](#_Toc23490)

[1.3. Composite trong OOP 1](#_Toc4470)

[2. Các thành phần trong Composite (Đăng) 1](#_Toc29023)

[2.1. Các thành phần 1](#_Toc9483)

[2.2. Mô hình UML và cấu trúc cây 2](#_Toc19583)

[3. Cách thức hoạt động của Composite (Đăng) 3](#_Toc6309)

[4. Đặt vấn đề và giải pháp 4](#_Toc3803)

[4.1. Đặt vấn đề (Thắng) 4](#_Toc30131)

[4.2. Giải pháp (Đăng) 6](#_Toc6256)

[5. Ví dụ về Composite 9](#_Toc20315)

[5.1. Ví dụ lý thuyết (Đăng) 9](#_Toc7780)

[5.2. Ví dụ thực tế (Thắng) 13](#_Toc8798)

[6. Ưu / Nhược điểm của Composite (Thắng) 24](#_Toc7643)

[6.1. Ưu điểm 24](#_Toc17682)

[6.2. Nhược điểm 24](#_Toc12947)

[7. Ứng dụng của Composite (Đăng) 24](#_Toc28690)

[8. Kết luận (Thắng) 26](#_Toc22337)

# Giới thiệu (Thắng)

## Định nghĩa

Composite là một mẫu thiết kế phân vùng và mô tả một nhóm đối tượng được xử lý giống như một phiên bản duy nhất của cùng một loại đối tượng.

## Công dụng

Mục đích của composite là "compose" các đối tượng thành các cấu trúc cây để thể hiện các hệ thống phân cấp một phần-toàn bộ. Nó cho phép bạn có cấu trúc cây và yêu cầu mỗi nút trong cấu trúc cây thực hiện một tác vụ.

Composite cho phép các đối tượng cùng chung một giao diện có thể được sử dụng như một đối tượng đơn giản.

## Composite trong OOP

Trong lập trình hướng đối tượng, một composite là một đối tượng được thiết kế như là một thành phần của một hoặc nhiều đối tượng tương tự, tất cả đều thể hiện chức năng tương tự. Đây được gọi là mối quan hệ “có-một” giữa các đối tượng.

# Các thành phần trong Composite (Đăng)

## Các thành phần

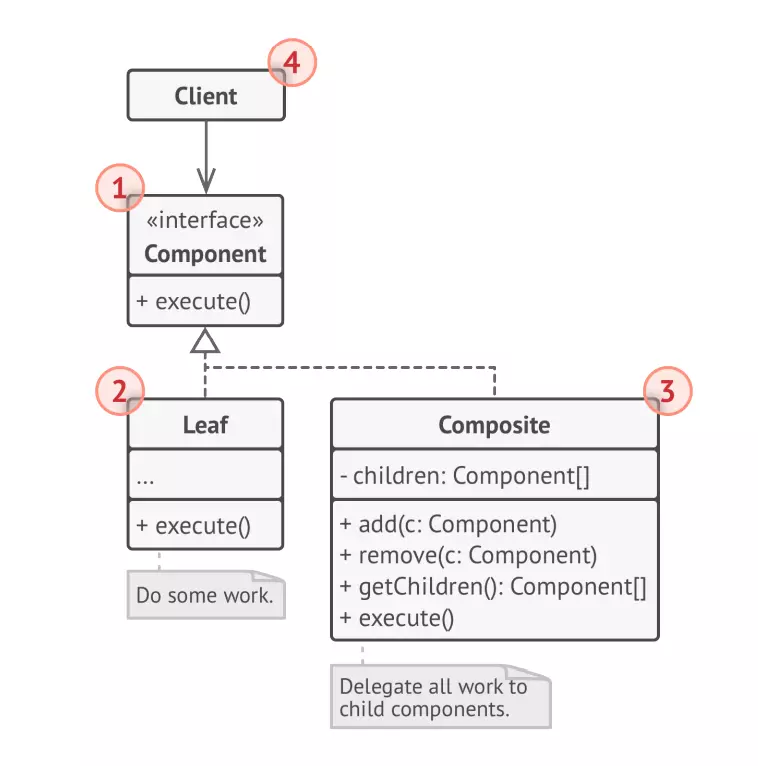
Component: là một interface hoặc lớp trừu tượng đại diện cho các thành phần trong cấu trúc Composite.

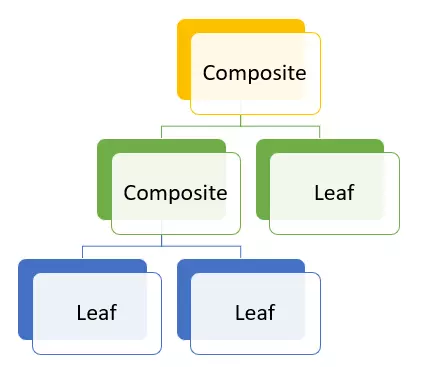
Leaf: là các đối tượng đơn giản, không có các thành phần con.

Composite: là các đối tượng phức tạp được tạo thành bằng cách kết hợp các thành phần con (bao gồm các đối tượng Leaf và Composite khác).

Client: thao tác với các đối tượng trong thành phần thông qua Component interface.

## Mô hình UML và cấu trúc cây





# Cách thức hoạt động của Composite (Đăng)

Tạo ra một giao diện (interface) chung cho tất cả các đối tượng trong cấu trúc, bao gồm cả đối tượng lá và đối tượng cha.

Tạo ra một lớp cha (base class) để định nghĩa các hành vi chung của các đối tượng trong cấu trúc.

Tạo ra một lớp đối tượng lá (leaf class) để đại diện cho các đối tượng lá trong cấu trúc. Lớp này sẽ implement các hành vi cụ thể của đối tượng lá. Khi duyệt đến một thành phần Leaf, nó sẽ thực hiện một hành động nào đó trên đối tượng đó.

Tạo ra một lớp đối tượng cha (composite class) để đại diện cho các đối tượng cha trong cấu trúc. Lớp này sẽ bao gồm các đối tượng con (sub-objects) của nó, có thể là các đối tượng lá hoặc các đối tượng cha khác. Lớp này sẽ implement các phương thức để thêm, xoá hoặc truy cập các đối tượng con của nó.

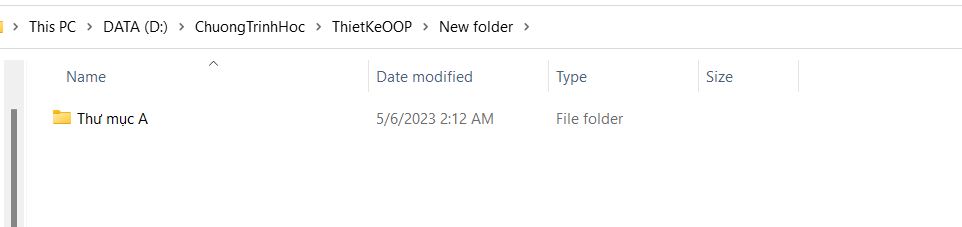
Trong lớp đối tượng cha, các phương thức sẽ gọi đệ quy để xử lý tất cả các đối tượng con của nó, bao gồm cả các đối tượng cha khác.

# Đặt vấn đề và giải pháp

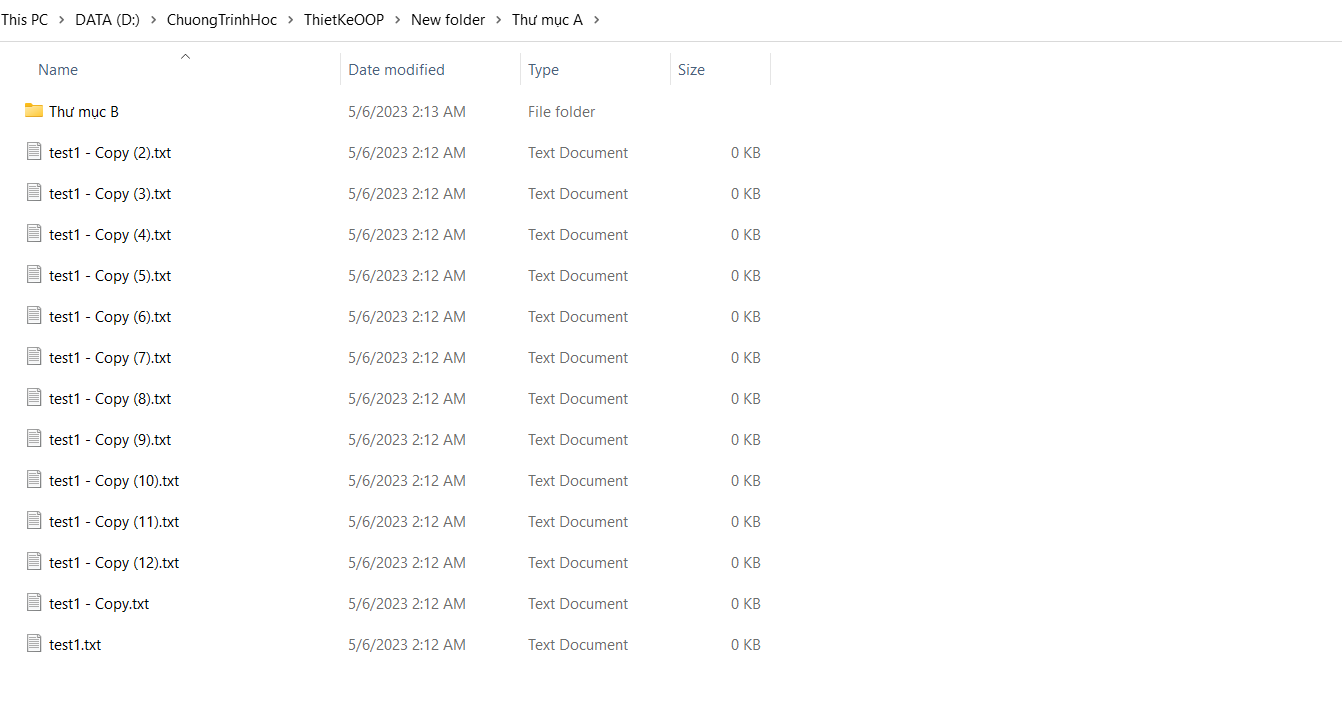
## Đặt vấn đề (Thắng)

Khi chưa sử dụng phương pháp composite trong lập trình hướng đối tượng:

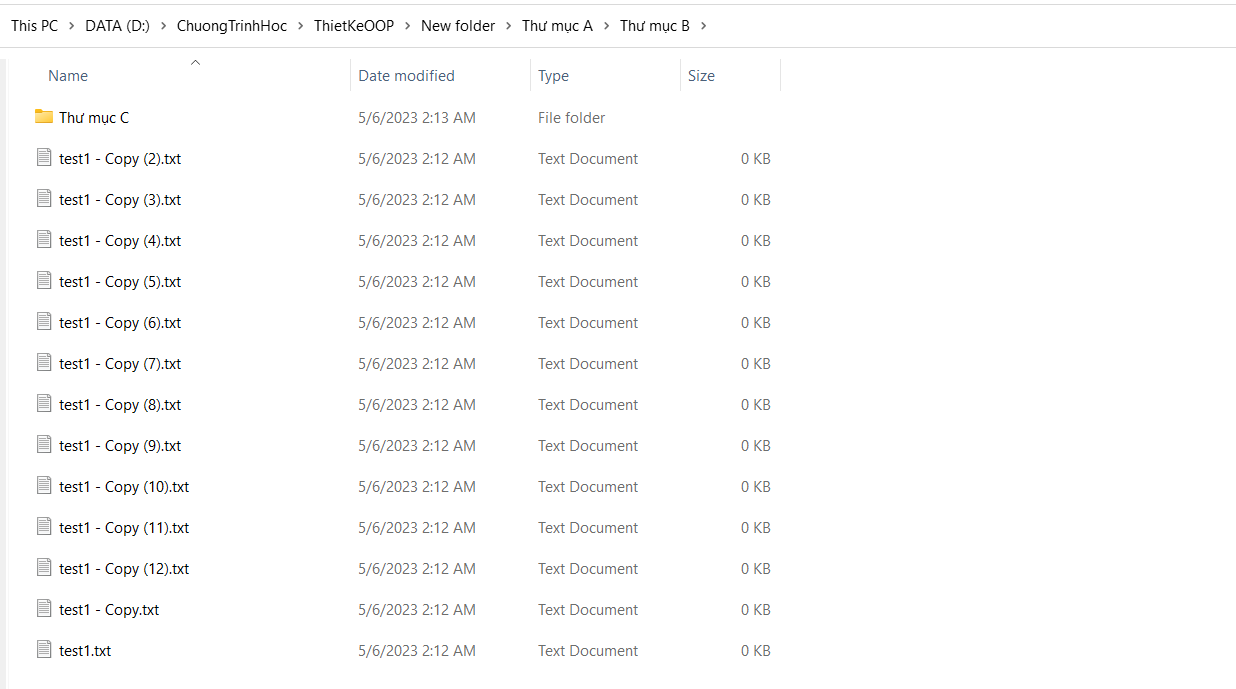
Giả sử ta muốn quản lý thư mục A



Trong thư mục A này có 100 file và có thêm 1 thư mục B

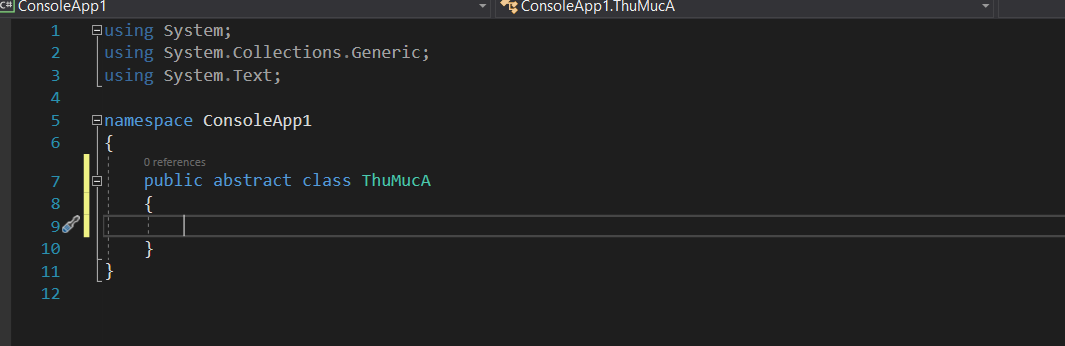


Tiếp đến trong thư mục B có 50 file (tiếp tục giả sử lại có tiếp tục thư mục C trong thư mục B và trong C lại có nhiều file khác).

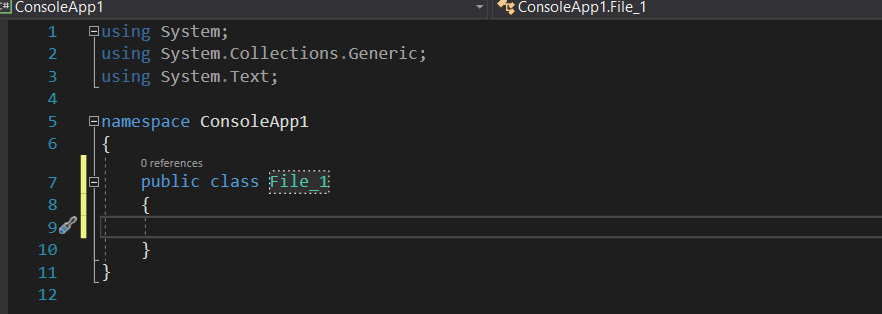


Tiến hành code:

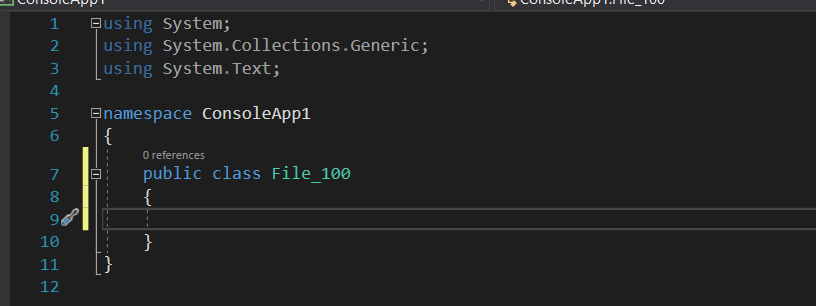
Tạo class ThuMucA



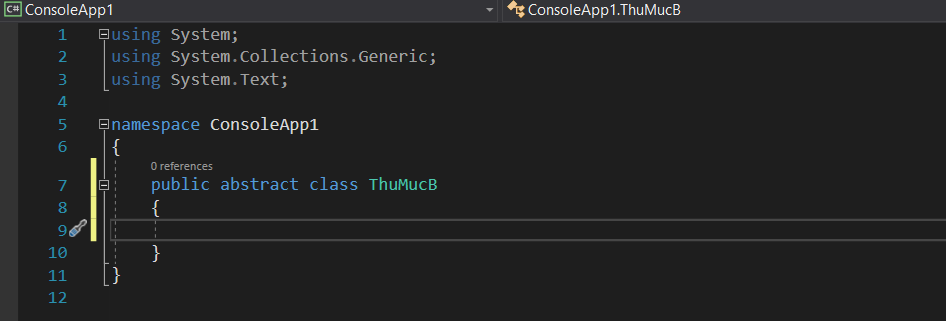
Tạo File\_1



Tiếp tục tạo đến File\_100



Và tạo tiếp ThuMucB

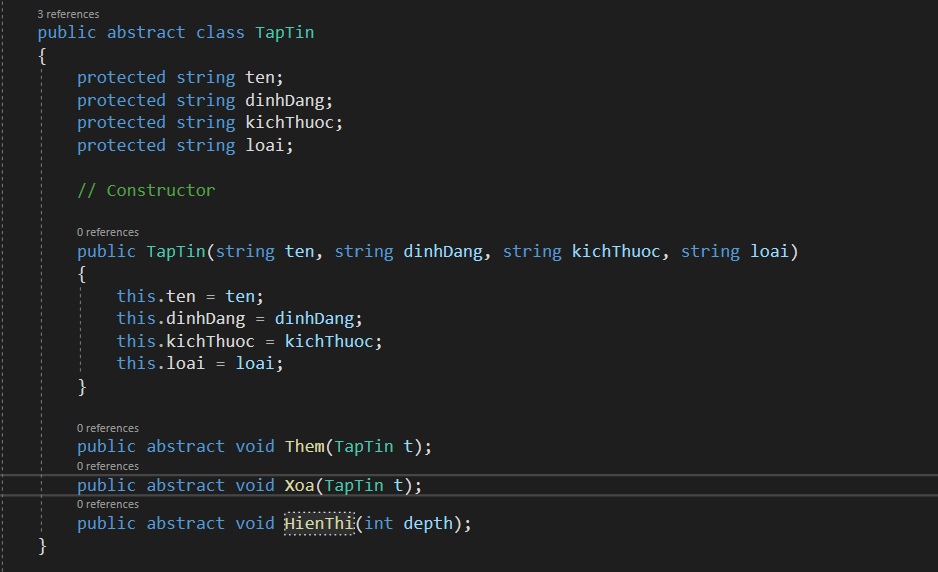


Vấn đề: nếu như vậy ta muốn có 1000 file và trong mỗi thư mục lại có 100 thư mục khác thì sao? Ta phải liên tục tạo class và nếu trong 1 class bất kỳ có lỗi thì ta phải tìm class đó là fix lỗi và class đó liên qua tới tất cả class còn lại thì ta phải fix tất cả các class đó.

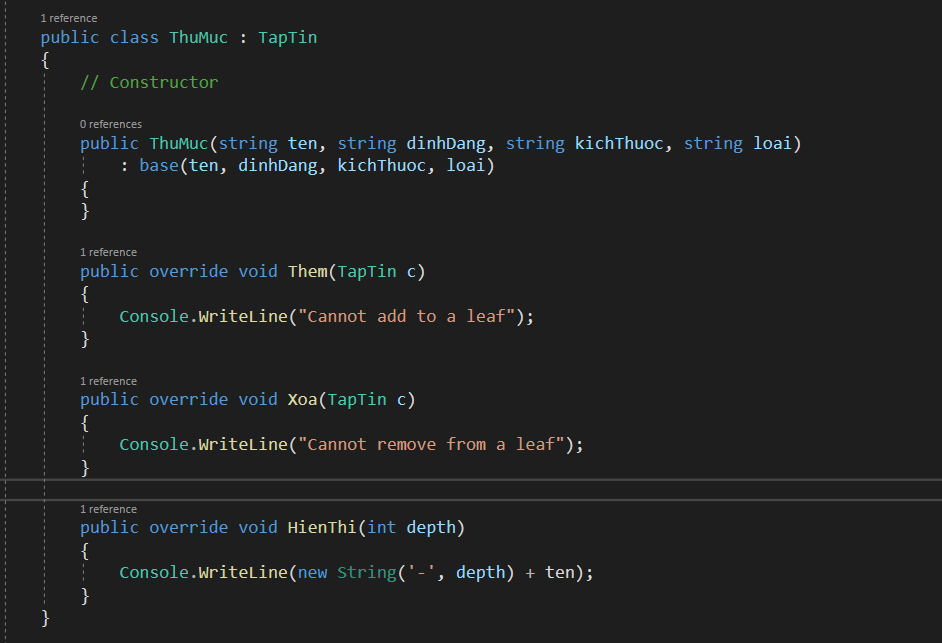
## Giải pháp (Đăng)

Khi sử dụng phương pháp composite trong lập trình hướng đối tượng:

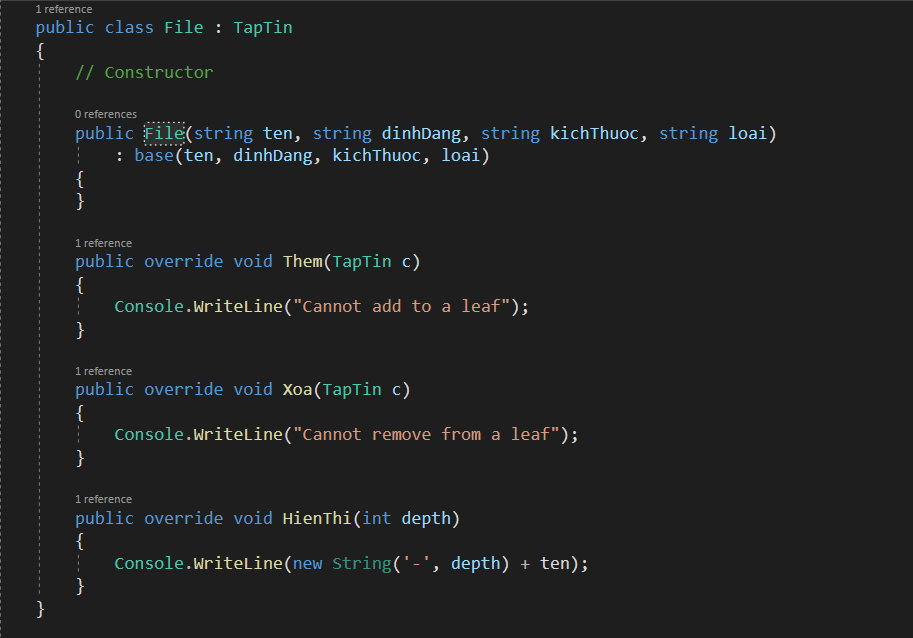
Tạo một component TapTin



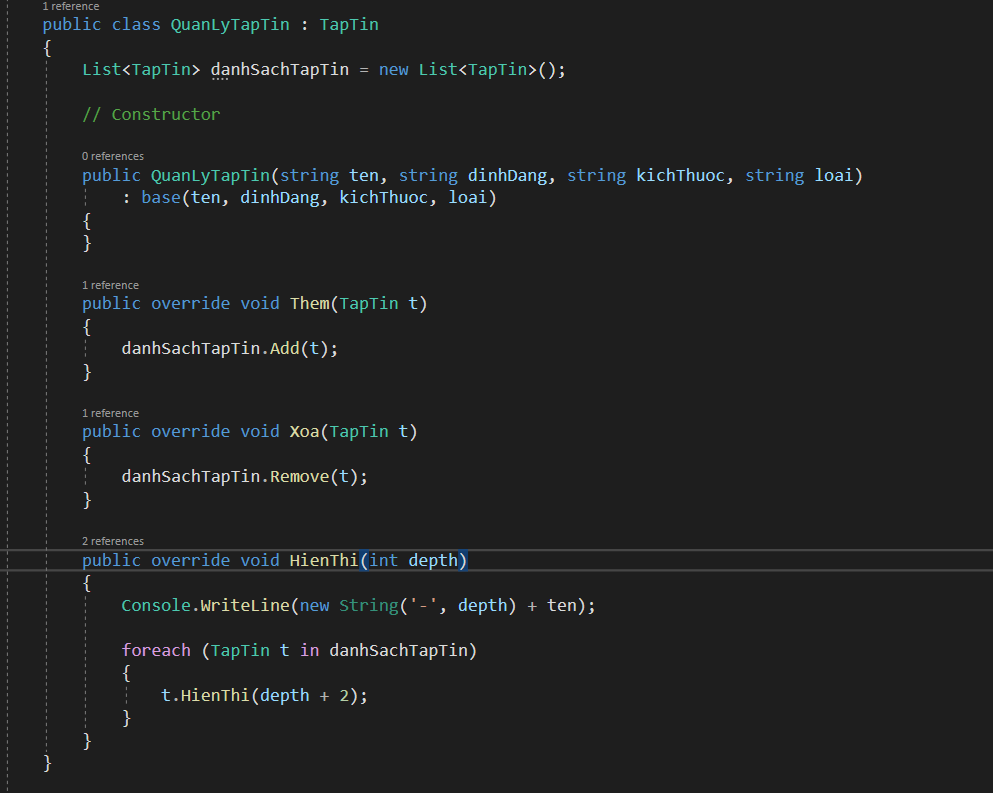
Tạo Leaf ThuMuc



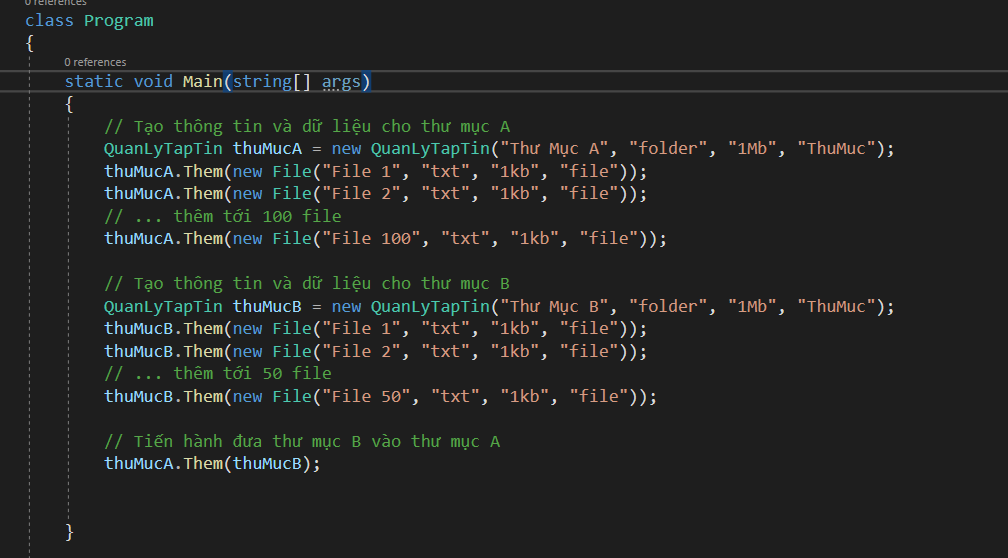
Tạo Leaf File



Tạo Composite QuanLyTapTin



Ở Client ta gọi là người truy cập thư mục



Như vậy vấn đề đã được giải quyết và ta không cần phải tạo nhiều class và nếu lỗi ta chỉ cần truy cập đến tập tin đó để sửa.

# Ví dụ về Composite

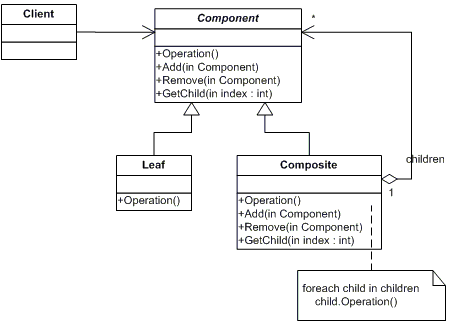
Giả sử chúng ta có một cấu trúc cây được tạo thành bởi các đối tượng Node và Leaf, trong đó Node là các đối tượng Composite và Leaf là các đối tượng đơn giản.

Chúng ta có thể duyệt qua toàn bộ cấu trúc cây bằng cách gọi phương thức tương ứng trên nút gốc của cây.

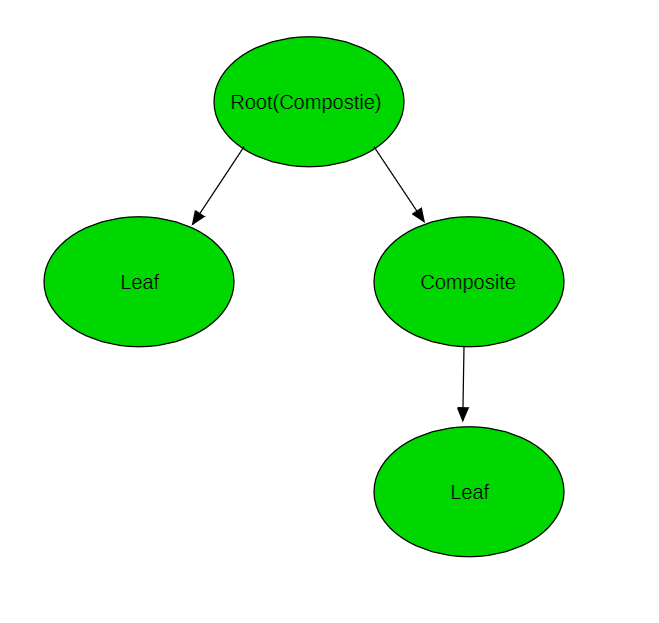
Như vậy, chúng ta có thể thực hiện các thao tác như tìm kiếm, thêm, xóa trên cấu trúc cây một cách dễ dàng.

## Ví dụ lý thuyết (Đăng)

Mô hình UML:

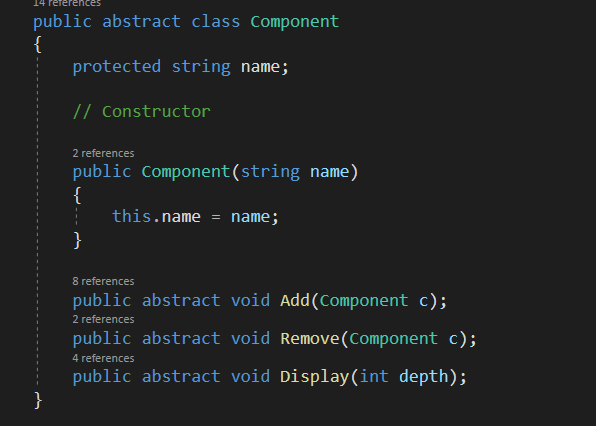


Cấu trúc cây:

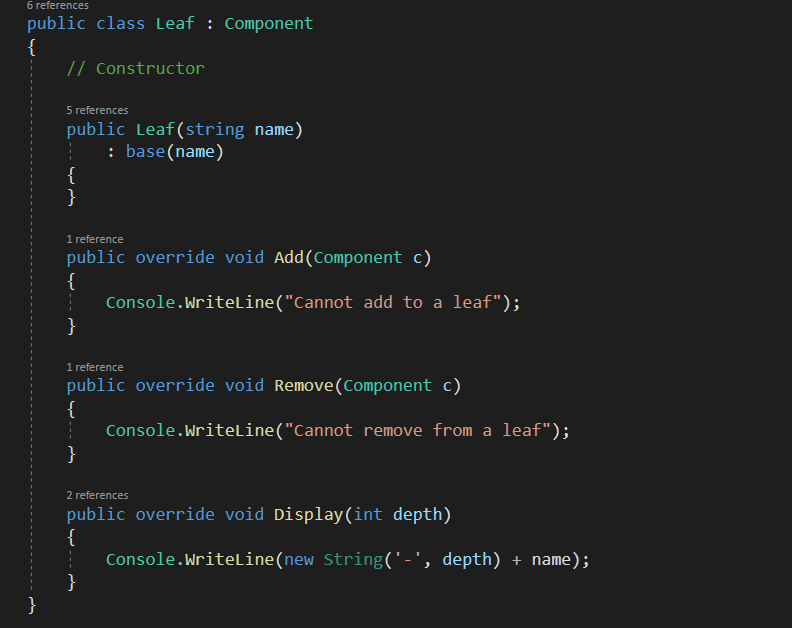


Code:

Tạo component



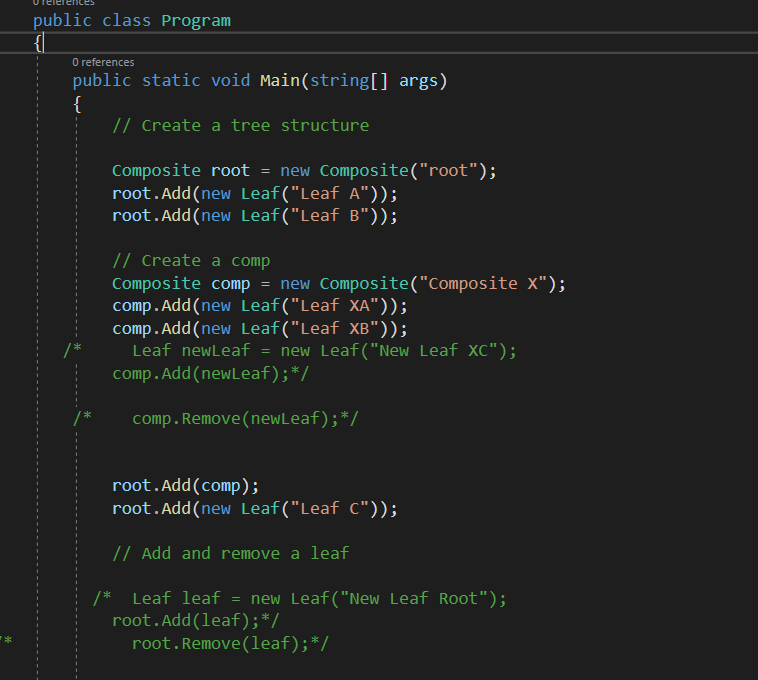
Tạo Leaf



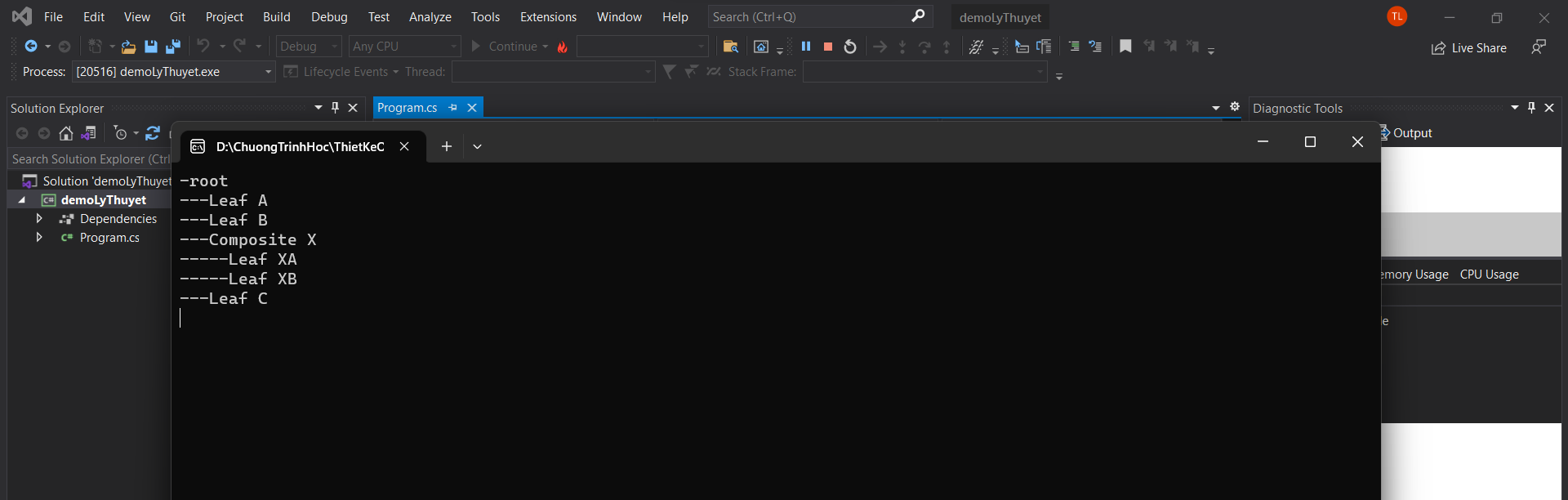
Tạo Composite



Thêm Leaf và sử dụng các chức năng thêm, xóa

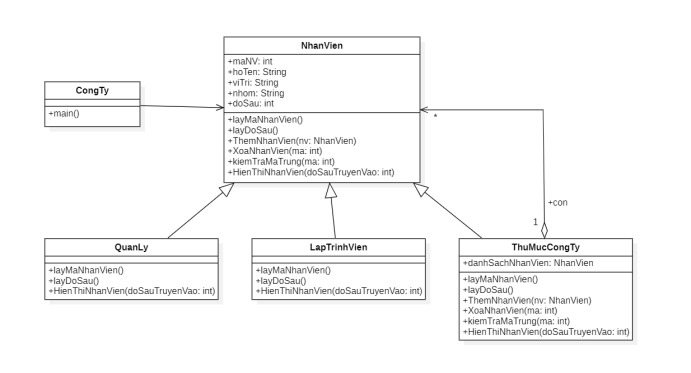


Kết quả

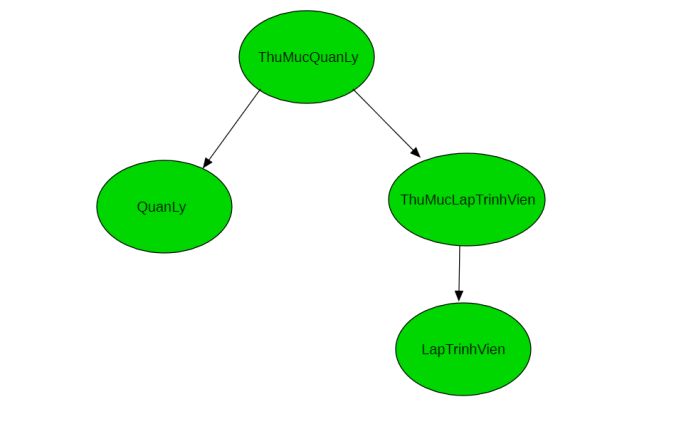


## Ví dụ thực tế (Thắng)

Mô hình UML:

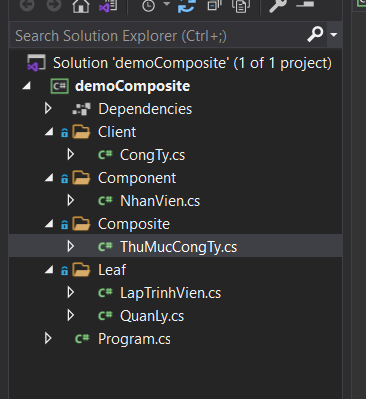


Cấu trúc cây:

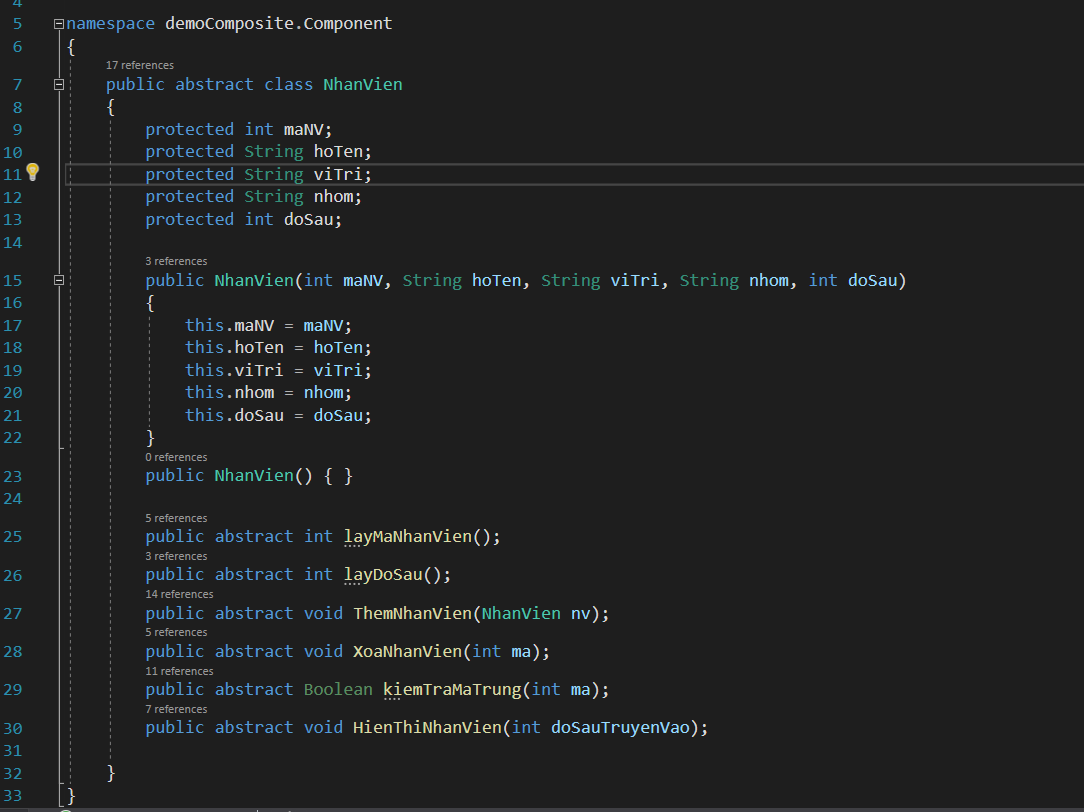


Code:

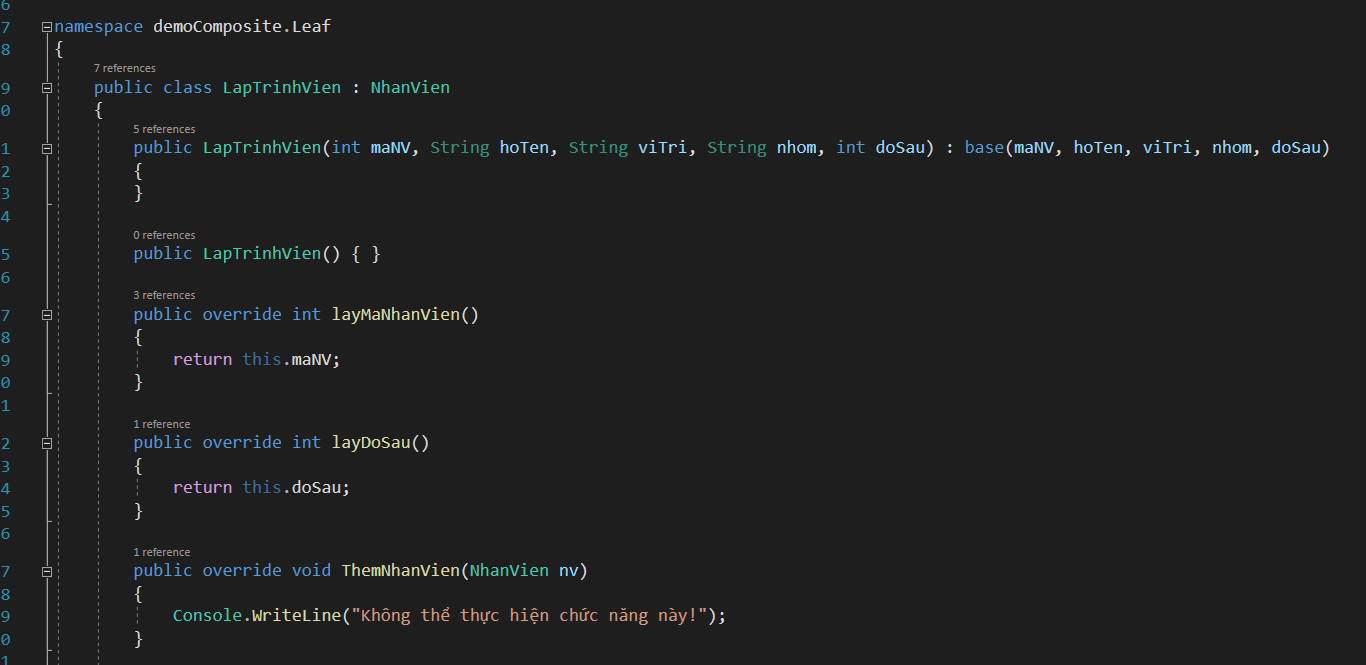
Cấu trúc project

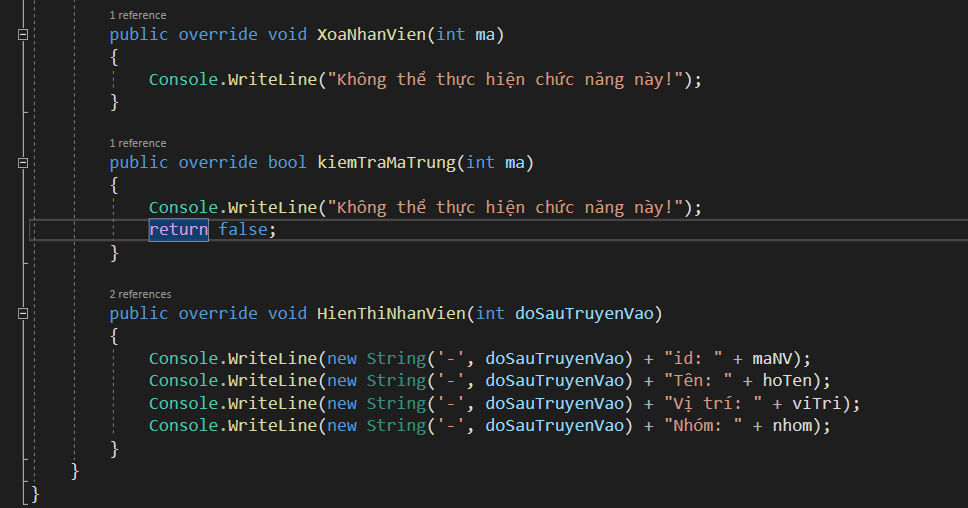


Tạo component NhanVien

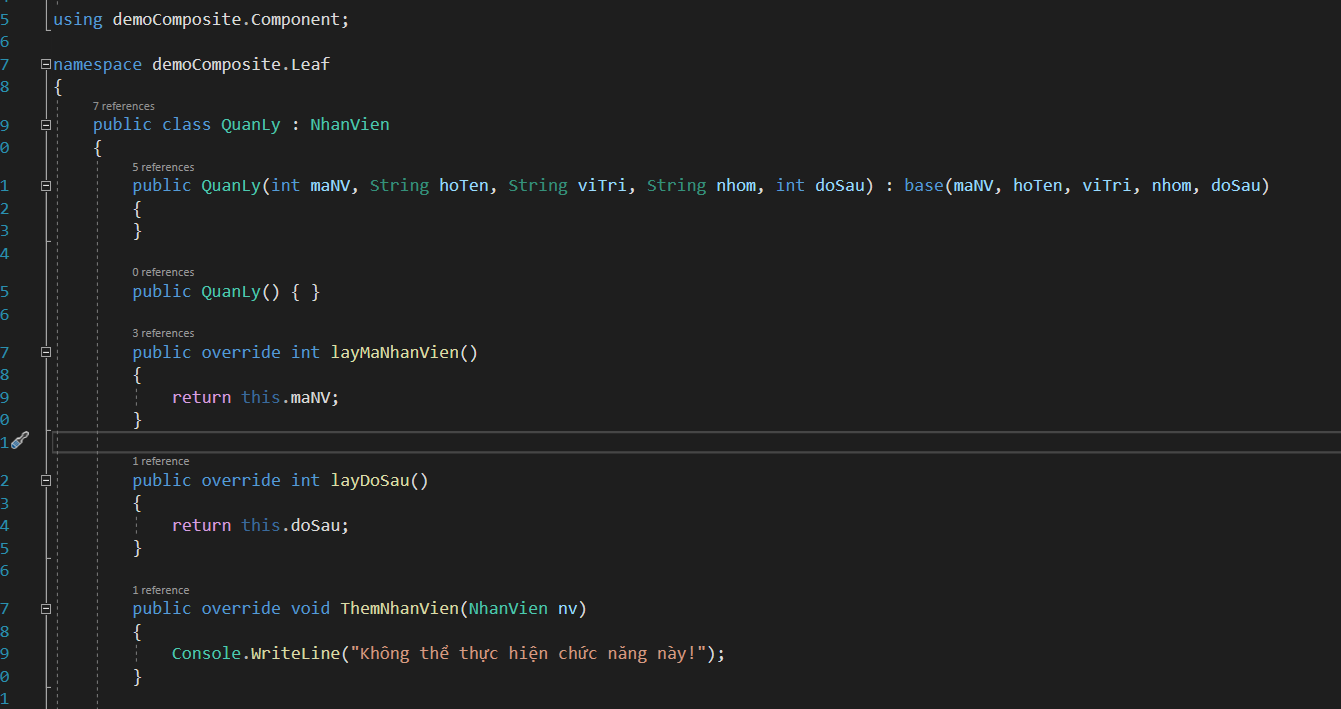


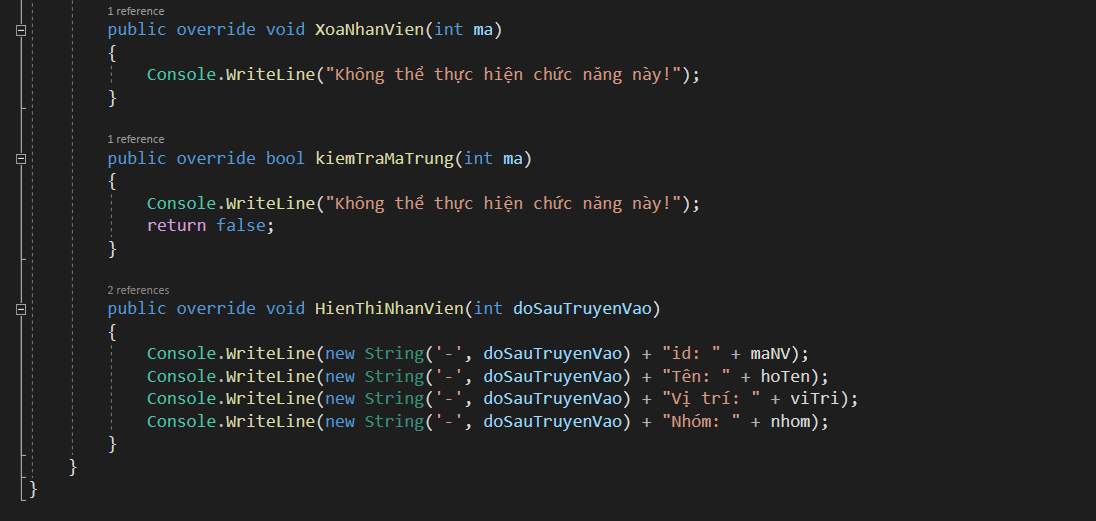
Tạo Leaf LapTrinhVien



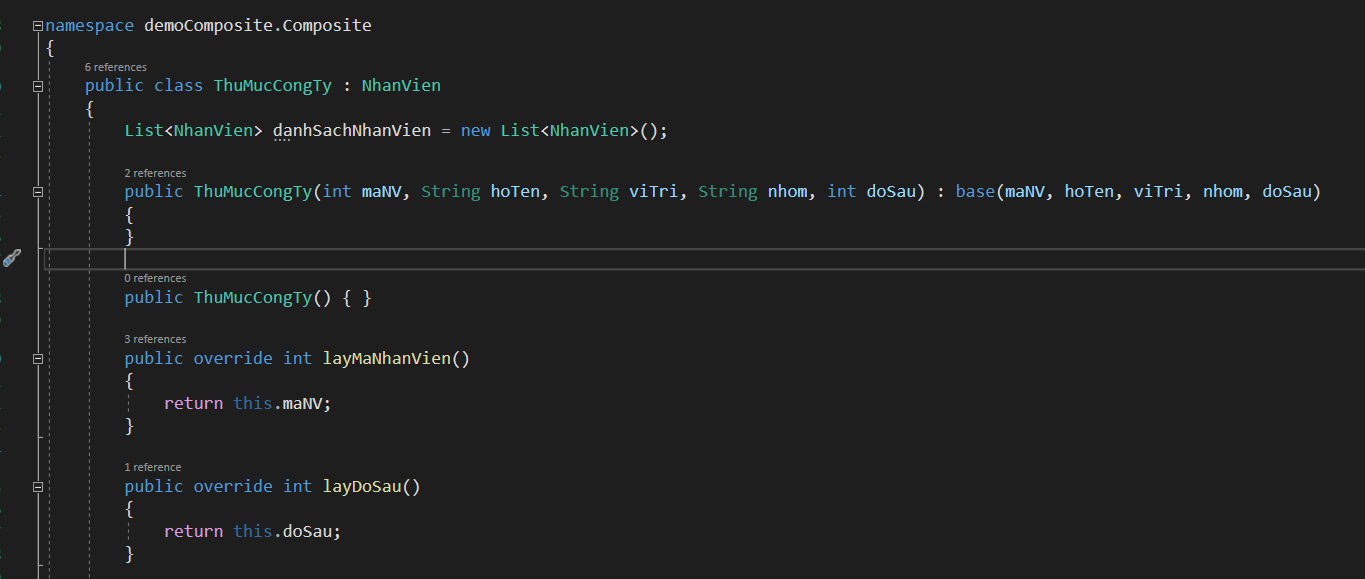


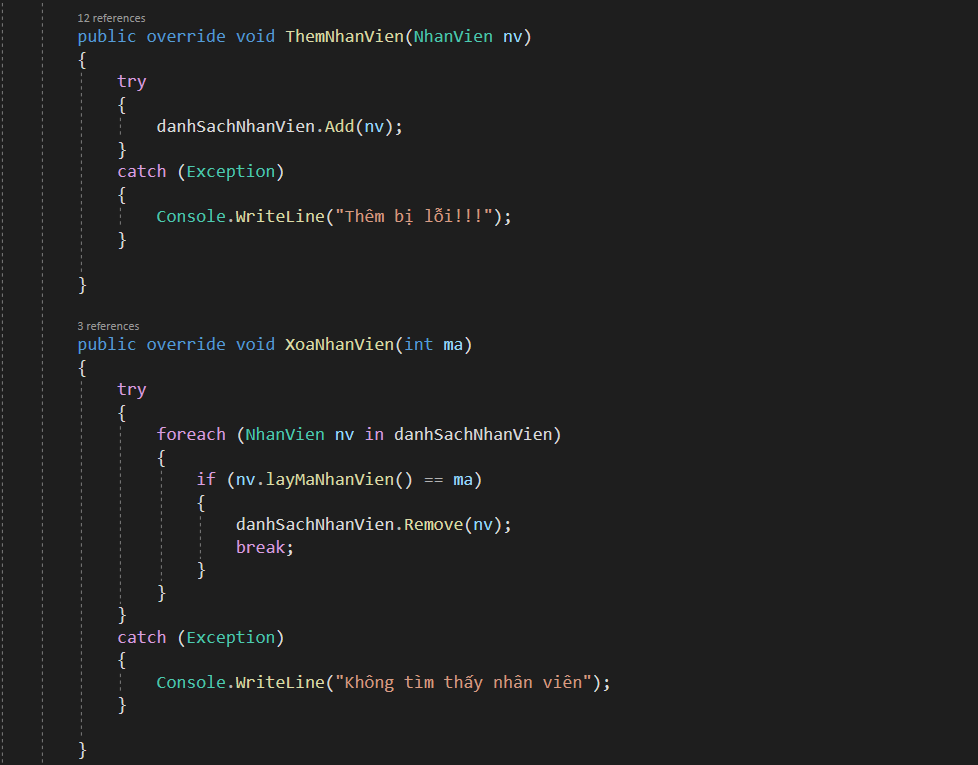
Tạo Leaf QuanLy

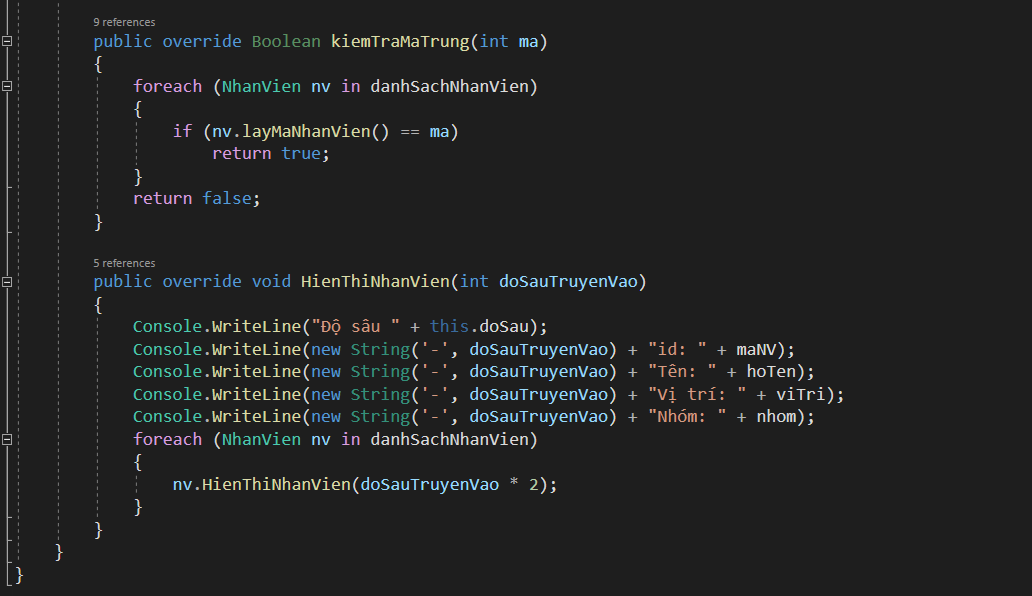




Tạo composite ThuMucCongTy





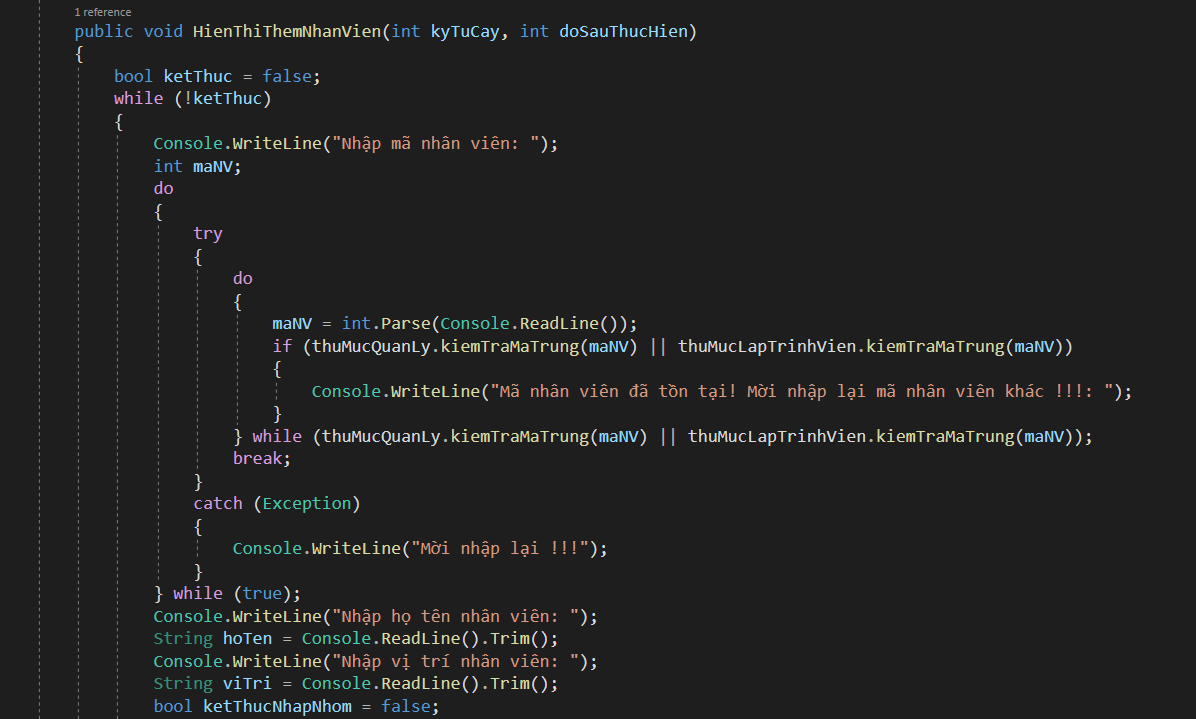


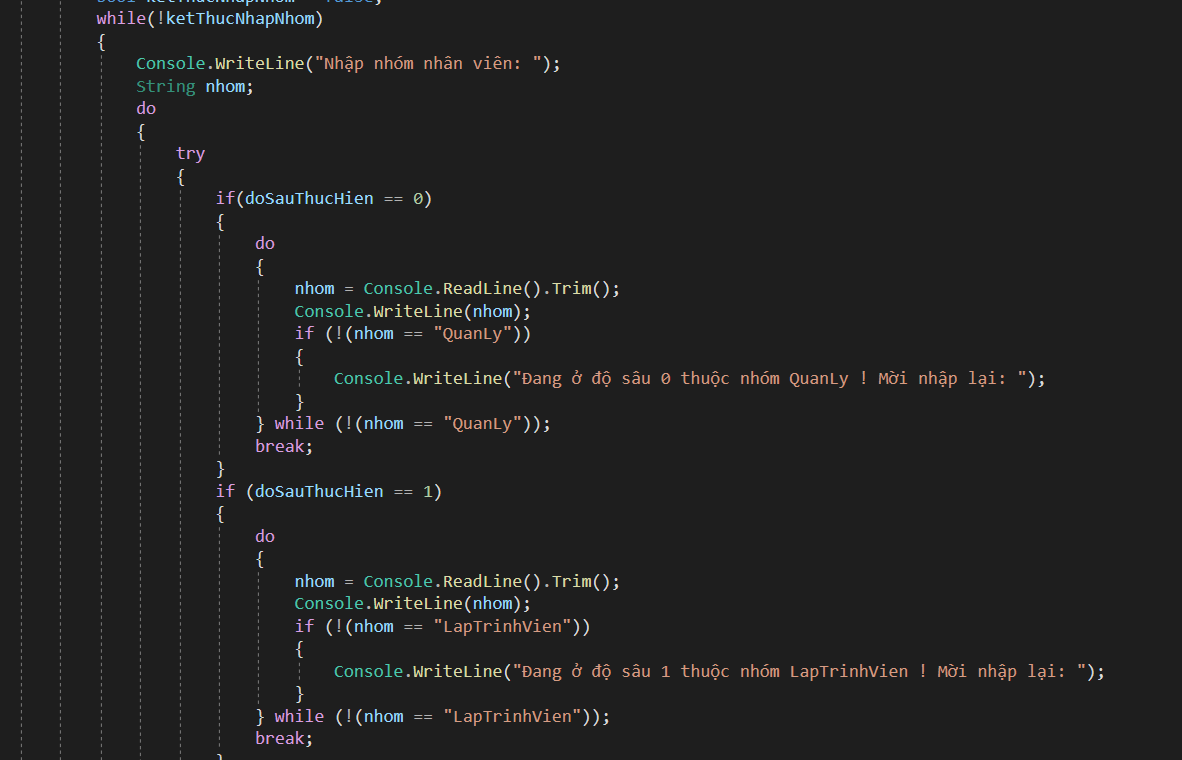
Ở Client CongTy sẽ tương tác giao diện

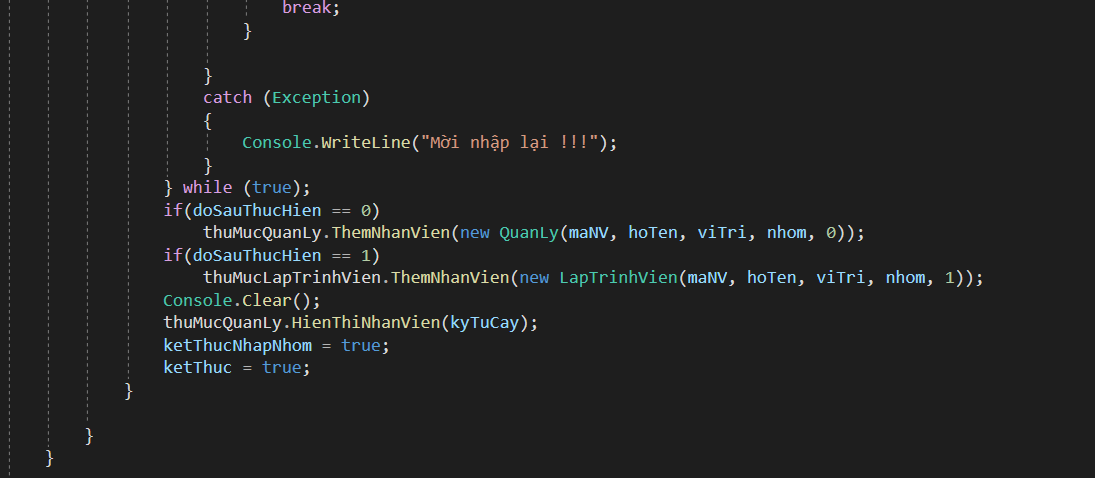
Thêm dữ liệu cho công ty



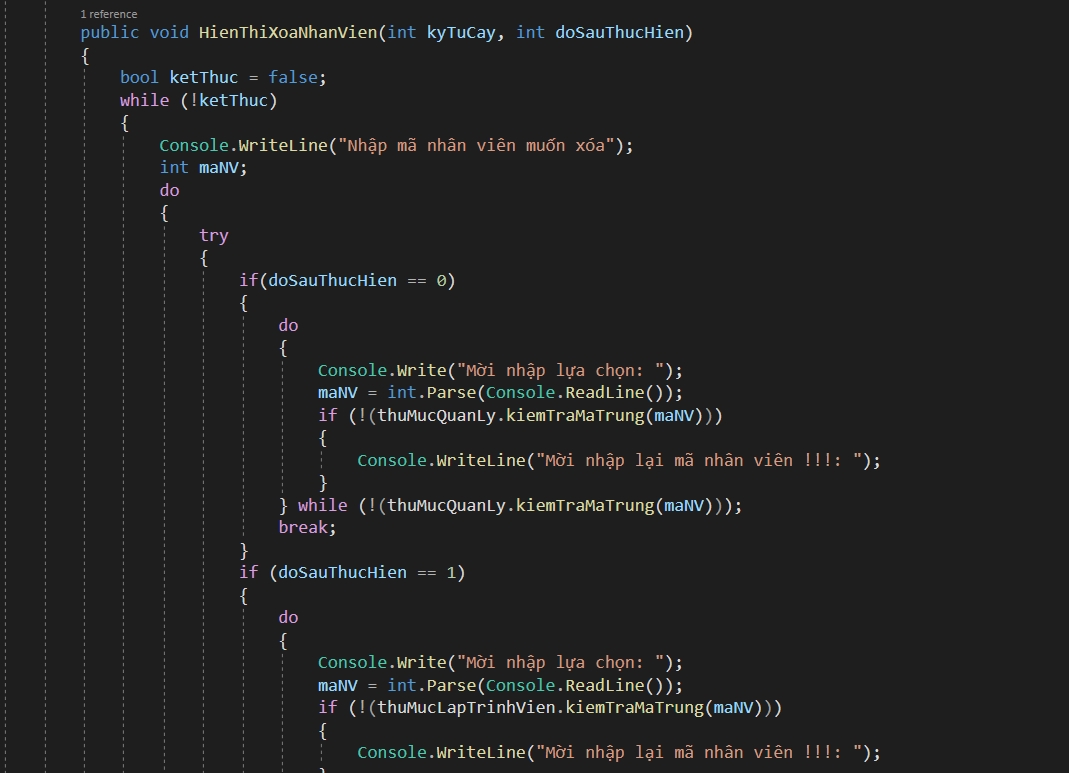
Code giao diện cho hiển thị thêm nhân viên





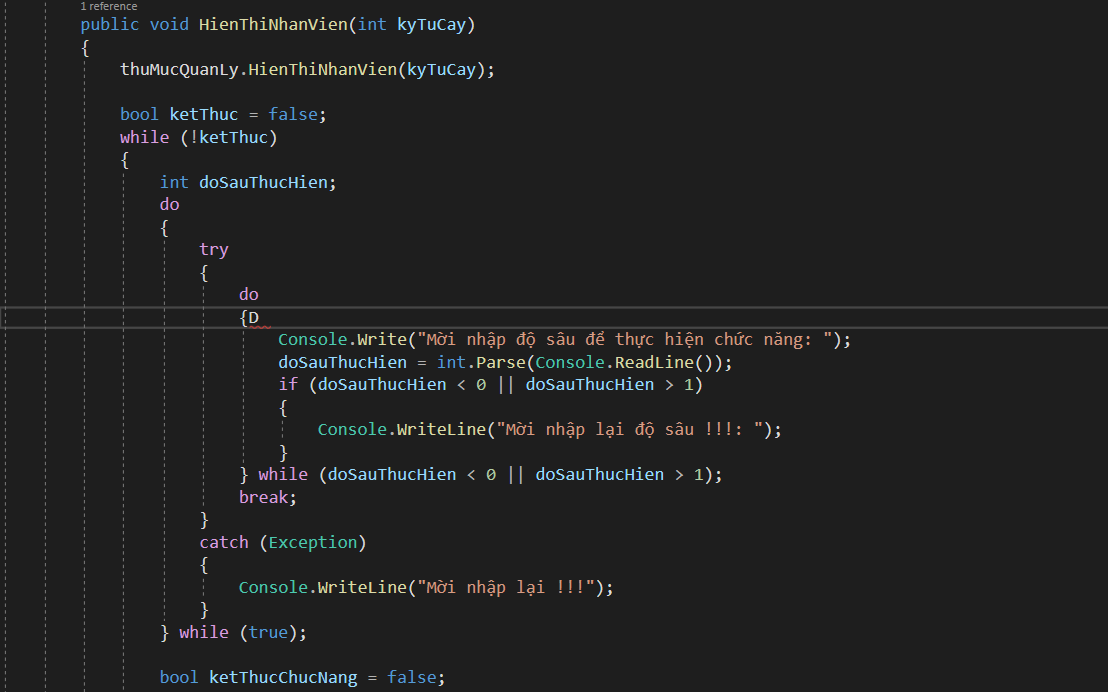


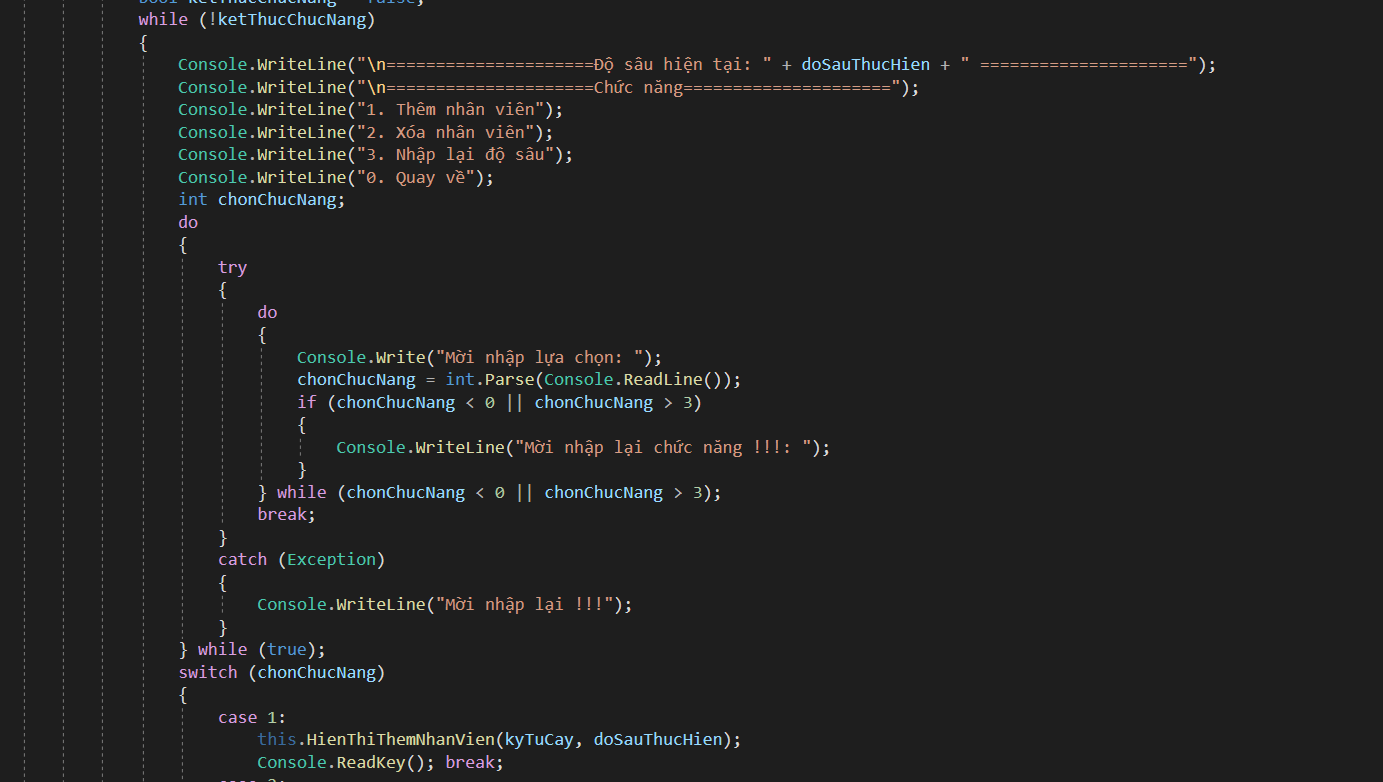
Code giao diện cho hiển thị xóa nhân viên

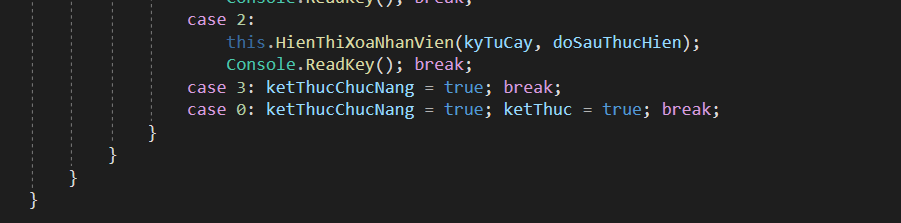




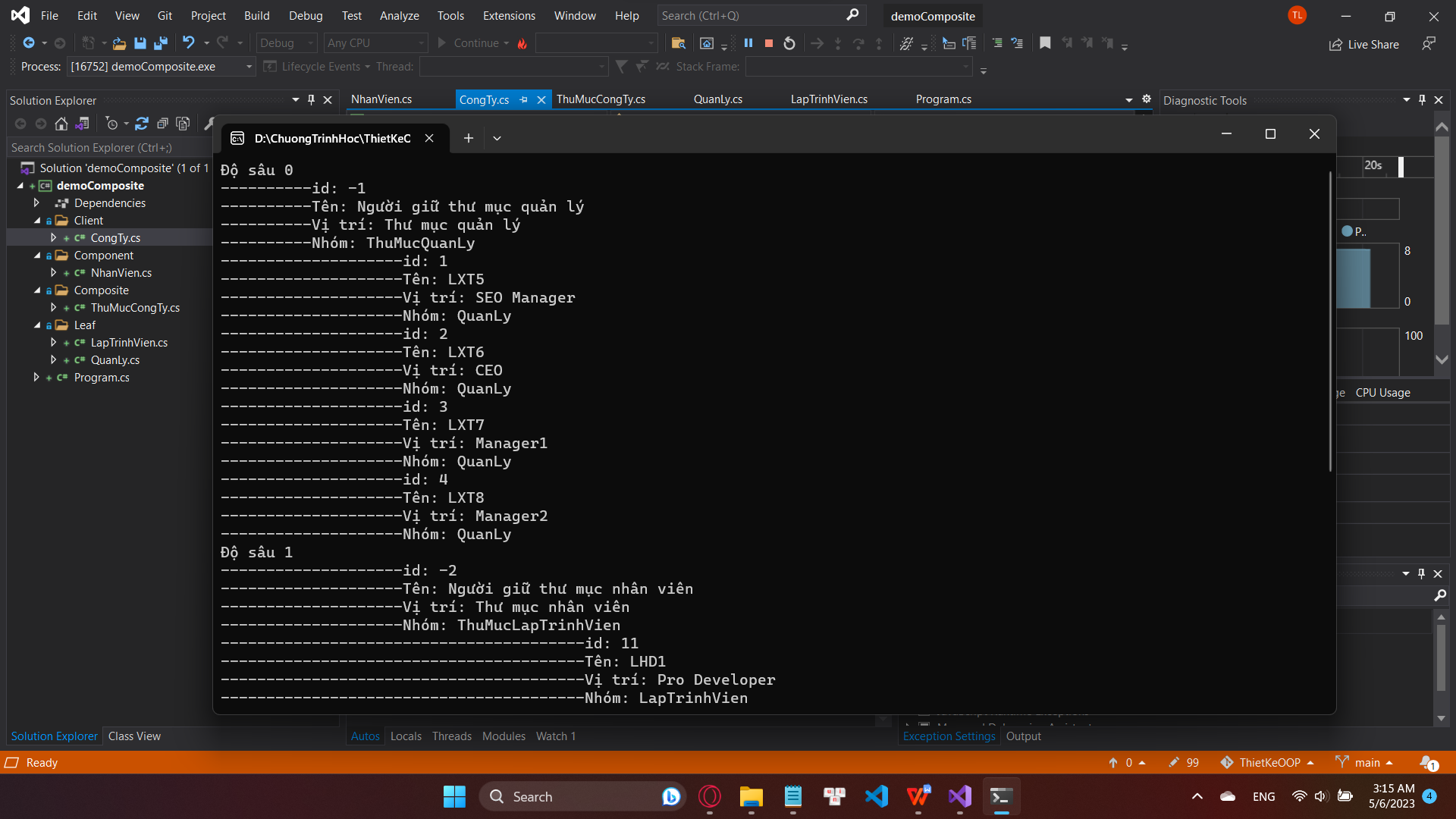
Code giao diện cho hiển thị nhân viên

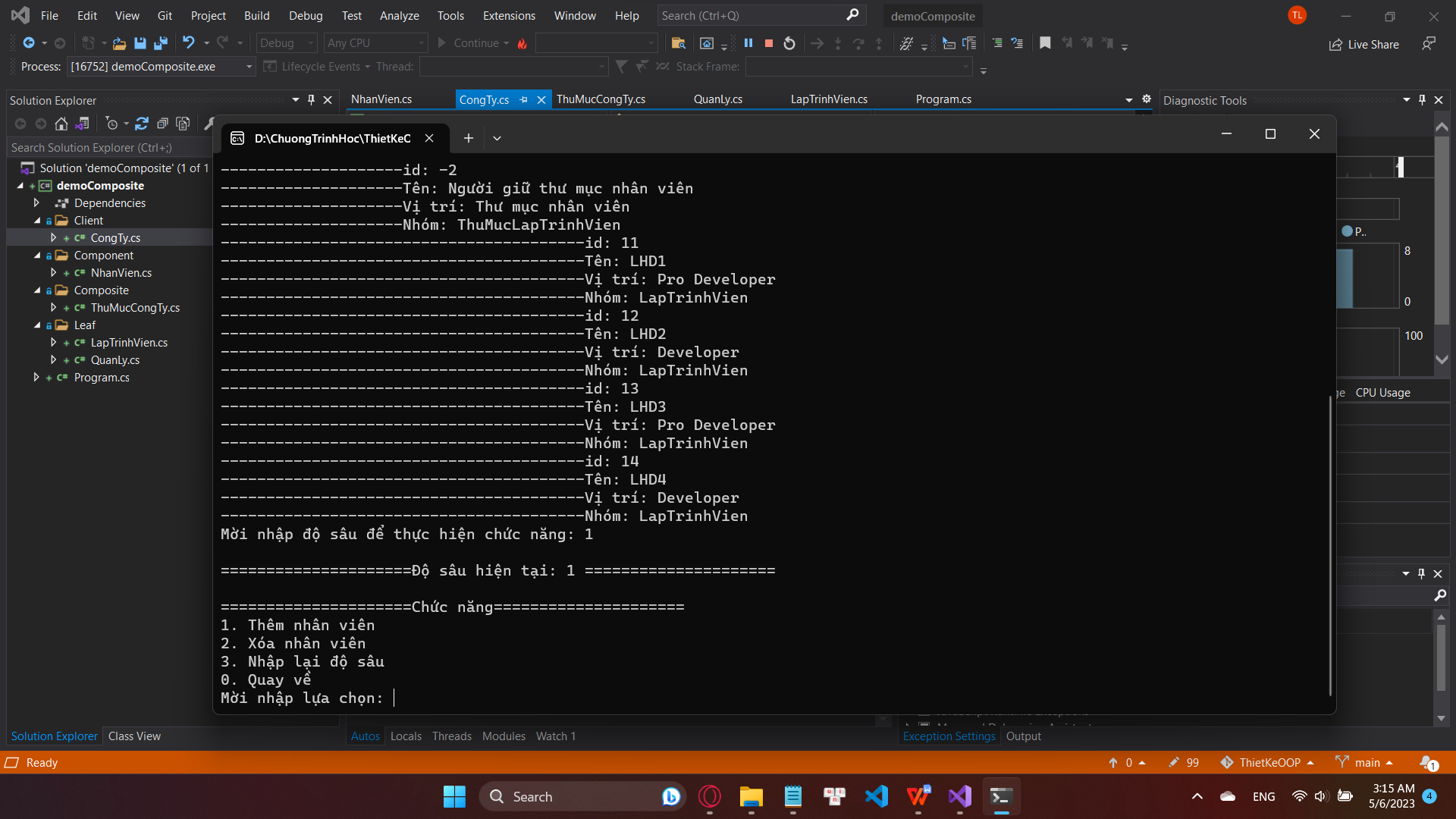






Kết quả:





# Ưu / Nhược điểm của Composite (Thắng)

## Ưu điểm

Composite cho phép xây dựng các cấu trúc phân cấp đối tượng phức tạp một cách dễ dàng và linh hoạt.

Nó cung cấp cho người lập trình một cách tiếp cận đơn giản và đồng nhất để xử lý các đối tượng phức tạp.

Khi thêm/xóa một đối tượng từ cấu trúc phân cấp, không cần thay đổi các đối tượng khác trong cấu trúc.

Composite cho phép người lập trình truy cập vào các thành phần của cấu trúc phân cấp dưới dạng một đối tượng duy nhất hoặc một tập hợp các đối tượng.

## Nhược điểm

Composite có thể làm cho mã của bạn trở nên phức tạp hơn nếu không được sử dụng đúng cách.

Nó có thể gây ra sự chậm trễ trong thời gian chạy nếu cấu trúc phân cấp của bạn có quá nhiều đối tượng hoặc nhiều cấp độ.

Composite cũng có thể gây ra sự phân tán quá mức, khiến cho việc bảo trì và sửa lỗi trở nên khó khăn hơn.

Nếu không được thiết kế đúng cách, composite có thể dẫn đến việc lặp lại mã và gây ra các vấn đề về hiệu suất.

# Ứng dụng của Composite (Đăng)

Mẫu thiết kế này giúp chúng ta dễ dàng thêm, xóa, sửa các thành phần trong cấu trúc mà không cần phải quan tâm đến cấu trúc tổng thể.

Composite cũng giúp chúng ta tăng tính linh hoạt và tái sử dụng mã nguồn.

Các ví dụ:

* Quản lý cấu trúc cây đa cấp: Composite pattern cho phép tạo ra một cấu trúc cây đa cấp các đối tượng, cho phép quản lý các đối tượng phức tạp và tương tác với chúng một cách dễ dàng. Ví dụ, trong các ứng dụng quản lý tài liệu, Composite pattern có thể được sử dụng để tạo ra một cây thư mục chứa các tệp tin và thư mục con.
* Xử lý đồ họa: Composite pattern có thể được sử dụng để tạo ra các đối tượng đồ họa phức tạp, bao gồm các hình dạng đơn giản và các hình dạng phức tạp hơn, và cho phép xử lý chúng một cách dễ dàng. Ví dụ, trong các ứng dụng vẽ, Composite pattern có thể được sử dụng để tạo ra một hình dạng bao gồm nhiều hình dạng con, và cho phép thực hiện các thao tác như di chuyển, xoay hoặc thay đổi kích thước của toàn bộ hình dạng đó.
* Quản lý danh sách các đối tượng: Composite pattern cũng có thể được sử dụng để quản lý danh sách các đối tượng, cho phép xử lý chúng một cách dễ dàng và linh hoạt. Ví dụ, trong các ứng dụng quản lý danh sách liên lạc, Composite pattern có thể được sử dụng để tạo ra một danh sách các liên lạc, bao gồm các liên lạc đơn giản và các nhóm liên lạc, và cho phép thực hiện các thao tác như thêm, xoá hoặc sửa đổi các liên lạc đó.
* Xử lý sự kiện: Composite pattern cũng có thể được sử dụng để xử lý các sự kiện phức tạp, bao gồm các sự kiện kết hợp từ nhiều nguồn khác nhau. Ví dụ, trong các ứng dụng quản lý sản xuất, Composite pattern có thể được sử dụng để xử lý các sự kiện phức tạp, bao gồm các thông tin từ nhiều nguồn khác nhau như máy móc, thiết bị và nhân viên.

# Kết luận (Thắng)

Composite là một mẫu thiết kế rất hữu ích trong lập trình hướng đối tượng. Nó cho phép chúng ta xây dựng các cấu trúc phức tạp bằng cách kết hợp các đối tượng đơn giản với nhau.

Composite giúp tăng tính linh hoạt và tái sử dụng mã nguồn, đồng thời giúp chúng ta dễ dàng thêm, xóa, sửa các thành phần trong cấu trúc mà không cần phải quan tâm đến cấu trúc tổng thể.