**Đề ôn tập 2:Tính đóng gói, kế thừa, đa hình**

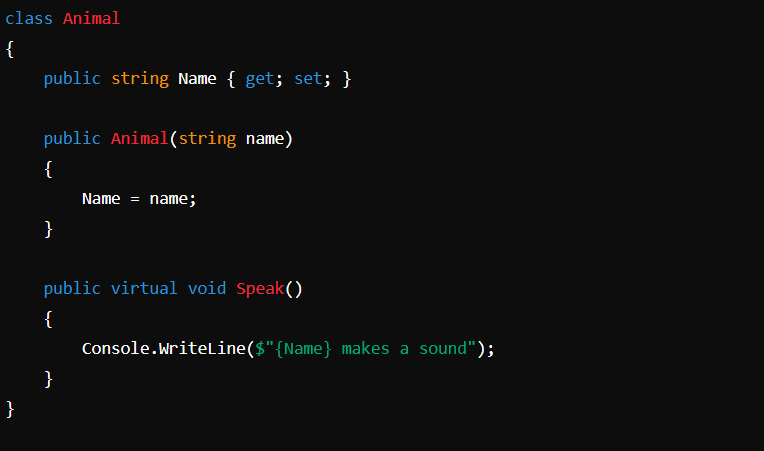
Yêu cầu trả lời trực tiếp kết quả vào file word và nộp

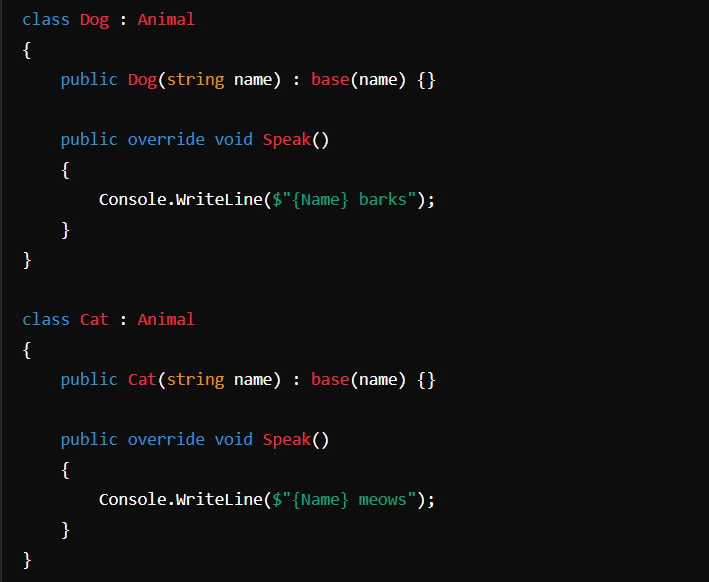
1. Định nghĩa tính kế thừa trong lập trình hướng đối tượng và cho biết lợi ích của nó. Cho ví dụ minh họa.

Tính kế thừa là một khái niệm trong lập trình hướng đối tượng (OOP), cho phép một lớp (class) kế thừa các thuộc tính (properties) và phương thức (methods) từ một lớp khác. Lớp kế thừa được gọi là lớp con (subclass) hoặc lớp dẫn xuất (derived class), trong khi lớp bị kế thừa được gọi là lớp cha (superclass) hoặc lớp cơ sở (base class).

**Mục đích:**

* **Tái sử dụng mã (Code Reusability):** Giúp tái sử dụng mã nguồn, giảm thiểu việc viết lại mã và tăng tính tái sử dụng.
* **Mở rộng tính năng (Extend Functionality):** Cho phép mở rộng và tùy biến các tính năng của lớp cơ sở mà không làm ảnh hưởng đến lớp cơ sở.
* **Duy trì và bảo trì (Maintenance):** Giúp quản lý, bảo trì và cập nhật mã nguồn dễ dàng hơn, vì các thay đổi trong lớp cơ sở sẽ tự động áp dụng cho tất cả các lớp con.
* **Đa hình (Polymorphism):** Cho phép một phương thức có thể có nhiều dạng khác nhau, cung cấp khả năng linh hoạt và dễ mở rộng của chương trình.





1. Mô tả các tính năng của interface và sự khác nhau giữa interface và abstract class. Cho ví dụ minh họa.

### Tính năng của Interface trong C#

**Interface** là một tập hợp các khai báo phương thức (methods), thuộc tính (properties), sự kiện (events), và chỉ mục (indexers) mà không có phần thân. Các lớp hoặc cấu trúc (structs) thực hiện (implement) interface phải cung cấp phần thân cho tất cả các thành viên của interface.

**Tính năng của Interface:**

1. **Định nghĩa nhưng không triển khai:** Interface chỉ cung cấp các khai báo mà không cung cấp phần thân cho các phương thức hoặc các thành viên khác. Các lớp thực hiện interface phải cung cấp phần thân cho các phương thức này.
2. **Không có trường (fields):** Interface không thể chứa các trường, cả tĩnh lẫn không tĩnh.
3. **Hỗ trợ đa kế thừa:** Một lớp hoặc một cấu trúc có thể thực hiện nhiều interface.
4. **Thúc đẩy tính đa hình (Polymorphism):** Interface cho phép các đối tượng của các lớp khác nhau có thể được xử lý thông qua một interface chung.
5. **Không có constructor:** Interface không thể có constructor.

### Sự khác nhau giữa Interface và Abstract Class

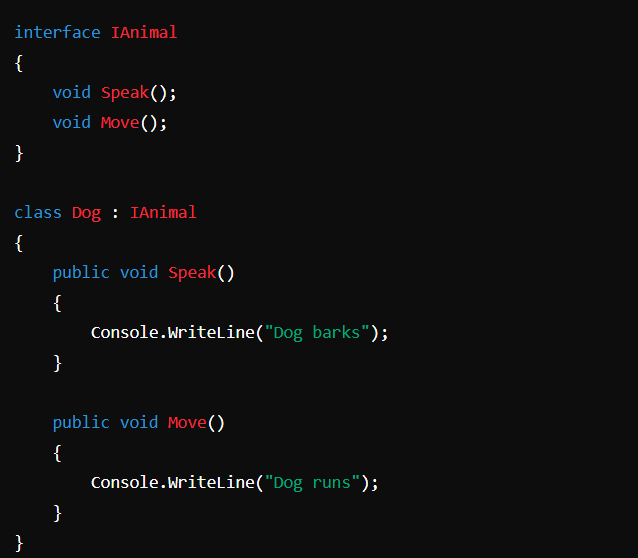
**Abstract Class:**

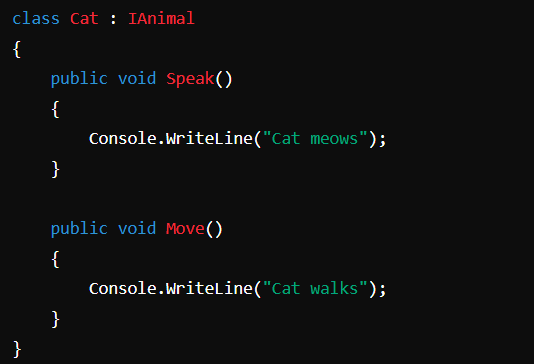
* **Có thể chứa triển khai:** Abstract class có thể chứa cả các phương thức có triển khai (implemented methods) và các phương thức trừu tượng (abstract methods) không có triển khai.
* **Có trường (fields):** Abstract class có thể chứa các trường.
* **Kế thừa đơn:** Một lớp chỉ có thể kế thừa từ một abstract class.
* **Có thể có constructor:** Abstract class có thể có constructor.
* **Khả năng cung cấp hành vi chung:** Abstract class thường được sử dụng để cung cấp một phần triển khai mặc định mà các lớp con có thể sử dụng hoặc ghi đè.

**Interface:**

* **Chỉ chứa khai báo:** Interface chỉ chứa các khai báo cho các phương thức, thuộc tính, sự kiện, và chỉ mục mà không có phần thân.
* **Không có trường:** Interface không thể chứa các trường.
* **Hỗ trợ đa kế thừa:** Một lớp hoặc một cấu trúc có thể thực hiện nhiều interface.
* **Không có constructor:** Interface không thể có constructor.
* **Chỉ cung cấp hợp đồng:** Interface cung cấp một hợp đồng mà các lớp thực hiện phải tuân thủ, nhưng không cung cấp bất kỳ hành vi nào.

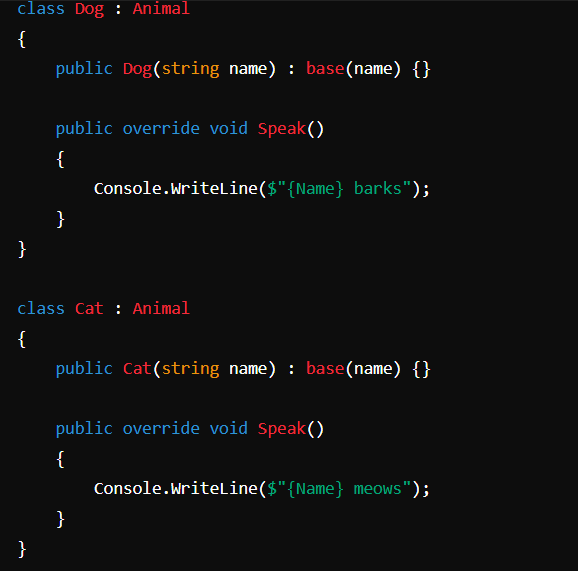
Interface:





Abstract Class:

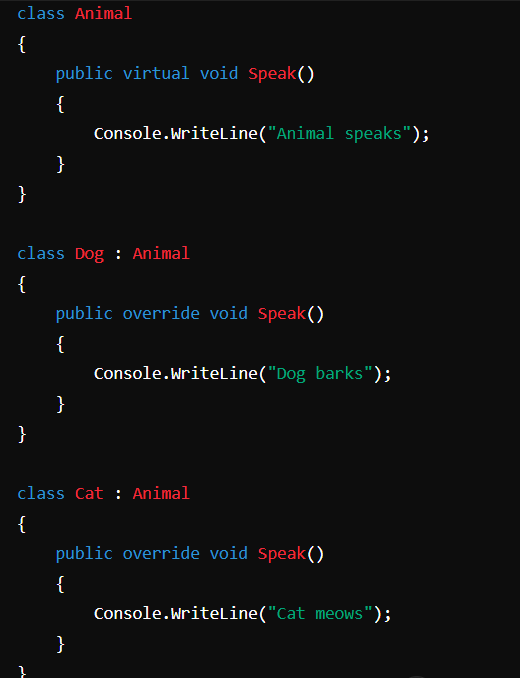




1. Giải thích từ khóa virtual, override, và abstract và cho ví dụ minh họa.

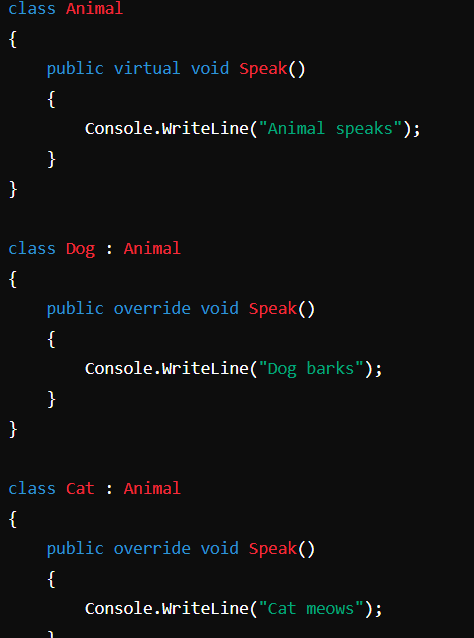
### Từ khóa virtual trong C#

**Giải thích:** Từ khóa virtual được sử dụng để chỉ định một phương thức hoặc một thuộc tính có thể được ghi đè trong các lớp dẫn xuất.



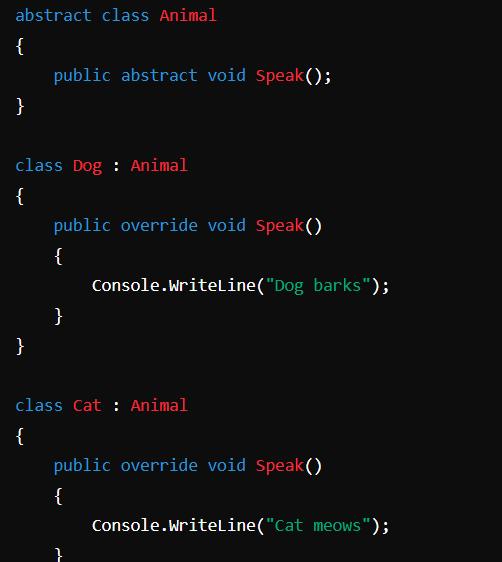
### Từ khóa override trong C#

**Giải thích:** Từ khóa override được sử dụng để ghi đè (override) một phương thức hoặc một thuộc tính đã được đánh dấu là virtual trong lớp cơ sở.



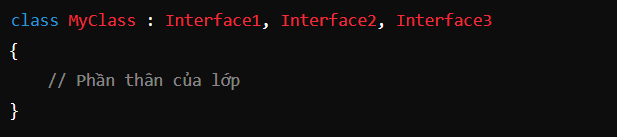
### Từ khóa abstract trong C#

**Giải thích:** Từ khóa abstract được sử dụng để khai báo một phương thức hoặc một lớp trừu tượng mà không cung cấp triển khai.

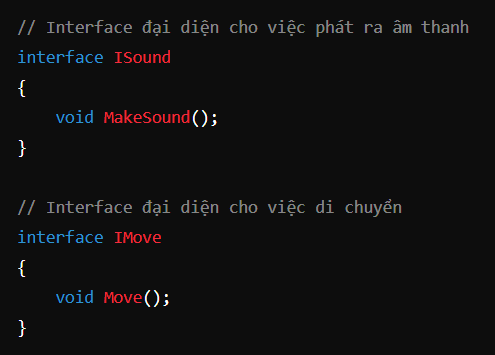


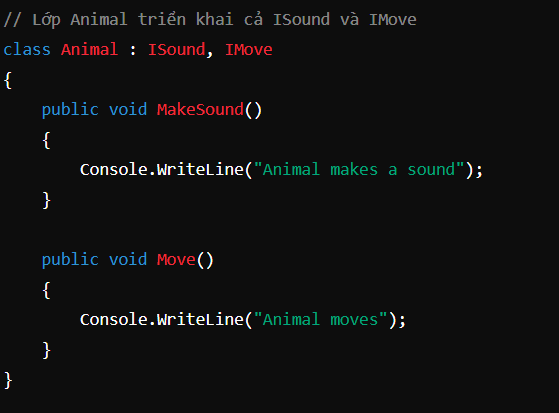
1. Làm thế nào để một lớp có thể triển khai nhiều interface? Cho ví dụ minh họa.

Trong C#, một lớp có thể triển khai nhiều interface bằng cách sử dụng cú pháp sau:



Ví dụ minh họa:





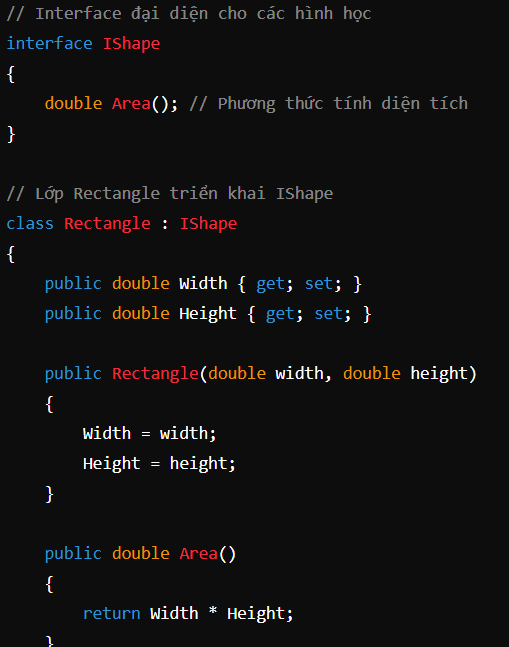
1. Tại sao một interface không thể chứa các phương thức với phần thân có định nghĩa thực thi?

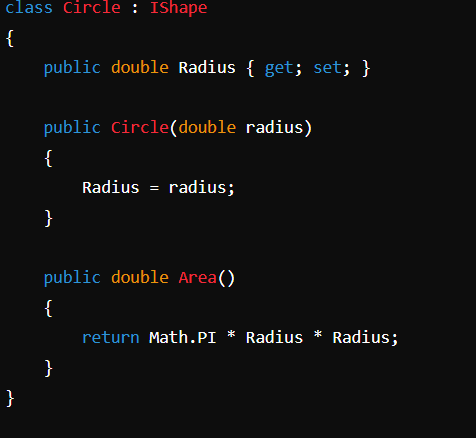
Trong ngôn ngữ lập trình như C#, một interface không thể chứa các phương thức với phần thân có định nghĩa thực thi vì các interface được thiết kế để chỉ định các hợp đồng (contracts) mà các lớp khác phải tuân thủ. Dưới đây là một số lý do tại sao interface không thể chứa các phương thức có định nghĩa thực thi:

* **Tính trừu tượng (Abstraction):** Interface là một phần của tính trừu tượng trong lập trình hướng đối tượng. Nó chỉ định giao diện (interface) mà các lớp khác phải tuân thủ mà không cung cấp triển khai cụ thể. Việc định nghĩa thực thi trong interface sẽ làm mất đi tính trừu tượng của nó.
* **Đa kế thừa (Multiple Inheritance):** Interface cho phép một lớp triển khai nhiều interface. Nếu các phương thức trong interface có định nghĩa thực thi, việc triển khai nhiều interface có thể dẫn đến xung đột giữa các triển khai.
* **Tính linh hoạt (Flexibility):** Interface cung cấp một giao diện mà các lớp có thể triển khai theo cách của riêng chúng. Nếu các phương thức trong interface có định nghĩa thực thi, nó sẽ hạn chế khả năng linh hoạt trong việc triển khai và tùy biến.
* **Tách rời hợp đồng và triển khai (Separation of Contract and Implementation):** Interface được sử dụng để chỉ định các hợp đồng mà các lớp khác phải tuân thủ, trong khi triển khai thực tế của các phương thức thuộc về lớp thực hiện. Việc định nghĩa thực thi trong interface sẽ làm mất đi sự tách rời này.

1. Giải thích về tính đa hình (polymorphism). Cho ví dụ sử dụng interface để minh họa tính đa hình.

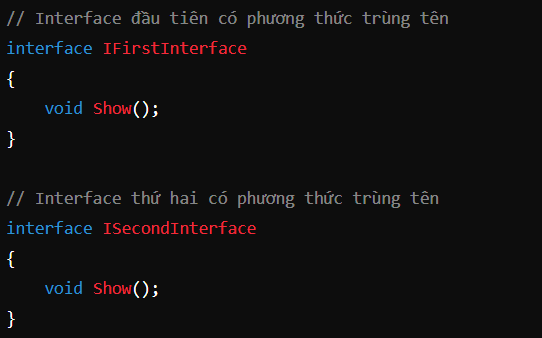
**Tính đa hình (Polymorphism)** là một khái niệm quan trọng trong lập trình hướng đối tượng, cho phép các đối tượng có thể hiểu và phản ứng với các phương thức gọi dựa trên loại của chính nó hoặc của lớp cơ sở của nó. Tính đa hình giúp tạo ra các lớp chung có thể xử lý các đối tượng theo cách phù hợp với từng loại đối tượng mà không cần biết chính xác loại của đối tượng đó.

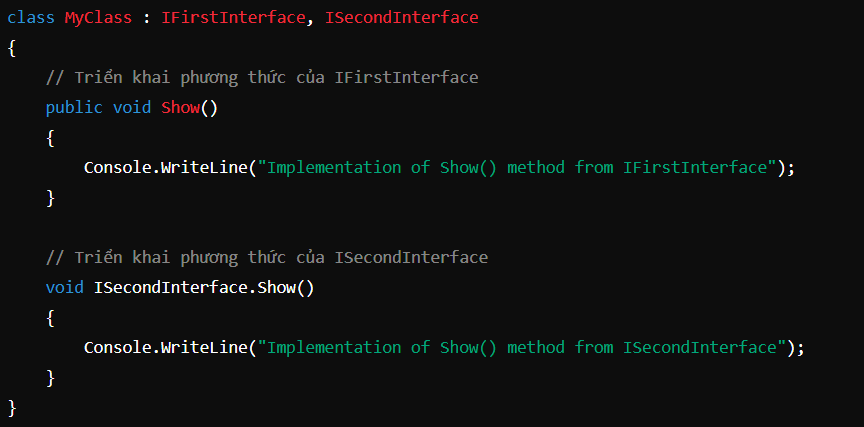




1. Làm thế nào để một lớp có thể triển khai hai interface có phương thức trùng tên? Cho ví dụ minh họa.

Khi một lớp triển khai hai interface có phương thức trùng tên, lớp đó phải cung cấp triển khai cho cả hai phương thức. Các phương thức này sẽ được phân biệt dựa trên tên của interface mà chúng thuộc về.





**Thực hành 1:**

A screenshot of a computer program

Description automatically generated

Từ lược đồ trên, tạo các lớp và bổ sung Phương thức tạo lập, toString

Trong lớp DanhSachNhanVien thực hiện các yêu cầu sau:

1. Phương thức nhập thủ công
2. Phương thức đọc và xuất dữ liệu từ file cho collection
3. Từ các thuộc tính của các lớp xây dựng các phương thức tương ứng như tìm kiếm, sắp xếp, cập nhật, thêm, xóa, sửa…

**Thực hành 2:** Từ bài thực hành 1 bổ sung các yêu cầu sau:

1. Thêm interface IKhaNangQuanLy với phương thức GanNhiemVu(string nhiemVu).
2. Triển khai interface này trong lớp QuanLy và viết phương thức để gán công việc cho nhân viên.
3. Tạo interface IBaoCao với phương thức TaoBaoCao().
4. Triển khai interface này trong lớp QuanLy và viết phương thức để tạo báo cáo về nhân viên.
5. Thêm tính năng lọc nhân viên theo phòng ban và hiển thị danh sách nhân viên thuộc phòng ban cụ thể.
6. Viết phương thức để lọc và hiển thị danh sách các nhân viên thuộc một phòng ban xác định.
7. Thêm thuộc tính Luong vào interface INhanVien và triển khai nó trong lớp QuanLy.
8. Cập nhật lớp QuanLy để lưu trữ và hiển thị thông tin lương của nhân viên.
9. Thêm interface IQuanLyLuong với phương thức TangLuong(double phanTram).
10. Triển khai interface này trong lớp QuanLy và viết phương thức để tăng lương cho nhân viên theo phần trăm.
11. Viết phương thức tìm kiếm và hiển thị thông tin của nhân viên theo tên.
12. Phương thức này sẽ tìm và hiển thị thông tin của các nhân viên có tên chứa từ khóa tìm kiếm.
13. Thêm tính năng tính toán và hiển thị tổng số lương của tất cả nhân viên.
14. Viết phương thức để tính toán và hiển thị tổng số lương của tất cả nhân viên trong danh sách.
15. Viết phương thức sắp xếp danh sách nhân viên theo tên và hiển thị danh sách sau khi sắp xếp.
16. Phương thức này sẽ sắp xếp danh sách nhân viên theo tên (tăng dần hoặc giảm dần) và hiển thị kết quả.
17. Từ các thuộc tính của các lớp mới bổ sung, xây dựng các phương thức tương ứng như tìm kiếm, sắp xếp, cập nhật, thêm, xóa, sửa…