Bài 34:

So sánh abstract và virtual:

| Tiêu chí | abstract | virtual |
| --- | --- | --- |
| Mục đích | Buộc các lớp con phải cài đặt lại. | Cung cấp phần thân mặc định nhưng cho phép ghi đè. |
| Cú pháp | Không có phần thân (implementation). | Có phần thân mặc định. |
| Sử dụng | Dùng trong lớp trừu tượng. | Dùng trong bất kỳ lớp nào. |
| Phải ghi đè | Lớp con bắt buộc phải ghi đè. | Lớp con có thể ghi đè hoặc không. |
| Ví dụ khi nào dùng | Khi bạn muốn mọi lớp con phải cung cấp cài đặt cụ thể. | Khi bạn muốn cung cấp hành vi mặc định mà có thể được tùy chỉnh. |

Kết luận:

* abstract được sử dụng khi bạn muốn bắt buộc tất cả các lớp con phải cài đặt lại thuộc tính hoặc phương thức. Điều này thường được dùng trong các lớp trừu tượng mà bạn muốn đảm bảo rằng mọi lớp con đều có cách triển khai cụ thể của riêng mình.
* virtual được sử dụng khi bạn muốn cung cấp một cách triển khai mặc định cho thuộc tính hoặc phương thức nhưng vẫn cho phép các lớp con có thể tùy chỉnh nếu cần. Điều này cung cấp sự linh hoạt hơn cho các lớp con, cho phép chúng chọn sử dụng hoặc ghi đè hành vi mặc định.

Bài tập 33: Nếu bỏ đi get, set ở khai báo public virtual string Name { get; set; } chuyển thành public string Name; trong lớp Person, thì sẽ xảy ra các vấn đề sau:

1.Tính đóng gói (Encapsulation): Khi chúng ta bỏ đi get và set, thì Name sẽ trở thành một trường (field) công khai. Điều này vi phạm nguyên tắc tính đóng gói, vì các lớp khác có thể trực tiếp truy cập và sửa đổi giá trị của Name mà không cần qua các phương thức trung gian. Việc này có thể dẫn đến các lỗi và mất đi sự kiểm soát trong việc quản lý dữ liệu.

2.Tính tái sử dụng (Reusability): Khi Name là một trường công khai, các lớp kế thừa từ Person sẽ không thể ghi đè (override) Name như trước đây. Điều này làm giảm tính tái sử dụng của lớp Person, vì các lớp con không thể tùy chỉnh hành vi của Name theo nhu cầu của mình.

3.Tính bảo mật (Security): Khi Name là một trường công khai, các lớp ngoài có thể trực tiếp gán giá trị không hợp lệ cho nó. Ví dụ, trong lớp Student, chúng ta không thể kiểm tra xem giá trị Name có hợp lệ hay không (như không được để trống).

4.Tính nhất quán (Consistency): Việc thay đổi Name từ thuộc tính sang trường công khai sẽ làm mất đi tính nhất quán trong thiết kế. Các lớp khác sử dụng Person có thể sẽ bị ảnh hưởng và phải thay đổi cách sử dụng Name.

Vì vậy, việc giữ lại get và set cho Name trong lớp Person là cần thiết để đảm bảo tính đóng gói, tái sử dụng, bảo mật và nhất quán trong thiết kế hướng đối tượng.