Bài 33 : Nếu bạn bỏ get và set, bạn sẽ mất nhiều tính năng và sự linh hoạt của thuộc tính trong C#. Việc sử dụng get và set là cách tốt nhất để duy trì tính chất trừu tượng và kiểm soát của dữ liệu trong các lớp kế thừa.

**Bài tập 34: So sánh khai báo abstract và khai báo virtual cho thuộc tính và phương thức của 1 lớp trong C#**

Trong C#, cả từ khóa abstract và virtual đều được sử dụng để chỉ ra rằng các thuộc tính hoặc phương thức của một lớp có thể được ghi đè trong các lớp con, nhưng chúng có những sự khác biệt quan trọng.

**Khai báo abstract**

* **Định nghĩa**: Một phương thức hoặc thuộc tính abstract chỉ có thể được khai báo trong một lớp trừu tượng (abstract class).
* **Không có triển khai**: Không có thân phương thức hoặc phần thân của thuộc tính. Nó chỉ định rằng lớp con phải cung cấp phần triển khai.
* **Bắt buộc override**: Lớp con phải ghi đè các phương thức hoặc thuộc tính abstract.

Ví dụ:

csharp

Sao chép mã

public abstract class Person

{

public abstract string Name { get; set; }

public abstract void DisplayInfo();

}

public class Student : Person

{

private string name;

public override string Name

{

get { return name; }

set { name = value; }

}

public override void DisplayInfo()

{

Console.WriteLine($"Student Name: {Name}");

}

}

**Khai báo virtual**

* **Định nghĩa**: Một phương thức hoặc thuộc tính virtual có thể được ghi đè nhưng không bắt buộc.
* **Có triển khai**: Có thể có phần thân phương thức hoặc phần thân của thuộc tính.
* **Không bắt buộc override**: Lớp con có thể ghi đè nếu cần thiết, nhưng không bắt buộc phải làm như vậy.

Ví dụ:

csharp

Sao chép mã

public class Person

{

public virtual string Name { get; set; }

public virtual void DisplayInfo()

{

Console.WriteLine($"Person Name: {Name}");

}

}

public class Student : Person

{

public override string Name { get; set; }

public override void DisplayInfo()

{

Console.WriteLine($"Student Name: {Name}");

}

}

**So sánh**

| **Tiêu chí** | **abstract** | **virtual** |
| --- | --- | --- |
| Khai báo | Chỉ trong lớp trừu tượng | Trong bất kỳ lớp nào |
| Phần triển khai | Không có | Có thể có |
| Bắt buộc override | Có | Không bắt buộc |
| Mục đích | Định nghĩa hợp đồng cho lớp dẫn xuất | Cung cấp triển khai mặc định có thể ghi đè |

**Bài tập 35: Viết chương trình C#, tạo lớp trừu tượng Shape và lớp kế thừa TamGiac**

**Lớp Shape:**

csharp

Sao chép mã

public abstract class Shape

{

private int soDinh;

public int SoDinh

{

get { return soDinh; }

set

{

if (value <= 0)

{

throw new ArgumentException("Số đỉnh phải lớn hơn 0.");

}

soDinh = value;

}

}

}

**Lớp TamGiac:**

csharp

Sao chép mã

public class TamGiac : Shape

{

public TamGiac()

{

SoDinh = 3;

}

public new int SoDinh

{

get { return base.SoDinh; }

set

{

if (value != 3)

{

throw new ArgumentException("Tam giác phải có đúng 3 đỉnh.");

}

base.SoDinh = value;

}

}

}

**Chương trình chính:**

csharp

Sao chép mã

class Program

{

static void Main(string[] args)

{

try

{

Shape shape = new TamGiac();

Console.WriteLine($"Số đỉnh của tam giác: {shape.SoDinh}");

}

catch (ArgumentException ex)

{

Console.WriteLine(ex.Message);

}

}

}

**Ý nghĩa của khai báo { get; set; } trong C#**

* **Khai báo tự động**: { get; set; } trong C# cho phép bạn tạo ra các thuộc tính với phần thân getter và setter tự động được triển khai. Điều này giúp mã nguồn ngắn gọn và dễ đọc hơn.
* **Getter và Setter**: Getter và Setter là các phương thức truy cập thuộc tính. Getter trả về giá trị của thuộc tính, trong khi Setter gán giá trị cho thuộc tính.
* **Validation**: Bạn có thể thêm logic vào getter hoặc setter để thực hiện các kiểm tra, xác thực hoặc bất kỳ logic bổ sung nào khi truy cập hoặc gán giá trị cho thuộc tính.

**Ví dụ với Getter và Setter:**

csharp

Sao chép mã

private string name;

public string Name

{

get { return name; }

set

{

if (string.IsNullOrWhiteSpace(value))

{

throw new ArgumentException("Name cannot be empty or null.");

}

name = value;

}

}

Những kiến thức này rất quan trọng trong lập trình C#, đặc biệt là khi làm việc với Entity Framework và Web API, nơi việc sử dụng các thuộc tính với { get; set; } rất phổ biến.