Method Hiding in C#

Introduction

In this section of the course, we have covered many important concepts related to Object-Oriented Programming (OOP), including inheritance, method overriding, and polymorphism. However, there is another technique called **Method Hiding** that often confuses developers when they first encounter it. Instead of including it in the video lectures, we are providing this written explanation so you can learn about it at your own pace.

This article will introduce **Method Hiding**, explain why it is useful, provide a simple analogy, and walk through its implementation with examples. We will also compare it with **Method Overriding** to ensure you understand the differences.

1. What is Method Hiding?

Method Hiding is a technique in C# that allows a derived class to define a method with the same name as a method in the base class, effectively **hiding** the base class method rather than overriding it. This is done using the new keyword.

Analogy

Imagine you are working in a company where employees have a standard ID card with a barcode. However, senior employees receive a special ID card with an embedded chip for additional access. The new ID card **hides** the old one but does not replace the concept of having an ID card.

Similarly, **Method Hiding** allows a derived class to introduce a new version of a method without completely replacing the functionality of the base class method.

2. Declaring and Using Method Hiding

Basic Syntax

To hide a method in the base class, we use the new keyword in the derived class:

```
class BaseClass
{
    public void ShowMessage()
    {
        Console.WriteLine("Message from BaseClass");
    }
}

class DerivedClass : BaseClass
{
    public new void ShowMessage()
    {
        Console.WriteLine("Message from DerivedClass");
    }
}
```

Step-by-Step Implementation

Step 1: Define a Base Class with a Method

```
class Animal
{
    public void Speak()
    {
        Console.WriteLine("The animal makes a sound.");
    }
}
```

This base class contains a Speak method.

Step 2: Create a Derived Class that Hides the Method

```
class Dog : Animal
{
    public new void Speak()
    {
        Console.WriteLine("The dog barks.");
    }
}
```

The Speak method in Dog uses new, which hides the base class method.

Step 3: Demonstrate Method Hiding in Action

```
Animal myAnimal = new Animal();
myAnimal.Speak();  // Output: The animal makes a sound.

Dog myDog = new Dog();
myDog.Speak();  // Output: The dog barks.

Animal myPet = new Dog();
myPet.Speak();  // Output: The animal makes a sound.
```

Here, myPet.Speak() calls the base class method because **Method Hiding does not support polymorphism.**

3. Comparing Method Hiding with Method Overriding

Feature Method Hiding (new) Method Overriding (override) Keyword
Used new override Supports Polymorphism No Yes Calls Base Method By Default Yes No (unless base.Method() is used) Compiler Warning Yes (if new is omitted) No

Example of Overriding (Comparison)

```
class Animal
{
    public virtual void Speak()
    {
        Console.WriteLine("The animal makes a sound.");
    }
}

class Dog : Animal
{
    public override void Speak()
    {
        Console.WriteLine("The dog barks.");
    }
}
```

Here, overriding allows **polymorphic behavior**, meaning myPet.Speak() would call Dog's version instead of Animal's.

4. When to Use Method Hiding?

You should use **Method Hiding** when:

- You want to introduce a new method in a derived class without affecting the base class method.
- You don't need polymorphic behavior (i.e., calling methods dynamically based on object type).
- The base class method should still be accessible if needed.

Use **Method Overriding** instead when:

- You want polymorphic behavior.
- The method in the derived class should replace the base class method completely.
- The base method is declared as virtual, abstract, or override.

5. Best Practices and Common Mistakes

Best Practices

✔ Always use the new keyword when hiding a method to make it explicit.
✔ Consider if method overriding is a better choice before using method hiding.
✔ Keep the base method accessible when appropriate to avoid confusion.

Common Mistakes

❌ **Forgetting the new keyword** – This causes a compiler warning, as C# assumes you unintentionally hid the method.

```
class Dog : Animal
{
    public void Speak() // Warning: hides base method
    {
        Console.WriteLine("The dog barks.");
    }
}
```

❌ **Expecting polymorphism to work with method hiding** – If you call the hidden method on a base class reference, it will execute the base class version instead.

6. Conclusion

Method Hiding allows a derived class to define a method with the same name as a method in the base class without overriding it. It is useful when you want to keep the base class method intact while providing a new implementation in the derived class.

While it might seem similar to **Method Overriding**, the key difference is that **Method Hiding does not support polymorphism**. If you need polymorphism, **overriding is the better approach**.

If you have any questions, feel free to ask in the Q&A.
Happy coding!