Method Hiding, Shadowing, and Overriding in C# Explained with Examples

Method hiding, shadowing, and overriding are important concepts in object-oriented programming using C#.

They involve the use of virtual, override, and new keywords to define and implement methods in derived classes that are related to methods in base classes. In this blog post, we will explore these concepts in detail and provide examples to illustrate how they work in C#.

Method Hiding:

- Method hiding is a concept where a derived class defines a new method with the same name as a method in the base class, effectively hiding the base class method.
- Method hiding is achieved by using the new keyword in the derived class method.
- The base class method should be marked as virtual to be hidden by the derived class method.
- The derived class method must have the same name as the base class method, but it can have a different return type or method signature.
- Example:

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

public class Cat : Animal
{
    public new void MakeSound()
    {
        Console.WriteLine("Cat makes a sound.");
    }
}
```

```
public class Program
{
    static void Main()
    {
        // Method Hiding
         Animal animal1 = new Animal();
         Animal cat1 = new Cat();
         Cat animal2 = new Animal(); // Compilation error - cannot implicitly
convert Animal to Cat
         Cat cat2 = new Cat();

         animal1.MakeSound(); // Output: "Animal makes a sound."
         cat1.MakeSound();    // Output: "Animal makes a sound." (hiding)
         cat2.MakeSound();    // Output: "Cat makes a sound."
    }
}
```

Method Shadowing:

- Method shadowing is similar to method hiding, where a derived class defines a new method with the same name as a method in the base class.
- However, in method shadowing, the base class method is not marked as virtual , and the derived class method does not use the new keyword.
- This can lead to unexpected behavior as the base class method may still be invoked when calling the method on an object of the derived class.
- Example:

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

public class Cat : Animal
{
    public void MakeSound()
    {
        Console.WriteLine("Cat makes a sound.");
    }
}
```

```
public class Program
{
    static void Main()
    {
        // Method Shadowing
        Animal animal1 = new Animal();
        Animal cat1 = new Cat();
        Cat animal2 = new Animal(); // Compilation error - cannot implicitly
convert Animal to Cat
        Cat cat2 = new Cat();

        animal1.MakeSound(); // Output: "Animal makes a sound."
        cat1.MakeSound();    // Output: "Animal makes a sound." (shadowing)
        cat2.MakeSound();    // Output: "Cat makes a sound."
    }
}
```

Method Overriding:

- Method overriding is a concept where a derived class provides a new implementation of a base class method, and the new implementation is used when calling the method on an object of the derived class.
- Method overriding is achieved by using the override keyword in the derived class method.
- The base class method should be marked as virtual and the derived class method must have the same name, return type, and method signature.
- Example:

```
public class Animal
{
    public virtual void MakeSound()
    {
        Console.WriteLine("Animal makes a sound.");
    }
}
public class Cat : Animal
{
    public override void MakeSound()
    {
        Console.WriteLine("Cat makes a sound.");
    }
}
```

```
public class Program
{
    static void Main()
    {
        // Method Overriding
        Animal animal1 = new Animal();
        Cat animal2 = new Animal(); // Compilation error - cannot implicitly
convert Animal to Cat
        Animal cat1 = new Cat();
        Cat cat2 = new Cat();

        animal1.MakeSound(); // Output: "Animal makes a sound."
        cat1.MakeSound();    // Output: "Cat makes a sound." (overriding)
        cat2.MakeSound();    // Output: "Cat makes a sound."
    }
}
```

Differences between Method Hiding, Shadowing, and Overriding:

1. **Method Hiding**:

- Derived class defines a new method with the same name as a method in the base class, effectively hiding the base class method.
- new keyword is used in the derived class method.
- Base class method should be marked as virtual or override.
- Derived class method can have a different return type or method signature.

2. Method Shadowing:

- Derived class defines a new method with the same name as a method in the base class, but the base class method is **not marked** as virtual or override.
- Base class method may still be invoked when calling the method on an object of the derived class.
- No new keyword is used in the derived class method.

3. Method Overriding:

- Derived class provides a new implementation of a base class method.
- override keyword is used in the derived class method.
- Base class method should be marked as virtual, abstract, or override.

- Derived class method must have the same name, return type, and method signature as the base class method.

In conclusion, method hiding, shadowing, and overriding are important concepts in C# that allow for customization and extension of base class methods in derived classes. By following the proper guidelines for creating objects and implementing these concepts, you can ensure correct and expected behavior in your object-oriented programming code.

I hope this detailed blog post on Method hiding, shadowing and overriding in C# with examples and best practices has been helpful to you. Remember to apply these concepts wisely in your code and make use of the code examples provided to enhance your understanding. Happy coding!