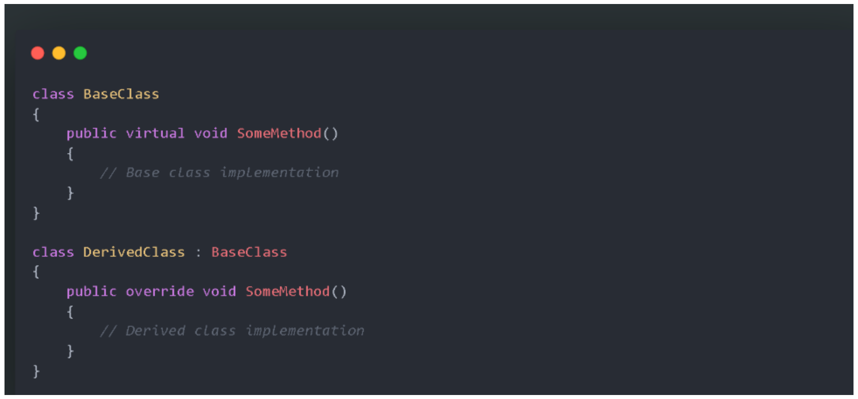
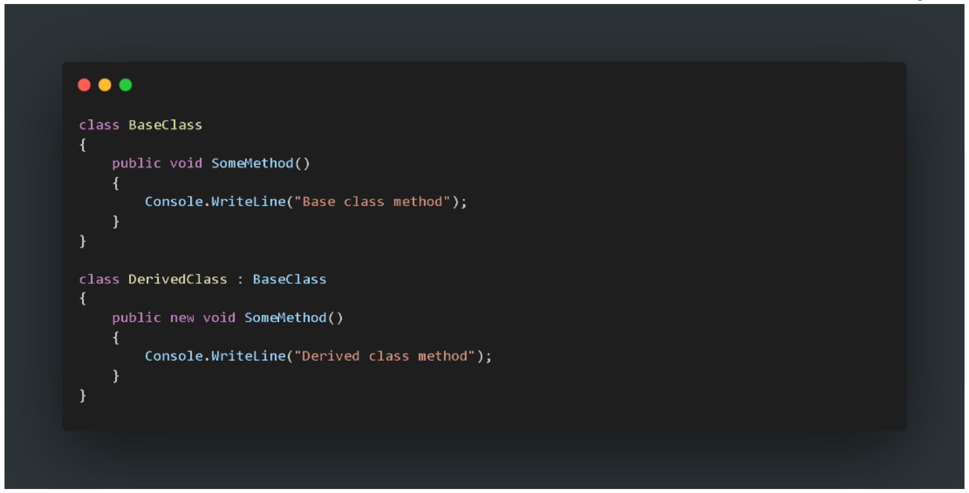
Access Modifier	Accessibility
Public	• *Everywhere*
Private	• *Only within the class*
Protected	• *Within the class and any subclasses*
Internal	• *Within the assembly*
Protected Internal	• *Within the class, subclasses, and the same assembly*

- ***Inheritance.***
- *Inheritance creates a hierarchical relationship between classes, where a derived class (also known as a child class or subclass) inherits members from a base class (also known as a parent class or superclass).*
- *The child class can access the public and protected members (fields, properties, and methods) of the base class.*
- ***NOTES***
  - ***Method Overriding***
- *allows the child class to provide a different implementation for a method that is already defined in the base class.*
- *use the override keyword in the child class method declaration.*
- *The base class method must be marked as virtual or abstract to allow overriding.*

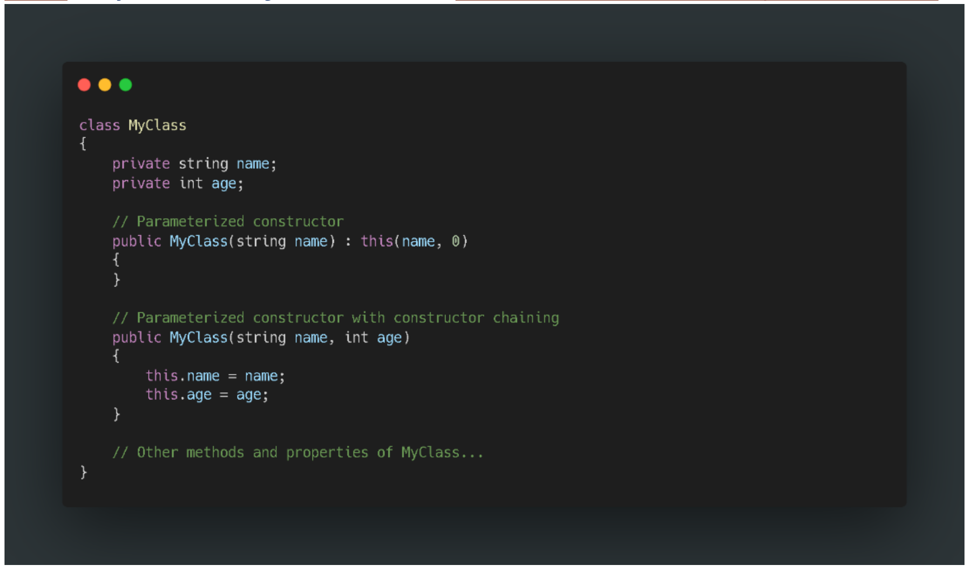


```
class BaseClass
{
    public virtual void SomeMethod()
    {
        // Base class implementation
    }
}

class DerivedClass : BaseClass
{
    public override void SomeMethod()
    {
        // Derived class implementation
    }
}
```

2.Method Hiding

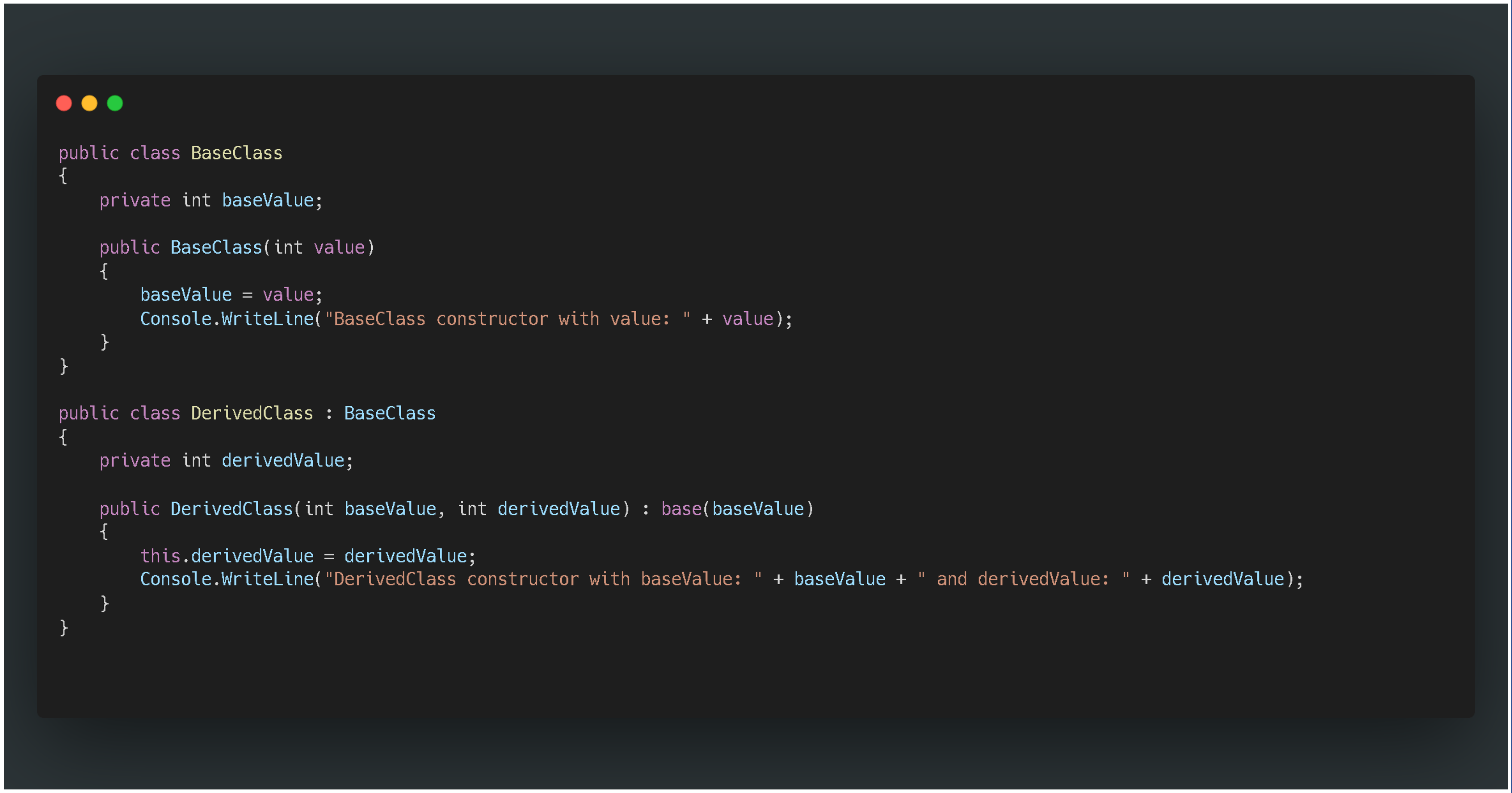
- *If a child class has a member with the same name as a member in the base class, the derived class member can hide the base class member using the new keyword.*
- *create a new member in the derived class without any relationship to the base class member.*
- 

```
class BaseClass
{
    public void SomeMethod()
    {
        Console.WriteLine("Base class method");
    }
}

class DerivedClass : BaseClass
{
    public new void SomeMethod()
    {
        Console.WriteLine("Derived class method");
    }
}
```

- ***Base Class Constructors:***
- *When a child class is instantiated, the base class constructor is called first to initialize the inherited members.*
- *If the base class has multiple constructors, the derived class constructor can use the base keyword to explicitly invoke a specific base class constructor.*
- ***Constructor chaining in Same Class.***
- *allows one constructor to call another constructor within the same class or in the base class.*
- *To call another constructor from within a constructor, you use the this keyword .*
- *this keyword refers to the current instance of the class.*
- 

```
class MyClass
{
    private string name;
    private int age;

    // Parameterized constructor
    public MyClass(string name) : this(name, 0)
    {
    }

    // Parameterized constructor with constructor chaining
    public MyClass(string name, int age)
    {
        this.name = name;
        this.age = age;
    }

    // Other methods and properties of MyClass...
}
```

- ***Chaining to Base Class Constructor.***
- *When a child class constructor is called, it can chain to a constructor in the base class using the base keyword.*
- *The base class constructor is called before the child class constructor initializes its own members.*

- 

```
public class BaseClass
{
    private int baseValue;

    public BaseClass(int value)
    {
        baseValue = value;
        Console.WriteLine("BaseClass constructor with value: " + value);
    }
}

public class DerivedClass : BaseClass
{
    private int derivedValue;

    public DerivedClass(int baseValue, int derivedValue) : base(baseValue)
    {
        this.derivedValue = derivedValue;
        Console.WriteLine("DerivedClass constructor with baseValue: " + baseValue + " and derivedValue: " + derivedValue);
    }
}
```

◦ ***sealed class.***

- *class that cannot be inherited by other classes.*
- *cannot serve as a base class for other classes.*
- *Once a class is sealed, all its methods are implicitly sealed and cannot be further overridden.*
- *It allows for better encapsulation.*

- ***Sealed Function.***

- *sealed method is a method that cannot be overridden by derived classes.(Stop For Extension Of Virtually).*
- *Once a method is sealed in a base class, it cannot be overridden by derived classes.*
- *sealed method in the base class is the final implementation.*
- *To sealed a method, it must be declared as virtual or override in the base class.*
- ***NOTES***
- ***Static variables***
- *Variable shared among all instances of a class.*
- *must be initialized at the time of declaration or within a static constructor.*
- *Static variables are accessible within the entire class and can be accessed using the class name followed by the variable name.ClassName.StaticVariable)*
- *Static variables are initialized before any instance of the class is created*
- *Access to static variables from multiple threads can cause race conditions and concurrency issues.*
- *Must Use synchronization mechanisms, such as locks or other thread-safe constructs, should be used when accessing or modifying static variables in a multi-threaded environment.*
- *Static variables are useful for maintaining shared state or storing data that needs to be shared across all instances of a class.*
- ***Static Method.***
- *static method is a method that belongs to the class itself rather than to instances of the class.*
- *They can be accessed directly using the class name followed by the method name.*
- *static methods do not require an instance of the class to be called.*
- *called directly using the class name without creating an object of the class.[ClassName.StaticMethod();]*
- *Static methods can access other static members (variables, methods) within the same class without the need for an instance reference.*
- *They cannot be marked as virtual, abstract, or override.*
- *Static methods cannot be marked as async or await.*

- ***Static Class***
- *Static classes cannot be instantiated (Sealed Behavior).*
- *Static classes cannot create Object using the new keyword because they are implicitly sealed.*
- *Static classes can only contain static members, including static methods, properties, fields, and events*
- *Static classes are defined at the namespace level and are accessible throughout the same namespace.*
- *Static classes are commonly used to group together utility functions and helper methods that provide common functionality without requiring object-specific data.*
- ***Static constructor***
- *static constructor (also known as a type initializer) is a special constructor that initializes the static members of a class.*
- *A static constructor does not have any parameters and is declared using the static keyword followed by the class name.*
- *has no access modifiers, return type, or method name.*
- *It is called only once during the lifetime of the program.*
- *Only one static constructor is allowed per class.*
- *Static constructors are typically used to initialize static members, including static variables, static properties, and other static data structures.*