Learning C-Sharp

Sealed keyword

What is Sealed keyword an when to use?

- It is one of the important keywords in C#. It can be used with classes, methods and properties.
- If you don't want derived classes to customize your class or change the behaviour of your class, you can use '*sealed*' keyword

How Sealed is used?

- When used with classes, a sealed class cannot be inherited or in other words it is used to restrict inheritance of a class.
- In C#, we should not use abstract modifier with sealed class, because an abstract class must be inherited by a class that provides an implementation of the abstract methods or properties.
- Local variables cannot be sealed.
- In C#, structs are implicitly sealed.

Let us consider two classes namely '*Employee'* and '*Manager*'.

'*Manager*' class is a derived class which is inheriting from '*Employee'* which is a base class.

```
class Employee
{

}

class Manager: Employee
{

}
```

```
class Employee
{

}

sealed class Manager: Employee
{

}
```

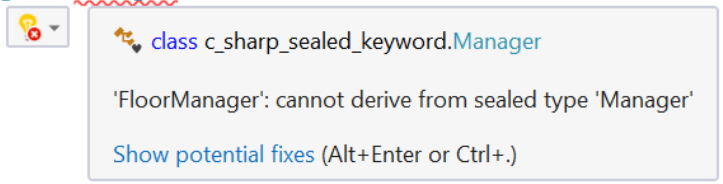
We don't want our '*Manager*' class to be inherited by any other classes

so we have to declare this class as a '**Sealed'** class as shown in the picture below.

As we can see, sealed class lets you inherit from other classes. So, there is no compiler error for this.

Now, consider we have a third class '*FloorManager*'. We try to inherit from '*Manager*' class as shown below.

```
class Employee
{

}

sealed class Manager: Employee
{

}

class FloorManager: Manager
{

}
```



'sealed' keyword can be used before or after the access modifier as shown below.

```
public sealed class Manager: Employee
{

}
```

```
sealed public class Manager : Employee
{

}
```

- When you define a class as 'sealed', its methods and properties automatically become '*sealed*'.

Let’s us consider the following code.

```
public class Employee
{
    public void DoSomething()
    {

    }
}

public class Manager : Employee
{

}
```

We cannot add ‘sealed’ keyword to a method in base class like this.

```
public sealed void DoSomething()
{

}
```

void Employee.DoSomething()

'Employee.DoSomething()' cannot be sealed because it is not an override

Compiler will show you an error. This means Base class lets you override a method at least once which makes sense. Let’s make some changes to the code.

```
public class Employee
{
    public virtual void DoSomething()
    {

    }
}

public class Manager : Employee
{
    public sealed void DoSomething()
    {

    }
}
```

void Manager.DoSomething()

'Manager.DoSomething()' cannot be sealed because it is not an override

As you can see, if 'DoSomething()' method in the derived class expects that method to be 'overridden'.

```
public sealed override void DoSomething()
{

}
```

Now, if I try to inherit this method in a 'FloorManager' which is deriving from 'Manager' class, then it will show a compiler error.

```
public class Employee
{
    public virtual void DoSomething()
    {

    }
}

public class Manager : Employee
{
    public sealed override void DoSomething()
    {

    }
}

public class FloorManager : Manager
{
    public sealed override void DoSomething()
    {

    }
}
```

```
void FloorManager.DoSomething()

'FloorManager.DoSomething()': cannot override inherited member 'Manager.DoSomething()' because it is sealed
```

So as expected, since the method in '*Manager*' class is marked as 'sealed', compiler does not allow '*FloorManager*' class to override that method.

- Sealed keyword can be used with properties as well.

```
public class Employee
{
    public virtual int BaseSalary { get; set; }
}

public class Manager : Employee
{
    public sealed override int BaseSalary { get; set; }
}
```