

ACCESS MODIFIERS

C# programlama dilinde oluşturulan uygulamaların güvenliğini artırmak için, kullanılan sınıfların (class) erişilebilirliğinin kısıtlanması gerekmektedir. Bu anlamda, erişim belirleyiciler (access modifiers) koda dışardan yapılmak istenen müdahalenin sınırlarını belirlemek amacıyla kullanılan anahtar ifadelerdir.

- **public:** Uygulayan nesnelere projenin her yerinden erişilebilir. Bu nedenle, herhangi bir erişilebilirlik kısıtlaması yoktur.

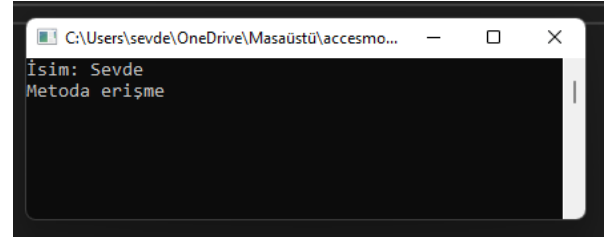
```
2 references
class Student
{
    public string name = "Sevde";

    1 reference
    public void print()
    {
        Console.WriteLine("Metoda erişme");
    }
}

0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        // Student class ından nesne oluşturma
        Student student1 = new Student();

        // name alanına erişme ve yazdırma
        Console.WriteLine("İsim: " + student1.name);

        //Student class ından print() metoduna erişme
        student1.print();
        Console.ReadLine();
    }
}
```



```
C:\Users\sevde\OneDrive\Masaüstü\accessmo...
İsim: Sevde
Metoda erişme
```

- **private:** Uygulayan nesnelere yalnızca bir sınıf veya yapı içinde erişilebilir. Sonuç olarak, oluşturuldukları sınıfın dışında onlara erişemiyoruz.

```
2 references
class PrivateTest
{
    private string name = "Shantosh Kumar";

    1 reference
    private void Msg(string msg)
    {
        Console.WriteLine("Hello " + msg);
    }
}

0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        PrivateTest privateTest = new PrivateTest();
        // private değişkene erişim
        Console.WriteLine("Hello " + privateTest.name);
        // private metoda erişim
        privateTest.Msg("Gökkuş Yıldız");

        //Hata alıyoruz çünkü name değişkenine ve Msg metoduna private olduğu için ulaşamıyoruz.
        //Msg metoduna ve name değişkenine sadece PrivateTest class ı içerisinden erişebiliriz.
    }
}
```

- **protected:** Yalnızca aynı sınıftan ve o sınıftan türetilen tüm sınıflarda erişilebilir olduğunu ima eder.

Aşağıdaki örnekte alan korumalı olduğu için main den erişemiyoruz.

```

3 references
class Student
{
    protected string name = "Sevde";
}

0 references
class Program : Student
{
    0 references
    static void Main(string[] args)
    {
        // Student class ından nesne türetme
        Student student1 = new Student();

        // name alanına erişme ve yazdırma
        Console.WriteLine("Name: " + student1.name);
        Console.ReadLine();
    }
}

```

- Alana türetilmiş sınıftan erişme:

```

1 reference
class Student
{
    protected string name = "Sevde";
}

2 references
class Program : Student
{
    0 references
    static void Main(string[] args)
    {
        // Türetilmiş sınıfın nesnesini oluşturma
        Program program = new Program();

        // name alanına erişme ve yazdırma
        Console.WriteLine("Name: " + program.name);

        Console.ReadLine();
    }
}

```

NOT: Alan, türetilmiş sınıflara erişebildiği için protected alana erişebiliyoruz.

- **internal:** Yalnızca aynı derleme içinde erişilebilir.
 - Kendi projesi içerisinde public, farklı bir projeden/dışarıdan çağırılmak istenildiğinde ise private özelliklerini taşır.
 - Birlikte çalışmak ve mantıksal bir işlevsellik birimi oluşturmak için inşa edilmişlerdir.
 - Bu nedenle, bir derlemeyi çalıştırdığımızda, derleme içindeki tüm sınıflar ve arayüzler birlikte çalışır.

Aşağıdaki örnekte alan ve metod internal olduğu için Program sınıfı içerisinden erişebiliyoruz.

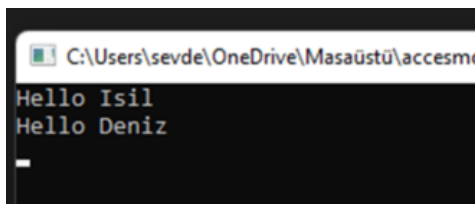
```
2 references
class InternalTest
{
    internal string name = "Isil";
    1 reference
    internal void Msg(string msg)
    {
        Console.WriteLine("Hello " + msg);
    }
}

0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        InternalTest internalTest = new InternalTest();

        // internal değişkene erişme
        Console.WriteLine("Hello " + internalTest.name);

        // internal metoda erişme
        internalTest.Msg("Deniz");

        Console.ReadLine();
    }
}
```



```
C:\Users\sevde\OneDrive\Masaüstü\accessmı
Hello Isil
Hello Deniz
_
```

- **protected internal** : Aynı protected gibi kendi bulunduğu class üzerinde ve bu class ı miras alan classlar üzerinden çağrılabilir. Artı olarak aynı proje (assembly/dll) üzerinde olmasalar dahi, tanımlandığı class'tan türetilmiş diğer class'ların içinden de çağrılabilirler.

Özel bir durum olmadıkça varsayılan olarak değişkenler ve methodlar private, classlar ise internal dır.

```

2 references
class InternalTest
{
    protected internal string name = "Taha";
    1 reference
    protected internal void Msg(string msg)
    {
        Console.WriteLine("Hello " + msg);
    }
}

0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        InternalTest internalTest = new InternalTest();

        // protected internal değişkene erişme
        Console.WriteLine("Hello " + internalTest.name);

        // protected internal metoduna erişme
        internalTest.Msg("Seyda");

        Console.ReadLine();
    }
}

```

```

C:\Users\sevde\OneDrive\Masaüstü\accesmodif\acc...
Hello Taha
Hello Seyda

```

Bu erişimlerin özelliklerini kısaca özetleyecek olursak;

Caller's location	public	protected internal	protected	internal	private protected	private
Within the class	✓	✓	✓	✓	✓	✓
Derived class (same assembly)	✓	✓	✓	✓	✓	✗
Non-derived class (same assembly)	✓	✓	✗	✓	✗	✗
Derived class (different assembly)	✓	✓	✓	✗	✗	✗
Non-derived class (different assembly)	✓	✗	✗	✗	✗	✗