

Nesne Tabanlı Programlama

Bölüm 2

Sınıflar ve Nesneler

Dr. Öğr. Üyesi Murat TAŞYÜREK (kayubmprogramlama1@gmail.com)

9 Ekim 2023

Kayseri Üniversitesi, Bilgisayar Mühendisliği Bölümü

- C# programlama dilinde nesneleri oluşturmak için sınıf (class) adı verilen veri türü kullanılmaktadır.
- Bu veri türü nesneyi oluşturacak olan verileri ve fonksiyonları birlikte barındırmaktadır.
- Sınıflar nesnelerin modelidir.
- Nesne tabanlı programlama mantığından bir defa sınıf yazılır ve ihtiyaç olduğunda tekrar tekrar kullanılır.

- Bir kişi (insan) türünde bir nesne oluşturmak istiyoruz.
- Bunun için öncelikle bir sınıf oluşturunuz.
- İnsan çocuk olarak dünya geldiğinde öncelikle adı ve soyadı belirlenir.
- Adi ve soyadi adında iki özelliği olmalı.
- Gerçek hayattaki adının konunmasına benzer şekilde metot (olay, etkinlik) tanımlanmalı ?
- Senin adın nedir ? Adını söyleyebilmeli (bilgisayar dilinde ekrana yazdırmalı)
- Senin adın var mı ? Adın konuldu mu konulacak mı ?

Visual Studio ile C# dilinde Console App Oluşturma

Create a new project

Recent project templates

- Console App C#
- Windows Forms App (.NET Framework) C#
- Windows Desktop Application C++
- Console App (.NET Framework) C#
- Console App C++
- Empty Project C++

console x

Clear all

C# All platforms All project types

Console App
A project for creating a command-line application that can run on .NET Core on Windows, Linux and macOS
C# Linux macOS Windows **Console**

Console App (.NET Framework)
A project for creating a command-line application
C# Windows **Console**

Other results based on your search

Console App
A project for creating a command-line application that can run on .NET Core on Windows, Linux and macOS
Visual Basic Linux macOS Windows **Console**

Console App
Run code in a Windows terminal. Prints "Hello World" by default.
C++ Windows **Console**

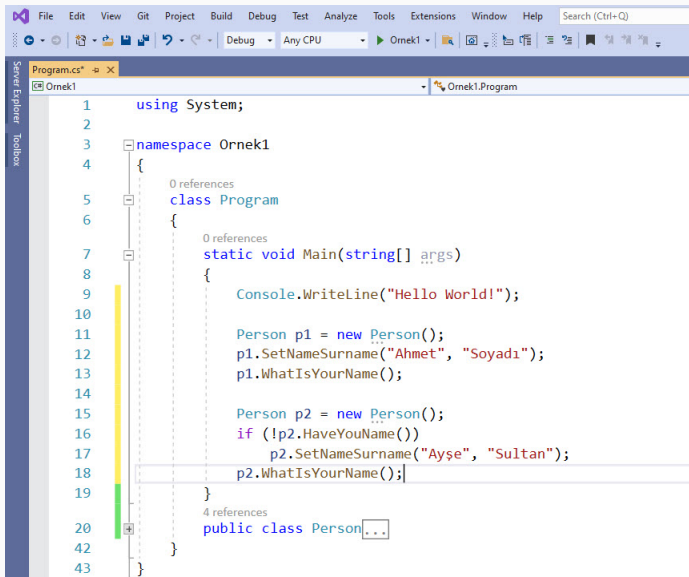
Console App (.NET Framework)

Next

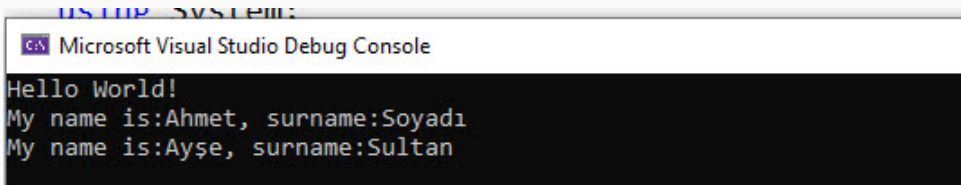
Person Class

```
,  
4 references  
public class Person  
{  
    string name, surname;  
    2 references  
    public void SetNameSurname(string name, string surname)  
    {  
        this.name = name;  
        this.surname = surname;  
    }  
    2 references  
    public void WhatIsYourName()  
    {  
        Console.WriteLine("My name is:" + this.name + ", surname:" + this.surname);  
    }  
    1 reference  
    public bool HaveYouName()  
    {  
        bool found = false;  
        if (string.IsNullOrEmpty(this.name))  
            found = false;  
        else  
            found = true;  
        return found;  
    }  
}
```

Source Code



```
1  using System;
2
3  namespace Ornek1
4  {
5      0 references
6      class Program
7      {
8          0 references
9          static void Main(string[] args)
10         {
11             Console.WriteLine("Hello World!");
12
13             Person p1 = new Person();
14             p1.SetNameSurname("Ahmet", "Soyadı");
15             p1.WhatIsYourName();
16
17             Person p2 = new Person();
18             if (!p2.HaveYouName())
19                 p2.SetNameSurname("Ayşe", "Sultan");
20             p2.WhatIsYourName();
21         }
22     }
23
24     4 references
25     public class Person...
```



The screenshot shows a window titled "Microsoft Visual Studio Debug Console". The console has a black background with white text. The output consists of three lines: "Hello World!", "My name is:Ahmet, surname:Soyadı", and "My name is:Ayşe, surname:Sultan".

```
DEBUG SYSTEM:  
Microsoft Visual Studio Debug Console  
Hello World!  
My name is:Ahmet, surname:Soyadı  
My name is:Ayşe, surname:Sultan
```

- Programda p1 ve p2 olmak üzere iki adet nesne tanımlanmıştır.
- Bu nesnelerin üye fonksiyonları çağrılarak nesnelerin belirli davranışlarda bulunması sağlanmıştır.
- Nesnelerin metotlarının çağırılması (canlandırılması) o nesneye mesaj göndermek olarak da ifade edilmektedir.
- Sınıflar aktif veri türüdür. Sınıflardan nesneler üretilir.
- Nesnelere mesaj gönderildiğinde nesneler bir davranışta bulunarak cevap verir.

- Sınıflar hem verileri hemde bu veriler ile işlem yapan fonksiyonları aynı paket içinde barındırır.
- Bu barındırma işlemi encapsulation (kapsülleme) olarak tanımlanır.
- Kapsülleme, C# bağlamında, bir nesnenin, kullanıcısı için gerekli olmayan verileri ve davranışları gizleme yeteneğini ifade eder. Kapsülleme, bir grup özellik, yöntem ve diğer üyelerin tek bir birim veya nesne olarak kabul edilmesini sağlar.

Tür belirtmeden nesne tanımlama

- Derleyiciler int, float, string gibi veri türleri için nasıl kullanılıyorsa nesne tanımlamak için aynı şekilde kullanılır.
- C# diline nesne oluşturmak için new terimi kullanılır.

```
1 using System;
2
3 namespace Ornek1
4 {
5     0 references
6     class Program
7     {
8         0 references
9         public void OrnekTanimla()
10        {
11            int i = 0;
12            Person p = new Person();
13
14            var i1 = 0;
15            var p1 = new Person();
16        }
17
18        0 references
19        static void Main(string[] args)
20        {
21        }
22
23        7 references
24        public class Person
25        {
26        }
27    }
28
29
30
31 }
```

Tür belirterek tanımlama

Tür belirtmeden tanımlama

Nesnelerin Dizi Olarak Tanımlanması

The screenshot shows a C# IDE with the following code in `Program.cs`:

```
1 using System;
2
3 namespace Ornek1
4 {
5     0 references
6     class Program
7     {
8         0 references
9         public void DiziTanimla()
10        {
11            int[] i = new int[2];
12            i[0] = 0;
13            i[1] = 1;
14
15            Person[] p = new Person[2];
16            p[0] = new Person();
17            p[1] = new Person();
18        }
19        0 references
20        static void Main(string[] args) {...}
21        8 references
22        public class Person {...}
23    }
24 }
```

Annotations in the image:

- A red arrow points from `new int[2];` to the text: **New teriminden dolayı i değişkeni aslında bir nesnedir.**
- A red arrow points from `p[0] = new Person();` and `p[1] = new Person();` to the text: **Dizinin her bir ögesi için yeni nesne üretilir**

Sınıf üyelerine erişimin denetlenmesi

- Yazılım geliştiriciler (programcılar) sınıfların oluşturulması ve kullanılması açısından ikiye ayrılır.
- İlk grup bu sınıfları yazar. İkinci grup ise bu sınıfları kullanarak nesne oluşturur. İkinci grup sınıfların orjinal yapısına asla müdahale edemez. Sadece sınıf tanımlayıcısının yani birinci grubun izin verdiği ölçüde nesneleri kullanabilir.
- Günümüz şartlarındaki backend, frontend terimine benzetilebilir. Backend arkaplanda servis yazar veritabanına bağları verileri çeker. Veritabanı vb. bütün işlemleri backend yapar.
- Frontend tarafında ise servis kullanılır. Yani aslında servis (class) türünden bir nesne üretilir ve o nesnenin ilgili evantı çağrılır (mesaj) gönderilir. Gelen bilgi frontend tarafından tasarlanan kullanıcı dostu arayüz ile kullanıcılara sunulur.
- Sınıfların kullanımı da benzer şekildedir. İlk grup oluşturur. İkinci grup kullanır.

Public, Protected, private

- Sınıfların özelliklerin sağlamak için sınıfı yazan yazılımcı sınıfın bazı üyelerini (özellik, fonskiyon, procedure vb.) gizleyerek (data hiding) onlara sınıf dışından erişilmesini engelleyebilir.
- Bir sınıfın içindeki üyelere erişimi denetleyen üç farklı etiket bulunmaktadır. Bu özellikler, **public** (açık), **protected** (korumalı), **private** (özel).
- **Public** olarak tanımlanan üyerele bu sınıftan türetilen bütün nesneler ve sınıflar erişebilir.
- **Protected** olarak tanımlanan üyelere sadece bu sınıftan kalıtım (inheritence) yoluyla üretilen sınıflar erişebilir. Inheritance ilerleyen haftalarda full detayıcı ile anlatılacak.
- **Private** olarak tanımlanan üylere sadece bu sınıfın içerisinden erişebilir. Bu sınıftan türetilen nesneler ve sınıflar erişemez.

- Bir öğrenci olduğunu ve öğrencilerinin sadece isim bilgisinin tutulduğunu varsayalım.
- Öğrencinin adı dışardan set edilebilsin ama dışardan hiç bir şekilde okunamasın. Sadece kalıtım yoluyla öğrencinin adının ne olduğu öğrenilebilsin.
- Öğrencinin adının kaç karakter olduğunu öğrenilebilsin.
- Öğrenci türünde bir sınıfımızın olması gerektiğini anlıyoruz.
- adı isminde bir özellik olmalı ve erişim olmaması için **private** olması gerektiği anlıyoruz. İsmi set etmek için **public** bir procedüre, ismini göstermek için **protected** bir procedure ve isminin kaç karakter olduğunu göstermek için de **public** bir fonksiyon yazmamız gerektiğini anlıyoruz.

Öğrenci classı

Controls in
1 onto
toolbox.

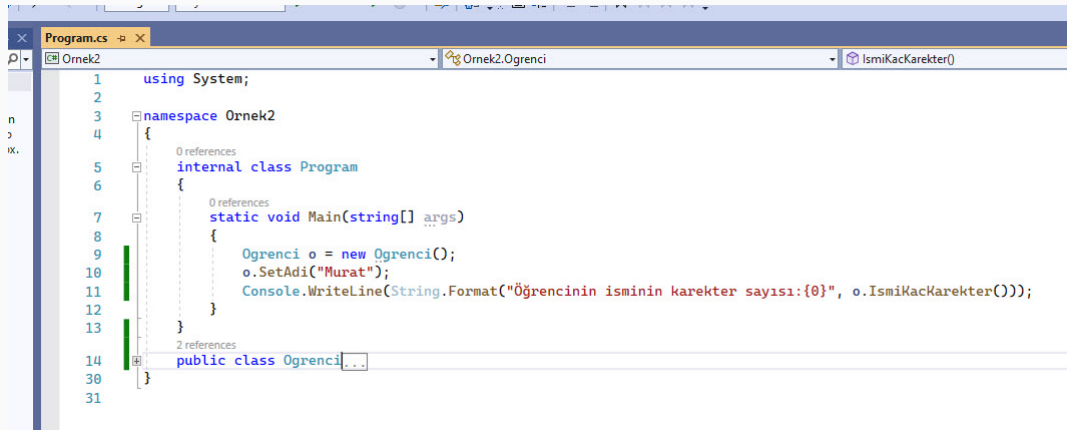
```
1 using System;
2
3 namespace Ornek2
4 {
5     internal class Program...
6     public class Ogrenci
7     {
8         private string adi;
9         public void SetAdi(string adi)
10        {
11            this.adi = adi;
12        }
13        protected void ShowName()
14        {
15            Console.WriteLine(String.Format("Öğrencinin adı:{0}", adi));
16        }
17        public int IsmiKacKarakter()
18        {
19            return this.adi.Length;
20        }
21    }
22 }
23
24
25
26
27
28
29
30
```

Öğrenci classının üyelerine erişim

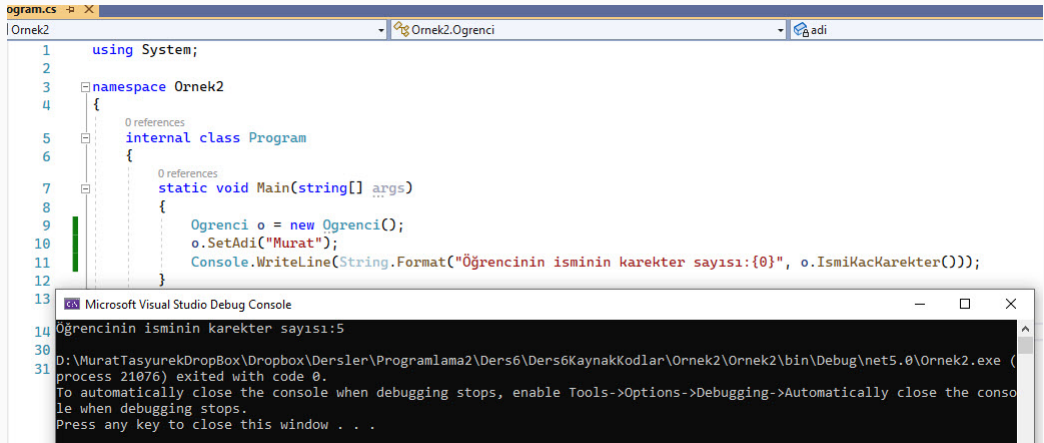
The screenshot shows the Visual Studio IDE with a C# project named 'Ornek2'. The code is as follows:

```
1 using System;
2
3 namespace Ornek2
4 {
5     internal class Program
6     {
7         static void Main(string[] args)
8         {
9             Öğrenci o = new Öğrenci();
10            o.
11        }
12    }
13    public class Öğrenci
14    {
15        private string _adi;
16        public string Ad { get { return _adi; } set { _adi = value; } }
17        public void SetAdi(string adi)
18        {
19            _adi = adi;
20        }
21        protected void ShowName()
22        {
23            Console.WriteLine(String.Format("Öğrencinin adı:{0}", _adi));
24        }
25    }
26 }
```

A red arrow points from the 'SetAdi' method in the 'Öğrenci' class to a tooltip that says 'void Öğrenci.SetAdi(string adi)' and '★ IntelliCode suggestion based on this context'. A text box with a red border and shadow is overlaid on the right side of the image, containing the text: 'adi özelliği ve ShowName fonksiyonu görünmüyor.'



```
1  using System;
2
3  namespace Ornek2
4  {
5      0 references
6      internal class Program
7      {
8          0 references
9          static void Main(string[] args)
10         {
11             Ogrenci o = new Ogrenci();
12             o.SetAdi("Murat");
13             Console.WriteLine(String.Format("Öğrencinin isminin karakter sayısı:{0}", o.IsmiKacKarakter()));
14         }
15     }
16     2 references
17     public class Ogrenci { ..
18
19
20
21
22
23
24
25
26
27
28
29
30
31 }
```



The screenshot shows a Visual Studio window with a C# file named `ogram.cs`. The code defines a namespace `Ornek2` containing an internal class `Program` with a `Main` method. The `Main` method creates an `Ogrenci` object, sets its `Adi` property to "Murat", and prints the character count of the name. The output window shows the result of the execution.

```
1 using System;
2
3 namespace Ornek2
4 {
5     0 references
6     internal class Program
7     {
8         0 references
9         static void Main(string[] args)
10        {
11            Ogrenci o = new Ogrenci();
12            o.SetAdi("Murat");
13            Console.WriteLine(String.Format("Öğrencinin isminin karakter sayısı:{0}", o.IsmiKacKarakter()));
14
15        }
16    }
17 }
```

Microsoft Visual Studio Debug Console

```
Öğrencinin isminin karakter sayısı:5
D:\MuratTasyurekDropBox\Dropbox\Dersler\Programlama2\Ders6\Ders6KaynakKodlar\Ornek2\Ornek2\bin\Debug\net5.0\Ornek2.exe (
process 21076) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

- Bir sınıfın üyelerinin dışarıya erişim tipi normalde **private** (özel) moddadır.
- Sınıf oluşturan kişi sınıfın üyelerini dışarıya açmak istiyorsa **public** veya **protected** ile **yetki** çerçevesinde açar.
- Benzer şekilde sınıf tanımlarken **public** ibaresi kullanılmadan **class Öğrenci** formatında sınıf tanımlarsanız bu sınıfta **private** modda olur. Böyle olduğu durumda kullanıldığı blok dışında diğer sınıf veya kod blokları tarafından erişilemez.
- Sınıfın içerisine başka bir sınıf değişken olarak tanımlanabilir. Burada aynı özellikler geçerlidir. Sonuçta sınıfta bir tür değişkendir.

- Önceki öğrenci bilgilerine ilaveten bir öğrencinin iki tane velisinin olduğunu varsayalım.
- Veli bilgileri okunabilir ancak daha sonra değiştirilemez olduğunu farzedelim.
- Veli adının sadece kendi içerisinde değiştirebildiğini düşünelim.
- Bir öğrencinin velisinin bilgisinin dışardan erişilebilir olduğunu varsayalım.

```
2 references
public class Veli
{
    string veliAdi="Ahmet";
    0 references
    public string GetVeliAdi()
    {
        return this.veliAdi;
    }
    0 references
    private void SetVeliAdi(string adi)
    {
        this.veliAdi = adi;
    }
}
```

Public terimi kullanıldığından dışardan erişebilir.

Terim kullanılmadığından private'dır.

Öğrenci Sınıfı

2 references

```
public class Öğrenci
```

```
{
```

```
    private string adi;
```

```
    public Veli veli1= new Veli();
```

```
    public Veli veli2= new Veli();
```

0 references

```
    public void SetAdi(string adi)
```

```
    {
```

```
        this.adi = adi;
```

```
    }
```

0 references

```
    protected void ShowName()
```

```
    {
```

```
        Console.WriteLine(String.Format("Öğrencinin adı:{0}", adi));
```

```
    }
```

0 references

```
    public int IsmiKacKarakter()
```

```
    {
```

```
        return this.adi.Length;
```

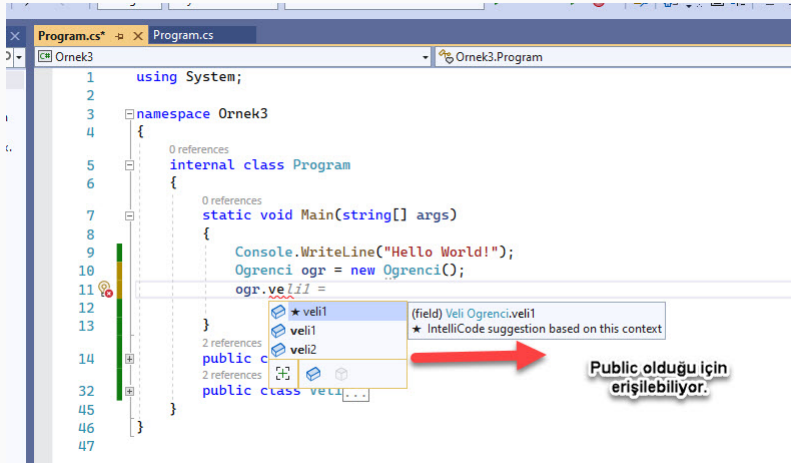
```
    }
```

```
}
```

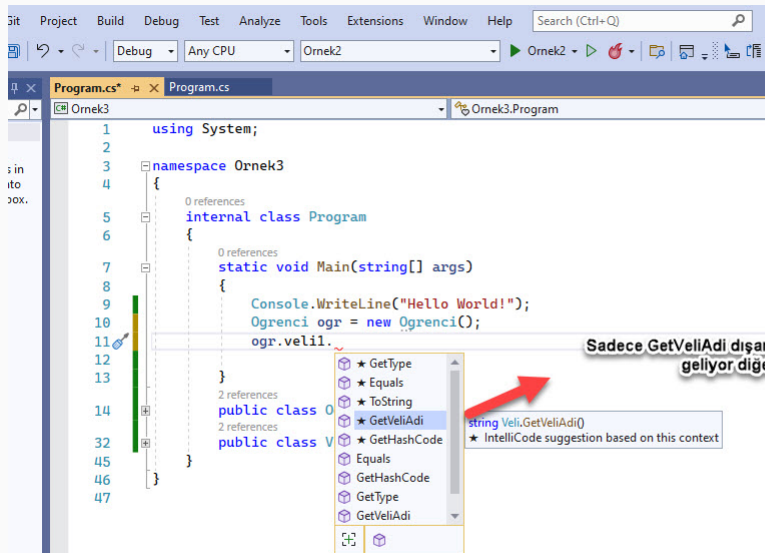


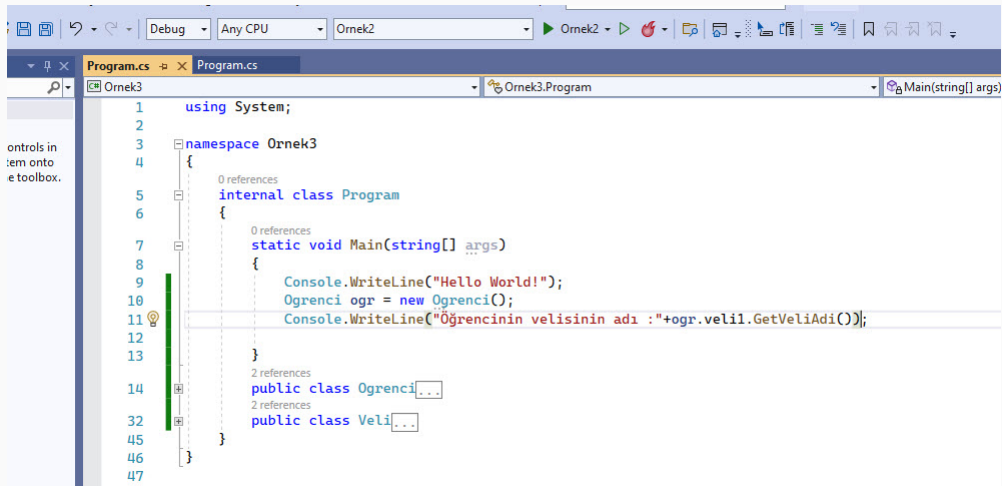
Veli türünde iki adet öznitelik. Public olduğundan dışardan erişilebilir.

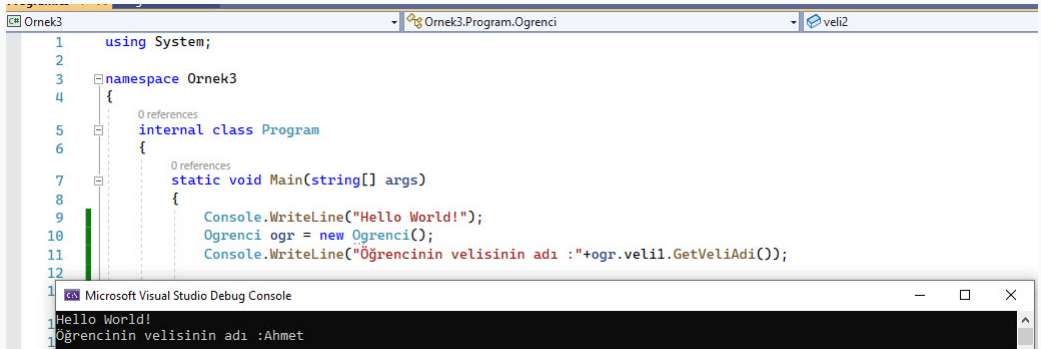
Öğrencinin velisine erişim



Öğrencinin velisinin adına erişim







The screenshot shows a Visual Studio window with a C# file named 'Ornek3'. The code defines a namespace 'Ornek3' containing an internal class 'Program' with a static 'Main' method. The 'Main' method prints 'Hello World!' and the name of a student object 'ogr' (Ahmet). The output window at the bottom shows the execution results.

```
1 using System;
2
3 namespace Ornek3
4 {
5     0 references
6     internal class Program
7     {
8         0 references
9         static void Main(string[] args)
10        {
11            Console.WriteLine("Hello World!");
12            Ogrenci ogr = new Ogrenci();
13            Console.WriteLine("Öğrencinin velisinin adı :"+ogr.veli1.GetVeliAdi());
14        }
15    }
16 }
```

Microsoft Visual Studio Debug Console

```
1 Hello World!
1 Öğrencinin velisinin adı :Ahmet
```

- Ödev: kişi ve araç bilgilerinin tutulduğu bir sistem isteniyor. Bir kişinin birden çok arabası olabilir.
- Araç ise taksi, kamyon, otobüs vb. bir çok araç olabilir. Ayrıca, araç ile ilgili modeli, yılı, markası, edinme tarihi ve edinme fiyatı tutulmalıdır.
- Kişi bilgisi olarak kişinin adı, soyadı, doğum yılı, TC Kimlik numarası tutulmak isteniyor. Ancak bu bilgiler gizli olmalı.
- Dört adet farklı türden aracınızın olduğunu ilk aracın 4., ikinci aracın 5., üçüncü aracın 6. ve dördüncü aracın 3. sahibi yani sizden önceki 2 sahibi vardı.
- Kişi arabalarını listele dediğinde aracın şasi numarası: ***A****9 şeklinde, Murat Taşyürek, Edinme Tarihi:01.01.2020, Model Yılı: Bir önceki sahibinin adı:... şeklinde göstermeli
- Bütün veriler dinamik olarak girilmeli, ödevler özgün olmalı, bütün işlemler class içerisinde olmalı
- Aynı ödevler **sıfır** puan olarak değerlendirilecektir.
- 16 Ekim 06.59'a kadar gönderilenler **8+2** puan, bu tarihten sonra vize sınavına kadar gönderenler **8** puan (kayubmprogramlama1@gmail.com).