

SINIFLARDA ALAN VE ÖZELLİK KULLANIMI

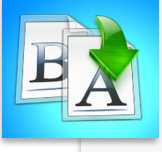


Atatürk Üniversitesi
Açıköğretim Fakültesi

NESNE TABANLI PROGRAMLAMA I

Öğr. Gör.
Gökhan TUTAR

İÇİNDEKİLER



- Sınıflarda Alanlar
- Alanların Kullanımı
- Sınıflarda Özellikler
- Salt Okunur Özellik Oluşturma
- Sadece Yazılabilir Özellik Oluşturma
- Doğrulamalı Özellik Oluşturma
- Otomatik Eşlenen Özellik Oluşturma



HEDEFLER

- Bu üniteyi çalıştıktan sonra;
 - Sınıflarda alanların kullanımı öğrenebilecek,
 - Özellikler hakkında bilgi sahibi olabilecek,
 - Salt okunur ve sadece yazılabilir özellikler oluşturulabilecek,
 - Doğrulamalı özellikler oluşturulabilecek,
 - Otomatik eşlenen özellikler oluşturabileceksiniz.

ÜNİTE 6



GİRİŞ

Önceki ünitelerde nesne tabanlı programlamada sınıf yapısı, sınıf türleri, sınıf nesne ilişkisi anlatılmıştı. Bu ünite de ise sınıflar içerisinde bulunan alanlar ve özelliklerin kullanımları anlatılacaktır. Alanlar ve özellikler sınıfta en çok kullanılan yapılar arasındadır. Nesne tabanlı programlamada kullanılan alan ve özellikler sınıf yapısı ile birlikte kullanılmaktadır.

Alanlar bir sınıf içerisinde bulunan değişkenlere erişmek için kullanılmaktadır. Özellikler ise sınıf içindeki değişkenlere kontrollü erişimi sağlamaktadır. Bundan dolayı güvenlik ve kullana bilirlik açısından alanlar ve özellikler konusu nesne tabanlı programlamada büyük önem taşımaktadır.

Nesne tabanlı programlamada her şey nesnelerden ibarettir. Bu ifadeyi gerçek dünyada düşünürsek bir kalemde silgide birer nesnedir. Ancak bu kalem ile silginin aynı şey olduğunu anlamına gelmez. Kalem ile silgiyi ayırt eden şey özelliklerdir. Kalem özellikleri genellikle ince uzundur, el ile tutulabilecek boyuttur. Bu özellikleri istediğimiz kadar arttırabiliriz. Silgi ise genellikle kaleme göre kısadır, kaleme göre daha kalındır. Silginin de bu özellikleri istediğimiz kadar arttırabiliriz. Nesne tabanlı programlamada da nesnelerin kendilerine has özellikleri bulunmaktadır. Tabi gerçek dünya ile bilgisayar arasındaki fark, bilgisayarda nesneyi ve özelliklerini programcı tarafından oluşturulmasıdır. Bilgisayarda herhangi bir sınırlayıcı bulunmamaktadır. İstenilen isimde ve özellikte nesne oluşturulabilir.

Bilgisayar ortamının bu kadar özgür olması bazen problemler meydana getirebilmektedir. Yani bir projede birden fazla kişi aynı anda çalışabilir. Herkesin bütün nesnelere erişim nesne özelliklerini değiştirebilmesi karışıklığa ve yetkisiz erişimlere sebep olabilir. Dolayısıyla nesne ve özellik kullanımı büyük önem teşkil etmektedir. Bu ünite de sınıf içerisinde alan ve özelliklerin kullanımı örnekler verilerek anlatılacaktır.

SINIFLARDA ALANLAR


Nesne tabanlı programlamada sınıflar soyut yapılardır. Yani başka bir yapı ile ilişkilendirilmedikleri sürece tek başlarına koddurlar. Tetikleme olmadıkça veri taşımaz ve çalışmazlar. Nesne tabanlı programlamada nesneyi oluşturan iki temel yapı vardır. Bunlardan birisi veri diğeri ise metottur. Veri bir değişenin değeridir. Yani ad, soyadı, yaş veya kilo gibi değerlerin bulunduğu yapıdır. Metot ise belirlenen işlem veya işlemleri gerçekleştiren kod bloklarıdır.

Nesne tabanlı programda alan (fields), sınıf içerisinde tanımlanan değişkenlerdir. Alan kelimesi İngilizcede “fields” olarak geçmektedir. Sınıflarda tanımlanan değişkenin tipi (string, integer gibi) fark etmeksizin bütün değişkenler alandır. Metotların içerisinde tanımlanan değişkenler alan değildir. Alanlar birden fazla metotta kullanılmak için tanımlanır. Zaten sadece bir metotta kullanılacak değişken metottun içerisinde tanımlanmalıdır. Alan tanımlamanın en basit örneği



Nesne tabanlı programda alan, sınıf içerisinde tanımlanan değişkenlerdir.

aşağıda verilmiştir. Örnekte “Ogrenciler” adındaki sınıfa “yas” adında bir alan tanımlanmıştır.



Örnek

- Örnek alan:
- `public class Ogrenciler`
- `{`
- `public int yas = 7; // Alan`
- `}`

Yukardaki örneğe dikkat edilirse “Ogrenciler” sınıfı ve “yas” alanı tanımlanırken önüne “public” kelimeleri konmuştur. Bunlar erişim belirleyicilerdir. Yani sınıfa veya alana kimlerin erişebileceğini belirlerler. Örnekte kullanılan public erişim belirleyicisi herkese açık anlamındadır. Yani “Ogrenciler” sınıfına ve “yas” alanına herkes erişebilmektedir. Erişim belirleyiciler konusunu daha detaylı olarak sonraki ünitelerde işlenecektir.

Alanlara Erişim Şekilleri

Alanlara hem sınıf içerisinde hem de sınıf dışından erişim verebilmek mümkündür. Yani sınıfın içerisinde bir metod alana ulaşabilmektedir. Alanları sınıf içerisinde tanımlanan değişkenler olarak tanımlamıştık. Ancak metod içerisinde tanımlanan değişkenler alan değildir.

Sınıf içerisinde alanlara erişim

Alanlara sınıf içerisinde erişmek için “this” ifadesi kullanılmaktadır. This ifadesi İngilizcede “bu” anlamındadır. This ifadesini kullanımındaki amaç alanın bulunduğu sınıfı ifade etmektedir. Sınıf içerisinde alanlara erişimde herhangi bir engel bulunmamaktadır. Sınıfın içerisindeki bütün metodlar bulundukları sınıfın bütün alanlarına (fields) erişebilirler.

```
public class Ogrenciler
{
    public int yas = 7;
    public int getirYas()
    {
        return this.yas;
    }
}
```

Yukarıdaki örnekte “Ogrenciler” adında bir sınıf tanımlanmıştır. Bu sınıfın altında da “yas” adında bir alan ve “getirYas” adında da bir metod tanımlanmıştır. Metodun içerisinde ise “yas” alanı geri döndürülmüştür. Dikkat edilirse “yas” alanını kullanmak için önüne “this” ifadesi kullanılmıştır. Eğer “this” ifadesi kullanılmazdı aynı zamanda da metodun içerisinde “yas” değişkenini olsaydı program metodun içerisindeki “yas” değişkenini kullanacaktır.



Alanlara sınıf içerisinde erişmek için “this” ifadesi kullanılmaktadır.

Sınıfın içerisinde tanımlanan alanın adı ile metodun içerisinde tanımlanan değişkenin adı aynı olabilir. Bu durumda kod her hangi bir hata vermeyecektir.

Ancak sınıfın içerisinde tanımlanan alanı çağırmak için “this” ifadesi kullanılmalıdır. Eğer kullanılmazsa değer metot içerisindeki değişken olarak algılanır.

Alanlara sınıf içerisinde erişilebildiği gibi değerleri de değiştirilebilmektedir. Sınıf içerisinde bulunan bir alanın değerini değiştirmek için tıpkı erişmek için kullanılan “this” ifadesi kullanılmaktadır.

```
public class Ogrenciler
{
    public int yas = 7;//Alan
    public int getirYas()
    {
        this.yas = 9;
        return this.yas;
    }
}
```

Yukardaki örnekte “Ogrenciler” adında bir sınıf tanımlanmış, sınıfı içerisinde de “yas” adından bir alan ve “getirYas” adında da bir metot tanımlanmıştır. Daha sonra “getirYas” metodu içerisinde “yas” alanının değeri “9” olarak değiştirilmiştir.

```
public class Ogrenciler
{
    public int yas = 7;//Alan
    public int getirYas()
    {
        int yas = 8;//Metot değişkeni
        return yas;
    }
}
```



Alanlara sınıf dışından erişmek için öncelikle sınıf tanımlanmalıdır.

Yukardaki örnekte iki tane “yas” adında değişken bulunmaktadır. Bunlardan bir tanesi “Ogrenciler” sınıfının içerisinde diğer ise “getirYas” metodunun içerisinde. Alan (field) olan “yas” değişkenine her yerden (erişim belirleyicilere bağlı olarak) erişilebilmektedir. Ancak metodun içerisindeki “yas” değişkenine sadece metot içerisinde erişilebilir. Ayrıca “getirYas” metodunun dönüş değeri olan “yas” değişkeni, “Ogrenciler” sınıfının alanı (field) değildir. Bundan dolayı “getirYas” metodunun dönüş değeri “8”dir. Eğer “getirYas” metodunun dönüş değerini “return this.yas” yazılıyorsa bu defa dönüş değeri “7” olurdu.

Sınıf dışından alanlara erişim

Alanlara erişim sınıf içerisinde direkt yapılmaktadır. Ancak sınıf dışından alana erişim için biraz daha farklıdır. Çünkü önceden de belirtildiği gibi sınıflar başlı başına kod parçalarıdır çağrılmadıkça çalışmazlar. Sınıfta bulunan alana erişmek için öncelikle o sınıf tanımlanmalıdır.

```
namespace WinFormsApp1
{
    public partial class Form1 : Form
    {
        public class OgrencilerSinif
        {
            public int yas = 7; //Alan
        }

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            OgrencilerSinif OgrencilerDegisken = new OgrencilerSinif();
            MessageBox.Show(OgrencilerDegisken.yas.ToString());
        }
    }
}
```

1

2

3

Yukarıdaki örnekte 1 numaralı alanda “OgrencilerSinif” adında bir sınıf tanımlanmış ardından sınıf içerisine de “yas” adında bir alan tanımlanmış bu alana “7” değeri atanmıştır. Örnekteki amacımız “OgrencilerSinif” sınıfı içerisinde tanımlanan “yas” alanına sınıf dışından erişmektedir. Örnekte 2 numaralı alanda da önceden tanımlanan “OgrencilerSinif” adındaki sınıf oluşturularak “OgrencilerDegisken” adındaki değişkenine atanmıştır. Yani artık “OgrencilerSinif” adındaki sınıfın bütün elemanlarına (erişim belirleyiciler doğrultusunda) “OgrencilerDegisken” aracılığıyla erişebiliriz. Son olarak 3 numaralı alanda “yas” alanına erişmek için “OgrencilerDegisken.yas” yazılmıştır. İfadenin sonunda bulunan “.ToString” ifadesinin amacı “yas” alanını string yani metin haline

dönüştürmektedir.

```
namespace WinFormsApp1
{
    public partial class Form1 : Form
    {
        public class OgrencilerSinif
        {
            public int yas = 7; //Alan
            public int getirYas()
            {
                return this.yas;
            }
        }
        public Form1()
        {
            InitializeComponent();

            private void Form1_Load(object sender, EventArgs e)
            {
                OgrencilerSinif OgrencilerDegisken = new OgrencilerSinif();
                OgrencilerDegisken.yas = 10;
                MessageBox.Show(OgrencilerDegisken.yas.ToString());
            }
        }
    }
}
```



Alanlarda yalnızca okunabilir veriler oluşturmakta mümkündür.

Alanların değerleri dışardan erişilerek değiştirilme imkânı da bulunmaktadır. Bunun için öncelikle sınıf tanımlanmalı ardından sınıftaki alana değer atanmalıdır. Yukardaki örnekte “OgrencilerDegisken” değişkenine “OgrencilerSinif” sınıfı tanımlanmış ardından “yas” alanının değeri “10” olarak güncellenmiştir.

Alanlarda yalnızca okunabilir veriler oluşturmakta mümkündür. Yani sınıf içerisinde tanımlanan alana tanımlanırken değer atanır daha sonra bu değer değiştirilmesi engellenebilmektedir. Sınıfta bulunan bir alanı sadece okunabilir yapmak için alanın önüne “readonly” yazılması yeterlidir.

```

namespace WinFormsApp1
{
    public partial class Form1 : Form
    {
        public class OgrencilerSinif
        {
            public readonly int yas = 7; //Alan
            public int getirYas()
            {
                return this.yas;
            }
        }
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            OgrencilerSinif OgrencilerDegisken = new OgrencilerSinif();

            OgrencilerDegisken.yas = 10;

            Console.WriteLine(OgrencilerDegisken.yas.ToString());
        }
    }
}

```

Yukardaki “OgrencilerSinif” sınıfında tanımlanan “yas” değişkeni “readonly” olarak tanımlandığı için bu alanın değeri daha sonradan değiştirilemez. Bundan dolayı yukardaki örnek hata verip çalışmayacaktır.

Sınıf içerisinde tanımlanan alan isimleri eşsiz olmak zorundadır. Alan isimleri eşsiz olmazsa program hata verip çalışmayacaktır.

<pre> public class Ogrenciler { public int yas = 7; //Alan public int yas = 9; //Alan } </pre>	1
<pre> public class OgrencilerYeni { public int yas = 11; //Alan } </pre>	2



Özellikler, alanlara erişmek ve bunlara değer atamak için erişimciler olarak da bilinen “get” ve “set” yapılarını kullanırlar.

Yukardaki örnekte “Ogrenciler” sınıfında iki tane “yas” alanı tanımlanmıştır. Bu yanlış kullanımdır ve program hata verecektir. Ancak iki farklı sınıfta aynı adda alanlar bulunabilir. Yani “Ogrenciler” sınıfında bir tane “yas” adında bir alan ve “OgrencilerYeni” sınıfında da bir tane “yas” alanı tanımlanabilmektedir. Böyle bir durumda herhangi bir sorun oluşmaz.

SINIFLARDA ÖZELLİKLER

Özellikler, alanların değerlerini okumak, yazmak veya hesaplamak için esnek bir mekanizma sağlayan sınıfların bir üyesidir. Başka bir deyişle, özellikleri kullanarak özel alanlara erişebilir ve değerlerini ayarlayabiliriz. Özellikler her

zaman genel veri üyeleridir. Özelliklerin İngilizcede “properties” olarak geçmektedir.

Özellikler, alanlara erişmek ve bunlara değer atamak için erişimciler olarak da bilinen “get” ve “set” yapılarını kullanırlar. Bir özelliğin get ve set bölümlerine veya bloklarına erişimciler denir. Bunlar, bir özelliğin erişilebilirliğini kısıtlamak için kullanışlıdır, set erişimcisi, bir özellikteki özel bir alana bir değer atayabileceğimizi belirtir ve set erişimcisi özelliği olmadan, salt okunur bir alan gibidir.

Nesne tabanlı programlamada özellikler verilere yetkili ve kontrollü erişim için büyük önem taşımaktadır.

```
namespace WinFormsApp1
{
    public partial class Form1 : Form
    {
        public class Kisiler 1
        {
            private string AlanIsim; //Alan
            public string Isim //Özellik
            {
                get { return AlanIsim; }
                set { AlanIsim = value; }
            }
        }

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            Kisiler k = new Kisiler();
            k.Isim = "Ahmet"; 2
            MessageBox.Show(k.Isim);
        }
    }
}
```

Özelliklerin (properties) en basit şekli yukardaki örnekte 1 numaralı kısımda gösterilmektedir. Örnekte 1 numaralı kısımda, “AlanIsim” adında bir tane alan (field) bir tane de “Isim” adında bir özellik (property) tanımlanmıştır. Örneğe dikkat edilirse “AlanIsim” alanının erişim belirleyicisi “private” olarak tanımlanmıştır. Yani “AlanIsim” alanına sınıf dışından erişim mümkün değildir. Bu alana yazılımcıların kontrollü şekilde erişmesini sağlamak için “Isim” özelliği tanımlanmıştır. Özelliğin (properties) iki yapısı bulunmaktadır. Bunlar get ve set erişimcileridir. Örnekte de görülebileceği gibi “get” erişimcisi veriyi döndürürken “set” erişimcisi de değer atamak için kullanılmaktadır.



Özelliğe değer atanırken ayrı bir satırda yapılmak zorunda değildir.

Özellikler verilere kontrollü yetkili ve kontrollü erişim için kullanıldığı daha önceden de belirtilmişti. Yukardaki örnekte de “AlanIsim” adlı alana sınıf dışından erişmek mümkün olmadığı için özellik kullanılarak “AlanIsim” adlı alana erişilmiştir. Yukardaki örnekte 2 numaralı alanda öncelikle daha önceden oluşturulan “Kisiler” sınıfı “k” değişkenine atanmış daha sonra “Isim” adlı özelliğin değerine “Gökhan” değeri atanmıştır.

Yukardaki örnekte 2 numaralı kısımda “k.Isim = “Ahmet”;;” yazan satır çalıştırıldığı anda; “Kisiler” sınıfının “Isim” özelliğinin “set” erişimcisi çalıştırılır. Yani “AlanIsim” adlı alana “Ahmet” değeri atanır.

Yukardaki örnekte 2 numaralı kısımda “MessageBox.Show(k.Isim);” yazan satır çalıştırıldığı anda; “Kisiler” sınıfının “Isim” özelliğinin “get” erişimcisi çalıştırılır. Yani “AlanIsim” adlı alanın değeri geri döndürerek mesaj kutusunda gösterilir.

Özelliğe değer atanırken ayrı bir satırda yapılmak zorunda değildir. Sınıf tanımlanırken sınıfın özelliklerine değer atanabilir. Sınıf tanımlanırken özelliğe değer atamak için süslü parantezler “{}” içerisinde yapılmalıdır. Süslü parantezler arasında önce özelliğin adı yazılır daha sonra da alanın değeri yazılır. Tabi alan string tipinde ise değer tırnak işaretleri arasında yazılmalıdır.

namespace WinFormsApp1

```
{
    public partial class Form1 : Form
    {
        public class Kisiler
        {
            private string AlanIsim; //Alan
            private string AlanSoyIsim; //Alan
            public string Isim //Özellik
            {
                get { return AlanIsim; }
                set { AlanIsim = value; }
            }

            public string SoyIsim //Özellik
            {
                get { return AlanSoyIsim; }
                set { AlanSoyIsim = value; }
            }
        }
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            Kisiler k = new Kisiler() { Isim = "Ahmet", SoyIsim = "Yılmaz" };
            MessageBox.Show(k.Isim);
        }
    }
}
```



Salt okunur özellikler de oluşturulabilir.

Yukardaki örnekte “Kisiler” sınıfı “k” değişkenine atanırken aynı anda “Kisiler” sınıfında bulunan “Isim” ve “SoyIsim” özelliklerine değer atanmıştır.

Salt Okunur Özellik Oluşturma

Bazı durumlarda salt okunur (readonly) özellikler de oluşturulabilir. Salt okunur, bir özelliğin değerine erişilebilen ancak ona bir değer atanamayan anlamına gelir. Bir özelliğin “set” erişimcisi yoksa salt okunur bir özelliktir. Örneğin, “Kisiler” sınıfında yalnızca “get” erişimcisi varsa ve “set” erişimcisi olmayan bir “Isim” özelliği oluşturmak için aşağıdaki örneği inceleyelim.

```
namespace WinFormsApp1
{
    public partial class Form1 : Form
    {
        public class Kisiler
        {
            public string Isim //Özellik 1
            {
                get { return "Ahmet"; }
            }
        }

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            Kisiler k = new Kisiler();

            k.Isim = "Hasan" 2

            MessageBox.Show(k.Isim);
        }
    }
}
```

Yukardaki örnekte 1 numaralı kısımda “Kisiler” adında bir sınıf oluşturulmuştur. Bu sınıfta içerisinde de “Isim” adında bir özellik oluşturulmuştur. Ancak “Isim” özelliğinin sadece “get” erişimcisi tanımlanmıştır. Bundan dolayı yukardaki örnekte 2 numaralı kısım hata verecektir.

Sadece Yazılabilir Özellik Oluşturma

Sadece yazılabilir bir özellikler de oluşturulabilir. Sadece yazılabilir, bir özelliğe değer atanabilir ancak onun değerini alınamaz anlamına gelir. Bir özelliğin “get” erişimcisi yoksa sadece yazılabilir bir özelliktir. Örneğin, “Kisiler” sınıfında yalnızca “set” erişimcisi varsa ve “get” erişimcisi olmayan bir “Isim” özelliği oluşturmak için aşağıdaki örneği inceleyelim.

```

namespace WinFormsApp1
{
    public partial class Form1 : Form
    {
        public class Kisiler
        {
            private string AlanIsim; //Alan
            public string Isim //Özellik
            {
                set { AlanIsim = value; }
            }
        }

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            Kisiler k = new Kisiler();

            k.Isim = "Ahmet";

            MessageBox.Show(k.Isim);
        }
    }
}

```

1

2

Yukardaki örnekte 1 numaralı kısımda “Kisiler” adında bir sınıf oluşturulmuştur. Bu sınıfında içerisinde de “Isim” adında bir özellik oluşturulmuştur. Ancak “Isim” özelliğinin sadece “set” erişimcisi tanımlanmıştır. Bundan dolayı yukardaki örnekte 2 numaralı kısım hata verecektir.

Doğrulamalı Özellik Oluşturma

Özelliklerin diğer bir katkısı değer atanmadan önce bazı kontroller oluşturularak değerler doğrulanabilmektedir. Yani özelliğe değer atanmadan önce kontrol edilecek eğer değer kontrolden geçerse atanacak kontrolden geçemezse hata veya uyarı verdirilebilir.



Özelliklerin diğer bir katkısı değer atanmadan önce bazı kontroller oluşturularak değerler doğrulanabilmektedir.

```

namespace WinFormsApp1
{
    public partial class Form1 : Form
    {
        public class Notlar
        {
            private int AlanNot = 60; //Alan
        }

        public int Not //Özellik
        {
            get { return AlanNot; }
            set
            {
                if (value >= 50)
                {
                    AlanNot = value;
                }
                else
                {
                    MessageBox.Show("Not 50'den küçük olamaz");
                }
            }
        }
    }

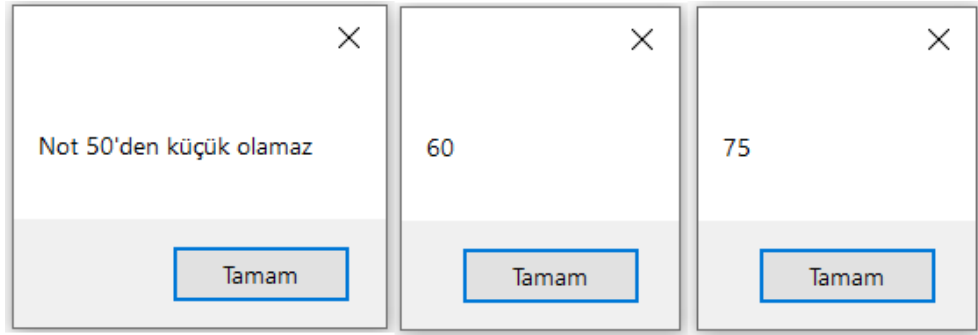
    public Form1()
    {
        InitializeComponent();
    }

    private void Form1_Load(object sender, EventArgs e)
    {
        Notlar k = new Notlar();

        k.Not = 40;
        MessageBox.Show(k.Not.ToString());
        k.Not = 75;
        MessageBox.Show(k.Not.ToString());
    }
}

```

Yukardaki örnekte 1 numaralı kısımda “AlanNot” adında bir alan tanımlanarak “60” değeri atanmıştır. Örneğin 2 numaralı kısmında ise “Not” özelliğinin “set” erişimcisinde atanan değer “50” den küçükse ekrana “Not 50’den küçük olamaz” mesajı verilmesi için gerekli kodlar oluşturulmuştur. Örneğin 3 numaralı alanında “Not” özelliğine “40” değerinin atanması istenmiştir. Ancak “set” erişimcisinde atacak değer “50” den küçük olduğu için ekrana “Not 50’den küçük olamaz” mesajı verilmiştir ve “Not” özelliğinin değeri değiştirilmeyerek tanımlanırken atanan değer olan “60” tır. Bundan dolayı örnekte 4 numaralı kısımda mesaj olarak “Not” özelliğinin değeri “60” olacaktır. Örnekteki 5 numaralı alanda atanan “75” değeri “50” den büyük olduğu için kontrole takılmadan değer olarak atanır. Son olarak 6 numaralı alanda “Not” özelliğinin son değeri olan “75” mesajı verilir. Yukarda bulunan örneğin ekran çıktısı Şekil 6.1.’de gösterilmiştir.



Şekil 6.1. Değer Atanmadan Önce Doğrulama Ekran Çıktısı

Özelliklere sadece değer atanmadan önce değil aynı zaman değer çağrılmadan öncede kontroller eklenebilmektedir. Yani değer çağrılmadan önce gerekli kontroller yapılarak farklı bir değer döndürülebilmektedir.



Özelliklere sadece değer atanmadan önce değil aynı zaman değer çağrılmadan öncede kontroller eklenebilmektedir.

```
namespace WinFormsApp1
{
    public partial class Form1 : Form
    {
        public class Notlar
        {
            private int AlanNot; //Alan
            public int Not //Özellik
            {
                get
                {
                    if (AlanNot > 90)
                    {
                        AlanNot = AlanNot + 5;
                    }
                    return AlanNot;
                }
                set
                { AlanNot = value; }
            }
        }

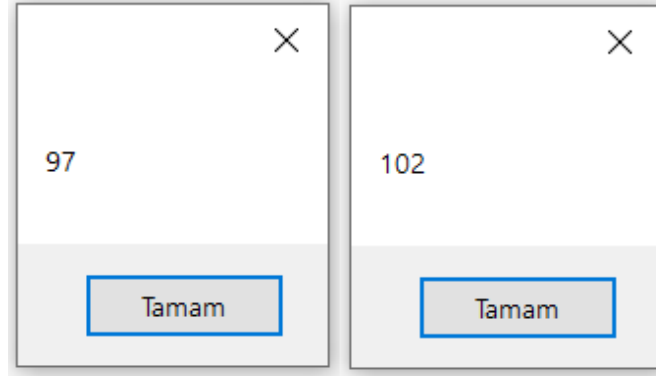
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            Notlar k = new Notlar();
            k.Not = 92; 2
            MessageBox.Show(k.Not.ToString()); 3
            MessageBox.Show(k.Not.ToString()); 4
        }
    }
}
```

Yukardaki örnekte 1 numaralı kısmında “Not” özelliğinin “get” erişimcisinde çağrılan değer “90” dan büyükse (yani 91 veya daha fazla) “AlanNot” alanının değeri 5 arttırılmıştır. Diğer durumlarda herhangi bir işlem yapılmadan “AlanNot”

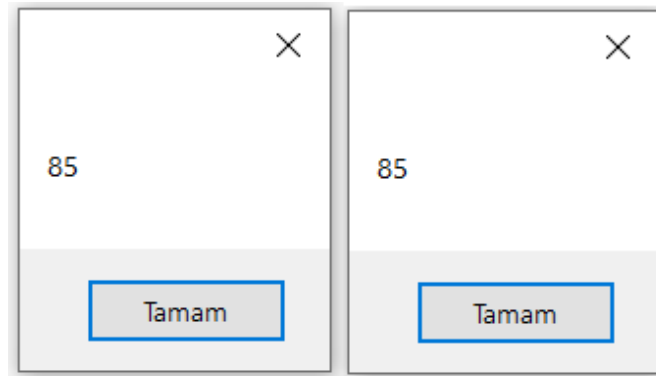
değeri döndürülmüştür. Örneğin 2 numaralı “Not” özelliğine “92” değeri atanmıştır. Örneğin 3 numaralı kısmında “Not” özelliği ekrana yazdırılmıştır.

Burada “Not” özelliğine atanan değer “90” dan büyük olduğu için atanan değere “5” eklenerek mesaj verilir. Bu işlem “Not” özelliği her çağrıldığında yapılacaktır. Yani yukardaki örnekte 4 numaralı kısımda “Not” özelliğine herhangi bir değer atanmadan sadece tekrar çağrılmıştır ancak burada da “Not” özelliğinin en son değeri “5” daha arttırılacaktır. Yani ilk uyarıda “Not” özelliğine atanan değer “5” fazlası olan “97” verilecek ardından da “Not” özelliğinin son değeri olan “97” ye “5” eklenerek “102” mesajı verilecektir.



Şekil 6.2. Değer Çağrılmadan Önce Doğrulama Ekran Çıktısı - 1

Şekil 6.2.’de örnek kodlar doğrultusunda “Not” özelliğine “92” değerinin atanmış halinin ekran çıktısı bulunmaktadır. Şekilde de görülebileceği gibi ilk olarak “Not” özelliğine atanan değer “5” fazlası yani “97” uyarısı verilmiştir. Ardında da “Not” özelliğinin son değerinden “5” fazlası olan “102” uyarısı verilmiştir.



Şekil 6.3. Değer Çağrılmadan Önce Doğrulama Ekran Çıktısı - 2

Şekil 6.3.’te örnek kodlar doğrultusunda 2 numaralı alanda “Not” özelliğine atanan değer “92” yerine “85” olarak kabul edilmiş ve buna göre ekran çıktısı bulunmaktadır. Şekilde de görülebileceği gibi ilk olarak atanan değer yani “85” değeri “90” değerinden küçük olduğu herhangi bir değişim olmadan “85” uyarısı verilmiştir. “Not” özelliğinin değeri değişmediği için ikinci mesajda da yine “85” verilmiştir.



Özellikler (properties) kullanılırken illa alanların (fields) beraberinde kullanılması zorunlu değildir.

Otomatik Eşlenen Özellik Oluşturma

Özellikler (properties) kullanılırken illa alanların (fields) beraberinde kullanılması zorunlu değildir. Yani özellikler tek başlarına kullanılarak değerler atanabilmektedir. Bunu yapılabilmesi için “get” ve “set” erişimcilerinde herhangi bir işlem yapmaya ihtiyaç yoktur. Bu şekilde özellik otomatik olarak eşlenerek değer atanır veya değere erişilebilir. Özelliklerin bu şekilde kullanımı diğer kullanımlara göre daha kısa ve hızlı yapılabilir.

Aşağıdaki örnekte 1 numaralı kısımda “Isim” adında bir özellik oluşturulmuştur. Özelliğe gelen değerler ve çekilecek değerler otomatik yapılması için “get” ve “set” erişimcilerine herhangi bir şey yazılmamıştır. Örneğin 2 numaralı kısımda ise “Notlar” sınıfı “k” değişkenine atanmış ardından “Isim” özelliğine “Ahmet” değeri atanmıştır. Son olarak da “Isim” özelliği uyarı olarak verilmiştir.

```
namespace WinFormsApp1
{
    public partial class Form1 : Form
    {
        public class Notlar
        {
            public string Isim //Özellik
            {
                get;
                set;
            }
        }
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            Notlar k = new Notlar();
            k.Isim = "Ahmet";
            MessageBox.Show(k.Isim);
        }
    }
}
```

1

2



Bireysel Etkinlik

- Ad, Soyad, Öğrenci Numaralarını alan özellik oluşturarak, özelliğe değer atarken öğrenci numarasının 1-100 arasında olma şartı ekleyiniz.



Özet

- Nesne tabanlı programlamada sınıflar soyut yapılardır. Yani başka bir yapı ile ilişkilendirilmedikleri sürece tek başlarına koddurlar.
- Nesne tabanlı programlamada nesneyi oluşturan iki temel yapı vardır. Bunlardan birisi veri diğeri ise metottur.
- Nesne tabanlı programda alan, sınıf içerisinde tanımlanan değişkenlerdir.
- Alan kelimesi İngilizcede “fields” olarak geçmektedir.
- Sınıfta tanımlanan değişkenin tipi (string, integer gibi) fark etmeksizin bütün değişkenler alandır.
- Alanlara hem sınıf içerisinde hem de sınıf dışından erişim verebilmek mümkündür.
- Alanlara sınıf içerisinde erişmek için “this” ifadesi kullanılmaktadır.
- Sınıf içerisinde alanlara erişimde herhangi bir engel bulunmaktadır.
- Sınıfın içerisindeki bütün metotlar bulundukları sınıfın bütün alanlarına (fields) erişebilirler.
- Sınıfın içerisinde tanımlanan alanın adı ile metodun içerisinde tanımlanan değişkenin adı aynı olabilir.
- Alanlara sınıf içerisinde erişilebildiği gibi değerleri de değiştirilebilmektedir. Sınıf içerisinde bulunan bir alanın değerini değiştirmek için tıpkı erişmek için kullanılan “this” ifadesi kullanılmaktadır.
- Alanlara erişim sınıf içerisinde direk yapılmaktadır.
- Ancak sınıf dışından alana erişim için biraz daha farklıdır. Çünkü önceden de belirtildiği gibi sınıflar başlı başına kod parçalarıdır çağrılmadıkça çalışmazlar.
- Sınıfta bulunan alana erişmek için öncelikle o sınıf tanımlanmalıdır.
- Alanlarda yalnızca okunabilir veriler oluşturmakta mümkündür.
- Sınıfta bulunan bir alanı sadece okunabilir yapmak için alanın önüne “readonly” yazılması yeterlidir.
- Sınıf içerisinde tanımlanan alan isimleri eşsiz olmak zorundadır. Alan isimleri eşsiz olmazsa program hata verip çalışmayacaktır.
- Özellikler, alanların değerlerini okumak, yazmak veya hesaplamak için esnek bir mekanizma sağlayan sınıfların bir üyesidir.
- Özellikler, alanlara erişmek ve bunlara değer atamak için erişimciler olarak da bilinen “get” ve “set” yapılarını kullanırlar.
- Bir özelliğin get ve set bölümlerine veya bloklarına erişimciler denir.
- Nesne tabanlı programlamada özellikler verilere yetkili ve kontrollü erişim için büyük önem taşımaktadır.
- Özelliğe değer atanırken ayrı bir satırda yapılmak zorunda değildir. Sınıf tanımlanırken sınıfın özelliklerine değer atanabilir. Sınıf tanımlanırken özelliğe değer atamak için süslü parantezler “{}” içerisinde yapılmalıdır.
- Salt okunur özellikler de oluşturulabilir.
- Salt okunur, bir özelliğin değerine erişilebilen ancak ona bir değer atanamayan anlamına gelir.
- Sadece yazılabilir bir özellikler de oluşturulabilir.
- Sadece yazılabilir, bir özelliğe değer atanabilir ancak onun değerini alınamaz anlamına gelir.
- Bir özelliğin “get” erişimcisi yoksa sadece yazılabilir bir özelliktir.
- Özelliklere sadece değer atanmadan önce değil aynı zaman değer çağrılmadan öncede kontroller eklenebilmektedir.
- Özellikler (properties) kullanılırken illa alanların (fields) beraberinde kullanılması zorunlu değildir.

DEĞERLENDİRME SORULARI

1. Nesne tabanlı programlamada kullanılan alan (field) ve özellik (property) yapıları aşağıdakilerden hangisi ile beraber kullanılır?
 - a) Sınıf (Class)
 - b) Buton (Buton)
 - c) Fonksiyon
 - d) Parametre
 - e) Dönüş değeri

- I. Metotların içerisinde tanımlanan değişkenler alan değildir.
- II. Sınıflarda tanımlanan bütün değişkenler alandır.
- III. Alanlar metotları içerir.
- IV. Alanlar sınıflardan bağımsız yapılardır.

2. Yukardakilerden hangisi veya hangileri alan(fields) için doğrudur?
 - a) Yalnız I
 - b) I ve II
 - c) I, II ve III
 - d) II ve IV
 - e) Yalnız IV

Sınıfın içerisindeki bütün metotlar bulundukları sınıfın alanlarına (fields) erişebilirler.

3. Yukardaki cümlede boş bırakılan yere aşağıdakilerden hangisi getirilmelidir?
 - a) bütün
 - b) bazı
 - c) özel
 - d) kısmi
 - e) genel
4. Bir alanı sadece okunabilir hale getirmek için aşağıdaki ifadelerden hangisi kullanılmalıdır?
 - a) this
 - b) class
 - c) field
 - d) not
 - e) readonly

- I. get
 - II. set
 - III. put
 - IV. push
5. Alanlara erişmek ve değer atamak için, özelliklerde (properties) yukardakilerden hangisini veya hangilerini kullanırlar?
- a) Yalnız I
 - b) I ve II
 - c) I, II ve III
 - d) III ve IV
 - e) Yalnız IV
6. Bir özelliğin erişimcisi yoksa o özellik salt okunurdur. Yukarda boş bırakılan yere aşağıdakilerden hangisi getirilmelidir?
- a) set
 - b) get
 - c) put
 - d) tut
 - e) push
7. Bir özelliğe değer atanırken bazı kontroller eklenmek isteniyorsa kod hangi erişimciye yazılmalıdır?
- a) get
 - b) properties
 - c) fields
 - d) class
 - e) set
8. Nesne tabanlı programlamada herkese açık erişim belirleyicisi aşağıdakilerden hangisidir?
- a) private
 - b) public
 - c) protected
 - d) mono
 - e) read
9. Alanlara sınıf dışından erişmek için öncelikle aşağıdakilerden hangisi yapılmalıdır?
- a) Sınıf tanımlanmalı
 - b) Alanı metot içerisinde tanımlanmalıdır
 - c) Metot oluşturulmalı
 - d) Erişim sınırlayıcı oluşturulmalı
 - e) Sınıfta metot oluşturulmalı

10. Sınıf içerisindeki alan adı ile metod içerisindeki değişken adı aynı olduğu durumlarda alanın değerine erişmek için aşağıdakilerden hangisi kullanılır?
- a) class
 - b) method
 - c) process
 - d) this
 - e) public

Cevap Anahtarı

1.a, 2.b, 3.a, 4.e, 5.b, 6.a, 7.e, 8.b, 9.a, 10.d

YARARLANILAN KAYNAKLAR

Strauss, D. (2020). Getting to Know Visual Studio 2019. In Getting Started with Visual Studio 2019 (pp. 1-60).

Containers, G., & Johnson, B. Essential Visual Studio 2019.

Price, M. J. (2019). C# 8.0 and .NET Core 3.0—Modern Cross-Platform Development: Build applications with C#, .NET Core, Entity Framework Core, ASP.NET Core, and ML.NET using Visual Studio Code. Packt Publishing Ltd.

Reimer, J. (2005). A History of the GUI. Ars Technica, 5, 1-17.