T.C.
MİLLÎ EĞİTİM BAKANLIĞI

BİLİŞİM TEKNOLOJİLERİ

NESNE TABANLI PROGRAMLAMADA BİLEŞENLER OLUŞTURMA

482BK0156

Ankara, 2012

- Bu modül, mesleki ve teknik eğitim okul/kurumlarında uygulanan Çerçeve Öğretim Programlarında yer alan yeterlikleri kazandırmaya yönelik olarak öğrencilere rehberlik etmek amacıyla hazırlanmış bireysel öğrenme materyalidir.
- Millî Eğitim Bakanlığınca ücretsiz olarak verilmiştir.
- PARA İLE SATILMAZ.

İÇİNDEKİLER

AÇIKLAMALAR

KOD	**482BK0156**
ALAN	**Bilişim Teknolojileri**
DAL/MESLEK	**Veritabanı Programcılığı**
MODÜLÜN ADI	**Nesne Tabanlı Programlamada Bileşenler Oluşturma**
MODÜLÜN TANIMI	Nesne Tabanlı Programlamada Bileşen Oluşturma ile ilgili bilgi, beceri ve tutumların kazandırıldığı bir öğrenme materyalidir.
SÜRE	40/24
ÖN KOŞUL	"Nene Tabanlı Programlamada Kalıtım ve Arayüzler" modülünü tamamlamış olmak.
YETERLİK	Bileşenler oluşturabilmek
MODÜLÜN AMACI	**Genel Amaç** Gerekli ortam sağlandığında bileşenler oluşturabileceksiniz. **Amaçlar** **1.** Alanlara erişmek için özellikler uygulayabileceksiniz. **2.** Ara birim özellikleri ile çalışabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	**Ortam:** Bilgisayar labaratuarı **Donanım:**Tabanlı programlama dili için gerekli donanıma sahip bilgisayar
ÖLÇME VE DEĞERLENDİRME	Modül içinde yer alan her öğrenme faaliyetinden sonra verilen ölçme araçları ile kendinizi değerlendireceksiniz. Öğretmen modül sonunda ölçme aracı (çoktan seçmeli test, doğru-yanlış testi, boşluk doldurma vb.) kullanarak modül uygulamaları ile kazandığınız bilgi ve becerileri ölçerek sizi değerlendirecektir.

GİRİŞ

Sevgili Öğrenci,

Bileşen oluşturma nesneye dayalı programlamanın temel görevlerinden bir tanesidir. Nesneye dayalı programlama kavramındaki nesne sözcüğü, uygulamada kullanacağınız bileşenleri kastetmektedir. Uygulamanız için bileşenleri hazırladıktan sonra uygun yöntemlerle bu bileşenleri organize ederek uygulamayı tamamlarsınız. Bu bileşenler olmadan uygulamayı tamamlamak imkânsızdır. Bileşenlerin bazıları size hazır olarak sunulur. Ancak birçoğunu da sizin oluşturmanız gerekir.

İşte bu modül size bileşen oluşturma becerisi kazandıracaktır.

ÖĞRENME FAALİYETİ–1

AMAÇ

Bu faaliyetle gerekli araç, gereç ve ortam sağlandığında "özellikler" ile çalışabileceksiniz.

ARAŞTIRMA

- Nesne tabanlı programlamada sınıf tanımlamada kullanılan metot ve özellik arasında ne farkı vardır? Araştırınız.
- Gerçek hayattaki nesnelerin özellikleri ile bir sınıf tanımlamada kullanılan özelliklerin benzerliği var mıdır?

1. ÖZELLİKLER

Bu bölümde, bir sınıftaki alanları gizlemek için özellikleri nasıl tanımlayacağınız ve kullanacağınız ele alınacaktır. Önceki bölümlerde, bir sınıf içindeki alanları özel yapmanız ve bu alanlardaki değerleri saklamak ve yeniden elde etmek için yöntemler sağlamanız gerektiği vurgulanmıştı. Bu yaklaşım, alanlara güvenli ve kontrollü bir erişim ve izin verilen değerlere ilişkin ek mantık ile kuralları kapsüllemenizi sağlar. Bununla birlikte, bu yolla bir alana erişim için söz dizimi doğal değildir. Bir değişken okumak ya da yazmak istediğinizde, normal olarak bir atama ifadesi kullanırsınız. Bu nedenle bir alanda (sonuç olarak bir değişken) aynı etkiye erişmek için bir yöntem çağırmak biraz acemiliktir. Özellikler bu uygunsuzluğu gidermek için tasarlanmıştır.

1.1. Metotlar İle Kapsülleme

Alanları gizlemenin (kapsüllemenin) yollarından bir tanesi de metotları kullanmaktır. Bilgisayar ekranındaki bir noktayı x ve y koordinatları ile gösteren aşağıdaki şekilde bir yapımız olsun. x koordinatı için geçerli aralığın 0 ile 1280 arasında olduğunu y koordinatı için ise geçerli aralığın 0 ile 1024 arasında olduğunu varsayalım.

```
struct EkranPozisyonu
{
    public EkranPozisyonu(int x, int y)
    {
        this.X = aralikDenetleX(x);
        this.Y = aralikDenetleY(y);
    }

    public int X;
    public int Y;
```

```
    private static int aralikDenetleX(int x)
    {
        if (x < 0 || x > 1280)
        {
            throw new ArgumentOutOfRangeException("X");
        }
        return x;
    }

    private static int aralikDenetleY(int y)
    {
        if (y < 0 || y > 1024)
        {
            throw new ArgumentOutOfRangeException("Y");
        }
        return y;
    }
}
```

Bu yapı ile ilgili en önemli problem kapsüllemenin altın kuralına uymamasıdır. Yani veri tipi özel (private) değildir. Ortak (public) veri kullanımı sınırlılıkları olan bir uygulamadır. Çünkü kullanımı denetlenmez. Örneğin Ekran Pozisyonu kurucu aralığı parametrelerini denetler. Ancak böyle bir denetim ortak alanlara (X ve Y) uygulanmaz. Uygulamada programlama hatası sonucu olarak X veya Y konum aralığının dışına düşer.

```
EkranPozisyonu orijin = new EkranPozisyonu(0, 0);
int xPos = orijin.X;
orijin.Y = -100; //hata! Y 0-1024 aralığında olmalı
```

Bu problemi çözmek için kullanılan yöntemlerden bir tanesi, alanları özel yapmak ve her özel alanın değerini okumak ve yazmak için birer erişimci ve değiştirici metot eklemektir. Değiştirici yöntemler daha sonra yeni alan değerlerinde aralık denetimi yapabilir. Örneğin aşağıdaki kod X alanı için bir erişimci (OkuX) ve bir değiştirici (YazX) metot içerir. YazX'in parametre değerini nasıl denetlediğine dikkat ediniz.

```
struct EkranPozisyonu
{

    ...

    public int OkuX()
    {
        return X;
    }

    public void YazX(int yeniX)
    {
        this.X = aralikDenetleX(yeniX);
    }

    ...
}
```

Kod artık aralık denetlemelerini başarıyla uygular. Bununla birlikte Ekran Pozisyonu artık doğal bir alan benzeri söz dizimine sahip değildir. Metot temelli bir söz dizimi ortaya çıkmıştır. Aşağıdaki örnek X'in değerini 10 artırır. Bunu yapmak için OkuX ile önce X'in değerini okumalı, arttırdıktan sonra da YazX ile X'in değerini yazmalıyız.

```
int xpos = orijin.OkuX();
orijin.YazX(xpos + 10);
```

Bu işlemi X alanı ortak (public) olduğunda aşağıdaki örnekte görüldüğü gibi gerçekleştiriyoruz:

```
orijin.X += 10;
```

Şüphesiz bu durumda alanları kullanmak daha kısa, daha açık ve daha kolaydır. Ne yazık ki ortak alanları kullanmak kapsüllemeyi kırar. Özellikler bu noktada devreye girerek her iki örneğin iyi yönlerini birleştirir. Yani alan benzeri bir söz dizimi kullanımına izin verirken kapsüllemeyi de sürdürür.

1.2. Özellikleri Kullanma

Bir özelliği bir deyimde kullandığınızda onu ya bir okuma bağlamında (değerini değiştirmediğiniz durumlar) ya da bir yazma bağlamında (değerini değiştirdiğiniz durumlar) kullanırsınız. Ekran Pozisyonu 2 (Ekran Pozisyonu yapısının özellik kullanan sürümü) yapısının özelliklerini kullanarak okuma ve yazma örneğine ait kodlar aşağıda verilmiştir:

```
struct EkranPozisyonu2
{
    public EkranPozisyonu2(int x, int y)
    {
        this.x = aralikDenetleX(x);
        this.y = aralikDenetleY(y);
    }

    private int x;

    public int X
    {
        get { return this.x; }
        set { this.x = aralikDenetleX(value); }
    }
    private int y;

    public int Y
    {
        get { return this.y; }
        set { this.y = aralikDenetleX(value); }
    }
```

```
private static int aralikDenetleX(int x)
    {
        if (x < 0 || x > 1280)
        {
            throw new ArgumentOutOfRangeException("X");
        }
        return x;
    }

    private static int aralikDenetleY(int y)
    {
        if (y < 0 || y > 1024)
        {
            throw new ArgumentOutOfRangeException("Y");
        }
        return y;
    }
}
```

Aşağıdaki örnek bir Ekran Pozisyonu 2 yapısının X ve Y özelliklerindeki değerlerini okumayı gösterir.

```
EkranPozisyonu2 orijin2 = new EkranPozisyonu2(0, 0);
int xpoz = orijin2.X; //orijin.X.get'i çağırır
int ypoz = orijin2.Y; //orijin.X.get'i çağırır
```

Özellik ve alanlara aynı söz dizimini kullanarak eriştiğinize dikkat ediniz. Bir özelliği okuma bağlamında kullandığınızda derleyici alan benzeri kodunuzu otomatik olarak o özelliğin get erişimcisine yapılan bir çağrıya dönüştürür. Benzer biçimde de bir özelliği yazma bağlamında kullanırsanız derleyici kodu otomatik olarak set erişimcisine yapılan bir çağrıya dönüştürür.

Bir özelliği hem okuma hem de yazma bağlamında kullanmak da olasıdır. Örneğin;derleyici aşağıdaki gibi ifadeleri otomatik olarak hem get hem de set erişimcilerine yapılan çağrılara dönüştürür.

```
orijin2.X += 10;
```

1.2.1. Sadece Okunabilir

Sadece get erişimcisi olan özellikleri bildirmenize izin verilir. Bu durumda özelliği yalnızca okuma bağlamında kullanabilirsiniz. Aşağıdaki örnekte Ekran Pozisyonu 2 yapısının X özelliğinin sadece okunabilir bir özellik olarak bildirilmiş biçimini görebilirsiniz:

```
struct EkranPozisyonu2
{
    ...
    public int X
    {
        get { return this.x; }
    ...
    }
}
```

X özelliği bir set erişimcisi içermez. Bu yüzden X değişkenine değer atayarak bir yazma bağlamında kullanmak için yapılan her girişim başarısız olur. Örneğin;

```
orijin2.X = 140; //derleme zamanı hatası
```

1.2.2. Sadece Yazılabilir

Benzer biçimde yalnızca set erişimcisi olan bir özellik bildirebilirsiniz. Bu durumda özelliği yalnızca yazma bağlamında kullanabilirsiniz. Örneğin aşağıda Ekran Pozisyonu 2 yapısının X özelliğinin sadece yazılabilir bir özellik olarak bildirilmiş biçimini görebilirsiniz.

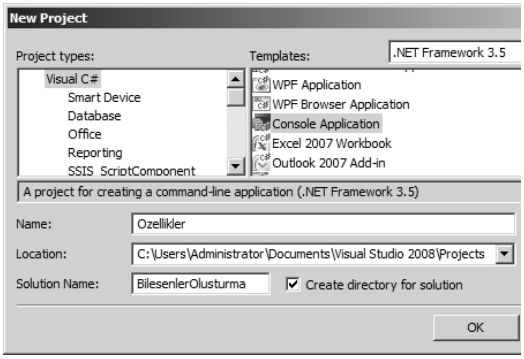
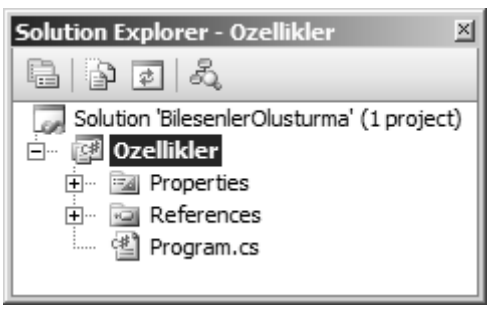
```
struct EkranPozisyonu2
{
    ...
    public int X
    {
        set { this.x = aralikDenetleX(value); }
    }
    ...
}
```

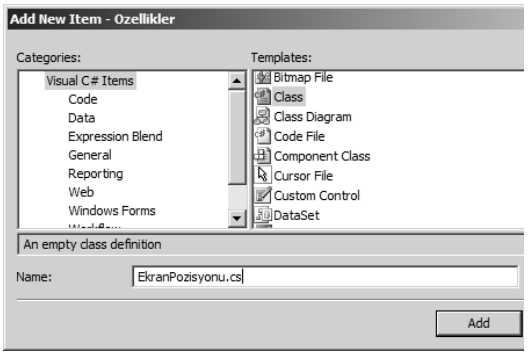
X özelliği bir get erişimcisi içermez. X'i okuma bağlamında kullanmak için yapılan her girişim başarısız olur. Örneğin;

```
Console.WriteLine(orijin2.X);    // derleme zamanı hatası
orijin2.X = 200;                 // derlenir
orijin2.X += 10;                 // derleme zamanı hatası
```

UYGULAMA FAALİYETİ

Aşağıda verilen işlem basamaklarını takip ederek (Öneriler kısmı, uygulama faaliyeti için yönlendirici olacaktır.) konuyu daha iyi pekiştiriniz.

İşlem Basamakları	Öneriler
➢ Nesne tabanlı programlama yazılımı uygulamasını açınız.	➢ Başlat – Çalıştır seçeneğini tıklayınız. Çalıştır penceresi açılır. Aç kutusuna "devenv" yazıp Tamam düğmesine tıklayınız.
➢ Yeni bir Konsol Uygulaması (Console Application) projesi oluşturunuz.	➢ Dosya (File) menüsünden Yeni (New) alt menüsünü, ardından Proje… (Project…) seçeneğini tıklayınız. Yeni Proje (New Project) penceresi açılır.  **Proje types** : Visual C# **Templates** : Console Application **Name** : Özellikler **Location** : Değiştirmeyiniz **Solution Name** : Bileşenler Oluşturma ➢ Yeni Proje (New Project) penceresinde üstteki seçim ve girişleri yaptıktan sonra Tamam düğmesini tıklayınız. Çözüm Gezgini (Solution Explorer) penceresi aşağıdaki gibi görünmelidir: 

➢ Projeye yeni bir sınıf ekleyiniz.	➢ Project menüsünden Add Class… seçeneğini tıklayınız. Dilerseniz alternatif olarak Shift + Alt + C tuşlarına birlikte basabilirsiniz. Add New Item penceresi açılır.  ➢ Kategoriler (Categories) bölmesinden Visual C# Items'ın, Şablonlar (Templates) bölmesinden Class'ın seçili olduğundan emin olduktan sonra Ad (Name) kutusuna "EkranPozisyonu.cs" yazıp Ekle (Add) düğmesini tıklayınız. Çözüm Gezgini (Solution Explorer) penceresine Ekran Pozisyonu.cs dosyası eklenerek editör bölmesinde Ekran Pozisyonu.cs sekmesi açılmalıdır.

➢ Ekran Pozisyonu sınıfının tipini Yapı (struct) olarak değiştiriniz.	`class EkranPozisyonu`
	`{`
	`}`
	sınıfını aşağıdaki şekilde değiştiriniz.
	`struct EkranPozisyonu`
	`{`
	
	`}`

➢ Ekran Pozisyonu.cs dosyasındaki yapıyı tamamlayınız.	Ekran Pozisyonu Yapısını aşağıdaki şekilde tamamlayınız.
	
	`struct EkranPozisyonu`
	`{`
	`public EkranPozisyonu(int x, int y)`
	`{`
	`this.X = aralikDenetleX(x);`
	`this.Y = aralikDenetleY(y);`
	`}`
	
	`public int X;`
	`public int Y;`
	
	`public int OkuX()`
	`{`
	`return X;`
	`}`
	
	`public void YazX(int yeniX)`
	`{`
	`this.X = aralikDenetleX(yeniX);`
	`}`
	
	`private static int aralikDenetleX(int x)`
	`{`
	`if (x < 0 || x > 1280)`
	`{`
	`Throw new ArgumentOutOfRangeException("X");`
	`}`
	`return x;`
	`}`
	
	`private static int aralikDenetleY(int y)`
	`{`
	`if (y < 0 || y > 1024)`
	`{`
	`throw new ArgumentOutOfRangeException("Y");`
	`}`
	`return y;`
	`}`
	`}`

➢ Ekran Pozisyonu.cs dosyasına Ekran Pozisyonu 2 yapısını ekleyiniz.	➢ Ekran Pozisyonu yapısının kapama küme parantezinden sonra Ekran Pozisyonu 2 yapısının kodlarını aşağıda görüldüğü gibi ekleyiniz.
	
	`struct EkranPozisyonu2`
	`{`
	`  public EkranPozisyonu2(int x, int y)`
	`  {`
	`  this.x = aralikDenetleX(x);`
	`  this.y = aralikDenetleY(y);`
	`  }`
	
	`  private int x;`
	
	`  public int X`
	`  {`
	`  set { this.x = aralikDenetleX(value); }`
	`  }`
	`  private int y;`
	
	`  public int Y`
	`  {`
	`  get { return this.y; }`
	`  set { this.y = aralikDenetleX(value); }`
	`  }`
	
	`  private static int aralikDenetleX(int x)`
	`  {`
	`  if (x < 0 || x > 1280)`
	`  {`
	`  throw new ArgumentOutOfRangeException("X");`
	`  }`
	`  return x;`
	`  }`
	
	`  private static int aralikDenetleY(int y)`
	`  {`
	`  if (y < 0 || y > 1024)`
	`  {`
	`  throw new ArgumentOutOfRangeException("Y");`
	`  }`
	`  return y;`
	`  }`
	`}`

➢ Program.cs dosyasındaki Main metoduna kodlar ekleyerek denemeler yapınız.	➢ Program.cs dosyasını editörde açıp aşağıdaki kodları girerek denemeler yapınız. Main metodu aşağıdaki gibi olmalıdır.
	
	`static void Main(string[] args)`
	`{`
	`EkranPozisyonu orijin = new EkranPozisyonu(0, 0);`
	`int xPos = orijin.X;`
	`orijin.Y = -100;`
	
	`int xpos = orijin.OkuX();`
	`orijin.YazX(xpos + 10);`
	
	`orijin.X += 10;`
	
	`EkranPozisyonu2 orijin2 = new EkranPozisyonu2(0, 0);`
	`int xpoz = orijin2.X;`
	`int ypoz = orijin2.Y;`
	
	`orijin2.X = -140;`
	`Console.WriteLine(orijin2.X);`
	`orijin2.X = 200;`
	`orijin2.X += 10;`
	`Console.ReadKey();`
	`}`

➢ Nesne tabanlı programlama yazılımını kapatınız.	➢ Uygulama faaliyeti bitti. File menüsünden Exit seçeneğini tıklayarak nesne tabanlı programlama yazılımını kapatınız.

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız becerileri **Evet**, kazanamadığınız becerileri **Hayır** kutucuğuna (X) işareti koyarak kendinizi değerlendiriniz.

Değerlendirme Ölçütleri		Evet	Hayır
1.	Kapsülleme tekniğini kavradınız mı?		
2.	Kapsülleme işlemi için alanların değerini metotlarla okuyup yazabildiniz mi?		
3.	Kapsülleme işlemi için alanların değerini özelliklerle okuyup yazabildiniz mi?		
4.	Sadece okunabilir özellikler oluşturabildiniz mi?		
5.	Sadece yazılabilir özellikler oluşturabildiniz mi?		

DEĞERLENDİRME

Değerlendirme sonunda **"Hayır"** şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız **"Evet"** ise "Ölçme ve Değerlendirme"ye geçiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

1. Kapsülleme kavramını aşağıdakilerden hangisi en iyi tanımlamaktadır?
- **A)** Alanları özel yapmak
- **B)** Yapı veya sınıfa alan değiştirme metodu eklemek
- **C)** Yapı veya sınıfa alan okuma metodu eklemek
- **D)** Yapı veya sınıftaki alanları özel yapıp değiştirme ve okumayı denetleme için metotlar kullanmak

2. Sadece okunabilir özellik eklemek için aşağıdakilerden hangi erişimci kullanılır?
- **A)** get
- **B)** set
- **C)** private
- **D)** public

3. Sadece yazılabilir özellik eklemek için aşağıdakilerden hangi erişimci kullanılır?
- **A)** get
- **B)** set
- **C)** private
- **D)** public

4. Bir özellik okuma ve değiştirme kullanımı bakımından aşağıdakilerden hangisine benzer?
- **A)** alan
- **B)** metot
- **C)** yapı
- **D)** sınıf

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ–2

AMAÇ

Bu faaliyetle gerekli araç, gereç ve ortam sağlandığında arayüz özelliklerini (interface properties) kullanabileceksiniz.

ARAŞTIRMA

- Arayüzlerde özellikler nasıl tanımlanır? Araştırarak rapor hâlinde sınıfa sununuz.

2. ARAYÜZ ÖZELLİKLERİ

Arayüzler yöntemlerin yanında özellikler de belirtebilir. Bunu yapmak için get ve/veya set anahtar sözcüğünü kullanırsınız. Ancak arayüzler uygulama kodları barındıramayacakları için get ve set erişimcilerinin gövdeleri yerine noktalı virgül (;) kullanılır. Örneğin;

```
interface IEkranPozisyonu
{
    int X { get; set; }
    int Y { get; set; }
}
```

Bu arayüzü kullanan tüm sınıf ve yapılar get ve set erişimci yöntemleri ile birlikte X ve Y özelliklerini de kullanmak zorundadır. Örneğin;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ArayuzOzellikleri
{
    struct EkranPozisyonu: IEkranPozisyonu
    {
        int x, y;
        public int X
        {
            get
            {
                return x;
            }
            set
            {
```

```
                x = value;
            }
        }
        public int Y
        {
            get
            {
                return y;
            }
            set
            {
                y = value;
            }
        }
    }
}
```

2.1. Özellikleri Uygulamada Kullanma

Nesne tabanlı programlama yazılımında Özellikler (Properties) penceresini kullandığınızda çeşitli uygulama bileşenlerinin [Metin Kutusu (TextBox), Düğme (Button) gibi bileşenler] özellik değerlerini ayarlayan alan bir kod oluşturursunuz. Bazı bileşenlerin çok sayıda özelliği vardır ve bazı özellikler diğerlerinden daha fazla kullanılır. Aşağıdaki örnekte gördüğünüz söz dizimini kullanarak bir çok özelliği çalışma zamanında değiştirebilirsiniz. Merkez adında bir Ekran Pozisyonu nesnesi yaratıp X özelliğini 0'a ayarlayan ve onu alıp ekrana yazan basit bir örnek aşağıdaki gibi olabilir:

```
...
static void Main(string[] args)
{
    EkranPozisyonu merkez = new EkranPozisyonu();
    merkez.X = 0;
    Console.WriteLine(merkez.X);
}
...
```

2.2. Otomatik Özellikler

Özelliklerin başlıca amacı alanların kullanımını uygulamasını dış dünyadan saklamaktır. Özelliklerin amacına yönelik kullanımı önemlidir. Çünkü get ve set erişimciler, sadece alana bir değer atamak ya da değer okumak işlemlerini yerine getiriyorsa bunun doğru bir yaklaşım olup olmadığı sorusu akla gelebilir. Veriyi ortak (public) alanlar olarak göstermek yerine özellikler tanımlamanız için en az iki neden vardır:

- **Uygulamalarla Uygunluk:** Alanlar ve özellikler, derlemelerde farklı meta veri kullanarak kendilerini ortaya koyar. Bir sınıf geliştiriyorsanız ve ortak alanlar kullanmaya karar verdiyseniz bu sınıfı kullanan bir uygulama, alanlar olarak bu öğelere başvuracaktır. Bir özelliği okurken ve yazarken kullandığınız söz dizimini bir alanı okumak ve yazmak için kullanabilseniz de derlenen kod gerçekte çok farklıdır. Daha sonra bu alanları özellikler olarak değiştirmeye

karar verirseniz mevcut uygulamalar yeniden derlenmeden sınıfın güncellenmiş sürümünü kullanamaz. Uygulamayı bir şirketin çok sayıda kullanıcısına uyguladıysanız bu bir dezavantajdır.

- **Arayüzlerle Uygunluk:** Bir arayüz gerçekleştiriyorsanız ve arayüz özellik olarak bir öğe tanımlıyorsa özellik sadece özel (private) alanlardaki veriyi okuyor ve yazıyor olsa bile arayüzdeki şartlara uyan bir özellik yazmanız gerekir. Sadece aynı adla bir ortak alan göstermek yoluyla bir özellik geliştiremezsiniz.

Programlama dili tasarımcıları, programcıların gereğinden fazla kod yazacak kadar zamanı olmayan meşgul insanlar olduklarını bilir. Bu amaçla derleyici sizin için otomatik özellikler kodunu üretebilir.

```
class Daire
{
    public int YariCap { get; set; }

}
```

Bu örnekte Daire sınıfı YariCap adında bir özellik içerir. Bu özellik, türü dışında herhangi bir bilgi içermez. Derleyici bu tanımı otomatik olarak aşağıda görüldüğü şekle dönüştürür.

```
class Daire
{
    private int _yariCap;

    public int YariCap
    {
        get
        {
            return this._yariCap;
        }
        set
        {
            this._yariCap = value;
        }
    }
}
```

2.3. Özellikler İle Nesneleri Başlatma

Bir nesneyi başlatmak için kurucuları tanımlamayı öğrenmiş olmalısınız. Bir nesne birden çok kurucuya sahip olabilir ve bir nesnedeki farklı öğeleri başlatmak için çeşitli parametrelerle kurucular tanımlayabilirsiniz. Ancak bu pratik bir yaklaşım değildir. Bir sınıfın kaç adet alan içerdiğine ve alanları başlatmak için istediğiniz çeşitli bileşimlere bağlı olarak çok sayıda kurucu yazmaya son verebilirsiniz. Bir sınıf oluşturulduğunda, set erişimcilere sahip ortak (public) özellikler için değerler belirleyerek bu oluşumu başlatabilirsiniz. Yani Ucgen adında bir sınıfın nesnelerini oluşturup bunları herhangi bir bileşimle belirleyebilirsiniz. Ucgen sınıfı aşağıdaki şekilde tanımlanmış olsun:

```
class Ucgen
{
    private int kenar1Uzunluk = 10;
    private int kenar2Uzunluk = 10;
    private int kenar3Uzunluk = 10;

    public int Kenar1Uzunluk
    {
        set { this.kenar1Uzunluk = value; }
    }
    public int Kenar2Uzunluk
    {
        set { this.kenar2Uzunluk = value; }
    }
    public int Kenar3Uzunluk
    {
        set { this.kenar3Uzunluk = value; }
    }
}
```

Bu sınıfa ait özellik ile nesne başlatma bileşimleri aşağıda görüldüğü gibi olabilir:

```
1. Ucgen ucgen1 = new Ucgen { Kenar1Uzunluk = 20 };
2. Ucgen ucgen2 = new Ucgen { Kenar1Uzunluk = 20, Kenar2Uzunluk = 15
   };
3. Ucgen ucgen3 = new Ucgen { Kenar1Uzunluk = 20, Kenar2Uzunluk = 15,
   Kenar3Uzunluk = 10 };
```

1 . örnekte **ucgen1** nesnesi oluşturulurken sadece **Kenar1Uzunluk** özelliğine 20 değeri verilmiş diğer iki özellik ayarlanmamıştır (Kenar2Uzunluk ve Kenar3Uzunluk).

Diğer özelliklere biz değer vermediğimiz için oluşturulan ucgen1 nesnesinin alanlarının değerleri aşağıdaki gibi olacaktır:

kenar1Uzunluk - 20
kenar2Uzunluk - 10
kenar3Uzunluk - 10

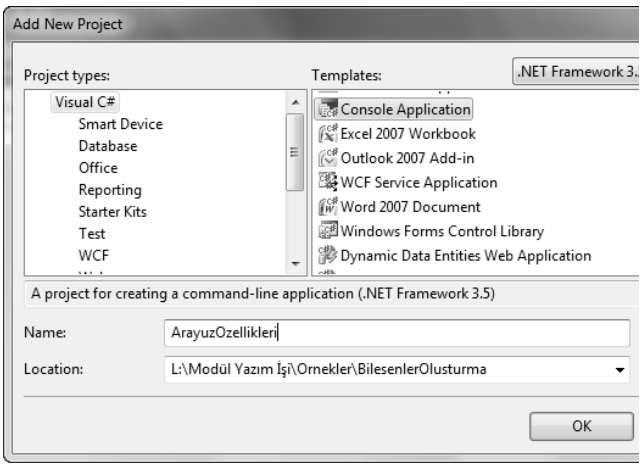
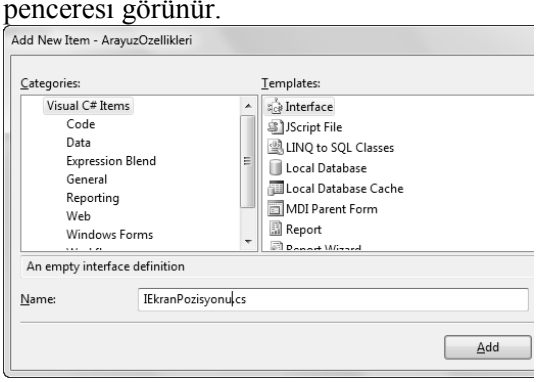
2 . örnekte nesne oluşturulurken Ucgen sınıfında yer alan kenar1Uzunluk 20, kenar2Uzunluk 15 olarak belirtildiğinden, kenar3Uzunluk sınıf içerisinde ayarlandığı hâliyle yani 10 olarak kalacaktır.

3 . örnekte tüm uzunluklar özellikler yardımıyla ayarlandığı için tüm uzunluklarımız verilen değerlerle ayarlanacaktır.

UYGULAMA FAALİYETİ

Aşağıda verilen işlem basamaklarını takip ederek (Öneriler kısmı, uygulama faaliyeti için yönlendirici olacaktır.) konuyu daha iyi pekiştiriniz.

İşlem Basamakları	Öneriler
➢ Nesne tabanlı programlama yazılımı uygulamasını açınız.	➢ Başlat – Çalıştır seçeneğini tıklayınız. Çalıştır penceresi açılır. Aç kutusuna "devenv" yazıp Tamam düğmesine tıklayınız.
➢ Öğrenme Faaliyeti- 1'de oluşturduğunuz BilesenlerOlusturma çözümünü açınız.	➢ Nesne tabanlı programlama yazılımını başlattığınızda varsayılan olarak Başlangıç Sayfasının (Start Page) görüntülenmesi gerekir. Bu sayfadaki Geçmiş Projeler (Recent Projects) bölümünden BilesenlerOlusturma çözümünü tıklayınız. ➢ Başlangıç Sayfasını görüntülemek için Görünüm (View) menüsünden Diğer Pencereler (Other Windows) alt menüsü içinde yer alan Başlangıç Sayfasını (Start Page) tıklayabilirsiniz. 

➢ Çözüme yeni bir proje ekleyiniz.	➢ Dosya (File) menüsünden Ekle (Add) alt menüsü içindeki Yeni Proje… (New Project…) seçeneğini tıklayınız. Yeni Proje Ekle (Add New Project) penceresi açılır.  ➢ Şablonlar (Templates) bölümünden Konsol Uygulamasını (Console Application) seçiniz,.Ad (Name) kutusuna "ArayuzOzellikleri" yazınız ve Tamamı (OK) tıklayınız.
➢ Projeye yeni bir arayüz ekleyiniz.	➢ Proje (Project) menüsünden Sınıf Ekle… (Add Class…) seçeneğini tıklayınız. Dilerseniz Shift + Alt + C tuşlarına da basabilirsiniz. Yeni Öğe Ekle – Arayuzler (Add New Item - ArayuzOzellikleri) penceresi görünür.  ➢ Şablonlar (Templates) bölümünden Arayüzü (Interface) seçiniz . Ad (Name) kutusuna "IEkranPozisyonu" yazınız ve Ekle (Add) düğmesine tıklayınız. Çözüm Gezgininde (Solution Explorer) "ArayuzOzellikleri" projesine "IEkranPosizyonu.cs" dosyası eklenir ve editör penceresinde "IEkranPosizyonu.cs" sekmesi açılır.

➢ "IEkranPosizyonu.cs" sekmesindeki kodu düzenleyiniz.	➢ "IEkranPosizyonu.cs" kod editörü başlangıçta şu şekilde görünür.
	`using System;`
	`using System.Collections.Generic;`
	`using System.Linq;`
	`using System.Text;`
	
	`namespace ArayuzOzellikleri`
	`{`
	`  interface IEkranPozisyonu`
	`  {`
	`  }`
	`}`
	
	`interface IEkranPozisyonu`
	`{`
	`}`
	
	➢ Üstteki kodu aşağıdakiyle değiştiriniz.
	
	`interface IEkranPozisyonu`
	`{`
	`  int X { get; set; }`
	`  int Y { get; set; }`
	`}`
	
	➢ IEkranPozisyonu adında bir arayüz oluşturdunuz .Hem değeri değiştirilebilir hem de elde edilebilir X ve Y özelliklerini bildirdiniz.

➢ Projeye EkranPozisyonu öğesi ekleyerek IEkranPozisyonu arayüzünü EkranPozisyonu yapısı ile uygulayınız.	➢ Aşağıdaki kodları ArayuzOzellikleri projesine EkranPozisyonu.cs öğesini ekledikten sonra yazınız. Yeni öğeyi daha önceki şekilde eklemek için Ctrl + Shift + A tuşlarına basabilirsiniz. Ad kutusuna Ekran Pozisyonu yazıp Tamamı tıklayarak sınıfı ekleyebilir ve içindeki kodları aşağıdakilerle değiştirebilirsiniz.
	
	`using System;`
	`using System.Collections.Generic;`
	`using System.Linq;`
	`using System.Text;`
	
	`namespace ArayuzOzellikleri`
	`{`
	`struct EkranPozisyonu: IEkranPozisyonu`
	`{`
	`int x, y;`
	`public int X`
	`{`
	`get`
	`{`
	`return x;`
	`}`
	`set`
	`{`
	`x = value;`
	`}`
	`}`
	`public int Y`
	`{`
	`get`
	`{`
	`return y;`
	`}`
	`set`
	`{`
	`y = value;`
	`}`
	`}`
	`}`
	`}`
	➢ Burada IEkran Pozisyonu arayüzünün X ve Y özelliklerini get ve set erişim belirteçleri ile birlikte uygulayan bir yapı oluşturdunuz.

➢ Program.cs dosyasına Ucgen sınıfını ekleyiniz.	➢ Program.cs dosyasını açınız .
	`class Program`
	`{`
	`...`
	`}`
	Altına
	`class Ucgen`
	`{`
	`private int kenar1Uzunluk = 10;`
	`private int kenar2Uzunluk = 10;`
	`private int kenar3Uzunluk = 10;`
	`public int Kenar1Uzunluk`
	`{`
	`set { this.kenar1Uzunluk = value; }`
	`}`
	`public int Kenar2Uzunluk`
	`{`
	`set { this.kenar2Uzunluk = value; }`
	`}`
	`public int Kenar3Uzunluk`
	`{`
	`set { this.kenar3Uzunluk = value; }`
	`}`
	`}`
	sınıfını ekleyinİZ.

➢ Main metodunda Ucgen sınıfının oluşumlarını özellikler kullanarak gerçekleştiriniz.	➢ Main metodunun küme ({ }) parantezleri içine aşağıdaki kodları ekleyiniz.
	`Ucgen ucgen1 = new Ucgen { Kenar1Uzunluk = 20 };`
	`Ucgen ucgen2 = new Ucgen { Kenar1Uzunluk = 20, Kenar2Uzunluk = 15 };`
	`Ucgen ucgen3 = new Ucgen { Kenar1Uzunluk = 20, Kenar2Uzunluk = 15, Kenar3Uzunluk = 10 };`

➢ Uygulamayı test ediniz.	➢ Uygulamayı Ctrl + F5 tuşlarına basarak test ediniz. Uygulamanın hatasız olarak çalıştığını görmeniz gerekir.
➢ Nesne tabanlı programlama yazılımını kapatınız.	➢ Uygulama faaliyeti bitti. Dosya (File) menüsünden Çıkış (Exit) seçeneğini tıklayarak Nesne tabanlı programlama yazılımı kapatınız.

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız becerileri **Evet,** kazanamadığınız becerileri **Hayır** kutucuğuna (X) işareti koyarak kendinizi değerlendiriniz.

Değerlendirme Ölçütleri		Evet	Hayır
1.	Bir yapı ya da sınıf içerisinde uygun olan erişim özelliklerini hatasız olarak tanımlayabildiniz mi?		
2.	Bir yapı ya da sınıf içerisinde uygun olan arayüz özelliğini tanımlayabildiniz mi?		
3.	Tanımlanan arayüz özelliğinde derleyicinin otomatik özellik oluşturduğunu kavradınız mı?		
4.	Özellikler ile bir nesneyi hatasız ve amacına uygun olarak başlatabildiniz mi?		

DEĞERLENDİRME

Değerlendirme sonunda **"Hayır"** şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız **"Evet"** ise "Ölçme ve Değerlendirme"ye geçiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

1. Bir arayüzde özelliğin set ve get erişimcisinin gövdesi yerine aşağıdaki işaretlerden hangisi konur?

A) :
B) .
C) ;
D) ,

2. Özellikler ile nesneleri başlatırken özellikleri aşağıdaki hangi işaret içine yazmak gerekir?

A) Köşeli parantezler
B) Tırnak işaretleri
C) Normal parantezler
D) Küme parantezleri

3. Bir kısmı programcı tarafından yazılınca derleyici tarafından otomatik olarak tamamlanan öğe aşağıdakilerden hangisidir?

A) Arayüz
B) Sınıf
C) Otomatik özellik
D) Yapı

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise "Modül Değerlendirme" ye geçiniz.

MODÜL DEĞERLENDİRME

Aşağıdaki cümlelerin başında boş bırakılan parantezlere, cümlelerde verilen bilgiler doğru ise D, yanlış ise Y yazınız.

1. ( ) Özellikler bir sınıf ve/veya yapıdaki alanları gizlemek için kullanılır.

2. ( ) Alanları gizlemeye "kapsülleme" denir.

3. ( ) Ortak (public) veri kullanmanın dezavantajı, derleyicinin buna izin vermemesidir.

4. ( ) Özelliklerin kullanımı metotların kullanımına çok benzemektedir.

5. ( ) Sadece okuma amaçlı bir özellik oluşturulabilinir.

6. ( ) Sadece yazma amaçlı bir özellik isteniyorsa özelliğin tanımında set erişimcisi kullanılmalıdır.

7. ( ) Arayüzler özellik tanımı içeremez.

8. ( ) Özellikler uygulamada aynen bir değişken gibi kullanılır.

9. ( ) Özellik ayrıntılarının derleyici tarafından yazılmasına "otomatik özellik" denir.

10. ( ) Gerekirse özellikler kullanılarak nesneler başlatılabilinir.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki modüle geçmek için öğretmeninize başvurunuz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ-1'İN CEVAP ANAHTARI

1	**D**
2	**A**
3	**B**
4	**A**

ÖĞRENME FAALİYETİ-2'NİN CEVAP ANAHTARI

1	**C**
2	**D**
3	**C**

MODÜL DEĞERLENDİRMENİN CEVAP ANAHTARI

1	**Doğru**
2	**Doğru**
3	**Yanlış**
4	**Yanlış**
5	**Doğru**
6	**Doğru**
7	**Yanlış**
8	**Doğru**
9	**Doğru**
10	**Doğru**

KAYNAKÇA

- Çeviren: TEZCAN Ümit**, Adım Adım Microsoft C# 2008,** Arkadaş Yayınevi, Ankara, 2009. (Orijinal Kaynak: SHARP John, Microsoft C# 2008 Step By Step, Microsoft Press, Redmond, 2008).
- http://msdn.microsoft.com/en-us/library/67ef8sbd(v=VS.90).aspx, C# Programlama Kılavuzu (C# Programming Guide) (15.08.2011/ 13.00).