T.C.
MİLLÎ EĞİTİM BAKANLIĞI

BİLİŞİM TEKNOLOJİLERİ

NESNE TABANLI PROGRAMLAMADA DEĞER TÜRLERİ

482BK0157

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	**482BK0157**
ALAN	**Bilişim Teknolojileri**
DAL/MESLEK	**Veri Tabanı Programcılığı**
MODÜLÜN ADI	**Nesne Tabanlı Programlamada Değer Türleri**
MODÜLÜN TANIMI	Nesne Tabanlı Programlamada, Numaralandırma ve Yapılara yönelik becerilerin kazandırıldığı bir öğrenme materyalidir.
SÜRE	40/24
ÖN KOŞUL	Nesne Tabanlı Programlamada Değerler ve Başvurular modülünü başarmış olmak
YETERLİK	Numaralandırma ve Yapılar ile Değer Türleri Oluşturmak
MODÜLÜN AMACI	**Genel Amaç** Gerekli ortam sağlandığında numaralandırma ve yapılar ile değer türleri oluşturabileceksiniz. **Amaçlar** **1.** Numaralandırmalarla çalışabileceksiniz. **2.** Yapılarla (struct) çalışabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	**Ortam:** Nesne tabanlı programlama dili için bilgisayar **Donanım:** 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,

Bu modül size Nesne Tabanlı Programlamada hazırladığınız programlarda ihtiyaç duyabileceğiniz Numaralandırma ve Yapılar hakkında bilgi verecektir.

Burada öğreneceğiniz bilgilerle programlarınız içerisinde ihtiyaç duyduğunuzda yeni çözüm yolları üretebilecek ve işlerinizi daha pratik bir şekilde tamamlayabileceksiniz.

ÖĞRENME FAALİYETİ–1

AMAÇ

Uygun ortam sağlandığında Numaralandırmalarla(Enum) çalışabileceksiniz.

ARAŞTIRMA

- Numaralandırmaların kullanım alanlarını araştırınız ve bir rapor halinde sununuz.

1. NUMARALANDIRMA (ENUM)

Numaralandırma (enumeration); belli sözcüklerin, belli tamsayıları temsili durumlarında kullanılan bir yapıdır. Değişkenlerin alabileceği değerlerin sabit olduğu durumlarda kullanılır.

1.1. Numaralandırma(Enum) Tanımlama

Numaralandırmaları tanımlamak için' enum' anahtar kelimesini kullanırız.

Örneğin, içinde ay isimlerini barındıran bir numaralandırma tanımlayalım.

```
enum aylar
  {
    Ocak, Şubat, Mart, Nisan, Mayıs, Haziran, Temmuz,
    Ağustos, Eylül, Ekim, Kasım, Aralık
  }
```

Sözcük	Değer	Sözcük	Değer
Ocak	0	Temmuz	6
Şubat	1	Ağustos	7
Mart	2	Eylül	8
Nisan	3	Ekim	9
Mayıs	4	Kasım	10
Haziran	5	Aralık	11

Tablo 1.1: Numaralandırma tutulan değişkenlere karşılık gelen değerler tablosu

Örnekte Ocak ayı 0 değerini alacak ve ondan sonra gelen sabitlerin değeri birer artarak devam edip gidecektir.

Duruma göre tanımladığımız sabitlere farklı sayısal değerler de atayabiliriz. Bu işlem için "=" işlecini kullanmalıyız.

```
enum aylar
{
   Ocak=1,Şubat, Mart, Nisan, Mayıs, Haziran, Temmuz,
   Ağustos, Eylül, Ekim, Kasım, Aralık
}
```

Sözcük	Değer	Sözcük	Değer
Ocak	1	Temmuz	7
Şubat	2	Ağustos	8
Mart	3	Eylül	9
Nisan	4	Ekim	10
Mayıs	5	Kasım	11
Haziran	6	Aralık	12

Tablo 1.2: Numaralandırma tutulan değişkenlere karşılık gelen değerler tablosu

```
enum aylar
{
   Ocak=1,Şubat,Mart,Nisan,Mayıs,Haziran,Temmuz=1,
   Ağustos, Eylül, Ekim, Kasım, Aralık
}
```

Yukarıda yazan örnekte ki tanımlamayı yaparsak sabitlerimizin değerleri aşağıda bulunan tablodaki gibi olacaktır.

Sözcük	Değer	Sözcük	Değer
Ocak	1	Temmuz	1
Şubat	2	Ağustos	2
Mart	3	Eylül	3
Nisan	4	Ekim	4
Mayıs	5	Kasım	5
Haziran	6	Aralık	6

Tablo 1.3: Numaralandırma tutulan değişkenlere karşılık gelen değerler tablosu

Numaralandırma yapılarımızın içinde tutulan değerler varsayılan olarak integer tipindedir. İhtiyaca uygun olarak diğer tamsayı tiplerinde de (byte, sbyte, short, ushort, uint, long ve ulong) tanımlama yapılabilir.

```
enum aylar : byte
{
   Ocak, Şubat, Mart, Nisan, Mayıs, Haziran, Temmuz,
   Ağustos, Eylül, Ekim, Kasım, Aralık
}
```

Burada dikkat etmemiz gereken parantezler içindeki değerler belirtilen türün kapasitesi ölçüsünde olmalıdır. Örneğin numaralandırmanın türünü byte olarak tanımladıysak en fazla 255 tane sözcük girmeliyiz.

1.2. Numaralandırma (Enum) Yapısını Kullanma

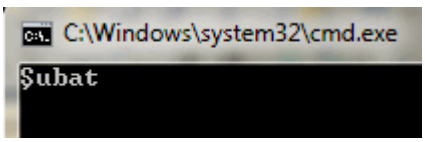
Bir önceki başlıkta numaralandırmaların nasıl tanımlandıklarını gördük. Şimdi de tanımladığımız numaralandırma yapılarını program içinde nasıl kullanacağımıza bakalım.

Öncelikle aylar adında bir enum tanımlayalım.

```
enum aylar
  {
    Ocak, Şubat, Mart, Nisan, Mayıs, Haziran, Temmuz,
    Ağustos, Eylül, Ekim, Kasım, Aralık
  }
```

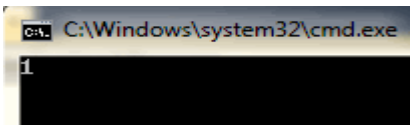
Şimdide enumu program içinde çağıralım.

```
class Program
  {
    static void Main(string[] args)
    {
      aylar ay = aylar.Şubat;
      Console.WriteLine(ay);
      Console.Read();
    }
  }
```



Resim 1.1: Ekran çıktısı

```
class Program
   {
      static void Main(string[] args)
      {
         aylar ay = aylar.Şubat;
         Console.WriteLine((int)ay); // sabitin integer karşılığını ekrana yazdırdık.
         Console.Read();
      }
   }
```



Resim 1.1: Ekran çıktısı

Şimdi de yapımız içindeki tüm sabitleri ekrana nasıl yazdıracağımıza bakalım. Enum içindeki tüm sabitleri almak istediğimiz de GetNames() metodunu kullanırız. GetNames() metodu, enum içindeki tüm sabitleri string tipinde bir dizi olarak almamızı sağlar.

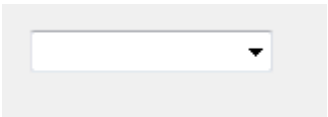
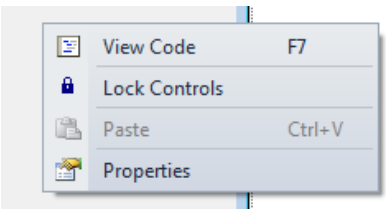
```
enum aylar
   {
      Ocak,Şubat,Mart,Nisan,Mayıs,Haziran,Temmuz,
Ağustos,Eylül,Ekim,Kasım,Aralık
   }
   class Program
   {
      static void Main(string[] args)
      {
         string[] ay = Enum.GetNames(typeof(aylar)); // GetNames() metoduyla
aldığımız değerleri string tipinde "ay" adında ki dizimize aktarıyoruz.

         foreach (string ayYaz in ay)
         {
            Console.WriteLine(ayYaz);
         }
         Console.Read();
      }
   }
```

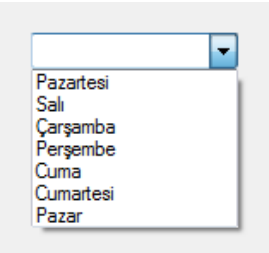
Bir sabitin enum içerisinde tanımlı olup olmadığını bulmak için IsDefined() metodunu kullanırız. Bu metot bool türünde bir sonuç döndürür.

```
class Program
   {
      static void Main(string[] args)
      {
         bool kontrol = Enum.IsDefined(typeof(aylar), "Ocak");
         if (kontrol)
         {
            Console.WriteLine("Bu ay tanımlıdır.");
         }
         else
         {
            Console.WriteLine("Bu ay tanımlı değildir.");
         }
         Console.Read();
      }
   }
```

UYGULAMA FAALİYETİ

İçinde haftanın günlerini barındıran bir numaralandırma tanımlayıp bir tane ComboBox içinde listeleyiniz.

İşlem Basamakları	Öneriler
➢ Nesne Tabanlı Programlama Yazılımını başlatıp yeni bir Windows form uygulaması oluşturunuz.	
➢ Form üzerine bir tane ComboBox ekleyiniz.	➢ Toolbox pençeresinde ComboBox elemanına çift tıklayınız. 
➢ Formun kod bölümüne geçiniz.	➢ Form üzerinde sağ tuşa tıklayıp açılan menüden View Code seçeneğini seçiniz. Klavyeden F7 tuşunu kullanarak ta yapabilirsiniz. 
➢ Gunler adında içinde haftanın günlerini barındıran bir numaralandırma tanımlayınız.	➢ Numaralandırmaların nerede tanımlandıklarını hatırlayınız. *enum Gunler { Pazartesi, Salı, Çarşamba, Perşembe, Cuma, Cumartesi, Pazar }*
➢ From1_Load bloğu içerisinde oluşturduğumuz numaralandırmadaki tüm değerleri içinde tutacak bir dizi tanımlayınız.	*string[] gunListe = Enum.GetNames (typeof(Gunler));*

➢ Foreach döngüsü yardımıyla dizide tuttuğumuz değerleri ComboBox aktarınız.	*foreach (string gun in gunListe)* *{* *comboBox1.Items.Add(gun);* *}*
➢ Programı çalıştırınız.	

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. Numaralandırmaların kullanım amacanı anladınız mı?		
2. Bir numaralandırma tanımlayabildiniz mi?		
3. Bir numaralandırma değişkeni tanımlayabildiniz mi?		
4. Tanımladığınız değişkene değer aktarabildiniz mi?		
5. Farklı numaralandırma örnekleri yapabildiniz 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 cümlelerin başında boş bırakılan parantezlere, cümlelerde verilen bilgiler doğru ise D, yanlış ise Y yazınız.

1. ( ) Enum yapısında tutulan değişkenlerin tümüne erişmek için GetNames metodu kullanılır.

2. ( ) Enum tipinin elemanlarına verilen sıra numaraları sadece Int türünden olabilir.

3. ( ) Enum tipinde ilk olarak yazılan elemana varsayılan olarak 0 numarası verilir.

4. ( ) Bir numaralandırma, (enum) anahtar sözcüğü ve arkasından küme parantezleri içerisinde türün sahip olabileceği geçerli değerleri tanımlayan ifadeler kullanarak oluşturulur.

5. ( ) Enum tipinin elemanlarına sayısal değer aktarırken, elemanların içeriği sıralı bir şekilde artıyorsa tüm elemanlara değer ataması yapılması zorunludur.

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

Uygun ortam sağlandığında Yapılarla (Struct) çalışabileceksiniz.

ARAŞTIRMA

- Sık kullanılan yapı türlerini araştırınız ve bir rapor halinde sununuz.
- Aralarında mantıksal ilişki bulunan 5 farklı grup oluşturup rapor halinde sununuz.

2. YAPILAR (STRUCT)

Nesne tabanlı programlama beş veri tipini destekler. Bunlar **Class, Struct, Enum, Interface** ve **Delegate** tipleridir. Bu öğrenme faaliyetinde Yapı (struct) veri tipini ele alacağız.

Aralarında mantıksal bir ilişki bulunan ama farklı türdeki verilerin bir arada bulunması gereken durumlarla karşılaşılabilir. İşte bu durumlarda belli bir grup verinin bir arada tutulması için oluşturulan yeni birime yapı (struct) denir.

Yapılar değer türündedir. Dolayısıyla belleğin yığın (stack) bölümünde tutulurlar.

Bellek kullanım bakımından yığın (stack) ve öbek (heap) olmak üzere ikiye ayrılır.

Yığın bölümünde veriler üst üste gelecek şekilde depolanırlar. Yeni bir veri eklendiğinde bu belleğin en üst bölgesine yerleştirilir. LIFO (Last In First Out) mantığına göre çalışır. Yani son giren ilk çıkar. Verilere erişim basit olduğu için hızlıdır.

Öbek kısmında ise boş bir alan oluşturulur ve veriler rastgele yerleştirilir. Verilere erişim karmaşık olduğundan daha yavaştır.

Değer türleri belleğin yığın (stack) bölümünde tutulurken;referans türler belleğin öbek (heap) bölümünde tutulur. Örneğin sınıflar(class) referans türünde olduklarından dolayı belleğin öbek (heap) bölümünde tutulur.

2.1. Yapı (Struct) Tanımlama

Yapıları tanımlamak için ‘struct’ anahtar sözcüğünü kullanırız. Bir yapının genel tanımlanma şekli aşağıdaki gibidir.

```
struct yapi-ismi
 {
      yapi-elemanlari-listesi;
  }
```

Örnek:

```
struct Ogrenci
      {
         int okulNo;
         string ad, soyad;
      }
```

Örnek:

```
struct Ogrenci
   {
      public int okulNo;
      public string ad, soyad;
   }
```

1. Bölüm

```
   class Program
   {
      static void Main(string[] args)
      {
         Ogrenci yeniOgrenci = new Ogrenci();
         Console.Write("Okul No giriniz : ");
         yeniOgrenci.okulNo = Convert.ToInt16(Console.ReadLine().ToString());
         Console.Write("Ad Giriniz :");
         yeniOgrenci.ad = Console.ReadLine().ToString();
         Console.Write("Soyad Giriniz :");
         yeniOgrenci.soyad = Console.ReadLine().ToString();
```

2. Bölüm

3. Bölüm

```
        Console.WriteLine("--------------------");
        Console.WriteLine("Yeni Öğrenci Bilgileri");

        Console.WriteLine("Okul No : {0}", yeniOgrenci.okulNo);
        Console.WriteLine("Adı : {0}", yeniOgrenci.ad);
        Console.WriteLine("Soyadı : {0}", yeniOgrenci.soyad);

        Console.ReadKey();
    }
```

4. Bölüm

- 1. bölümde içerisinde integer ve string tipinde üç tane değişken bulunduran bir yapı(struct) tanımladık.
- 2. bölümde Ogrenci isimli yapıdan yeniOgrenci isimli yeni bir nesne oluşturduk.
- 3. bölümde yeni oluşturduğumuz nesneye klavyeden değerler atadık.
- 4. bölümde ise nesneye atadığımız değerleri ekrana yazdırdık.

Yapılar da sınıflar gibi metotlar, değişkenler, özellikler, yapıcılar vb. içerebilir.

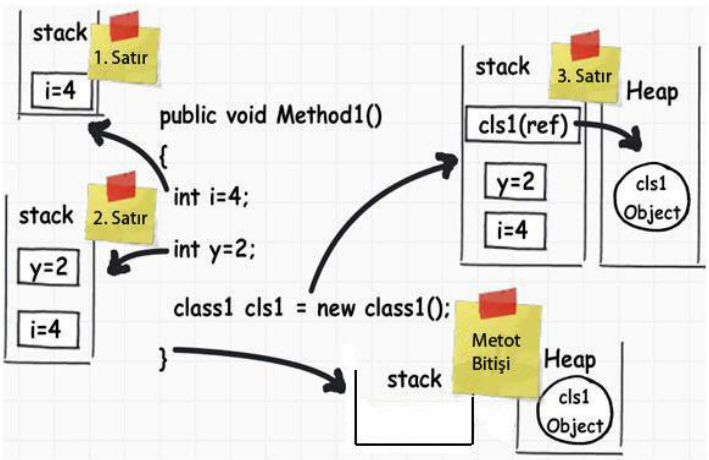
```
struct Dikdortgen
{
    public int alanHesapla(int kisaKenar, int uzunKenar)
    {
        int alan = kisaKenar*uzunKenar;
        return alan;
    }
}

struct Kare
{
    public int alanHesapla(int kenar)
    {
        int alan = kenar * kenar;
        return alan;
    }
}
```

```
class Program
{
   static void Main(string[] args)
   {
      Dikdortgen dikdortgenAlan = new Dikdortgen();
      Console.WriteLine("Dikdortgenin Alanı : {0}", dikdortgenAlan.alanHesapla(4,
5));

      Kare kareAlan = new Kare();
      Console.WriteLine("Karenin Alanı : {0}", kareAlan.alanHesapla(5));
      Console.ReadKey();
   }
}
```

Diktortgen ve Kare adında iki tane yapı (struct) tanımladık ve yapılar içinde alan hesabı yapan metot oluşturduk. Program içinde bu oluşturduğumuz yapılardan yeni nesneler türettiğimiz zaman bu yeni nesneler üzerinden metotlara erişim sağlayabiliyoruz. Şunu unutmamak gerekir ki metotlara erişim sağlayabilmemiz için public türünde tanımlanmaları gerekir.

2.2. Yapı ve Sınıf Arasındaki Farklar

- Yapılar değer türünde olduklarından belleğin yığın (stack) bölümünde, sınıflar ise referans türünde oldukları için belleğin öbek (heap) bölümünde depolanırlar.



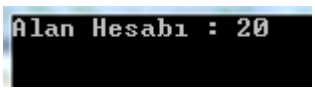
Resim 2.1: Değişken ve sınıfların bellek bölgesinde tutulma biçimleri

- Sınıflar için varsayılan (default) yapıcı (constructor) metotlar yazılabiliyorken yapılarda bunu yapamayız. Fakat yapılarda parametreli yapıcılar tanımlayabiliriz. Burada dikkat edilmesi gereken nokta ise yapılarda tanımladığımız parametreli yapıcılara ilk değer ataması yapmak zorundayız. Sınıflarda ise buna gerek yoktur. Çünkü derleyici bizim yerimize ilk değer atamasını yapacaktır.

Örnek:

```
class Dikdortgen
{
   public int kisaKenar;
   public int uzunKenar;

   public Dikdortgen()
   {
      kisaKenar = 4;
      uzunKenar = 5;
   }

   public int alanHesap()
   {
      int alan;
      alan = kisaKenar * uzunKenar;
      return alan;
   }
}

class Program
{
   static void Main(string[] args)
   {
      Dikdortgen dortgen = new Dikdortgen();
      Console.WriteLine(dortgen.alanHesap());
      Console.ReadKey();
   }
}
```

Varsayılan yapıcı metot



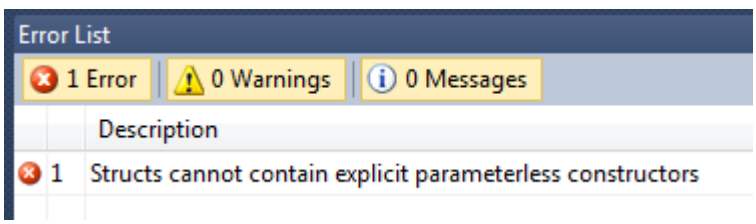
Resim 2.2: Ekran çıktısı

Yukarıdaki örnekte Diktorgen adında bir tane sınıf tanımladık. Bu sınıfa varsayılan yapıcı metot aracılığıyla kisaKenar ve uzunKenar değişkenlerine varsayılan bir değer atadık. Program içerisinde bu sınıftan yeni bir nesne türettiğimiz zaman bu değerler varsayılan olarak atanacaktır. Dikkat ettiyseniz alanHesap() metodumuza bir değer ataması yapmadığımız halde program doğru bir şekilde çalışmıştır.

```
struct Dikdortgen
{
    public int kisaKenar;
    public int uzunKenar;

    public Dikdortgen()
    {
        kisaKenar = 4;
        uzunKenar = 5;
    }

    public int alanHesap()
    {
        int alan;
        alan = kisaKenar * uzunKenar;
        return alan;
    }
}

class Program
{
    static void Main(string[] args)
    {
        Dikdortgen dortgen = new Dikdortgen();
        Console.WriteLine("Alan Hesabı : {0}", dortgen.alanHesap());
        Console.ReadKey();
    }
}
```

Varsayılan yapıcı metot

Aynı işlemi yapı (struct) kullanarak yapmaya çalışırsak aşağıdaki resimde görünen hatayı alırız. Hatanın sebebi,yapılarda parametresiz varsayılan yapıcı tanımlanamamasıdır.



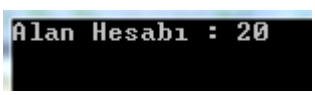
Resim 2.3: Hata penceresi

Fakat yapılarda parametre alan yapıcı metotlar tanımlayabildiğimizi söylemiştik.

```
struct Dikdortgen
{
   public int kisaKenar;
   public int uzunKenar;

   public Dikdortgen(int m_kisaKenar, int m_uzunKenar)
   {

   }
}
```

Yukarıdaki tanımlamada yapıcı metoda ilk değer ataması yapmadığımız için yine hata ile karşılaşılacaktır.

```
struct Dikdortgen
{
   public int kisaKenar;
   public int uzunKenar;

   public Dikdortgen(int m_kisaKenar, int m_uzunKenar)
   {
      kisaKenar = m_kisaKenar;
      uzunKenar = m_uzunKenar;
   }

   public int alanHesap()
   {
      int alan;
      alan = kisaKenar * uzunKenar;
      return alan;
   }
}

class Program
{
   static void Main(string[] args)
   {
      Dikdortgen dortgen = new Dikdortgen(4,5);
      Console.WriteLine("Alan Hesabı : {0}", dortgen.alanHesap());
      Console.ReadKey();
   }
}
```



Resim 2.4: Ekran çıktısı

- Yapılarda sınıflar gibi türetme yapamayız. Bir sınıf tanımlarken bir başka sınıfı temel alan yani kalıtım yoluyla oluşturabiliyoruz fakat bir yapıyı bir başka yapıyı temel alarak oluşturamayız.
- Yapılar işleri bittikleri zaman bellekten otomatik olarak silindikleri için yıkıcı metotları tanımlayamayız.

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
➢ Sizde Tarih adında bir yapı tanımlayarak, gun, ay ve yildan oluşan elemanlarını tanımlayınız.	➢ Modüldeki örneklerden faydalanabilirsiniz.
➢ Aynı örneği Tarih yapısı için bir kurucu oluşturarak geliştiriniz.	➢ Modüldeki örneklerden faydalanabilirsiniz.
➢ Bu yapıya oluşturduğunuz kurucuyu kullanarak başlangıç değerlerini aktarınız.	➢ Modüldeki örneklerden faydalanabilirsiniz.

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. Yapıların kullanım amacanı anladınız mı?		
2. Yapılar ile Sınıflar arasındaki farkları anladınız mı?		
3. Bir yapı tanımlayabildiniz mi?		
4. Yapınıza ait değişkenler tanımlayabildiniz mi?		
5. Yapı içindeki değişkenlere başlangıç değerleri verebildiniz mi?		
6. Farklı yapı örnekleri yapabildiniz 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 cümlelerin başında boş bırakılan parantezlere, cümlelerde verilen bilgiler doğru ise D, yanlış ise Y yazınız.

1. ( ) System.Object türü bir yapıdır.

2. ( ) Yapılar birer değer türüdür.

3. ( ) Yapıların küçük boyutlu veriler için kullanılması uygundur.

4. ( ) Yapılar için varsayılan yapıcı metod (default constructor) yazamayız.

5. ( ) Yapılar sınıfları kapsar ve özellikleri daha fazladır.

DEĞERLENDİRME

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

MODÜL DEĞERLENDİRME

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

1. ( ) Birçok numaralandırma değişkenine aynı değeri verebiliriz.

2. ( ) Bu_gun = (Gunler)1 ifadesi değişkene Gunler isimli numaralandırmanın 2. sıradaki değerini atar.

3. ( ) Numaralandırmalara (Enum) null değer de atanabilir.

4. ( ) Enum tiplerini tanımlarken sadece public kapsam belirteçlerini kullanabiliriz.

5. ( ) Enum tipinin elemanlarına verilen sıra numaraları char türünden olabilir.

6. ( ) Yapılar bazı özellikleri kırpılmış sınıflardır.

7. ( ) Bir yapı değişkenini diğer bir yapı değişkenine eşitleyemeyiz ya da atayamayız.

8. ( ) Oluşturulan bir yapı kurucusunun tüm alanlarına her zaman başlangıç değeri atamak zorundadır.

9. ( ) Yapılar öbek (heap) adı verilen alanda tutulurlar.

10. ( ) *Zaman simdi = new Zaman (14, 30);* şeklindeki bir kullanımla yapı değişkenlerine ilk değerleri aktarılabilir.

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	**Y**
3	**D**
4	**D**
5	**Y**

ÖĞRENME FAALİYETİ-2'NİN CEVAP ANAHTARI

1	**Y**
2	**D**
3	**D**
4	**D**
5	**Y**

MODÜL DEĞERLENDİRME CEVAP ANAHTARI

1	**D**
2	**D**
3	**D**
4	**Y**
5	**Y**
6	**D**
7	**Y**
8	**D**
9	**Y**
10	**D**

KAYNAKÇA

- Sharp John, **Adım Adım Microsoft C# 2008**, Arkadaş Yayınevi/ Ankamat Matbaacılık, Ankara, 2009.