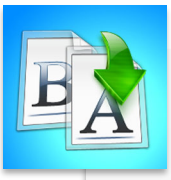
DEĞİŞMEZLER, SABİTLER VE NUMARALANDIRMA



İÇİNDEKİLER

- Değişmezler (Literals)
  - Tamsayı Değişmezleri
  - Kayan Noktalı Değişmezleri
  - Karakter Değişmezleri
  - Dize Değişmezleri
  - Boş - Değersiz Değişmezler
  - Mantıksal Değişmezler
- Sabitler (Constants)
- Numaralandırma (Enumerations)
  - Numaralandırma Temelleri

HEDEFLER

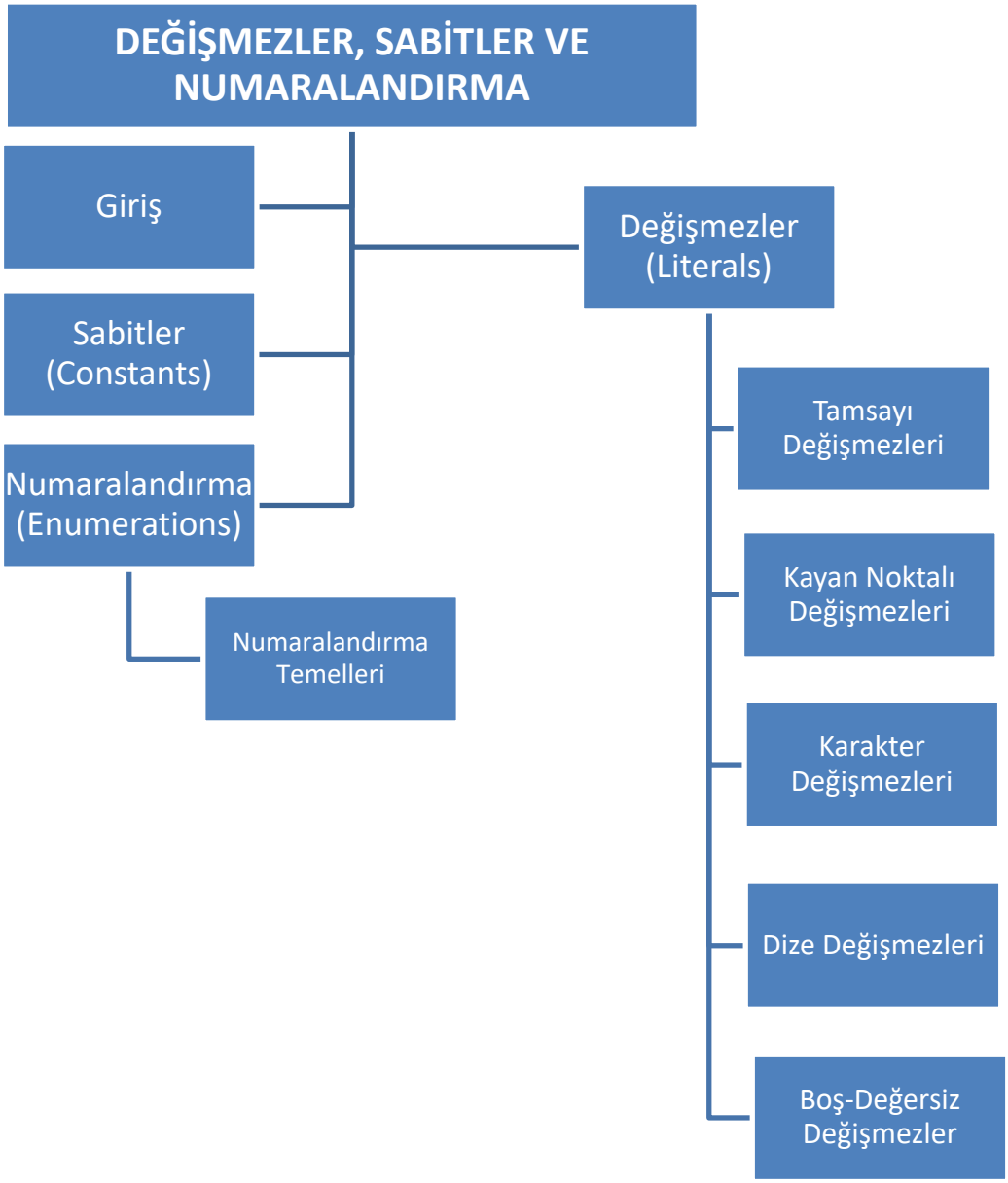
- Bu üniteyi çalıştıktan sonra;
  - Değişmezler, sabitler ve numaralandıma hakkında bilgi edinebilecek,
  - Nesne yönelimli programlama içinde kullanımını görebilecek,
  - C# programlama dili ile değişmezler, sabitler ve numaralandırma kullanımı hakkında örnek kodları görebilecek,
  - Kendi kod bloğunuz içinde ilgili ifadelerin kullanabilecek seviyeye gelebileceksiniz.

Atatürk Üniversitesi
Açıköğretim Fakültesi

NESNE TABANLI PROGRAMLAMA I

Dr. Muhammet DAMAR

ÜNİTE
14


DEĞİŞMEZLER, SABİTLER VE NUMARALANDIRMA
Giriş
Sabitler (Constants)
Numaralandırma (Enumerations)
Numaralandırma Temelleri
Değişmezler (Literals)
Tamsayı Değişmezleri
Kayan Noktalı Değişmezleri
Karakter Değişmezleri
Dize Değişmezleri
Boş-Değersiz Değişmezler

GİRİŞ

Bu bölüm içerisinde değişmezlerden (literals), sabitlerden (constants) ve numaralandırma (enumarations) kavramlarından bahsedilecek ve Visual Studio 2019 ortamında C# programlama diliyle kullanımı açıklanacaktır. Bu bölümü tamamladıktan sonra projelerinizde ihtiyaçlarınıza bağlı değişmezler, sabitler ve numaralandırma mantığı ile programlama yapabileceksiniz.

Sabitler, programın yürütülmesi sırasında değiştiremeyeceği sabit değerleri ifade eder. Sabitler, tamsayı sabiti, kayan sabit, karakter sabiti veya dize değişmezi gibi temel veri türlerinden herhangi biri olabilir.

Değişmezlere (literals), sabitlere (constants) ve numaralandırmalara (enumerations) bu bölüm içerisinde üzerinde duracağımız kavramlar olacaktır. Nesne yönelimli bir programlama dilleri olan, C#, Java, F#, C++ gibi dillerde proje ihtiyaçlarına bağlı sıklıkla kullanılan değişkenlerdir. Değişmezlere, sabitlere ve numaralandırmalar derleme zamanı oluşturulur ve değiştirilememeleri en önemli özellikleri olarak öne çıkmaktadır.

Sabitler, programın yürütülmesi sırasında değiştiremeyeceği sabit değerleri ifade eder. Bu sabit değerlere değişmez değerler de denir. Sabitler, tamsayı sabiti, kayan sabit, karakter sabiti veya dize değişmezi gibi temel veri türlerinden herhangi biri olabilir. Numaralandırmalar ise sabitler ile çalışmanın kolay bir yolunu sağlar atanan sabit değerleri adlar ile ilişkilendirir. Program yürütme sırasında değişkenlere atanan değerler değişebilirken, değişmezler ve sabitlere depolanan değerler ise sabit kalır. Bu ünitede değişmezler, sabitler ve numaralandırmalar ile ilgili konular verilen örnekler ile anlatılacaktır.

Bu ünite kapsamında değişmezler, sabitler ve numaralandırma konularına ve bu konulara ilişkin Visual Studio 2019 ortamında C# programlama dili ile yapılan örneklere yer verilecektir.

Değişmez, değişkenler tarafından kullanılan bir değerdir. Değerler tamsayı, kayan nokta veya dize vb. olabilir.

DEĞİŞMEZLER (LITERALS)

Değişmezler, değişkenler tarafından kullanılan bir değerdir. Değerler tamsayı, kayan nokta veya dize vb. olabilir.

Değişmez değerler, örtük olarak bir veri türüne atanan sabit değerlerdir. Genellikle değişkenleri başlatmak için kullanılır. Aşağıdaki C# kod bloğu içinde bir tamsayı değerinin değeri 2 artırmak için kullanılmaktadır.

```
CiftSayi = CiftSayi + 2;
```

Aşağıda bir değişmez kullanım örneği gösterilmektedir.

```
using System;

namespace ConsoleAppTest
{
    class Program
    {
        static void Main()
        {
            // Burada 2021 bir sabit/değişmezdir.
            int sabitDegiskenim = 2021;
            Console.WriteLine(sabitDegiskenim);
        }
    }
}
```

Ondalık Değişmezler (Onluk Tabanda): Bu gösterimde, izin verilen rakamlar 0-9 arasındadır.

Değişmezlerin kullanımı tüm platformlarda aynı özellikte çalışır. Bu durum değişmez kullanımının mantığından kaynaklanmaktadır. Derleyici, değişmezi inceleyerek, değişmeze bir veri türü atar. Ondalık değere sahip olanlar double veri tipi olarak atanır. True veya False anahtar sözcüklerine boolean veri türü atanır. Değişmez bilgi tırnak içindeyse, bir dize (string) veri türü olarak atanır. Aşağıdaki kod satırında, iki dize değişmezi birleştirilir ve bir dize değişkenine atanır:

```
AdSoyad = "Mustafa" + "Kemal" + "Atatürk";
```

Sabit değere bir tür karakteri ekleyerek değişmezin varsayılan veri türü atamasını geçersiz kılmak mümkündür (override işlemi). Örneğin, 12.25 değerine double veri türü atanacaktır, ancak 12.25f değeri derleyicinin buna tek bir veri türü atamasına neden olacaktır. Ondalık sayıların sonuna eklenen “F” ya da “f”, ondalık türü float olarak belirtilir.

Değişmezler aşağıdaki türlerden olabilir; Tamsayı Değişmezler (Integer Literals), Kayan Noktalı Değişmezler (Floating-point Literals), Karakter Değişmezler (Character Literals), Dize Değişmezleri (String Literals), Boş - Değersiz Değişmezler (Null Literals), Mantıksal Değişmezler (Boolean Literals).

Sabitler, tanımlarından sonra değerlerinin değiştirilememesi dışında normal değişkenler gibi ele alınır. Aşağıda sırasıyla örnekler ile değişmez türleri ifade edilecektir.

Tamsayı Değişmezleri

Sekizli değişmezler (Sekizlik Tabanda): Bu gösterimde, izin verilen rakamlar 0-7 arasındadır.

Tamsayı türünün değişmez değeri, tamsayı değişmezi olarak bilinir. Sekizli, ondalık, ikili veya onaltılık sabit olabilir. Ondalık sayılar için ön ek gerekmez. U veya u gibi tamsayı değişmezleri ile bir sonek, işaretsiz sayılar için kullanılırken l veya L uzun (long integer) sayılar için kullanılır. Varsayılan olarak, her değişmez değer int türündedir. İntegral veri türleri (byte, short, int, long gibi) için değişmez değerleri şu şekillerde belirtebiliriz:

Ondalık Değişmezler (Onluk Tabanda): Bu gösterimde, izin verilen rakamlar 0-9 arasındadır.

```
//Ondalık Değişmezler
int degizmez = 101;
```

Sekizli değişmezler (Sekizlik Tabanda): *Bu* gösterimde, izin verilen rakamlar 0-7 arasındadır.

```
// Sekizli sayı, 0 önekini almalıdır.
int degismez= 0146;
```

On Altılık Değişmezler (On Altılık Tabanda): Bu gösterimde izin verilen rakamlar 0-9 ve karakterler A-F'dir

On Altılık Değişmezler (On Altılık Tabanda): Bu gösterimde izin verilen rakamlar 0-9 ve karakterler A-F'dir. Harf ile gösterim için hem büyük hem de küçük harf kullanabiliriz. Bildiğimiz gibi, C# büyük/küçük harf duyarlı bir programlama dilidir. Ancak burada c# büyük/küçük harfe duyarlı değildir.

```
// Onaltılık sayı 0X veya 0x önek almalıdır
int degismez = 0X456Face;
```

İkili Değişmezler (İkilik Tabanda): Bu gösterimde izin verilen rakamlar sadece 1 ve 0'dır.

```
// İkili sayı, 0b ile önek olmalıdır.
int degismez = 0b100;
```

Aşağıda hatalı ve hatasız değişmez kullanım örnekleri gösterilmektedir.

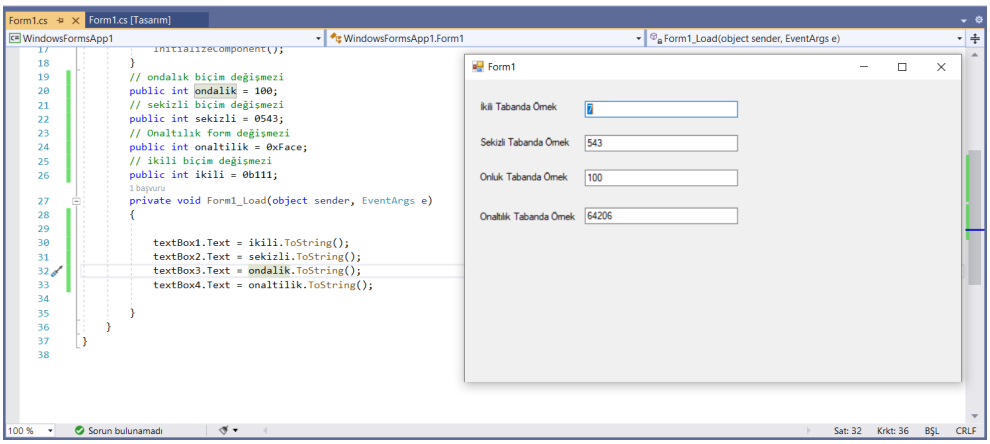
İkili Değişmezler (İkilik Tabanda): Bu gösterimde izin verilen rakamlar sadece 1 ve 0'dır.

Örnek

- 07887 // geçersiz: 8 sekizlik bir rakam değil
- 067uu // geçersiz: soneki (u) uygun değildir, tekrarlanmaktadır
- 0b107 // geçersiz: 7 sayısı ikili bir rakam değildir
- 0b101 // geçerli ikili değişmez
- 456 // geçerli ondalık değer
- 02453 // geçerli sekizlik değişmez
- 0x65d // geçerli onaltılık değişmez
- 16789 // geçerli int türünde değişmez
- 304U // geçerli negatif değeri almayan int türünde değişmez
- 3078L // geçerli long türünde değişmez
- 965UL // geçerli negatif değeri almayan long türünde değişmez

Aşağıda tamsayı değişmezlerin kullanımı ondalık, sekizli, onaltılık, ikili biçimde C# programlama kodu gösterilmektedir. Resim 14.1 üzerinde de elde edilen ekran görüntüsü paylaşılmaktadır.

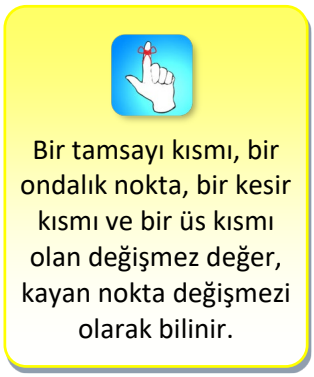
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        // ondalık biçim değişmezi
        public int ondalik = 100;
        // sekizli biçim değişmezi
        public int sekizli = 0543;
        // Onaltılık form değişmezi
        public int onaltilik = 0xFace;
        // ikili biçim değişmezi
        public int ikili = 0b111;
        private void Form1_Load(object sender, EventArgs e)
        {

            textBox1.Text = ikili.ToString();
            textBox2.Text = sekizli.ToString();
            textBox3.Text = ondalik.ToString();
            textBox4.Text = onaltilik.ToString();
        }
    }
}
```

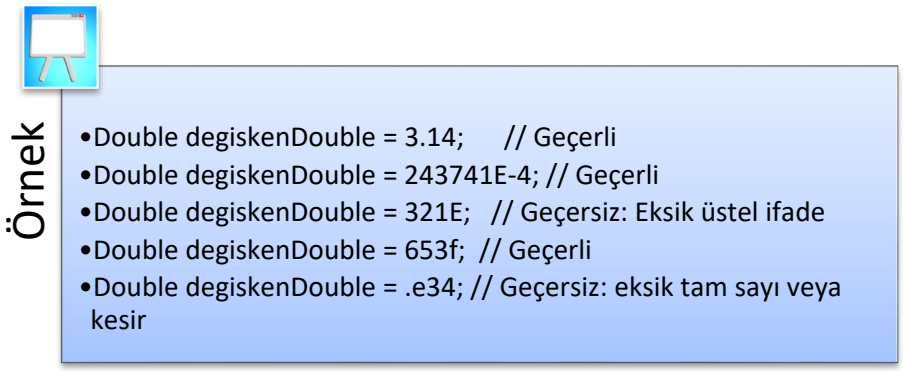


Resim 14.1. Tamsayı Değişmezlerinin C# Kod Bloğunda Ekran Çıktısı Görünümü

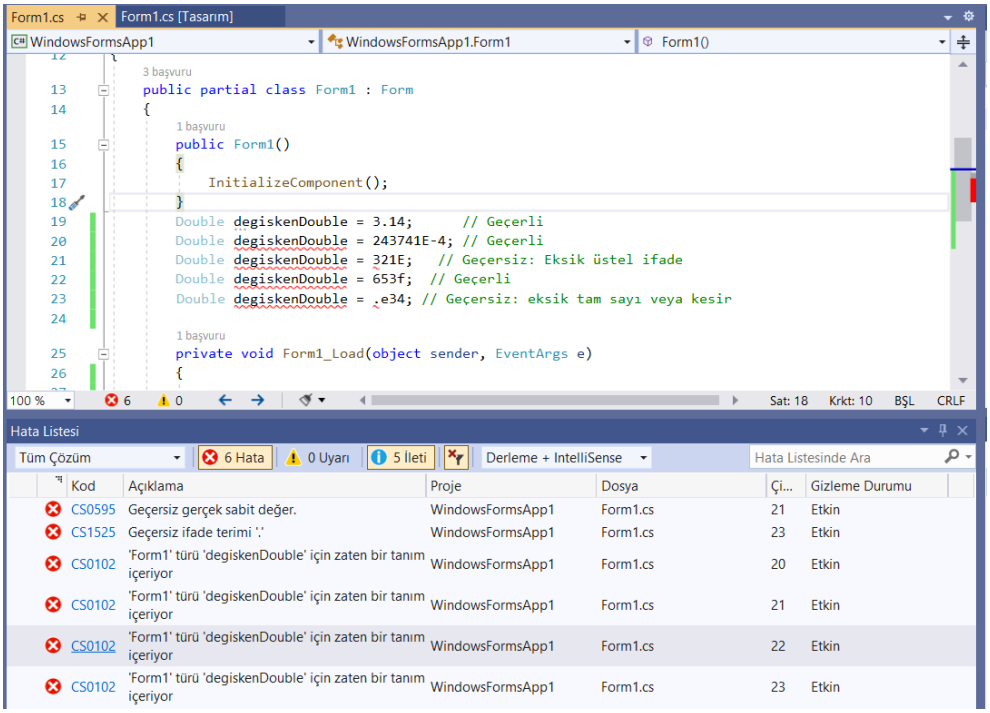
Kayan Noktalı Değişmezler

Bir tamsayı kısmı, bir ondalık nokta, bir kesir kısmı ve bir üs kısmı olan değişmez değer, kayan nokta değişmezi olarak bilinir. Bunlar, ondalık biçimde veya üstel biçimde temsil edilebilir. Varsayılan olarak, her kayan nokta değişmezi *double* türündedir ve doğrudan kayan değişkene atayamayız. Ancak kayan nokta değişmezini *f* veya *F* ile eklenerek kayan noktalı tip olarak belirtebiliriz. Kayan nokta değişmezini sonek *d* veya *D* ile çift tip olarak belirtebiliriz.





Yukarıda örnek kutusu içinde kayan nokta değişmezler ile ilgili Visual Studio 2019 ortamında yazılan örnek kodlardaki hataların görünümü aşağıda Resim 14.2 üzerinde gösterilmektedir.

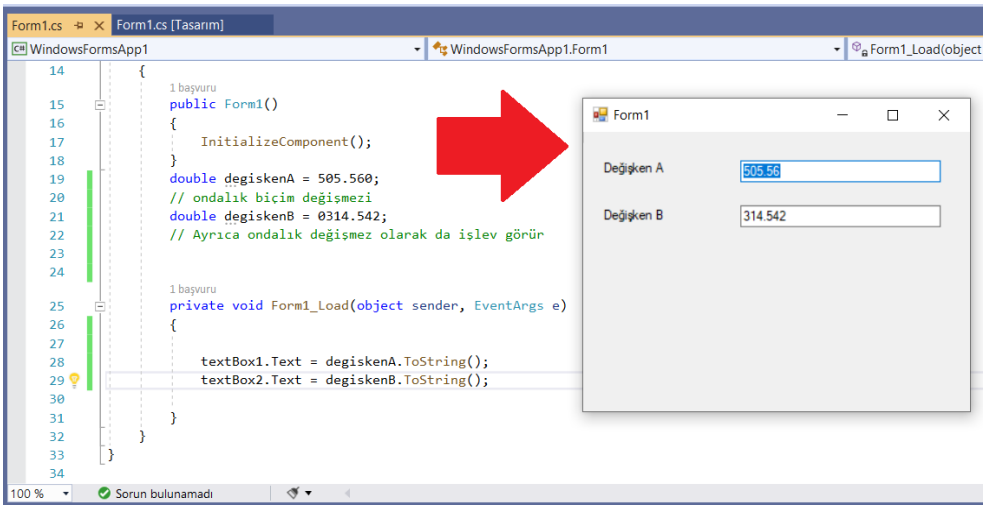


Resim 14.2. Derleyici Üzerinde Kayan Noktalı Değişmez Kod Bloğundaki Hata Görünümü

Kayan noktalı değişmezlerin C# programlama dili ile gösterimine bir başka örnek kod ise aşağıda sunulmaktadır.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        double degiskenA = 505.560;
        // ondalık biçim değişmezi
        double degiskenB = 0314.542;
        // Ayrıca ondalık değişmez olarak da işlev görür
        private void Form1_Load(object sender, EventArgs e)
        {

            textBox1.Text = degiskenA.ToString();
            textBox2.Text = degiskenB.ToString();
        }
    }
}
```

İlgili kodun ekran görünümü ve derleyici üzerinde kodların görünümü Resim 14.3 üzerinde gösterilmektedir. İlgili kod bloğunda yazan kodları öncelikle Visual Studio 2019 ortamına aktarıp kodlar üzerinde değişiklik yaparak konuyu daha iyi kavrayabilirsiniz.



Resim 14.3. Kayan Noktalı Değişmezler İle Geliştirilen Kod Bloğunun Çalıştığında Ekran Görünümü

Karakter Değişmezleri (Character Literals)

Karakter veri türleri için değişmez değerleri üç şekilde ifade edilebilir:

Tek Tırnak Gösterim: Tek tırnak içinde karakter değişmezleri veri tipini tek karakter olarak belirtebiliriz.

```
char karakter = 'a';
```

Karakter veri türleri için değişmez değerleri, tek tırnak gösterim, evrensel kod gösterim, özel gösterim olmak üzere üç şekilde ifade edilebilir.

Evrensel Kod (Unicode) Gösterim: Evrensel kod gösteriminde char değişmezlerini belirtebiliriz. Burada xxxx 4 onaltılık sayıyı temsil eder.

```
char karakter = '\u0061'; // Burada /u0061 a'yı temsil eder.
```

Özel Gösterim: Her kaçış karakteri char değişmezleri olarak belirtilebilir.

```
char karakter1 = '\n'; // Bir satır atlamak için kullanılır
char karakter2 = '\t'; // Bir tab atlamak için kullanılır
```

Tablo 14.1. Bazı Karakterlerin Özel Gösterimi

Özel Gösterim	Anlamı
\\	\
\'	'
\?	?
\"	"
\b	Geri Tuşu
\n	Yeni Satıra Geç
\t	Tab Yap
\f	Form Besleme
\r	Satır Başı
\v	Dikey Sekme
\xhh…	Bir veya daha fazla basamaktan oluşan onaltılık sayı

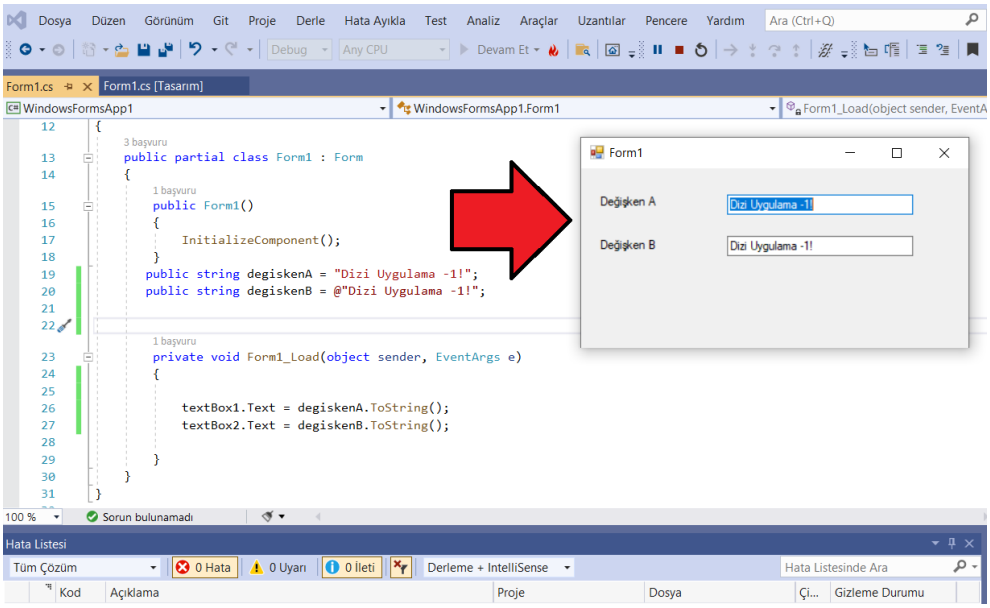
Örnek kodlarımızı, Visual Studio 2019 ortamına aktararak çalıştırmayı, kodlar üzerinde değişiklikler yaparak farklı sonuçlar elde etmeyi unutmayın!

Örnek

- String dizi1 = " Dizi Uygulama -1!";
- String dizi2 = @"Dizi Uygulama -1!";

Örnek Bloğunda da görüleceği üzere karakter değişmezlerinin C# kod bloğu üzerinde aynı ekran görüntüsünü verebilecek farklı gösterimleri söz konusudur. Resim 14.4 'de de görüleceği üzere karakter değişmezlerinin iki farklı gösterimi Visual Studio 2019 üzerindeki yazılan program kod bloğu ve programı çalıştırdığımızda karşımıza çıkan ekran görünümü görülmektedir.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        public string degiskenA = "Dizi Uygulama -1!";
        public string degiskenB = @"Dizi Uygulama -1!";
        private void Form1_Load(object sender, EventArgs e)
        {

            textBox1.Text = degiskenA.ToString();
            textBox2.Text = degiskenB.ToString();
        }
    }
}
```



Resim 14.4. Karakter Değişmezleri Dizi Değişkeninin İki Farklı Gösterimi

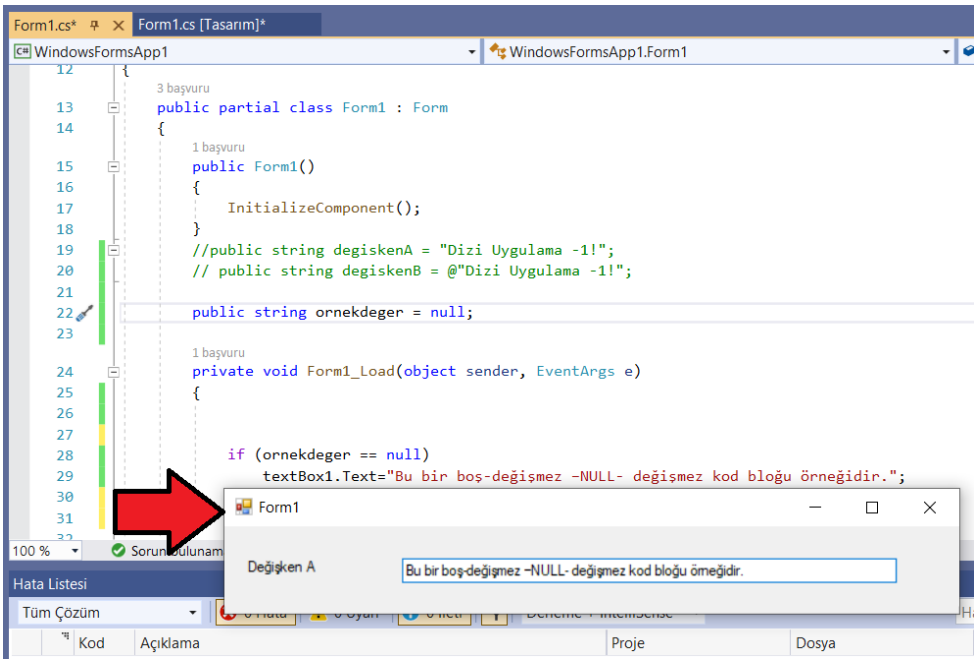
Boş - Değersiz Değişmezler (Null Literals)

Boş - Değersiz Değişmezler, null türünü belirten değişmez değerdir. Ayrıca, null herhangi bir başvuru türüne sığabilir. Bu nedenle polimorfizm için çok iyi bir örnektir.

Boş - Değersiz Değişmezler, null türünü belirten değişmez değerdir. Ayrıca, null herhangi bir başvuru türüne sığabilir.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        public string ornekdeger = null;

        private void Form1_Load(object sender, EventArgs e)
        {
            if (ornekdeger == null)
                textBox1.Text="Bu bir boş-değişmez –NULL- değişmez kod bloğu
örneğidir.";
        }
    }
}
```

Boş değersiz değişmez örnek kodunun derleyici üzerinden görünümü ve kodun çalıştırıldığında karşılaştığımız ekran görüntüsü Resim 14.5 üzerinde ifade edilmektedir.



Resim 14.5. Boş Değersiz Değişmezler Örnek Kodu ve Çalıştırıldığında Ekran Görüntüsü

Mantıksal Değişmezler (Boolean Literals)

Mantıksal yani Boolean değişmezleri için yalnızca iki değere izin verilir, yani true ve false. Aşağıda örnek bir gösterim görülmektedir.

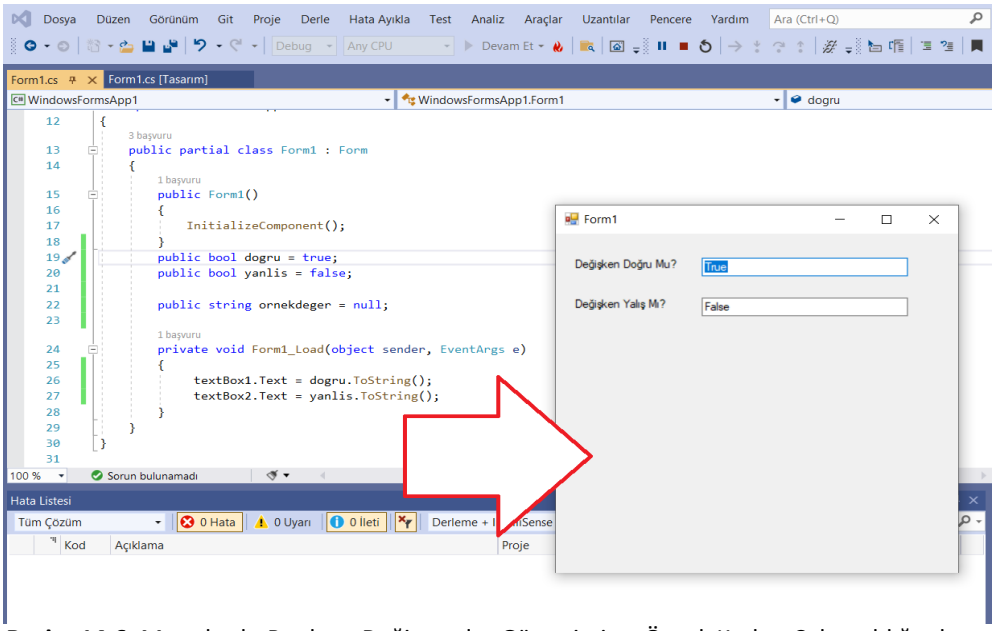
Mantıksal değişmezler için yalnızca iki değere izin verilir, yani true ve false.

Örnek

- bool dogru = true;
- bool yanlis = false;

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        public bool dogru = true;
        public bool yanlis = false;

        public string ornekdeger = null;

        private void Form1_Load(object sender, EventArgs e)
        {
             textBox1.Text = dogru.ToString();
             textBox2.Text = yanlis.ToString();
        }
    }
}
```

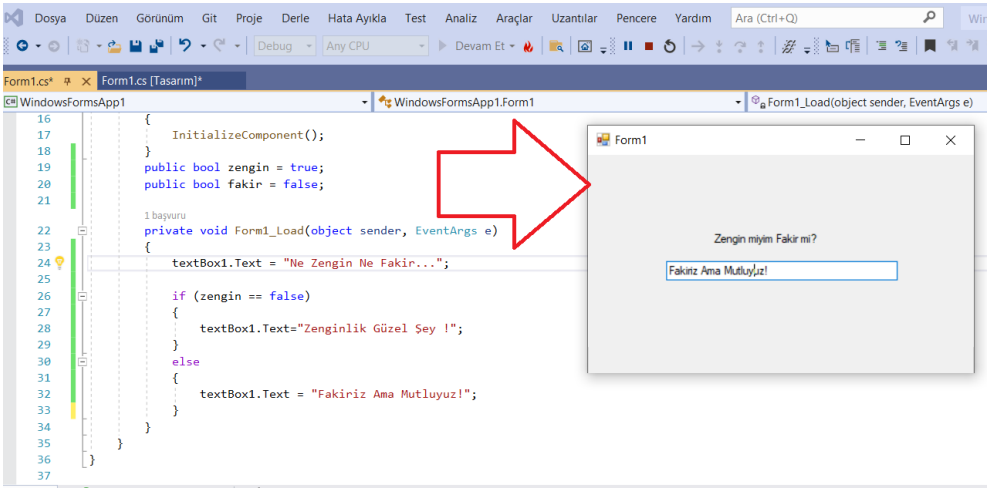
Mantıksal yani boolean değişmezler örnek kodunun derleyici üzerinden görünümü ve kodun çalıştırıldığında karşılaştığımız ekran görüntüsü Resim 14.6 üzerinde görülebilir. Örnek kodları Visual Studio ortamında çalıştırmayı ve kodlar üzerinde değişiklikler yapmayı unutmayınız.



Resim 14.6. Mantıksal - Boolean Değişmezler Gösterimi ve Örnek Kodun Çalıştırıldığında Ekran Görüntüsü

Mantıksal yani boolean değişmezlerin if yani eğer bloğu içinde tanımlanması ve kullanımını aşağıdaki kod bloğu üzerinde görebilirsiniz. Kodumuz çalıştırıldığında Resim 14.7 üzerinde gösterilen ekran ile karşılaşılmaktadır.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        public bool zengin = true;
        public bool fakir = false;

        private void Form1_Load(object sender, EventArgs e)
        {
            textBox1.Text = "Ne Zengin Ne Fakir...";

            if (zengin == false)
            {
                textBox1.Text="Zenginlik Güzel Şey !";
            }
            else
            {
                textBox1.Text = "Fakiriz Ama Mutluyuz!";
            }
        }
    }
}
```



Resim 14.7. Mantıksal - Boolean Değişmezlerin Eğer - If Bloğu İçerisinde Gösterimi ve Ekran Görünümü

İlgili Değişmezlerin C# İle Tanımlama

C# programlama dilinde sabit değişkenler için de değişken isimlendirme ile ilgili temel kuralları aşağıdaki şekilde özetlemek mümkündür.

- Visual C#, programlama dili Case sensitivity özelliğine sahip yani büyük küçük harfe duyarlı bir dildir. Yani *DEGISKEN, degisken, degisKen* tanımlamalarının hepsi program içinde farklı bir değişkeni ifade edecektir.
- Değişken tanımlamada Türkçeye özgü harflerin kullanımı (*ı,İ,ş,Ç,Ğ*, vb.) uygun bir kullanım değildir. Bu harfler program kodlarınız içinde kullanmayınız. Programcılık kültürü adına ve İngilizce yatkınlığı için de mümkün olduğu kadar İngilizce değişken kullanmanız da ayrıca tavsiye edilir.
- Bir değişkende harfler, rakamlar ve alt çizgi olabilir. Fakat değişken adı sadece *İngilizce* alfabeden bir harf ile ve yalnızca alt çizgi ile başlayabilir. Bir rakamla başlayamaz. Her değişkenin adı her zaman bir harfle başlamalıdır. *1degisken, 8BuyukHarf, 123, 1SAYAC* gibi değişken isimleri hatalı kullanıma örnektir. *Degisken1, Variable1, Value_1, _value, _degiskenX* gibi kullanımlar doğru kullanıma örnektir.
- Değişken adında boşluk bırakılamayacağı (Degisken X, Buyuk Harf_Degisken, Yeni Degisken gibi kullanımlar hatalıdır.) unutulmanalıdır. Ayrıca özel karakterler *(+, /, ? vb.)* değişkenler değişken tanımlamada kullanılamaz.
- Programlama diline özgü olan veya tüm programlama dillerinde geçerliliğe sahip değişkenler (static, int, char, if, for, else, while gibi) veya belirli bir sınıfa özgü (C# math sınıf üyelerinin C# kod yazım sırasında kullanımı, math, cos, sin, tanh gibi).

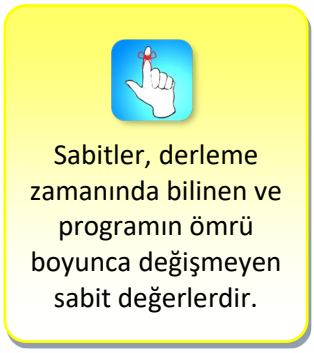
Aşağıda ayrıca birkaç örnek ile C# kod bloğu içinde tanımlama hataları gösterilmektedir. Bu tür kullanımlardan kodlarınızın hata vermemesi için sakınmanız gerekmektedir.

Const, değişkenleri sabitler olarak tanımlamak için kullanılan anahtar sözcüktür.

Örnek

```
•int 07DEĞİŞKENTANIMI= 0;
•// Türkçe karekter ve sayı ile başlanmıştır. Geçersiz kullanım.
•int +sayac=50;
•// Özel karekter kullanımına örnektir. Geçersiz kullanım.
•int degisken=1; int Degisken=2; int DEGISKEN=3;
•// Her biri farklı bir değişkeni göstermektedir. geçerli kullanım örneğidir.
•int char=5; int class=0; ,string static=25;
•// Geçersiz kullanım örnekleridir.
```

SABİTLER (CONSTANTS)



Sabitler, derleme zamanında bilinen ve programın ömrü boyunca değişmeyen sabit değerlerdir.

Sabitler, derleme zamanında bilinen ve programın ömrü boyunca değişmeyen sabit değerlerdir. Bir sınıfın veya yapının içerebileceği üye türlerinden birisidir. Çoğu zaman kodunuzda aynı sabit değeri tekrar tekrar kullanmanız gerekir. Örneğin, bir dizi geometrik hesaplamanın pi değerini kullanması gerektiğinde, kodunuzdaki değişmeyecek bir değer olan pi değerini tekrarlamak yerine, bir sabit kullanarak bu durumu engelleyebiliriz. *Const* anahtar sözcüğünü ve ardından veri türünü ve sabit adını kullanarak bir sabit değer tanımlanmaktadır. Visual C# programlama dilinde Sabit tanımlama işlemi, aşağıda gösterildiği şekilde yapılmaktadır.

```
const  <Veri Tipi> Değişken_İsmi;
```

Değişkenler program boyunca aynı değeri alabildiği gibi, ihtiyaca göre istenirse farklı değerler de alabilir. Fakat bazı özel durumlarda değişken değerlerinin program boyunca aynı kalması istenir. Program boyunca değerinin değişmemesi gereken veriler sabit olarak tanımlanırlar. Aşağıdaki şekilde tanımlanabilir.

```
const double piSayim = 3.14159265358979323846;
const string sabitAdı = "Sabitim";
```

Aynı türden birden çok sabiti tek seferde aşağıdaki gibi bildirebilirsiniz:

```
const int bir = 1, iki = 2, üç = 3;
```

Sabitler *const* değiştiricisi ile bildirilmiştir. Yalnızca C# yerleşik türleri (System.Object hariç) const olarak bildirilebilir. Sınıflar, yapılar ve diziler dahil olmak üzere kullanıcı tanımlı türler const olamaz. C#, const yöntemlerini, özelliklerini veya olaylarını desteklemez.

Const, değişkenleri sabitler olarak tanımlamak amacıyla kullanılan anahtar sözcüktür. Sabitler kullanılırken aşağıdaki temel kurallara dikkat etmek gerekir. Bunlar:

- Sabit, değeri bildirildiği anda başlatılması gereken, *const* anahtar sözcüğüyle bildirilen statik değişmez bir değişkendir.

- Sabit ifadeler kendi yapılarından dolayı *static* bir nesne oldukları için ayrıca *static* anahtar sözcüğü kullanılmaz.
- Sabitlerin hemen bildirilmesi gerekmekte ve daha sonra değiştirilememektedir. Bir sabite atadığınız değerin sabit bir ifade olması ve derleyicinin değeri derleme zamanında zaten değerlendirebilmesi gerekir.

Boolean değerlerinin ve dizelerin bir sabit için gayet iyi kullanılabilirken örneğin *DateTime* gibi tarih ve zaman nesnesi kullanımı için sabitler uygun bir kullanım değildir.

```
const DateTime suan = DateTime.UtcNow;
//hatalı kullanım
```

Bir yıl içindeki ay sayısı her zaman 12'dir. Sınıfın kendisi tarafından bile değiştirilemez. Aslında, derleyici C# kaynak kodunda bir sabit tanımlayıcıyla karşılaştığında (Örneğin, Aylar), değişmez değişken değerini doğrudan ürettiği ara dil koduna koyar. Aşağıda kod bloğu bu durumu yansıtan bir kod bloğudur.

```
public class benimTakvimim
{
    const int aylar = 12;
}
```

DLL dosyaları gibi başka bir kodda tanımlanan sabit değerlere başvurduğunuzda dikkatli hareket edilmesi gerekir. Yeni bir DLL sürümü sabit için yeni bir değer tanımlıyorsa, programınız yeni sürüme yeniden derlenene kadar eski değişmez değeri de

Çalışma zamanında bir sabitle ilişkili değişken adresi olmadığından, const alanlar başvuruya göre geçirilemez. Aynı türden birden fazla sabit aynı anda bildirilebilir. Bir yılın günleri, haftaları veya günleri standarttır. Sabit kullanımına idea bir örnektir. Bir uygulamada gün, ay, veya yıl referans alarak uygulama yazıyorsak programın çalışma anında bir defa değerler atanarak, sabit const değişken ile programınızda kullanabilirsiniz.

```
public class benimTakvimim2
{
    public const int aylar = 12;
    public const int haftalar = 52;
    private const int gunler = 365;
}
```

Sabitler genel, özel, korumalı, iç, korumalı iç veya özel korumalı olarak işaretlenir. Bu erişim değiştiricileri, sınıf kullanıcılarının sabite nasıl erişebileceğini tanımlar. Aşağıdaki kod bloğu da bu kullanıma örnektir. *Public* ile tanımlanmış aylar, haftalar değişkenlerine benimTakvimim3 sınıfı dışında erişilebilirken, günler değişkeni *private* kullanıma sahip olduğu için benimTakvimim3 sınıfı dışında kullanımı mümkün değildir.

```
public class benimTakvimim3
{
  public const int aylar = 12;
  public const int haftalar = 52;
  private const int gunler = 365;
  const double haftaBasinaGun = (double)gunler/(double)haftalar;
  const double ayBasinaGün = (double)gunler / (double)aylar;
}
```

Sabitlere statik alanlar gibi erişilebilir. Çünkü sabitin değeri türün tüm örnekleri için aynıdır. Bunları bildiren anahtar *static* sözcüğünü kullanmazsınız.

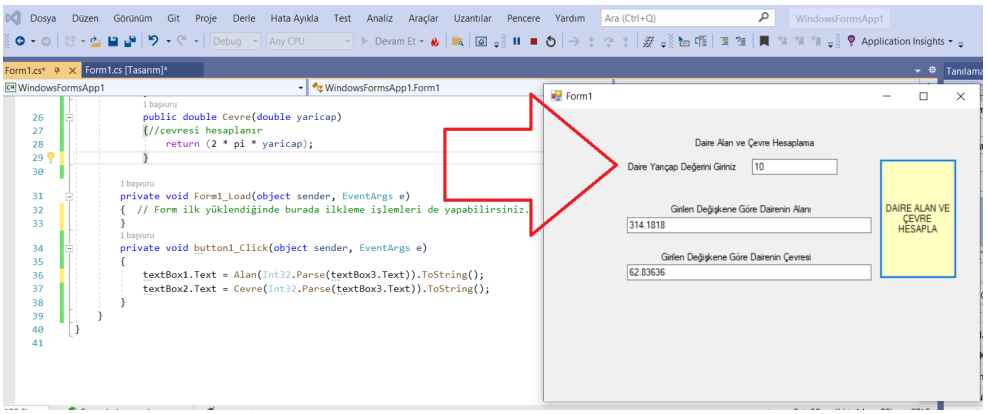
Sabiti tanımlayan sınıfta yer alan ifadeler, sabite erişmek için sınıf adı sonrasında nokta ve sabitin adını kullanmak gerekir. Aşağıda kod bloğu üzerinde görülebilir.

```
int geciciAyDegeri = benimTakvimim2.aylar;
// Dogru – public, sınıf dışında bu değer kullanılabilir.
int geciciGunDegeri = benimTakvimim2.gunler;
// Yanlış – private sadece o sınıf içinde kullanılır.
```

Sabiti tanımlayan sınıfta yer alan ifadeler, sabite erişmek için sınıf adı sonrasında nokta ve sabitin adını kullanmak gerekir.

Sınıf adı niteleyicisi kullanımı, siz ve sabiti kullanan diğer kullanıcıların sabit olduğunu ve değiştirilememesi için emin olmasını sağlar. Aşağıda kod bloğu üzerinde örnek bir dairenin alan ve çevre hesabı verilmektedir. Yukarıdaki örneklerde nasıl gün, ay ve hafta değerleri bir yıl için sabit ise pi değeri de bir dairenin hesabında, alan, çevre veya hacim hesaplarında kullanmak için ideal bir *sabit* kullanımına örnektir. Aşağıdaki kod bloğunun derleyici üzerindeki görünümü ve çalıştırılan kodun ekran çıktısı Resim 14.8 üzerinde görülebilir.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
            public const double pi = 3.141818; // sabit değer olarak pi değeri
            public double Alan(double yaricap)
            { // alan hesaplanır
                return pi * yaricap * yaricap;
            }
            public double Cevre(double yaricap)
            {//çevresi hesaplanır
                return (2 * pi * yaricap);
            }
        private void Form1_Load(object sender, EventArgs e)
        {  // Form ilk yüklendiğinde burada ilkleme işlemleri de yapabilirsiniz.
        }
        private void button1_Click(object sender, EventArgs e)
        {
            textBox1.Text = Alan(Int32.Parse(textBox3.Text)).ToString();
            textBox2.Text = Cevre(Int32.Parse(textBox3.Text)).ToString();
        }
    }
}
```



Resim 14.8. Sabit Kullanımı Dairenin Çevre ve Alan Hesabı Üzerinden Gösterimi

NUMARALANDIRMA (ENUMERATIONS)

Numaralandırma; belli sözcüklerin, belli tamsayıları temsili durumlarında kullanılan bir yapıdır.

Numaralandırma; belli sözcüklerin, belli tamsayıları temsili durumlarında kullanılan bir yapıdır. Değişkenlerin alabileceği değerlerin sabit olduğu durumlarda kullanılır. C# programlama dili için numaralandırma (enumarations) işlemi yapabilmek için *enum* anahtar sözcüğü kullanmamız gerekmektedir.

Bir numaralandırma aşağıdaki türlerden herhangi birinden türetilebilir:

- byte,
- sbyte,
- short,
- ushort,
- int,
- uint,
- long,
- ulong.

Esasen, bir numaralandırma, yalnızca bir dizi sonlu seçeneğe izin veren bir türdür ve her seçenek bir sayıya karşılık gelir.

Genellikle bir değişkenin değerini önceden tanımlanmış birkaç ilgili sabitten birine atamanız gerekir. Bu durumlarda, değerleri gruplandırmak için bir numaralandırma türü oluşturabilirsiniz. Numaralandırmalar, bir tamsayı sabiti kümesini kodda kullanılabilecek adlarla ilişkilendirir. Ayrıca varsayılan olarak, numaralandırmadaki her öğenin temel türü int'dir.

Örneğin, Enum türünde bir değişken bildirilebilir ve enum sabitlerinden birine ayarlanabilir. Aşağıdaki kod, sırasıyla 0, 1, 2, 3, 4 ve 5 değerleriyle oturmaOdasi, calismaOdasi , bebekOdasi, giyinmeOdasi ve misafirOdasi adlarıyla ilişkili altı farklı odanın sabitini tanımlamak için kullanılan bir enum türü odalar oluşturur:

```
enum Odalar
{
    oturmaOdasi,
    calismaOdasi,
    bebekOdasi,
    giyinmeOdasi,
    misafirOdasi
}
```

Bir numaralandırma byte, sbyte, short, ushort, int, uint, long, ulong, türlerinden türetilebilir.

Numaralandırma Temelleri

Bir numaralandırma türü, bir değişkene atanabilecek bir dizi adlandırılmış integral sabiti tanımlamanın etkili bir yolunu sağlar. Esasen, bir numaralandırma, yalnızca bir dizi sonlu seçeneğe izin veren bir türdür ve her seçenek bir sayıya karşılık gelir. Varsayılan olarak, bu sayılar, sıfırdan başlayarak değerlerin bildirildiği sırayla artar. Haftanın günleri için bir numaralandırma şeklinde tanımlanabilir.

```
public enum HaftaninGunleri
{
        Pazartesi,
        Sali,
        Carsamba,
        Persembe,
        Cuma,
        Cumartesi,
        Pazar
}
```

Bu numaralandırma şu şekilde kullanılabilir:

```
// Değişkenleri belirli günlere karşılık gelen değerlerle tanımlayın
OzelGunler dogdugumGun = Day.Pazartesi;
OzelGunler sansliGun = Day.Cuma;
// dogdugumGun'e karşılık gelen int'yi alın örneğin,  Cuma 4 numara
int dogdugumGunIndeks = (int)dogdugumGun;
// 6 sayısını temsil eden gün haftanın son gününü işaret eder
OzelGunler sonGun = (HaftaninGunleri)6;
```

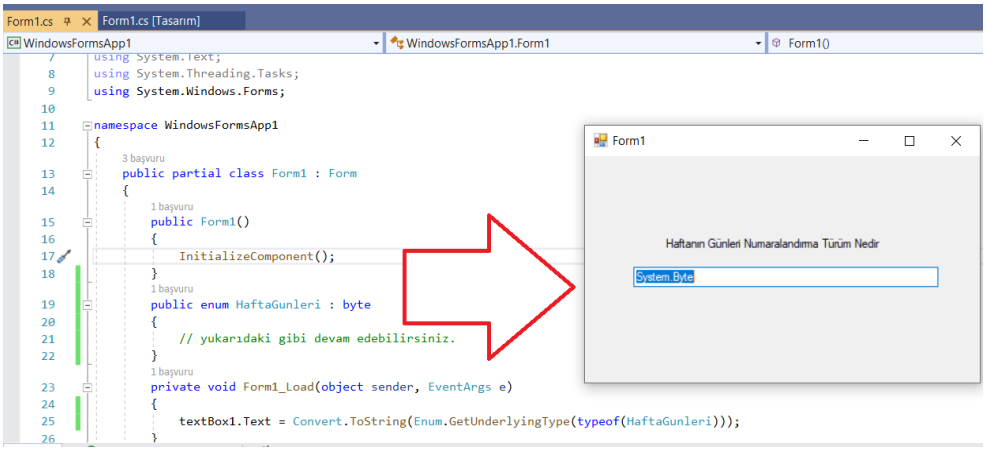
Varsayılan olarak, numaralandırmadaki her öğenin temel türü int'dir. Ancak byte, sbyte, short, ushort, uint, long ve ulong da kullanılabilir.

Int dışında bir tür kullanıyorsanız, numaralandırma adından sonra iki nokta üst üste kullanarak numaralandırma türünü belirtmelisiniz. Ayrıca Enum'un temel türünü aşağıdaki gibi alabilirsiniz. Çalıştırılan kodun ekran görüntüsü Resim 14.9 üzerinde ifade edilmektedir.

Her bir numaralandırma değerine atanan sayısal değer, bitsel operatörler kullanılarak numaralandırmalar işlenirken yardımcı olur.

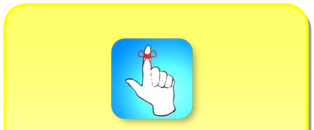
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        public enum HaftaGunleri : byte
        {
            // yukarıdaki gibi devam edebilirsiniz.
        }
        private void Form1_Load(object sender, EventArgs e)
        {
           textBox1.Text = Convert.ToString(Enum.GetUnderlyingType(typeof(HaftaGunleri)));
        }
    }
}
```

Çıktı Görünümü



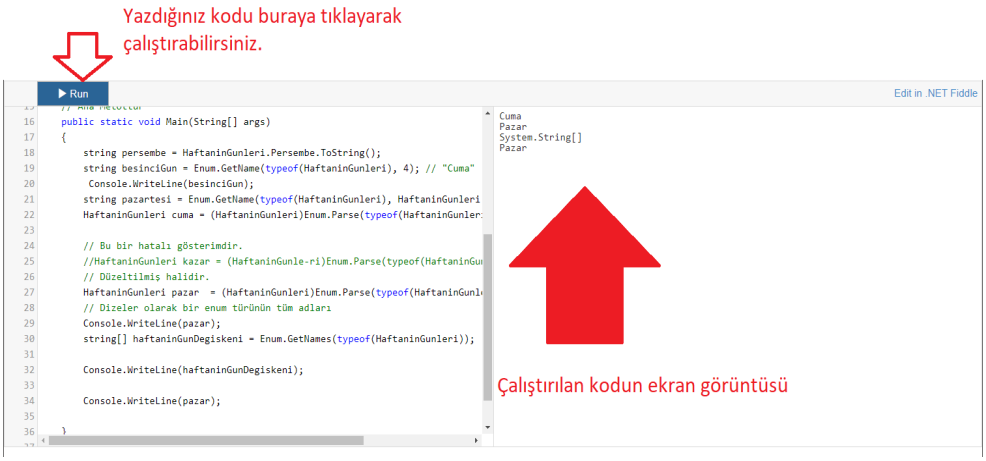
Resim 14.9. Numaralandırma Kullanımı ve Değişken Tipi Algılama ve Çalışan Kodun Ekran Görünümü

Numaralandırma, sabitler ve değişmezler üzerine kodlarınızı web üzerinden çalıştırabileceğiniz uygulamalar da söz konusudur. Bu bölümde Csharp-tutorials üzerinden kodlarınızı çalıştırmayı ve kodlarınızı çalıştırdıktan sonra elde edeceğiniz ekran görüntüsünü Resim 14.10 üzerinde göstermekteyiz. Bu internet üzerinde kodlarınızı çalıştırırken class isminizin önüne public tanımlaması ile kullanınız. Aksi halde derlerken hata ile karşılaşabilirsiniz.



Web üzerinden bu adresten kodlarınızı çalıştırabilirsiniz: *https://csharp-tutorials.com/tr-TR/CodeEditor/code/fuaRQG*

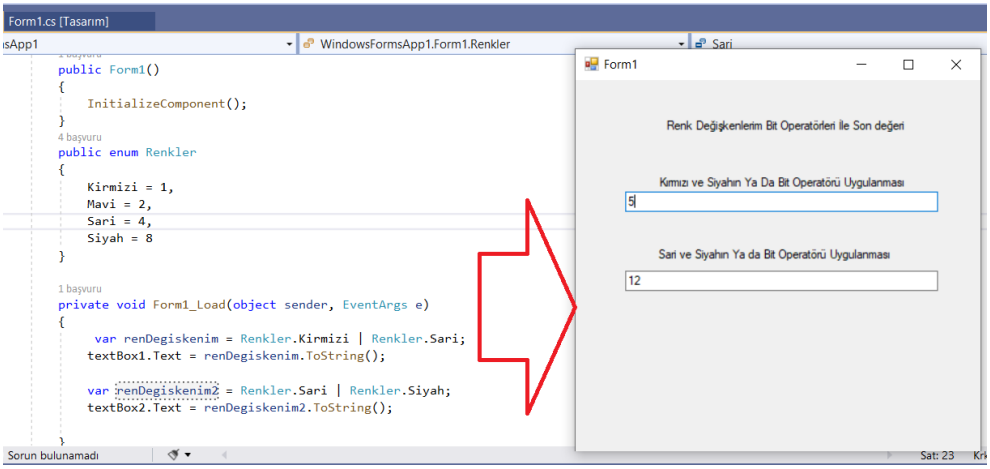
```
using System;
public class Sabitler
{
        public enum HaftaninGunleri
        {
                Pazartesi,
                Sali,
                Carsamba,
                Persembe,
                Cuma,
                Cumartesi,
                Pazar
        }
        // Ana Metottur
        public static void Main(String[] args)
        {
                string persembe = HaftaninGunleri.Persembe.ToString();
                string besinciGun = Enum.GetName(typeof(HaftaninGunleri), 4); // "Cuma"
                Console.WriteLine(besinciGun);
                string pazartesi = Enum.GetName(typeof(HaftaninGunleri), HaftaninGun-
leri.Pazartesi);
                HaftaninGunleri cuma = (HaftaninGunleri)Enum.Parse(typeof(HaftaninGun-
leri), " Cuma");

                // Bu bir hatalı gösterimdir.
        //HaftaninGunleri kazar = (HaftaninGunleri)Enum.Parse(typeof(HaftaniGunleri),
"Paztar");
                // Düzeltilmiş halidir.
                HaftaninGunleri pazar = (HaftaninGunleri)Enum.Parse(typeof(HaftaninGun-
leri), "Pazar");
                Console.WriteLine(pazar);
                // Dizeler olarak bir enum türünün tüm adları
                string[] haftaninGunDegiskeni = Enum.GetNames(typeof(HaftaninGunleri));
                Console.WriteLine(haftaninGunDegiskeni);

        }
}
```



Resim 14.10. Örnek Numaralandırma Kodunun Web Üzerinden Çalıştırılması ve Kod Çalışması Sonrası Elde Ettiğimiz Ekran Görüntüsü

FlagsAttribute, numaralandırılabilir tek bir değer yerine bir bayrak koleksiyonunu temsil ettiğinde kullanılmalıdır. Her bir numaralandırma değerine atanan sayısal değer, bitsel operatörler kullanılarak numaralandırmalar işlenirken yardımcı olur. Bitsel operatör kullanımı ile ilgili gösterim aşağıda kod bloğu üzerinde görebilirsiniz. Çalıştırılan kodun derleyici üzerinden görünümü de Resim 14.11 üzerinde gösterilmektedir.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        public enum Renkler
        {
            Kirmizi = 1,
            Mavi = 2,
            Sari = 4,
            Siyah = 8
        }
        private void Form1_Load(object sender, EventArgs e)
        {
             var renDegiskenim = Renkler.Kirmizi | Renkler.Sari;
            textBox1.Text = renDegiskenim.ToString();
            var renDegiskenim2 = Renkler.Sari | Renkler.Siyah;
            textBox2.Text = renDegiskenim2.ToString();

        }
    }
}
```



Resim 14.11. Numaralandırma ve Bitsel Operatör Kullanımı, Kodların Çalışma Anı Ekran Görünümü

Bireysel Etkinlik

- 22 Ocak 2018 günü Pazartesi ise, 19 Mayıs 2019 hangi güne denk gelmektedir. Enum ve Cost tanımalamarı ile bu problemi çözünüz.

Özet

- Bu bölüm içerisinde değişmezlerden (literal), sabitlerden (constants) ve numaralandırma (enumarations) kavramlarından bahsedilmiş ve Visual Sudio 2019 .Net ortamında C# programlama diliyle kulanımı açıklanmıştır.
- Bu bölümü tamamladıktan sonra projelerinizde ihtiyaçlarınıza bağlı değişmezler, sabitler ve numaralandırma mantığı ile programlama yapabileceksiniz. eğişmezlere (Literals), Sabitlere (Constants) ve Numaralandırmalara (Enumerations) Kısa bakış Sabitler, programın yürütülmesi sırasında değiştiremeyeceği sabit değerleri ifade eder. Bu sabit değerlere değişmez değerler de denir.
- *Sabitler*, tamsayı sabiti, kayan sabit, karakter sabiti veya dize değişmezi gibi temel veri türlerinden herhangi biri olabilir. Ayrıca numaralandırma sabitleri de vardır. Program yürütme sırasında değişkenlerin değerleri değişse de, değişmezler ve sabitler değişmeyen veri öğelerini içerir. Aşağıda sırasıyla bu kon başlıkları Visual Sudio 2019 .Net ortamında C# programlama dili ile uygulamalar sırasıyla gösterilmektedir.
- *Değişmezler (Literals),* sabit değerler konu başlığı altında değerlendireceğimiz bir diğer konu değişmezler yani literallerdir. Değişmez, değişkenler tarafından kullanılan bir değerdir. Değerler tamsayı, kayan nokta veya dize vb. olabilir.
- Değişmezler aşağıdaki türlerden olabilir:
- *Tamsayı Değişmezler (Integer Literals):* Tamsayı türünün değişmez değeri, tamsayı değişmezi olarak bilinir. Sekizli, ondalık, ikili veya onaltılık sabit olabilir.
- *Kayan Noktalı Değişmezler (Floating-point Literals):* Bir tamsayı kısmı, bir ondalık nokta, bir kesir kısmı ve bir üs kısmı olan değişmez değer, kayan nokta değişmezi olarak bilinir. Bunlar, ondalık biçimde veya üstel biçimde temsil edilebilir.
- *Karakter Değişmezler (Character Literals):* Karakter veri türleri için değişmez değerleri tek tırnak gösterim, evrensel kod ile gösterim ve özel gösterim olmak üzere üç şekilde ifade edilebilir.
- *Dize Değişmezleri (String Literals*): Çift tırnak (“”) içine alınmış veya @“” ile başlayan değişmez değerler, String değişmezleri olarak bilinir.
- *Boş - Değersiz Değişmezler (Null Literals):* Null türünü belirten değişmez değerdir. Ayrıca, null herhangi bir başvuru türüne sığabilir. Bu nedenle polimorfizm için çok iyi bir örnektir.
- *Mantıksal Değişmezler (Boolean Literals):* Değişmezleri için yalnızca iki değere izin verilir, yani true ve false.
- *Sabitler (Constants),* derleme zamanında bilinen ve programın ömrü boyunca değişmeyen sabit değerlerdir. Bir sınıfın veya yapının içerebileceği üye türlerinden birisidir. Çoğu zaman kodunuzda aynı sabit değeri tekrar tekrar kullanmanız gerekir. Örneğin, bir dizi geometrik hesaplamanın pi değerini kullanması gerektiğinde, kodunuzdaki değişmeyecek bir değer olan pi değerini tekrarlamak yerine, bir sabit kullanarak bu durumu engelleyebiliriz.
- *Numaralandırma (Enumerations),* belli sözcüklerin, belli tamsayıları temsili durumlarında kullanılan bir yapıdır. Değişkenlerin alabileceği değerlerin sabit olduğu durumlarda kullanılır. Bir numaralandırma aşağıdaki türlerden herhangi birinden türetilebilir: byte, sbyte, short, ushort, int, uint, long, ulong. Enum için varsayılan türümüz int’ dir ve enum tanımında tür belirtilerek yukarıda ifade ettiğimiz türlerden bir ile türetilebilir.

DEĞERLENDİRME SORULARI

"Derleme zamanında bilinen ve programın ömrü boyunca değişmeyen sabit değerlerdir. Bir sınıfın veya yapının içerebileceği üye türlerinden birisidir."

1. İlgili tanım aşağıdakilerden hangisine aittir?
   a) Dosya (File)
   b) Sınıf (Class)
   c) Nesne (Object)
   d) Sabitler (Constants)
   e) Değişken (Variable)

2. Aşağıdakilerden hangisi değişmezler (literals) örneği olarak verilemez?
   a) Dize Değişmezler (String Literals)
   b) Mantıksal Değişmezler (Boolean Literals)
   c) Karakter Değişmezler (Character Literals)
   d) Tamsayı Değişmezler (Integer Literals)
   e) Negatif Değişmezler (Negative Literals)

3. Numaralandırma için varsayılan tür aşağıdakilerden hangisidir?
   a) int
   b) sbyte
   c) byte
   d) long
   e) short

4. Aşağıda verilenlerden hangisi numaralandırma (enumaration) kullanımının sağladığı faydayı ifade etmektedir?
   a) Modülerlik
   b) Tekrar Kullanılabilirlik
   c) Bilgi Saklama
   d) Kolay Hata Ayıklama
   e) Derleme Zamanı Oluşturulur ve Değiştirilemez

5. Bir numaralandırma aşağıdaki türlerden herhangi birinden türetilemez?
   a) byte
   b) short
   c) int
   d) string
   e) long

6. Aşağıdakilerden hangisi Numaralandırma (Enumarations) için yanlış bir ifadedir?
   a) Belli sözcüklerin, belli tamsayıları temsili durumlarında kullanılan bir yapıdır.
   b) Değişkenlerin alabileceği değerlerin sabit olduğu durumlarda kullanılır.
   c) Bir numaralandırma long veya ulong türlerinden herhangi birinden türetilebilir.
   d) Bir numaralandırma varsayılan olarak int türündedir.
   e) Program içerisinde istediğimiz gibi değerlerini değiştirebiliriz.

7. Aşağıdakilerden hangisi nesne yönelimli bir programlama dili değildir?
   a) C#
   b) Java
   c) F#
   d) C++
   e) Assembly

8. C# programlama dili için numaralandırma (enumarations) tanımlamak için kullanılan anahtar sözcüktür?
   a) class
   b) protected
   c) enum
   d) public
   e) main

9. C# programlama dili için aşağıdakilerden hangisi sabit (constants) gösterimine örnektir?
   a) pivate int degisken;
   b) protected long degisken;
   c) public static degisken;
   d) const int degisken=3.14;
   e) static degisken=5;

10. Aşağıdakilerden hangisi sabit tanımlamakta kullanılan anahtar sözcüktür?
    a) new
    b) const
    c) convert
    d) parse
    e) object

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

YARARLANILAN KAYNAKLAR

Alagić, S. (2015). Object-Oriented Technology. London: Springer.

Anggoro, W. (2016). Functional C#. England: Packt Publishing Ltd.

C# Klavuzu (2021). C# Belgeleri. Erişim Tarihi: 27/08/2021, https://docs.microsoft.com/tr-tr/dotnet/csharp/

C# Notes for Professionals (2021). GoalKicker.com, C# Notes for Professionals 105. Erişim Tarihi: 27.08.2021, https://goalkicker.com/CSharpBook

Dathan, B., & Ramnath, S. (2015). Object-Oriented Analysis, Design and Implementation. ABD: Springer.

GeeksForGeeks (2020). C# | Literals. Erişim Tarihi: 27/08/2021, https://www.geeksforgeeks.org/c-sharp-literals/

Gross, C. (2006). Foundations of Object-Oriented Programming Using. NET 2.0 Patterns. USA: Apress.

Gunnerson, E., & Wienholt, N. (2012). A Programmer's Guide to C# 5.0. New York: Apress.

İTÜBİDB (2019). Nesne Yönelimli Programlama (Object-Oriented Programming). Seyir Defteri. İTÜ Bilgi İşlem Daire Başkanlığı. Erişim Tarihi: 27/08/2021, https://bidb.itu.edu.tr/seyir-defteri/blog/2019/02/05/object-oriented-programming

Karaçay T. (2021). Erişim Belirteçleri, Bölüm 11. Erişim Tarihi: 27/08/2021, http://mail.baskent.edu.tr/~tkaracay/etudio/ders/prg/java/ch11/accessmods.htm

Küçükkoç, İ. (2020). Algoritma ve Programlamaya Giriş, Ders Notları. Balıkesir Üniversitesi, Mühendislik Fakültesi Endüstri Mühendisliği Bölümü.

MEB Değer Türleri (2012). T.C. Millî Eğitim Bakanlığı Bilişim Teknolojileri Nesne Tabanlı Programlamada Değer Türleri (482BK0157), Ankara, 2012.

Nakov, S., & Kolev, V. (2013). Fundamentals of Computer Programming with C#: The Bulgarian C# Book. Sofia: Faber Publishing.

Numaralama Türleri (2021). Numaralama türleri (C# başvurusu). Erişim Tarihi: 27.08.2021, https://docs.microsoft.com/tr-tr/dotnet/csharp/language-reference/builtin-types/enum

Tutorialspoint (2021). C# - Constants and Literals. Erişim Tarihi: 27/08/2021, https://www.tutorialspoint.com/csharp/csharp_constants.htm

W3School (2021). Constants in C#, Erişim Tarihi: 27/08/2021, https://www.w3schools.in/csharp-tutorial/constants/