C#

Programlama Dili

Kadir ÇOLAK

2017

BU KİTAP AKADEMİK C# BİLGİSİ VERMEYİ AMAÇLAMAMIŞ TAM AKSİNE PROGRAMCININ İHTİYAÇ DUYDUĞU C# BİLGİSİNE GÖRE DERLENMİŞ VE YAZILMIŞTIR.

İÇİNDEKİLER

ÖNSÖZ

Sevgili okur,

Bilişim dünyası her geçen gün büyümekte ve inanılmaz bir şekilde ivme kazanmaktadır.

Bu hızlı ve inanılmaz değişim içerisinde şüphesiz ki ülkemizin geleceğinde bilişim kritik bir rol üstlenmektedir. Gelecekte birçok ülke için yazılım veyahut bilişim başlığı artık günümüzden de değerli olacağı gibi ülkeler arası ayrımda kilit taşı üstlenecek hale gelecektir.

İster kodlama ve yazılımın temel eğitimini lisede almış olun, ister üniversitede almış olun isterseniz de hiçbir eğitim kurumunda bu bilgileri almamış olun bu hiç fark etmez. Önemli olan bunu yapabilecek sabrı ve emeği gösterebilmenizdir. Geçen zamanın uzun olması sizi hiçbir zaman için caydırmasın veya pes etme noktasına götürmesin.

Programlamak göründüğü gibi kolay değildir. Bunu kitabın ilerleyen sayfalarındaki örneklerde daha net bir şekilde göreceksiniz. Ama bu zorluk sizde bir korku veya ürperti asla uyandırmasın. Düzenli ve gayretli bir çalışma ile sizde birçok farklı tür ve konuda tek başınıza programlar yazabilecek seviyeye gelebilirsiniz.

Programlama ve yazılım bilgisi için az çok matematik bilgisi olduğuna inanlardan biriyim. Burada istenen matematik bilgisi akademik derslerde gösterilen türev, integral, logaritma, limit gibi derin ve karmaşık yapıda olanlar değildir. Düşünme matematiği ve mantıksal matematik bir programcının olmazsa olmazlarındandır.

Bu kitabın içindeki teori ve pratik bilgiler size akademik kapıda yol açacak ve önünüzü genişletecek seviyede değildir. Bir bilgisayar programcısı ve yazılımcısının ihtiyaç duyduğu seviyede kritik ve değerli bilgilere yer verilmiştir.

SAYGILARIMLA!

Algoritma nedir?

Algoritma, herhangi bir soruna karşı izlenecek adımların ve yöntemlerin belirli bir sıralamada gitmesi durumudur. Hangi olayın daha önce geldiği veya gelmediği kritik önem taşımaktadır çünkü bazı bilgi ve olayların daha önceden yapılması bütün olaya etki edebildiği gibi istenilmeyen hatalara da yol açmaktadır.

Örnek olarak bir bilgisayar satın alma algoritmasını geliştirelim;

- Bilgisayar alabilmek için yeterli parayı biriktir.
- Bilgisayarı alacağın mağazaya git.
- Bilgisayarı satın al.

Yukarıdaki örnek gayet saf bir algoritma örneğidir. Bu algoritmada ikinci ve birinci adımı kendi içlerinde değiştirdiğimizde şöyle bir hata çıkmaktadır: *"Para olmadan bilgisayar alamazsın"*

Bu algoritmada farklı bir hata daha yaparak örnek sayımızı artıralım. İkinci ve üçüncü adımı kendi aralarında değiştirir isek mantıken şöyle bir hatayla karşılaşırız: *"Bilgisayarı satın alacağın yere gitmeden bilgisayarı satın alamazsın"*

Algoritma en temel haliyle böyledir. Sorun ve olaya karşı çözüm amaçlı izlenecek yolların sıralı ve düzgün bir şekilde derlenmesidir.

Değişken nedir?

Kullanıcıdan farklı çeşitlerde alınabilecek geniş kapsamlı verilerdir. Mesela çoğu uygulama ve sisteme kayıt olurken ad-soyad-doğum tarihi gibi bilgileri siz kendiniz girersiniz. Burada girdiğiniz bu her bilgi aslında bir değişken özelliği taşımaktadır.

string	Metinsel ifade
int	Tam sayı
double	Virgüllü Sayı
float	Virgüllü Sayı
char	Tek Karakter
bool	True / False

Yukarıdaki tabloda sol tarafta yazanlar programlama dilinde birebir yazılacak değişken kalıplarıdır. Her değişken aynı değişken türü ve kategorisine uyum sağlayamaz. Değişkenlerin içerisinden size uygun olanını o an için kullanmak zorundasınız.

Kullanıcının girdiği değişken değerlerinde Türkçe karakterlerin olmasında hiçbir sıkıntı yok.

Değişken isimleri oluştururken uyulması gereken bazı kurallar vardır;

Türkçe karakter içermemelidir.
Sayısal veri ile başlamamalıdır.
Nokta, virgül gibi özel karakterler içermemelidir.
İstenilen değere göre uygun ve mantıklı bir ad içermelidir.
Değişken isminde boşluk bırakılmamalıdır.

int sayi=5;	int sayi=”5”;
string tcno=”11111111111”;	string tcno=11111111111;
char karakter=’a’;	char karakter=”a”;
double pi=3.14;	double pi=3,14;
double pi_sayisi=3.14;	double pi sayisi=”3.14”;
bool=false;	bool=”false”;
int sayi=5;	int sayi=”5”;
string tcno;	string 1tcno;
int sayi;	int .sayi;
float virgullusayi;	float 1virgüllüsayi;
char ozel;	char özel;

Yukarıdaki tablonun sol tarafı doğru değişken örneklerini, sağ tarafı ise yanlış değişken örneklerini göstermektedir.

1. String Değişkeni

string kullanıcıdan alınan harflerle veya rakamlarla yazılmış bir bütünlemedir. Bu değişken ile matematiksel bir işlem yapılamaz. string değişken türüne örnek birkaç ifadeye aşağıda yer verilmiştir;

Ahmet
Yılmaz
34 FZ 458
İstanbul
Genel Müdür

Gördüğünüz gibi bazen sayısal verilerin de içine dahil olduğu ama genel itibariyle bir cümle, kelime veya karakter topluluğundan oluşan verilerdir.

T.C. Kimlik kartı, 11 haneli telefon numarası, öğrenci numarası ve pin şifreleri bir string değer özelliği taşımaktadır. Çünkü girilen değerler hep sayısal olmasına rağmen matematiksel işlem yapılmadığından dolayı kelime değeri görmektedir.

string değerleri atanırken " ve " içinde yazılmaktadır. Eğer bu veri char değişkeni ise ' ve ' karakterleri arasında yazılacaktır. String bir bütünü kaplarken char ise sadece tek karaktere odaklanmaktadır.

```
string adsoyad ="Ahmet YILMAZ";
```

2. İnt Değişkeni

int değişkeni kullanıcıdan alınan veya arka planda programcı tarafından oluşan ve 1'e tam bölünen bütün sayılardır. Virgüllü yani irrasyonel sayı olmaması durumunda bütün sayılar int değişken türüne dahil edilir.

12
45356
-64
1000000
0

int değişkeninde aranan en önemli özellik bu değerin dört temel işleme tabi tutulacak olmasıdır. Eğer bu değişken işlemlerde kullanılmayacak yani matematiksel kalıba sokulmayacaksa burada büyük bir yanılgıya düşeriz.

T.C. kimlik numarası, telefon numarası, öğrenci numarası, posta kodu, 4 haneli pin şifresi gibi değerlerin hepsi sırf sayılardan oluşmasına rağmen sadece cümle veya kelime gibi değer gördüğü için onlar string değişkeninde yazılırken bizim işleme tabi tutacağımız değişkenler ise int değişkeninde yazılır.

```
int sayi =1548;
```

3. Double Değişkeni

int örneğininin kalıplarıyla çok uyuşmaktadır. Aradaki temel tek fark ise int verileri 1'e tam bölünebilen sayılar olacakken double değerleri küsuratlı yani virgüllü çıkabilme özelliğine sahiptir. Tam sayı ve virgül sonrası değer nokta ile ayrılmaktadır.

Bir toplama işleminde birinci sayıyla ikinci sayıyı toplayacaksanız ikisi de ya double değişkeni olmalı ya da int değişkeni olmalıdır. int ve double değişkenleri birbirleriyle matematiksel işleme tabi tutulabilir. Değişkenler arası matematiksel işlem yapılamaz.

3.14
2.7
-1.5646464
6.3333333
0.1

double pi =3.14;

4. Float Değişkeni

double değişkeni ile birebir aynı özellikleri taşımaktadır. Aradaki tek fark değişken tanımlarken double yerine float yazılmasıdır. Geriye kalan bütün özellikleri aynıdır (virgüllü sayıları bünyesinde barındırması, sayıları nokta ile ayırması vb.)

float virgullusayi =3.14;

5. Char Değişkeni

İlk örnekte anlatılan string değişkenin en kısa hali olarak tanımlayabiliriz. String içinde birkaç kelimeyi veya birkaç harfi barındırabiliyor iken char sadece tek karakteri bünyesine alır.

A
.
1
a
*

Yukarıda örnek char değerleri verirken A ile a arasında fark olduğunu şüphesiz anlayabiliyorsunuz. Bu C# programlama dilinin küçük/büyük harf hassasiyetine dayandığını bizlere ipucu olarak vermektedir.

char değişkenleri ' ve ' karakterleri içersinde yazılır ve tek bir harf, tek bir karakter olma şartını mutlaka arar.

char nokta ='a';

6. Bool Değişkeni

bool değişkeninde maksimum 2 ihtimal vardır. Ya doğrudur ya da yanlıştır. Bazı şeylerin yapılıp yapılmadığı durumunu bool değişkeni vasıtasıyla öğrenebilir ve programın gidişatını ona göre yönlendirebilirsiniz.

true
false

bool durum =true;

Değişkenleri nasıl kullanabiliriz?

Bazı durumlarda değişkenlere 4 temel işlem ve artırma-azaltma işlemleri yapmanız gerekecek. Bunları örneklerle açıklayalım;

```
static void Main(string[] args)
{
    int sayi = 5;
    sayi++;
}
```

```
static void Main(string[] args)
{
    int sayi = 5;
    sayi = sayi + 1;
}
```

```
static void Main(string[] args)
{
    int sayi = 5;
    sayi += 1;
}
```

Yukarıda örnek olarak gösterilen üç kod aynı sonuca ulaşmaktadır. Başta 5 değerini verdiğimiz sayi değişkenini bir artırarak 6 sayısına ulaşmış oluyoruz.

```
static void Main(string[] args)
{
    int sayi = 5;
    sayi--;
}
```

```
static void Main(string[] args)
{
    int sayi = 5;
    sayi = sayi - 1;
}
```

```
static void Main(string[] args)
{
    int sayi = 5;
    sayi -= 1;
}
```

Yukarıda örnek olarak gösterilen üç kod aynı sonuca ulaşmaktadır. Başta 5 değerini verdiğimiz sayi değişkenini bir azaltarak 4 sayısına ulaşmış oluyoruz.

```
static void Main(string[] args)
{
    string ad = "Ahmet";
    string soyad = "YILMAZ";
    string adsoyad = ad + soyad;
}
```

Yukarıda örnek olarak verilen kodda string değişken olan adsoyad değişkeni Ahmet ve YILMAZ kelimelerini birleştirerek tek değişkene atamaktadır. Burada ki yaşanan tek teknik arıza ise boşluk bırakılmaması olacaktır.

AhmetYILMAZ

```
static void Main(string[] args)
{
    string ad = "Ahmet";
    string soyad = "YILMAZ";
    string adsoyad = ad +""+ soyad;
}
```

Bu kodda ise ad ve soyad değişkeni gene adsoyad değişkeninde birleştirilmektedir fakat aralarına bir boşluk karakteri dahil ederek doğru bir yazım elde edilmiştir.

Ahmet YILMAZ

```
static void Main(string[] args)
{
    string ad = "Ahmet";
    string kucuk= ad.ToLower();
}
```

Bu kod yapısında ise Ahmet olarak belirlenmiş ad değişkenin bütün harfleri küçülterek *ahmet* sonucuna ulaşılmaktadır.

ahmet

```
static void Main(string[] args)
{
    string ad = "Ahmet";
    string kucuk = ad.ToUpper();
}
```

Bu kod yapısında ise Ahmet olarak belirlenmiş ad değişkenin bütün harfleri büyülterek *AHMET* sonucuna ulaşılmaktadır.

AHMET

```
static void Main(string[] args)
{
    int sayi1 = 5;
    int sayi2 = 8;
    int top = sayi1 + sayi2;
}
```

Buradaki kod yapısında sayi1 ve sayi2 değişkeni toplanmaktadır. Aynı + işaretini string değişkenlerde kullandığınızda birleştirme görevi görürken int-double-float gibi sayısal değişkenlerde toplama değeri görmektedir.

13

```
static void Main(string[] args)
{
    string sayi1 = "5";
    string sayi2 = "8";
    string top = sayi1 + sayi2;
}
```

string değişkende değerleri gene tırnak içine aldık ve + ile birleştirme yapınca 5+8=13 sonucuna değil de karakterleri yan yana getirme mantığıyla 58'i buldu.

58

```
static void Main(string[] args)
{
    int sayi1 = 5;
    int sayi2 = 8;
    int fark = sayi1 - sayi2;
}
```

Burada ise iki int değişkenini – işaretiyle çıkarma mantığında sonuçlar. + işlemiyle yapılan toplama işlemi – işaretiyle yapıldığı müddet çıkarma işlemi değeri görecektir.

-3

```
static void Main(string[] args)
{
    int sayi1 = 5;
    int sayi2 = 8;
    int carp = sayi1 * sayi2;
}
```

+ ve – işaretleriyle yapılan toplama ve çıkarma işlemi * olduğu zaman çarpma işlemi değeri görecektir.

40

```
static void Main(string[] args)
{
    int sayi1 = 5;
    int sayi2 = 8;
    int bolum = sayi1 / sayi2;;
}
```

Buradaki / işaretiyle iki sayısal değerin bölme işlemi gerçekleştirilmektedir. Buradaki sayi1/sayi2 bölme işleminin sonucu ekranda 0 olarak çıkacaktır. Mantıken 0 çıkmamakta aksine 0.625 çıkmaktadır. Buradaki 0.625 sonucunun virgül sonrasını almadığından dolayı tam sayı bazında geri bildirim yapmaktadır.

0

```
static void Main(string[] args)
{
    double sayi1 = 5;
    double sayi2 = 8;
    double bolum = sayi1 / sayi2;;
}
```

Burada int yerine double değişkeni kullanılmıştır. İşlemde küsuratlar önem kazandığı için 5/8 işlemini 0.625 olarak en doğru şekliyle bulacaktır.

0.625

```
static void Main(string[] args)
{
    const double pi = 3.14;
}
```

Burada değişkenin başına *const* yazılarak o değişkenin hiçbir şekilde değişime uğrayamayacağı vurgulanmaktadır. 3.14 olan double pi değişkenine sayı ekleyemez ve sayı çıkaramazsınız. const özelliği ile bu değer dokunulmaz hale getirilmiştir.

```
static void Main(string[] args)
{
    int sayi = 5;
    string kelime = "araba";
    string birlesim = sayi.ToString() + " " + kelime;
}
```

Burada ise 5 olarak atanmış int sayi değişkeni ve “araba” olarak atanmış string kelime değişkeni birlikte başka bir string değişkeninde yan yana getirilmiştir. sayi değişkeni .ToString() özelliği ile string değere dönüştürülmüş ve yaşanacak hata kaldırılmıştır.

5 araba

```
static void Main(string[] args)
{
    char degisken1='A';
    string cumle = degisken1.ToString() + degisken1.ToString();
}
```

Bu kod yapısında char ile ‘A’ değişkeni atanmış ve string cumle değişkeninde char ifadesinin yer alabilmesi için .ToString() ile bir dönüşüm gerçekleştirilmiştir.

AA

```
static void Main(string[] args)
{
    int sayi1 = 5;
    int sayi2 = 8;
    int sayi3 = 12;
    int sonuc = ((sayi1 * sayi3) * sayi2) * (sayi2);
}
```

Bu kodda yapılan matematiksel işlemlerde hangisinin önce başlanacağıyla ilgili örnekler safhasına girmektedir. 5-8-12 değerlerinde üç farklı değişkenle bir takım işlemler gerçekleştirilecektir;

= ((5*12)*8)*(8)

= (60*8)*(8)

= 480*8

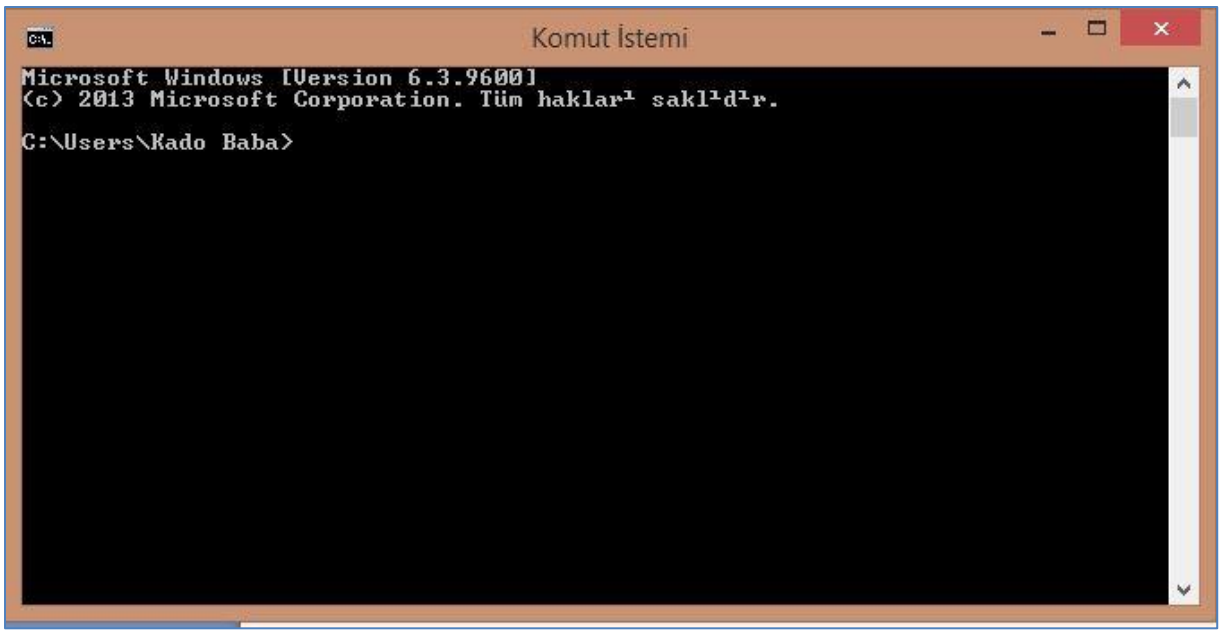
= ***3840***

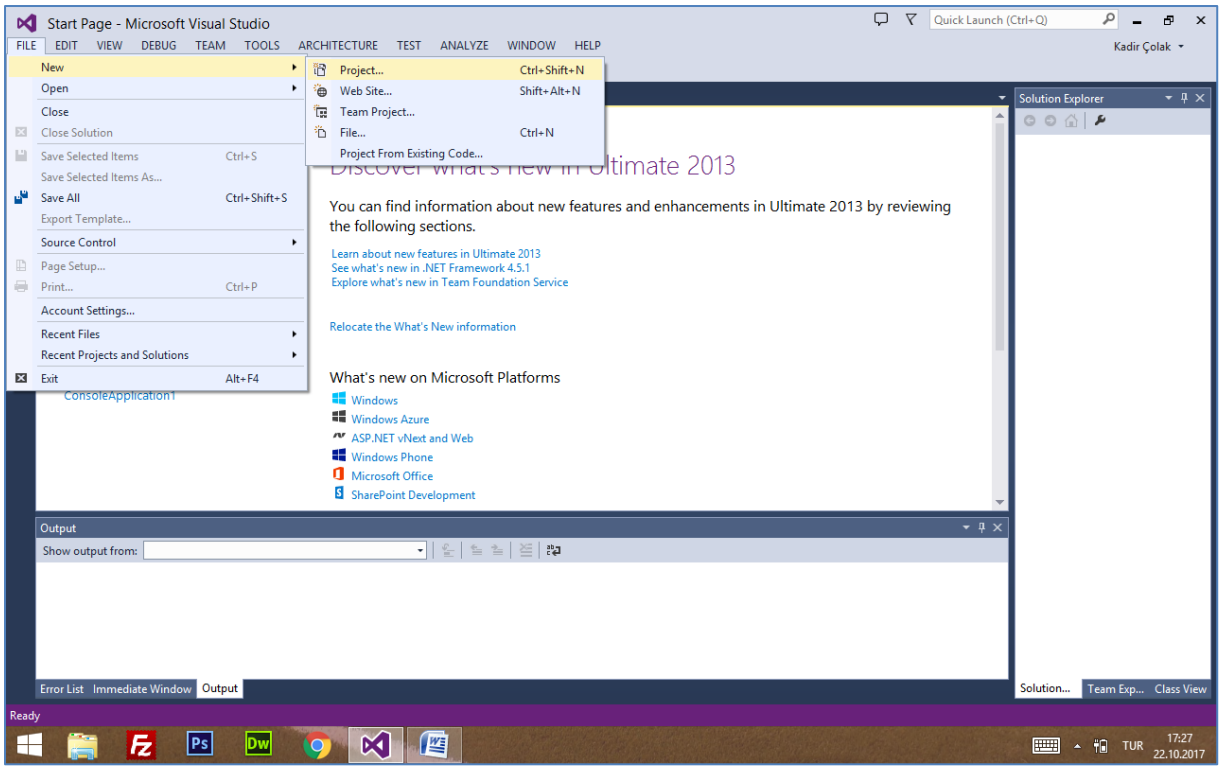
Console Application nedir?

Console Application, Windows işletim sistemlerinde herkesin rahatlıkla erişebileceği cmd programının tasarımına sahip pek fazla fonksiyon ve görselliğe sahip olmayan, temel işleri halletme güdüsü taşıyan program kalıbıdır.

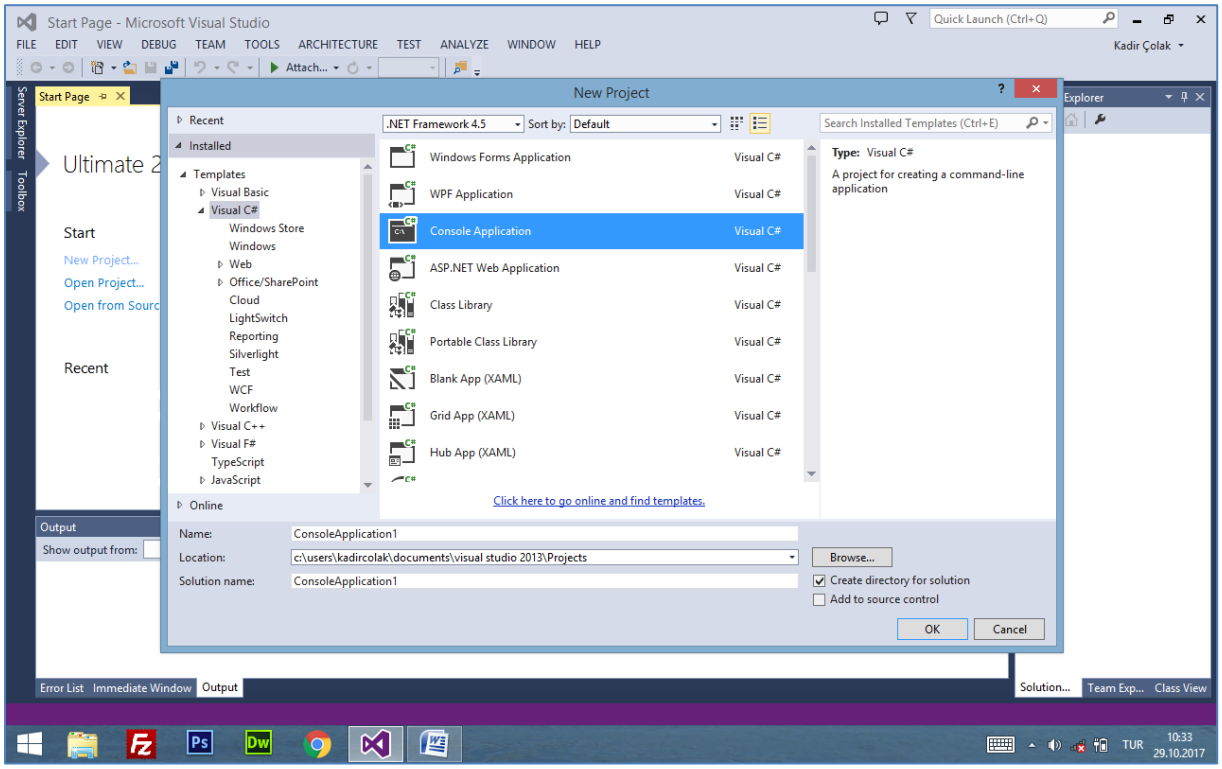
Console Application programlama, yazılım ve algoritma temellerinin kavrandığı antrenman yeri olarak görülebilir. Bu siyah ekranda çok şahane bir program kodlasanız dahi sırf tasarım ve görünümü itibarıyla programınız hak ettiği değeri ve ilgiyi göremeyecektir.

Komut istemi Console Application türünde en çok bilinen programlar arasında bilinmektedir;

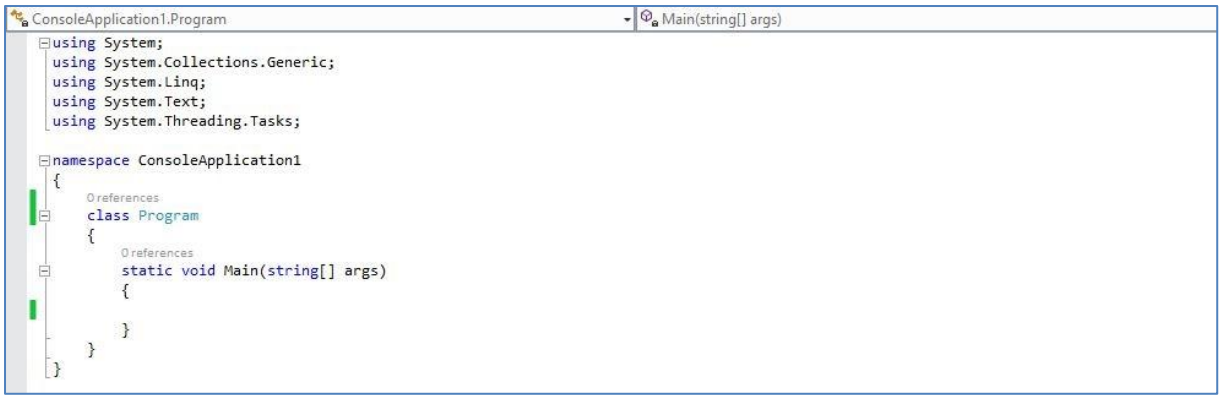




Yukarıdaki resimde göründüğü gibi adımlar izlendiğinde aşağıdaki menü karşınıza çıkacaktır;



İkinci adımda ise Console Application menüsü seçerek ilgili kodlama ve yazılım ekranına giriş yapacaksınız. Karşınıza görsel tasarımına karışabileceğiniz bir yönetim merkezi veyahut benzeri bir şey çıkmayacaktır.



Yukarıdaki görselde görünen bir yazılı şema çıkacak karşınıza. Console Application kodları burada ki static void Main(string[] args) parantezlerinin içine yazılacaktır.

Console Application bünyesinde yazılan kodların ve algoritmanın gerçek çıktısını görmek için F5 tuşuna basmalı veya “Start” tuşuna basmalısınız.

Temel Console kodları nedir?

```
static void Main(string[] args)
{
    Console.Write("Merhaba Dünya");
}
```

Yukarıdaki örnek kodu yazdığınızda ekrana Merhaba Dünya yazısı yazacaktır. Ama F5 tuşuyla açtığımız program ekranı saniyeler içinde kendi kendiliğinden kapanacaktır. Bunun için programı ayakta tutmanıza yarayacak Console.ReadKey(); kod satırını da dahil etmelisiniz.

```
static void Main(string[] args)
{
    Console.Write("Merhaba Dünya");
    Console.ReadKey();
}
```

İşte bu kod yapısıyla F5 tuşuna bastığınızda program sadece sizin kapatmanız ile kapanacak ve ayakta duracaktır. Ekrana sadece “Merhaba Dünya” yazısı yazılacaktır.

```
static void Main(string[] args)
{
    Console.Write("Merhaba Dünya \n");
    Console.ReadKey();
}
```

```
static void Main(string[] args)
{
    Console.WriteLine("Merhaba Dünya");
    Console.ReadKey();
}
```

Console.Write ile ekrana yazdırdığınız söz veya cümlenin bitimi sonrasında otomatik aşağıya inmesini yani paragraf satırı atlamasını isteyebilirsiniz. Böyle bir istekte yukarıda bulunan iki kod yapısından birini tercih etmeniz gerekecektir.

```
static void Main(string[] args)
{
    Console.Write("Merhaba Dünya");
    Console.ReadKey();
}
```

```
static void Main(string[] args)
{
    Console.WriteLine("Merhaba Dünya");
    Console.ReadKey();
}
```

Yukarıda örnek olarak verilmiş iki kodun aralarındaki tek fark ekrana yazdırma işlemi sonrasında imlecin nerede durduğudur. WriteLine yazan kodda ekrana yazdırma sonrasında paragraf satırı kadar aşağı inilirken Write ile yazan kısımda ekrana yazdırılanın sağından itibaren aşağı inilmeden devam edilir.

```
static void Main(string[] args)
{
    string ad;
    Console.Write("Adınızı giriniz: ");
    ad = Console.ReadLine();
    Console.ReadKey();
}
```

Bu kodda ise ekrana yazdırılan “Adınızı giriniz: “ mesajının sonrasında kullanıcıdan bir değer girilmesi beklenmektedir. Kullanıcının adınızı giriniz sorusuna yanıt olarak yazdığı söz dizisi ad değişkeni olarak atanacaktır.

```
static void Main(string[] args)
{
    string ad;
    Console.Write("Adınızı giriniz: ");
    ad = Console.ReadLine();
    Console.WriteLine("Merhaba "+ad);
    Console.ReadKey();
}
```

Bu örnek kodda kullanıcının adınızı giriniz sorusuna verdiği yanıt sonrasında girdiği ismiyle ona Merhaba hitabında bulunması işlemi yapılacaktır. Kullanıcı adınızı giriniz sorusuna eğer cevap olarak Hasan yazdıysa son aşamada ***Merhaba Hasan*** yazacaktır.

```
static void Main(string[] args)
{
    string ad,soyad;
    Console.Write("Adınızı giriniz: ");
    ad = Console.ReadLine();
    Console.Write("Soyadınızı giriniz: ");
    soyad = Console.ReadLine();
    Console.WriteLine("Adınız: " + ad);
    Console.WriteLine("Soyadınız: "+soyad);
    Console.ReadKey();
}
```

Bu örnekte ise kullanıcı iki farklı soruya (ad-soyad) kendi iradesiyle cevap vermiş ve verdiği cevapların sonrasında bir özet maiyetinde programcı tekrar aynı bilgileri ona gösterme işlemini yapmıştır. Kullanıcı yazdığı bilgilerin aynısı görecektir.

```
static void Main(string[] args)
{
    int sayi1, sayi2,toplam;
    Console.Write("Birinci sayıyı giriniz: ");
    sayi1 = Convert.ToInt32(Console.ReadLine());
    Console.Write("İkinci sayıyı giriniz: ");
    sayi2 = Convert.ToInt32(Console.ReadLine());
    toplam = sayi1 + sayi2;
    Console.WriteLine(sayi1 + "+" + sayi2 + "=" + toplam + " ");
    Console.ReadKey();
}
```

Burada kullanıcıdan iki sayı girmesi istenmiştir. İki sayı girmesi sonrasında girdiği sayıların toplamını matematiksel dille yazarak sonucu ekrana yazdırma yapılmıştır. string değer girişinden farklı olarak int değişken türünde girilen değer convert işlemi ile dönüşüm geçirmiştir.

```
static void Main(string[] args)
{
    Console.Title="Deneme Çalışması";
    Console.ReadKey();
}
```

Bu kodda ise Console ekranın en üstünde sol uzunluğu %100 olan çubukta ne yazılacağına karar verilmektedir. Bir nevi console ekranın dış görünüşündeki başlığı değiştirilebilmektedir.

```
static void Main(string[] args)
{
    bool durum = false;
    Console.WriteLine(durum);
    Console.ReadKey();
}
```

Bu kodda durum adı verilmiş bool değişkenine false değeri atanmıştır. Console.WriteLine ile ekrana yazdırıldığında false yerine ***False*** olarak yazacaktır.

```
static void Main(string[] args)
{
    double sayi1, sayi2, toplam, fark, carp, bol;
    Console.Write("Birinci sayıyı giriniz: ");
    sayi1 = Convert.ToDouble(Console.ReadLine());
    Console.Write("İkinci sayıyı giriniz: ");
    sayi2 = Convert.ToDouble(Console.ReadLine());
    toplam = sayi1 + sayi2;
    fark = sayi1 - sayi2;
    carp = sayi1 * sayi2;
    bol = sayi1 / sayi2;
    Console.WriteLine("TOPLAMA: "+sayi1 + "+" + sayi2 + "=" + toplam + " ");
    Console.WriteLine("ÇIKARMA: " + sayi1 + "-" + sayi2 + "=" + fark + " ");
    Console.WriteLine("ÇARPMA: " + sayi1 + "*" + sayi2 + "=" + carp + " ");
    Console.WriteLine("BÖLME: " + sayi1 + "/" + sayi2 + "=" + bol + " ");
    Console.ReadKey();
}
```

Burada int yerine double olarak atanan değişkenlerden dolayı convert işlemi ToDouble mantığıyla olmuştur. Bir önceki örnekteki gibi tek işlem yerine aynı 2 sayıyla 4 farklı işlemin sonucu birden alt alta yazılarak aktarılmıştır.

Burada sonuçlar küsuratlı olsa bile dahi küsurat ve virgül sonrası değerler dikkate alınarak en doğru şekilde sonuç verilecektir.

```
static void Main(string[] args)
{
    int sayi = 5;
    Console.Write(sayi + "");
    Console.ReadKey();
}
```

```
static void Main(string[] args)
{
    int sayi = 5;
    Console.Write(sayi.ToString());
    Console.ReadKey();
}
```

Burada int tipinde olan sayi değişkenini hiçbir işleme tabi tutmadan direkt ekrana yazdırmak istediğimizde düz şekilde yazamayız. int türünde olduğu için bunu farklı şekillerle string moduna dönüştürmemiz gerekiyor.

Eğer bu değişkenler string olsaydı direkt parantez içinde hiçbir eklemesiz yazdırabilirdik ama int-double-float gibi sayısal değişkenleri mutlak surette string görünümlü yapmalısınız.

```
static void Main(string[] args)
{
    Console.ForegroundColor = ConsoleColor.Blue;
    Console.ReadKey();
}
```

Bu kodda ise console ekranında yazan yazıların rengi *blue* (mavi) olarak değiştirilmiştir.

```
static void Main(string[] args)
{
    string ad, soyad;
    Console.Write("Adınızı giriniz: ");
    ad = Console.ReadLine();
    Console.Write("Soyadınızı giriniz: ");
    soyad = Console.ReadLine();
    Console.WriteLine("----------------------------------------");
    Console.WriteLine("----------------------------------------");
    Console.WriteLine("Adınız: " + ad.ToUpper());
    Console.Write("Soyadınız:" + soyad.ToUpper());
    Console.ReadKey();
}
```

Burada kullanıcıdan istenen string türünde ad ve soyad değişkenleri kullanıcının girdiği şekilde değilde tamamen büyük harflerle ekrana yazdırılmaktadır. Bunun dışında iki satırlık bir --------- eklentisi yapılarak da bir nevi ekrana yazdırılan kısım ile kullanıcıdan değer alınan kısım arasında ki sınır çizgisi tayin edilmiştir.

İf ve else komutları nelerdir?

Yazılımın kod süreci boyunca bazı durumlarda bazı değişken ve olayların olup olmadığı, yapılıp yapılmadığı sürecini kontrol etmek gerekir. İf ve else kod yapılarıyla olayların durumlarını evet-hayır mantığıyla kıyas edebilirsiniz.

==	Eşitse
!=	Eşit değilse
>	Büyükse
<	Küçükse
>=	Büyük eşitse
<=	Küçük eşitse
&&	Ve
||	Veya
%	Mod (Kalanı bulma)

Yukarıdaki tabloda if parantezlerinde kullanabileceğiniz mantıksal operatörlere yer verilmiştir.

```
static void Main(string[] args)
{
    int sayi = 5;
    if(sayi==5)
    {
        Console.Write("Sayı değişkeni 5'dir.");
    }
    else
    {
        Console.Write("Sayı değişkeni 5 değildir.");
    }
    Console.ReadKey();
}
```

Yukarıda örnek olarak int tipi sayi değişkenin değeri 5'e atanmıştır. Daha sonrasında değerin 5 olup olmadığı == ile kontrol edilmiştir. sayi değişkeni 5 olduğu için if parantezinin içindekileri gerçekleştirilmiştir. Eğer sayi değişkeni 5 olmasaydı da başka bir sayı olsaydı else bloğundaki işlemleri yapacaktı.

```
static void Main(string[] args)
{
    int sayi = 11;
    if (sayi == 1) { }
    else if (sayi == 2) { }
    else if (sayi == 3) { }
    else if (sayi == 4) { }
    else { }
    Console.ReadKey();
}
```

Kodladığınız programın herhangi bir kısmında ihtimal veya olasılık sayı 3 ve 3'den fazla ise *if-else if-else if-……-else* sıralamasıyla kodlamanız gerecektir. Sadece iki ihtimal olması durumunda if ve else kodları yeterli olacaktır.

```
static void Main(string[] args)
{
    int sayi = 8;
    if(sayi==7)
    {
        Console.WriteLine("Sayı değişkeni 7'dir");
    }
    Console.ReadKey();
}
```

Kodlamada yazdığınız if komutlarının illa birinin tutması zorunlu değildir. Örnek olarak yukarıdaki kodlamaya sahip bir programda ekranda hiçbir şey yazmadan boş bir şekilde çıktı verecektir. İf yapısıyla siz olası ihtimalleri süzgeçten geçiriyor ve programı ona göre şekillendiriyorsunuz.

```
static void Main(string[] args)
{
    int sayi = 5;
    if(sayi==5)
    {
        Console.WriteLine("Başarılı"); //1
    }
    else if(sayi==6)
    {
        if(sayi==5)
        {
            Console.WriteLine("Başarılı"); //2
        }
    }
    else
    {
        Console.WriteLine("Başarılı"); //3
    }
    Console.ReadKey();
}
```

Bu örnek kodda 1-2-3 numaralı ihtimallerden 1 numaralı ihtimal gerçekleşmiştir. 2 numaralı ihtimal içerisinde if(sayi==5) olması sizleri yanılgıya düşürebilir fakat o if parantezinin üstünde yer alan if koşulunda sayi değişkeni 6 olmadığı için o ihtimali baştan atlayacaktır.

```
static void Main(string[] args)
{
    string cevap;
    Console.WriteLine("Cinsiyetinizin ilk harfini giriniz:");
    cevap = Console.ReadLine();
    if(cevap=="E"||cevap=="e")
    {
        Console.WriteLine("Erkek");
    }
    else if(cevap =="K"||cevap =="k")
    {
        Console.WriteLine("Kadın");
    }
    else
    {
        Console.WriteLine("Yanlış değer girişi yaptınız");
    }
    Console.ReadKey();
}
```

Yukarıdaki örnekte ekrana yazdırılan yazı ile kullanıcıdan cinsiyetinin ilk harfini girmesi istenmiştir. Kullanıcının Caps Lock tuşunun açık olup olmama ihtimalini bilmediğimiz için || operatörüyle iki farklı sonuca göre uyarlama yaptık. E-e harflerinden biri yazarsa “ERKEK” sonucuyla karşılaşacaktır. K-k harflerinden birini yazarsa “KADIN” sonucuyla karşılaşacaktır. E-e veya K-k harflerinden hiçbirini yazmasa hatalı giriş yaptığına dair bir uyarıyla karşılaşacaktır.

```
static void Main(string[] args)
{
    string cevap;
    int yas;
    Console.WriteLine("Cinsiyetinizin ilk harfini giriniz:");
    cevap = Console.ReadLine();
    cevap = cevap.ToUpper();
    Console.WriteLine("Yaşınızı giriniz:");
    yas = Convert.ToInt32(Console.ReadLine());
    if(cevap=="E")
    {
        if(yas>=20)
        {
            Console.WriteLine("Askere gidebilirsiniz");
        }
        else
        {
            Console.WriteLine("20 yaş ve üzeri olmanız koşuluyla askere gidebilirsiniz");
        }
    }
    else
    {
        Console.WriteLine("Askere asla gidemezsiniz");
    }
    Console.ReadKey();
}
```

Bu örnekte ise kullanıcının cinsiyetinin hem erkek olması hem de yaşının 20 ve üzerinde olması koşulunda askerliğe uygun olduğu mesajı verilecektir. Aksi haldeki bütün ihtimallerde olumsuz bir mesajla karşılaşacaktır.

```
static void Main(string[] args)
{
    int sayi = 8;
    if(sayi==8)
    {
        Console.WriteLine("Ankara");
    }
    else if (sayi == 8)
    {
        Console.WriteLine("İstanbul");
    }
    else
    {
        Console.WriteLine("İzmir");
    }
    Console.ReadKey();
}
```

```
static void Main(string[] args)
{
    int sayi = 8;
    if(sayi==8)
    {
        Console.WriteLine("Ankara");
    }
    if (sayi == 8)
    {
        Console.WriteLine("İstanbul");
    }
    else
    {
        Console.WriteLine("İzmir");
    }
    Console.ReadKey();
}
```

Yukarıda örnek olarak verilmiş iki kod yapısındaki tek fark ikinci ihtimalin birinde if diğerinde else if kullanılmasıdır. Soldaki örnekte programın çıktısında Ankara yazarken ikinci yani sağdaki örnekte ekran çıktısında İstanbul yazmaktadır.

Else if ile sorguladığımız değişken eşitliği diğer ihtimallerin tutması sonrası aranan özel koşulları belirtmektedir. Eğer else if kullanmadan sürekli *if-if-if-….-if* ile koşullama yaparsanız aynı koşulu barındırmasına rağmen en son yazdığınız koşul parametresini tetikleyecektir.

For komutu nedir?

for komutuyla döngü mantığıyla bazı işlemleri ve olayları birden fazla çok kez yapabilirsiniz. En temel for örneği ile başlayalım;

```
static void Main(string[] args)
{
    for (int i = 0; i <= 10; i++ )
    {
        Console.WriteLine(i + "");
    }
        Console.ReadKey();
}
```

Bu örnekte 0 ve 10 dahil olmak üzere 0'dan başlayıp 10'a kadar bütün sayıları teker teker alt alta yazma işlemini yapmaktadır. Bu işlemi for kullanmadan yapsaydınız şüphesiz daha uzun bir kod yazmanız gerecekti.

```
static void Main(string[] args)
{
    for (int i = 0; i < 10; i++ )
    {
        Console.Write(i + "-");
    }
        Console.ReadKey();
}
```

Bu örnekte ise 0-1-2-3-4-5-6-7-8-9- şeklinde yan yana sonuç verecektir. Yukarıda WriteLine olan kod yapısı burada Write olduğu için yan yana yazma işlemi gerçekleştirilmiştir. 10 rakamının çıkarılması ise <= yerine < konmasından kaynaklanmaktadır.

```
static void Main(string[] args)
{
    int toplam = 0;
    for (int i = 0; i < 10; i++ )
    {
        toplam+=i;
    }
    Console.WriteLine(toplam+"");
    Console.ReadKey();
}
```

Burada 0 ve 9 dahil olmak üzere aradaki bütün sayıların toplamı hesaplanmaktadır. 0+1+2+3+4+5+6+7+8+9=45 mantığıyla ekrana 45 değeri yazdırılacaktır.

```
static void Main(string[] args)
{
    for (;;)
    {
        Console.WriteLine("Bitmeyen Yazı");
    }
    Console.ReadKey();
}
```

Bu örnek kodda ise for ile sınırsız bir döngü yapılmıştır. Bu sayede ekranda sonsuza kadar sürekli yenisi eklenen “Bitmeyen Yazı” ibaresi gözükecektir.

```
static void Main(string[] args)
{
    for (int i = 10; i >= 0;i-- )
    {
        Console.WriteLine(i+"");
    }
    Console.ReadKey();
}
```

for ile yapılan kodlarda illa bir artım yaşanması zaruri değildir. Yukarıdaki örnekte 10’dan başlayıp 0’a kadar sayılar büyükten küçüğe doğru alt alta yazdırılmıştır.

```
static void Main(string[] args)
{
    int sayi;
    Console.WriteLine("Bir sayı giriniz:");
    sayi=Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("-----------------------------------------");
    for (int i = 0; i <=sayi ; i++ )
    {
        Console.WriteLine(i+"");
    }
    Console.ReadKey();
}
```

Bu kod örneğinde ise kullanıcıdan bir sayı girmesi istenmiştir. Girdiği sayı kaç olursa olsun 0’dan başlayarak girdiği sayıya kadar olan bütün sayılar alt alta yazılı şekilde gelecektir. Orada yazan en son sayı ise kullanıcının girdiği değer olacaktır.

Yukarıdaki kod örneğinde şöyle mantıksal bir hata yaşanabilir;

- Değerin negatif veya 0’dan küçük girilmesi

Böyle bir durum yaşanabilecek olması programcının algoritmasında muhakkak yer olmalıdır. Bu olası ihtimal if komutu ile engellenebilir.

```
static void Main(string[] args)
{
    double ogrenci, not, ortalama = 0;
    double toplam=0;
    Console.WriteLine("Sınıfınızda kaç kişi var?");
    ogrenci=Convert.ToDouble(Console.ReadLine());
    for (int i = 1; i <=ogrenci ; i++ )
    {
        Console.WriteLine(i+". öğrencinin not ortalamasını giriniz:");
        not = Convert.ToDouble(Console.ReadLine());
        toplam += not;
    }
    ortalama = toplam / ogrenci;
    Console.WriteLine("Sınıfınızın not ortalaması: " + ortalama);
    Console.ReadKey();
}
```

Bu örnekte ise kullanıcıdan sınıflarında kaç kişi olduğu sorusuna yanıt aranmıştır. Girilen sayısal yanıt değeri kez her öğrenci arkadaşının not ortalamasını sormakta ve sonunda bu girdiği not ortalaması değerlerinin toplamının öğrenci sayısına bölünerek aritmetik ortalaması kullanıcıyla paylaşılmaktadır.

Girilen notların küsuratlı olmak gibi bir ihtimali yok evet fakat ortalamanın küsuratlı çıkma ihtimali olduğundan mütevellit hepsini double değişken tipinde yapmak zorundasınız.

```
static void Main(string[] args)
{
    int adet=0;
    for (int i = 0; i <= 100; i++ )
    {
        if(i%2==0)
        {
            adet++;
        }
    }
    Console.WriteLine("0-100 arasında toplam " + adet + " tane çift sayı değeri var");
    Console.ReadKey();
}
```

Bu örnekte 0 ve 100 dahil olmak üzere aradaki sayılardan çift sayılar 2'ye bölümünden kalan 0 ise koşuluyla bulunarak tane bazında artırılmıştır. Döngü bittiğinde yani i değeri artık 101'e geldiğinde ise teker teker saydığı adet değişkeninin string cümle içerisinde kullanıcıyla paylaşmaktadır.

Eğer tek sayıları bulmamız istense idi if bloğu içerisinde ki koşul 2'ye bölümünden kalan 1 ise şeklinde olacaktı. Çünkü icabında bir sayı 2 ile bölündüğünde ya kalanı olmaz ya da kalanı bir çıkar.

for ile sadece sayısal değişkenlerde döngü yapılmaz. char verisi ile de örnek bir uygulamaya aşağıda yer verilmiştir;

```
static void Main(string[] args)
{
    for (char i = 'z'; i >= 'a'; i-- )
    {
        Console.WriteLine(i);
    }
    Console.ReadKey();
}
```

Bu kodların çıktısında alfabenin en büyük harfinden başlayarak alt alta sıralanmış harfler a harfine geldiğinde duracak ve döngü burada sonlanacaktır.

```
static void Main(string[] args)
{
    for (char i = 'a'; i <= 'z'; i++ )
    {
        Console.WriteLine(i);
    }
    Console.ReadKey();
}
```

Yukarıdaki örnekte z'den a'ya gidilirken ikinci örnekte ise a'dan z'ye doğru sıralı bir şekilde harf ilerlemesi yapılmaktadır.

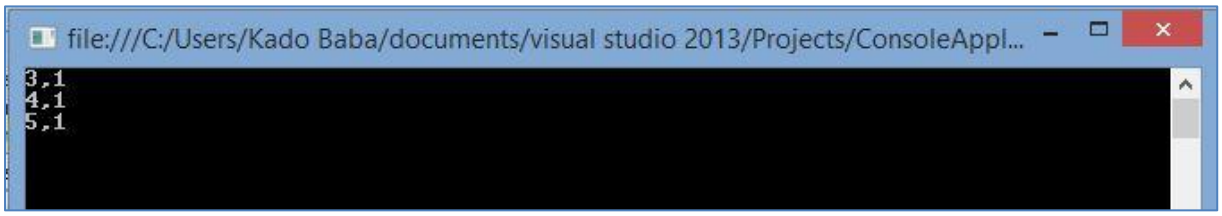
For döngüsünün bu denli ters yönlü olabilmesi için sondaki ++ veya – işaretinin değişmesi ve ikinci kısımdaki < ve > işaretlerinin değiştiği net bir şekilde görülmektedir.

```
static void Main(string[] args)
{
    for (int i = -20; i <= 5; i++ )
    {
        Console.WriteLine(i+"");
    }
    Console.ReadKey();
}
```

for döngüsünde sadece pozitif tam sayıların kullanması gibi bir zorunluluk bulunmamaktadır. Negatif tam sayıları da for döngüsünün içine dahil edebilirsiniz.

```
static void Main(string[] args)
{
    for (double i = 3.10; i <= 5.99; i++ )
    {
        Console.WriteLine(i+"");
    }
    Console.ReadKey();
}
```

Şu ana kadar for döngüsü içerisinde hep int veya tek örneğe mahsus char değişkenlerini kullandık. Eğer double türü değişken kullanıyorsanız başlangıçta verdiğiniz sayıyı +1 mantığıyla artıracak ve sondaki değere en yakın kısımda döngüyü sonlandıracaktır.



Yukarıdaki örneğin ekran çıktısı

3.10 ile başlayıp 5.99 ile bitirme aralığı koyduğumuz for kalıbında 3.10, 4.10 ve 5.10 değerlerine ulaşmıştır. Çünkü başlangıçta verilen sayının türü ve değişkeni ne olursa olsun +1.0 eklemesi yapmaktadır.

While komutu nedir?

while komutu ile aynı for mantığıyla döngü işlemi yapılmaktadır. Kodların yazılımı ve kullanılırlığı açısından for ile arasında bir takım farklar mecburen bulunmaktadır;

```
static void Main(string[] args)
{
    int i = 0;
    while(i<=20)
    {
        Console.WriteLine(i + "");
        i++;
    }
    Console.ReadKey();
}
```

Bu kodun ekran çıktısında başta 0 yazacak ve sürekli alt satıra inerek artan sayılar 20 olunca sonlanacak ve while döngüsünden böylece çıkış yapacaktır.

```
static void Main(string[] args)
{
    int i = 20;
    while(i>=0)
    {
        Console.WriteLine(i + "");
        i--;
    }
    Console.ReadKey();
}
```

0'dan başlayıp 20'ye kadar sıralı şekilde alt alta gitme örneği bir yukarıda verilmişti. Bu örnekte ise onun tam tersi verilmektedir. 20'den başlayıp 0'a kadar azaltma işlemi yaparak ekrana yazdırma komutunu uygulamaktadır.

```
static void Main(string[] args)
{
    int i = 20;
    while(i>=0)
    {
        Console.WriteLine(i + "");
        i--;
        if(i==10)
        {
            break;
        }
    }
    Console.ReadKey();
}
```

Bu örnekte ise gene 20'den başlayıp 0'a kadar azaltma işlemi yapılarak döngü mantığı devam etmektedir. i değişkeni 10 olunca if(i==10) parantezine takılacağı için break; kodu sayesinde döngü orada sonlanacaktır. Yani ekran çıktısında 20'den başlayıp 11'e kadar (11 dahil) bir sıralama gözükecektir.

```
static void Main(string[] args)
{
    int i = 0;
    while(i<=20)
    {
        Console.Write("a");
        i++;
    }
    Console.ReadKey();
}
```

Bu kod örneğinde 20-19-18-17-16-15-14-13-12-11-10-9-8-7-6-5-4-3-2-1-0 sayılarını döngü mantığıyla 21 kere tekrar ettiği için a harfini de yan yana 21 kez tekrarlayacak ve ekrana yazdıracaktır.

```
static void Main(string[] args)
{
    int i = 20;
    while(i>=0)
    {
        Console.WriteLine(i + "");
        i--;
        if (i == 0)
        {
            i = 20;
        }
    }
    Console.ReadKey();
}
```

Bu kod örneğinde ise 0'da while döngüsü bittiği anda i değeri tekrar 20 yapıldığı için sonsuza kadar 20'den başlayıp 0'a kadar uzanan döngü tekrarlanacak ve hiç bitmeyecektir.

```
static void Main(string[] args)
{
    int i = 0;
    while(i<=20)
    {
        if(i%2==0)
        {
            Console.Write("a");
        }
        if(i%2==1)
        {
            Console.Write("b");
        }
        i++;
    }
    Console.ReadKey();
}
```

Yukarıda yer alan örnekte 0 ve 20 dahil olmak üzere aradaki sayıların çift ve tek olma durumuna göre ekrana a ve b harfleri yazdırılmaktadır. Bu ekran çıktısında yan yana yazılı olan a ve b harflerinin kombinasyonunda 11 tane a harfi ve 10 tane b harfi yazılı olmak mecburiyetindedir.

switch ve case komutu nedir?

switch ve case kodlarıyla kullanıcı tarafından girilen bütün olası değerleri kontrol edebilir ve programı ona göre yönlendirebilirsiniz;

```
static void Main(string[] args)
{
    byte yanit;
    Console.WriteLine("1 ve 2 değerlerinden birini giriniz:");
    yanit = Convert.ToByte(Console.ReadLine());
    switch (yanit)
    {
        case 1:
            Console.WriteLine("Tek Sayı");
            break;
        case 2:
            Console.WriteLine("Çift Sayı");
            break;
        default:
            Console.WriteLine("Yanlış değer girişi yaptınız");
            break;
    }
    Console.ReadKey();
}
```

Yukarıda yer alan örnekte kullanıcıdan 1 ve 2 değerlerinden birisini girmesini istenmektedir. Kullanıcı 1 değerini girince “Tek Sayı” çıktısıyla 2 değerini girince ise “Çift Sayı” çıktısıyla karşılaşacaktır. Geriye kalan bütün olası değerleri yazdığında ise “Yanlış değer girişi yaptınız” şeklinde bir uyarı alacaktır.

switch parantezi içinde her case ve default altında mutlak surette break; eklemesinin yapılması zaruridir.

```
static void Main(string[] args)
{
    string yanit;
    Console.WriteLine("A harfine tıklayınız...");
    yanit = Console.ReadLine();
    yanit = yanit.ToUpper();
    switch (yanit)
    {
        case "A":
            Console.WriteLine("Tebrikler!");
            break;
        default:
            Console.WriteLine("A harfine tıklama yapmadınız");
            break;
    }
    Console.ReadKey();
}
```

Burada kullanıcıdan a harfine tıklanması istenmektedir. Kullanıcı ister Caps Lock tuşu açıkken bassın isterse de kapalıyken bassın her türlü olasılığa rağmen a harfine basması sonucunda “Tebrikler!” yazsını görecektir. Aksi haldeki bütün ihtimallerde ise “A harfine tıklama yapmadınız” uyarısıyla karışılacaktır.

```
static void Main(string[] args)
{
    string yanit;
    Console.WriteLine("4 köşesi olan herhangi bir geometrik şekil yazınız:");
    yanit = Console.ReadLine();
    yanit = yanit.ToUpper();
    switch (yanit)
    {
        case "KARE":
            Console.WriteLine("4 eşit kenara sahip geometrik şekildir.");
            break;
        case "DİKDÖRTGEN":
            Console.WriteLine("2 kısa ve 2 uzun kenara sahip geometrik şekildir.");
            break;
        default:
            Console.WriteLine("Geçersiz bir geometrik şekil adı girdiniz");
            break;
    }
    Console.ReadKey();
}
```

Bir önceki örnekte byte olduğu için tek karakterli girişleri “ “ içine almadan direkt olarak yazmıştık. Burada ise string değer olduğundan mütevellit tırnak içine alarak yazdık.

Bu örnekte yer alan uygulamanın aynısı if ve else başlıkları altında da yapabilirsiniz. Switch ve case komutu günümüzde değerini kaybetmeye yüz tutmaktadır. switch ve case yaptığınız her işlemin aynısı if ile yapabilirken if ile yaptığınız her işlemi switch ve case yapmanız mümkün değildir.

Random komutu nedir?

Random kod yapısıyla istediğiniz sayı aralığında istediğiniz kadar rastgele sayı üretebilir ve bunları ilgili yerlerde kullanabilirsiniz. Bunu basit örneklerle başlayarak açıklayalım;

```
static void Main(string[] args)
{
    Random rastgele = new Random();
    int sayi = rastgele.Next(5);
    Console.Write(sayi + "");
    Console.ReadKey();
}
```

Burada yeni bir rastgele adında random değeri oluşturulmuş ve bu değer int tipindeki sayi değişkenine bağlanarak 0-5 (5 dahil değil) arası bir rastgele değer üretimi yaptırılmıştır. Burada çıkması muhtemel 5 farklı sonuç vardır;

- 0
- 1
- 2
- 3
- 4

```
static void Main(string[] args)
{
    Random rastgele = new Random();
    int sayi = rastgele.Next(30);
    Console.Write(sayi + "");
    Console.ReadKey();
}
```

Burada ise çıkması muhtemel 30 farklı değer vardır. Bunlar sırasıyla şöyledir; 0-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29.

Random değerleri oluştururken her zaman başlangıç değerini 0 yapmak zorunda değiliz;

```
static void Main(string[] args)
{
    Random rastgele = new Random();
    int sayi = rastgele.Next(2,21);
    Console.Write(sayi + "");
    Console.ReadKey();
}
```

Burada oluşturulacak rastgele sayi değişkeninin değer aralığı ise 2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20 şeklindedir. Parantez içindeki ilk sayı dahil, ikinci sayının bir eksiği dahil olmak üzere şeklinde bir tanımlama yapılabilir.

```
static void Main(string[] args)
{
    Random rastgele = new Random();
    int sayi = rastgele.Next(2,21);
    int sayi2 = rastgele.Next(2,21);
    Console.Write(sayi + "-"+ sayi2);
    Console.ReadKey();
}
```

Bu örnekte oluşturulan sayi ve sayi2 değişkenleri her zaman aynı rastgele değerler olarak üretilmeyecektir. İkisinin de rastgele sayı üretimindeki başlangıç ve bitiş değerleri aynı olmasına rağmen her seferinde farklı bir olasılık çıkmaktadır.

```
static void Main(string[] args)
{
    Random rastgele = new Random();
    int sayi = rastgele.Next(2,21);
    int sayi2 = rastgele.Next(2,21);
    if(sayi+sayi2==30)
    {
        Console.Write(sayi + "+" + sayi2 +"= 30");
    }
    else
    {
        Console.WriteLine("Bu sefer ki denemenizde iki sayının toplamı 30 olarak çıkmadı");
    }
    Console.ReadKey();
}
```

Burada rastgele üretilen sayi ve sayi2 değişkeninin toplamı 30 olduğu takdirde ekrana matematiksel şekliyle verecektir. Diğer olasılıklarda ise olumsuz bir mesaj gösterilecektir.

```
static void Main(string[] args)
{
    Random rastgele = new Random();
    int sayi = rastgele.Next(2,21);
    int sayi2 = rastgele.Next(2,21);
    if(sayi>=sayi2)
    {
        Console.Write(sayi + " ve " + sayi2 );
    }
    else
    {
        Console.WriteLine("İlk üretilen sayı ikincisinden büyük çıkmadı");
    }
    Console.ReadKey();
}
```

Bu örnekte gene aynı kalıplarda üretilmiş sayi ve sayi2 değişkenlerinden sayi, sayi2'den büyük bir değer olduğu zamanlarda sonuçlar paylaşılacaktır. Aksi durumlarda olumsuz bir yanıt görünecektir.

```
static void Main(string[] args)
{
    Random rastgele = new Random();
    int zar = rastgele.Next(6);
    Console.WriteLine("Zardan size çıkan sayı " + (zar + 1));
    Console.ReadKey();
}
```

Bu örnekte üretilen değer aralığı ve 0-1-2-3-4-5 şeklindedir. Oysa bir zarda 0 rakamı yoktur ve en büyük değer 6'dır. Bunu sağlamak için üretilen sayıya ekrana yazdırma aşamasında 1 eklenerek değer aralığı aslında 1-2-3-4-5-6 şekline dönüştürülmüştür.

```
static void Main(string[] args)
{
    Random rastgele = new Random();
    int not = rastgele.Next(100);
    Console.WriteLine("Sınavdan aldığınız not " + (not + 1));
    Console.ReadKey();
}
```

Zar mantığının aynısı bu 100 lük sistemde alınan herhangi bir not olarak dönüştürülmüştür.

```
static void Main(string[] args)
{
    int deger;
    Random rastgele = new Random();
    int sayi = rastgele.Next(100,500);
    sayi = sayi + 1;
    Console.WriteLine("101 ve 500 arasında herhangi bir sayı giriniz: ");
    deger = Convert.ToInt32(Console.ReadLine());
    if(deger==sayi)
    {
        Console.WriteLine("Tebrikler!");
    }
    else
    {
        Console.WriteLine("Yanlış Bildiniz!");
        Console.Write("Doğru Yanıt: " + sayi);
    }
    Console.ReadKey();
}
```

Burada kullanıcıdan 101-500 aralığında bir değer girmesi istenmiş ve girdiği değerin arka planda rastgele üretilen sayi değişkenine eşit olup olmadığı if ve else ile kontrol edilmiştir. Bu kontrolün sonrasında kullanıcıya "Tebrikler!" ve "Yanlış Bildiniz!" olmak üzere iki farklı senaryo hazırlanmıştır.

Kullanıcının rastgele üretilen sayi değişkenini yanlış bilmesi halinde doğru yanıt ekranda gösterilmektedir.

goto komutu nedir?

goto yapısı ile programlamada belirli şekilde giden normal sıralamayı değiştirebilir ve ışınlama mantığı ile programın gidişatını hiçbir şeye bağlı olmadan gerçekleştirebilirsiniz;

```
static void Main(string[] args)
{
    Console.WriteLine("Adana");
    goto son;
    Console.WriteLine("Ankara");
    Console.WriteLine("İzmir");
    son:
    Console.WriteLine("Manisa");
    Console.ReadKey();
}
```

goto yapısı olmasaydı 4 şehri alt alta normal bir şekilde yazdıracaktı fakat goto eklemesi ile ekranda sadece Adana ve Manisa şehirlerinin yazıları gözükecektir.

goto son; ile son: noktasına hızlı bir geçiş yapılmıştır ve aradaki bütün kodlar gerçekleştirilmemiştir.

```
static void Main(string[] args)
{
    bas:
    int faktoriyel=1;
    Console.WriteLine("Bir sayı giriniz:");
    int sayi = Convert.ToInt32(Console.ReadLine());
    for (int i = sayi; i >= 1; i--)
    {
        faktoriyel *= i;
        if(i==1)
        {
            Console.WriteLine(sayi+"!= "+faktoriyel + "");
        }
    }
    Console.WriteLine("Tekrar değer girişi yapmak için 'T' tuşa basınız");
    string giris = Console.ReadLine();
    giris = giris.ToUpper();
    if(giris=="T")
    {
        goto bas;
    }
    Console.ReadKey();
}
```

Bu örnekte kullanıcının girdiği sayının faktöriyeli for ile bulunmuş ve kullanıcıya sonucu bildirilmiştir. Kullanıcının tekrar bir değer girme isteği ihtimali üzerine 'T' tuşuna basması dahilinde aynı işlemlerin komple baştan yapılması goto ile sağlanmıştır.

try ve catch komutu nedir?

Yazılım sürecinde programlarınızın kullanımı her zaman sizin istediğiniz ve beklediğiniz şekilde gerçekleşmeyecektir. Kullanıcıdan sayısal veri istediğiniz zaman metinsel ifade girilmesi gibi if ile kontrol edilemeyecek bazı sistemsel hataları try-catch kodları ile engelleyebilirsiniz;

```
static void Main(string[] args)
{
    bas:
    Console.WriteLine("Bir sayı giriniz:");
    try
    {
        int sayi = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("Doğru giriş yaptınız!");
    }
    catch
    {
        Console.WriteLine("Lütfen geçerli ve doğru bir sayı giriniz...");
        goto bas;
    }
    Console.ReadKey();
}
```

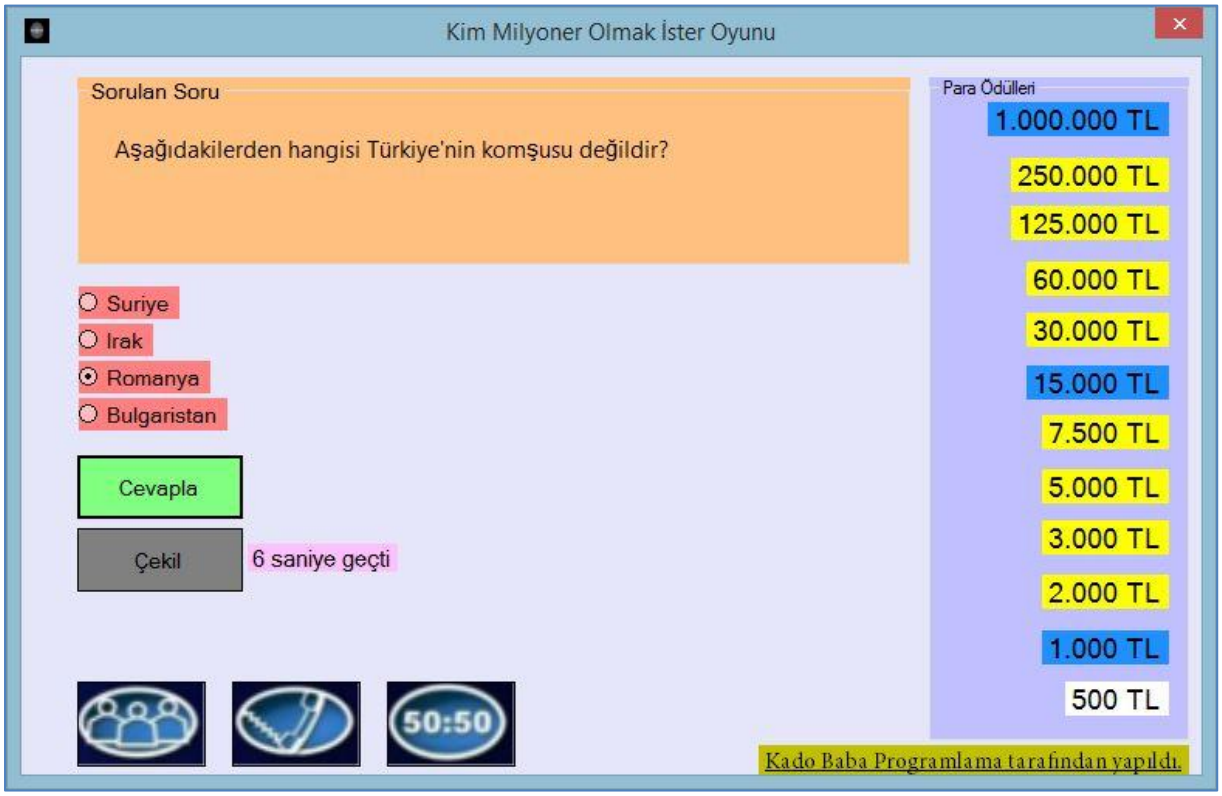
Yukarıda kullanıcıdan bir sayı girmesi istenmiştir. Kullanıcı burada metinsel ifade ve özel karakter girerse olumsuz senaryoları bünyesinde barındıran catch parantezine yönlendirilecek ve uyarı mesajı ile karşılaşacak. Doğru bir değer girmesi ihtimalinde ise zaten try parantezinde olduğu için hata yaşanmayacak.

```
static void Main(string[] args)
{
    bas:
    Console.WriteLine("Bir karakter girişi yapınız: ");
    try
    {
        char karakter = Convert.ToChar(Console.ReadLine());
        Console.WriteLine("Doğru giriş yaptınız!");
    }
    catch
    {
        Console.WriteLine("Lütfen geçerli ve doğru bir değer giriniz...");
        goto bas;
    }
    goto bas;
    Console.ReadKey();
}
```

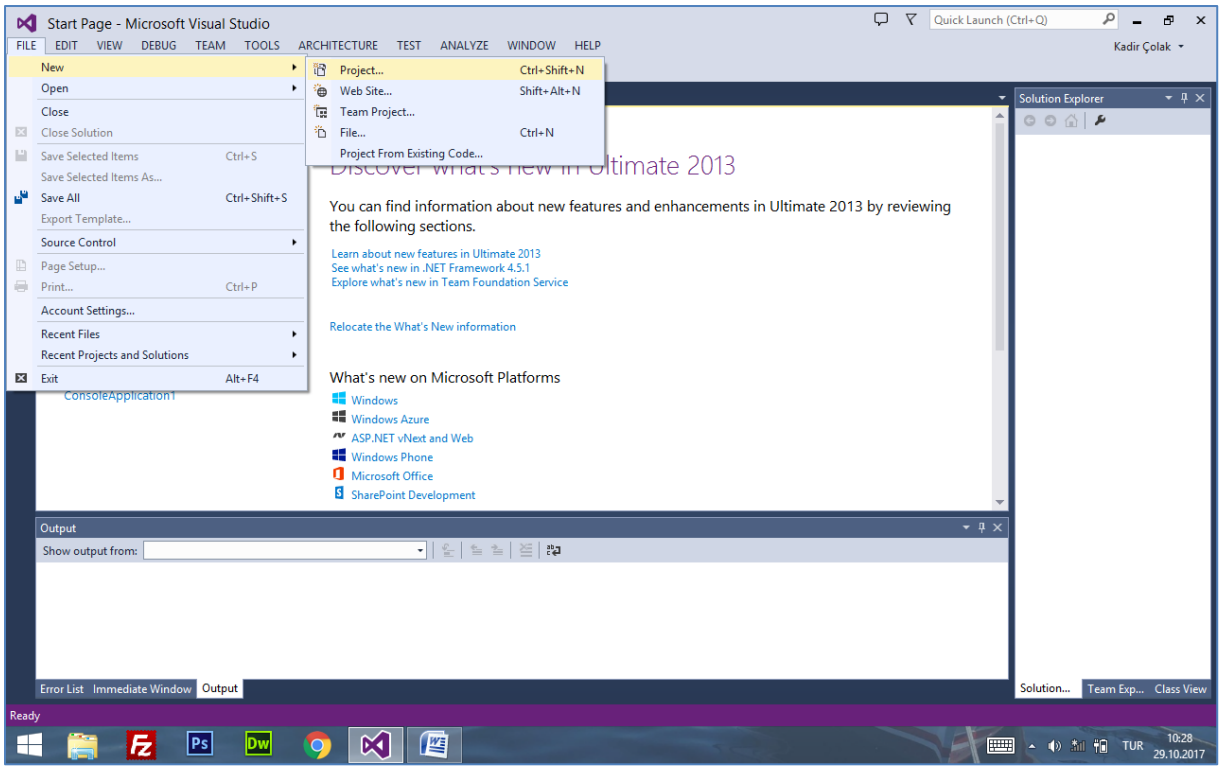
Aynı kalıp örnek bu sefer char değişkeni üzerinde örnek olarak yapılmıştır.

Windows Form Application nedir?

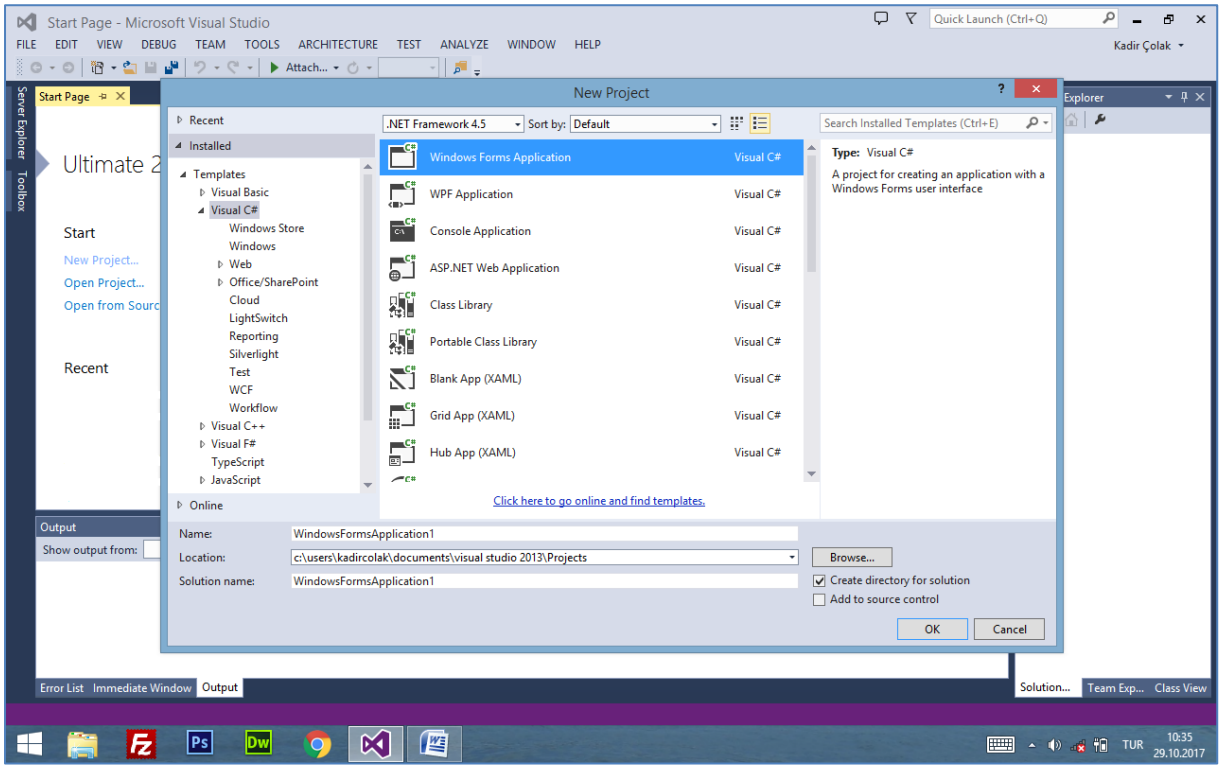
Windows Form Application, Windows işletim sistemlerinde herkesin rahatlıkla erişebileceği görsel ve tasarımsal bir içeriğe sahip daha geniş fonksiyonları ve özellikleri bulunan program türleridir. Console ekranında ekleyemediğimiz buton ve renklendirmeler gibi birçok ekstra özelliği burada istediğiniz gibi ayarlayabilirsiniz.



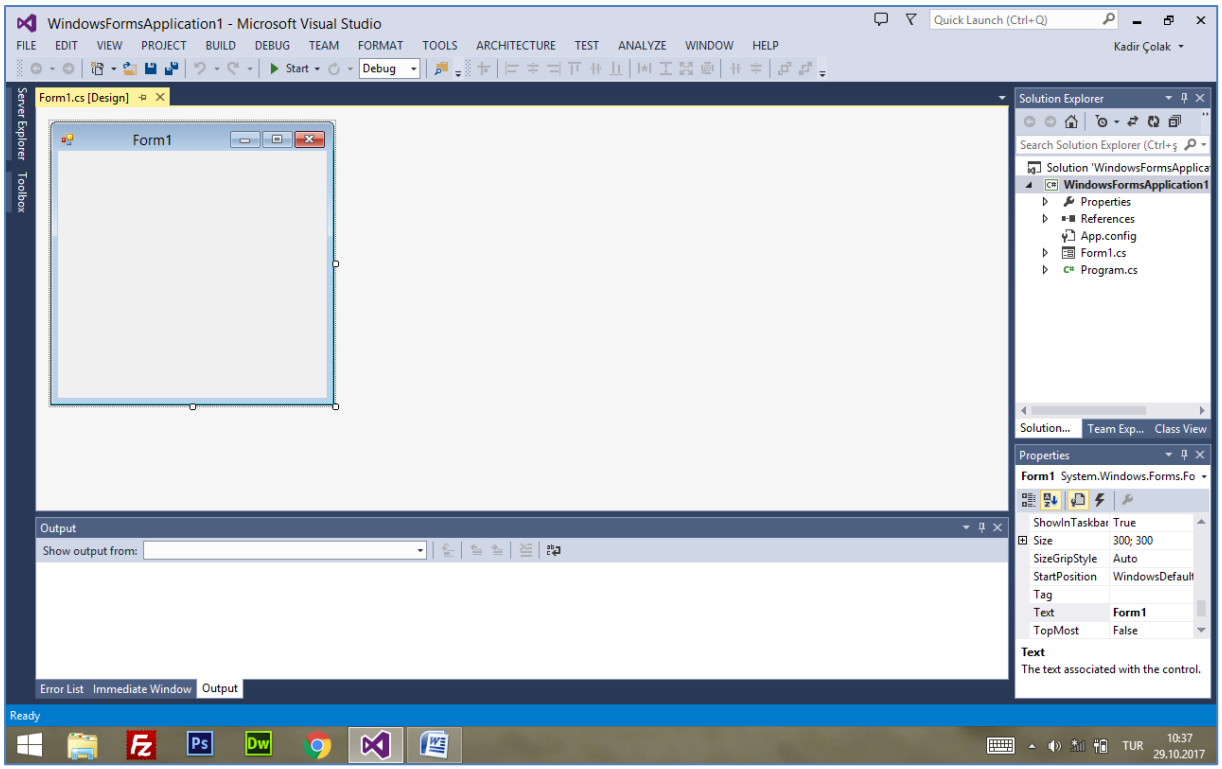
Yukarıda örnek bir Windows Form Application örneği olan *Kim Milyoner Olmak İster?* oyunu gösterilmiştir.



Yukarıdaki resimde göründüğü gibi adımlar izlendiğinde aşağıdaki menü karşınıza çıkacaktır;



İkinci adımda ise Windows Form Application menüsü seçerek ilgili kodlama ve yazılım ekranına giriş yapacaksınız. Karşınıza görsel tasarımına karışabileceğiniz bir yönetim merkezi çıkacaktır.



Yukarıdaki görselde yer alan bir yönetim merkezi çıkacak karşınıza. Buradan gerek formun ekranı gerek diğer ayarlar olsun onları değiştirebilir ve eklemeler yapabilirsiniz.

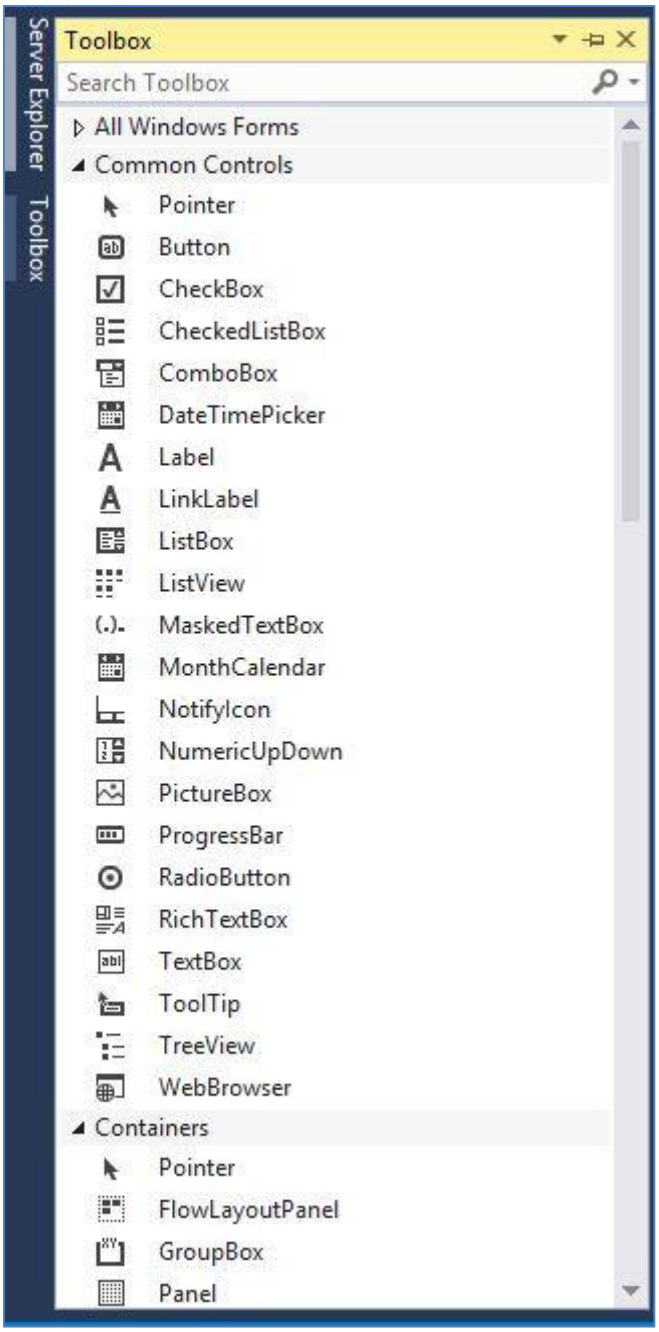
Windows Form Application bünyesinde yazılan kodların ve algoritmanın gerçek çıktısını görmek için F5 tuşuna basmalı veya “Start” tuşuna basmalısınız.

Toolbox nedir?

Toolbox, en genel tanımıyla Windows Form Application bünyesinde yeni nesneler eklememizi sağlayan yerdir. Buton, Textbox, CheckBox, RadioButton, ListBox vb. birçok örnek öğeyi Toolbox içerisinden seçer ve yerleştiririz.

Eğer Toolbox nesnesini en solda göremiyorsanız **Ctrl+Alt+X** tuşlarının kombinasyonu sayesinde program yönetim merkezinize Toolbox kısmını eklettirebilirsiniz.

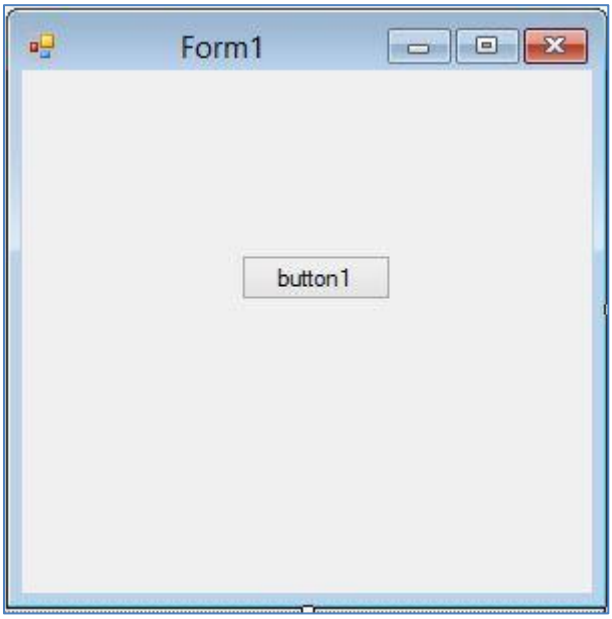
Tamamen yerli üretimidir.



Kadir ÇOLAK tarafından yazılmıştır.

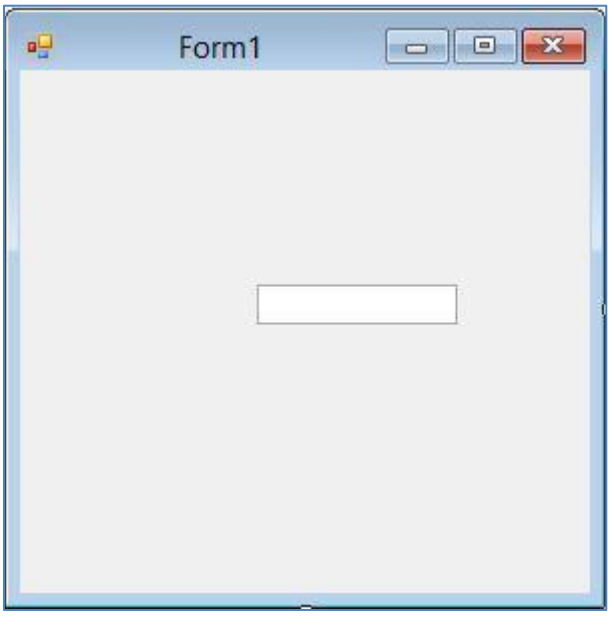
Button nedir?

button genellikle kullanıcıdan bir onay olmak, bir sonraki işleme geçebilmek ve diğer bazı kilit görevleri egale etmek amacıyla kullanılan ve yerleştirilen bir toolbox nesnesidir.



TextBox nedir?

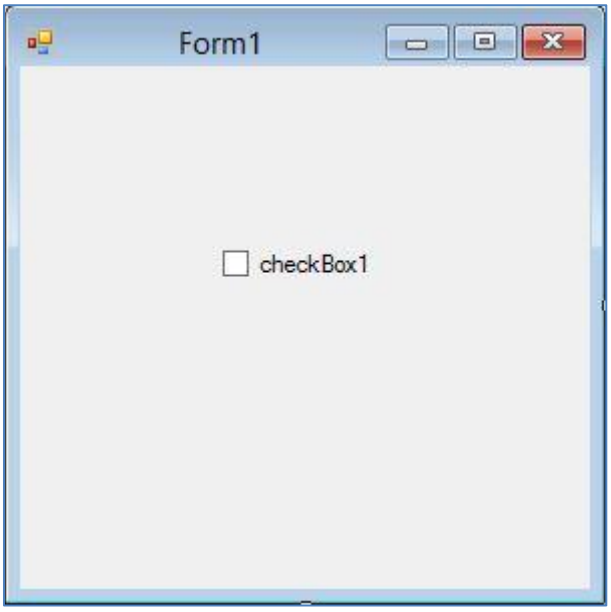
textBox, kullanıcıdan çeşitli türlerde veri ve değer almaya yarayan kutu şeklindeki toolbox nesnesidir. Örnek olarak kullanıcıdan istenen kullanıcı adı bilgisi veyahut T.C. kimlik numarası bilgisini textBox nesnesi aracılığı ile öğrenirsiniz.



CheckBox nedir?

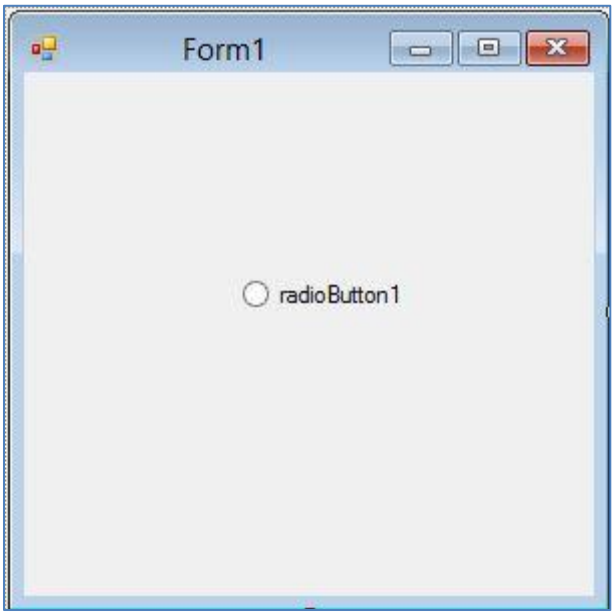
checkBox, kullanıcıdan onaylanması istenen durumlarda işaret bırakıp bırakmama seçeneğiyle hizmet gören toolbox nesnedir.

Bir Windows Form Application ekranına ne kadar checkBox koyarsanız koyun kullanıcı onların hepsini birden işaretleyebilmektedir. Kullanıcının hobileri ve fobileri örnek olarak checkBox ile öğrenilebilir.



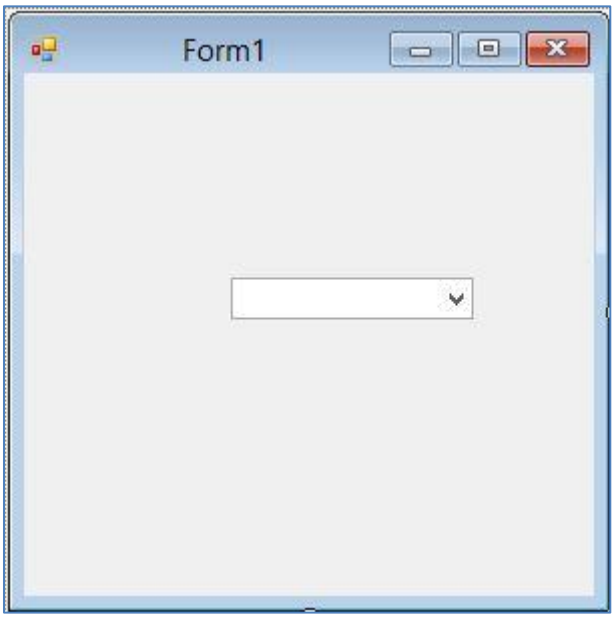
RadioButton nedir?

radioButton, kullanıcıya sorulan birkaç farklı seçeneğe tek cevap vermesi gerektiği durumlarda kullanılan toolbox nesnesidir. Aynı checkBox gibi işaretlenebilme özelliği bulunur fakat maksimum 1 tanesini seçili olur. Örnek olarak bir kişinin evli veya bekâr olduğunu radioButton ile öğrenebilirsiniz.



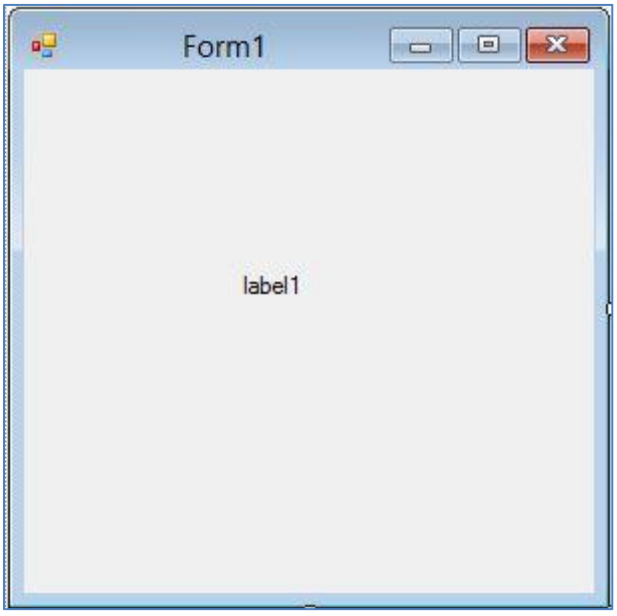
ComboBox nedir?

comboBox içinde bir sürü değeri barındıran ve kullanıcı tarafından bunların bir tanesinin seçilmesinin istendiği durumlarda kullanılan toolbox nesnesidir. Örnek olarak yaşadığınız şehri comboBox nesnesi içerisinde seçersiniz.



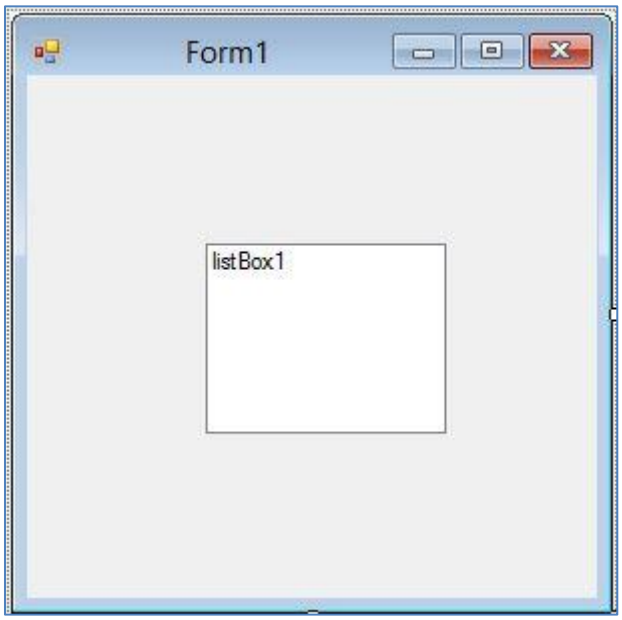
Label nedir?

label bazı durumlarda bilgi vermek amaçlı bazı durumlarda ise hata mesajını göstermek amaçlı kullanılır. Burada hangisini ne zaman ve nasıl kullanacağınız sizin öznel görüşünüze bağlı olarak değişiklik göstermektedir.



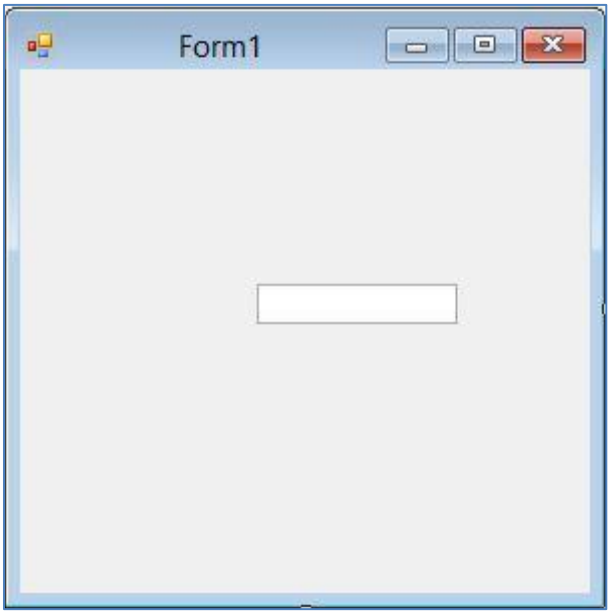
ListBox nedir?

listBox ihtiyaç duyulan bilgi ve değişkenlerin bir yerde düzenli şekilde toplandığı bölüm olarak düşünülebilir. Gelişen durumlara göre listBox bünyesinde eleman ekleme, çıkartma, güncelleme ve yerini değiştirme gibi işlemler yapılabilir.



MaskedTextBox nedir?

Birkaç sayfa önce anlatılan textBox nesnenin aşırı gelişmiş sürümüdür. Telefon numarası, posta kodu, telefon numarası, tarih tarzında girilen değeri filtreleyerek kullanıcının o şekilde giriş yapma koşulunu getirebilirsiniz. textBox ile maskedTextBox nesnelerinin görünümü aynıdır.

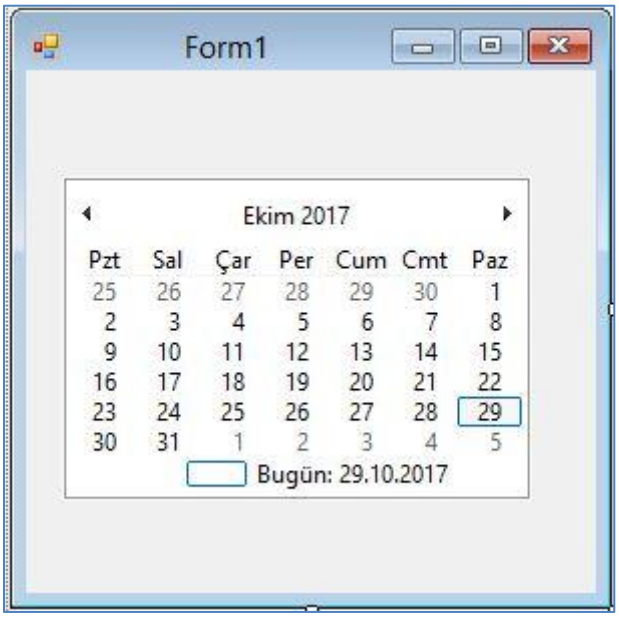


MenuStrip nedir?

menuStrip aslında her programda gördüğünüz bir yönetim merkezi olarak varsayılabilir. Office programlarında en yukarıda solda uzun şekilde duran menüden kaydetme-ekleme-düzenleme tarzı geniş çaplı ayarlamaları yapabilmektesiniz. İşte bu menuStrip adlı toolbox nesnesi sayesinde olur.

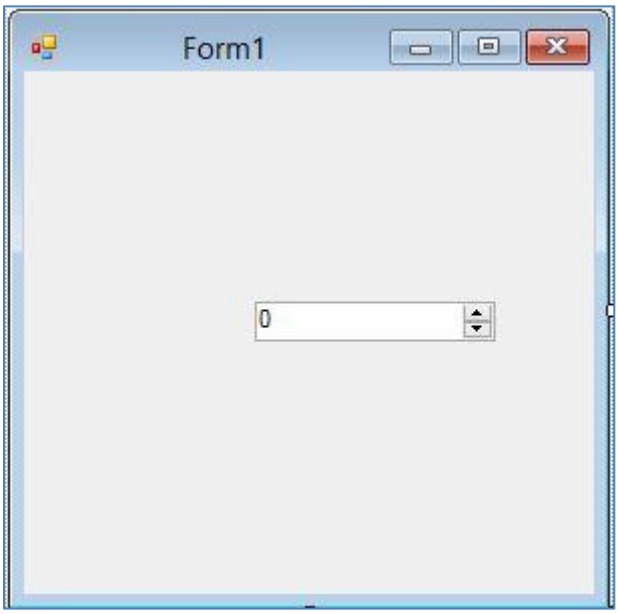
MonthCalendar nedir?

monthCalendar ile tarih bilgisinde o ayın bütün bilgilerini ve günlerini kullanıcıya gösterebilirsiniz. Bu nesne ile bir nevi takvim görevi gördürebilirsiniz.



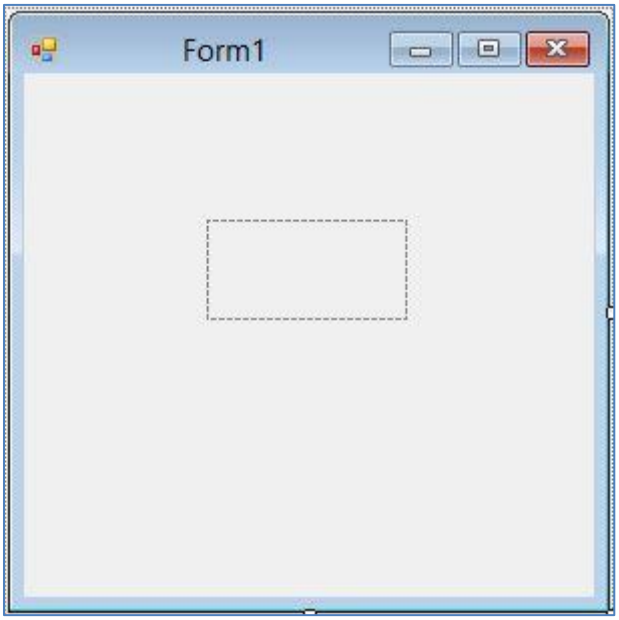
NumericUpDown nedir?

Kullanıcıdan sayısal bir değer istendiği zaman belirli sayı değerleri arasında bir tercih yapma işlemini numericUpDown nesnesi ile gerçekleştirir. numericUpDown nesnesi ile yapılabilecek çok geniş açıda işler bulunmaktadır.



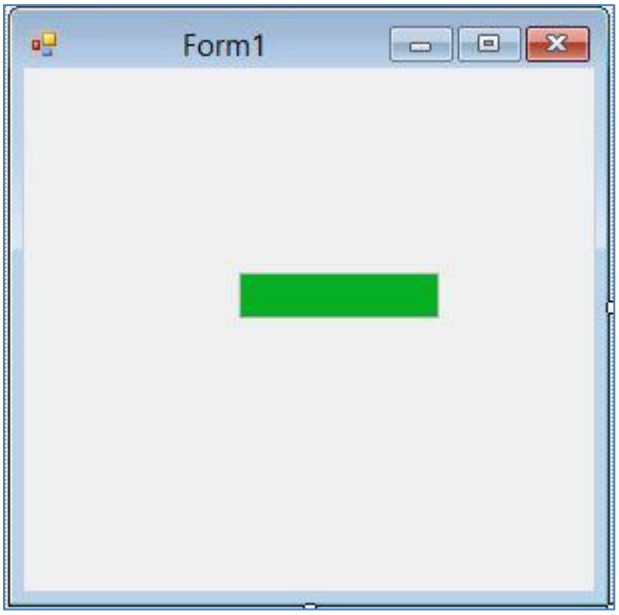
PictureBox nedir?

pictureBox ile programınızın istediğiniz kısmına ve bölümüne resim-fotoğraf ve gif tarzı öğeler eklemesi yapabilirsiniz. Resmin kaynağını alma aşamasında diğer bilgisayarlarda o resmin gözükmemesi gibi teknik bir arıza yaşanmayacaktır.



ProgressBar nedir?

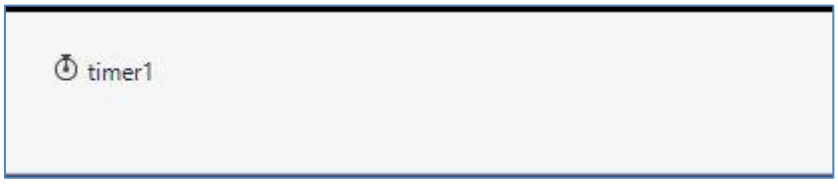
progressBar tam ve en net tanımıyla yükleme çubuğudur. Program kurulumlarında setup yüklemesi esnasında zamanla artan o nesne işte bu progressBar olarak tanımlandırılmıştır.



Timer nedir?

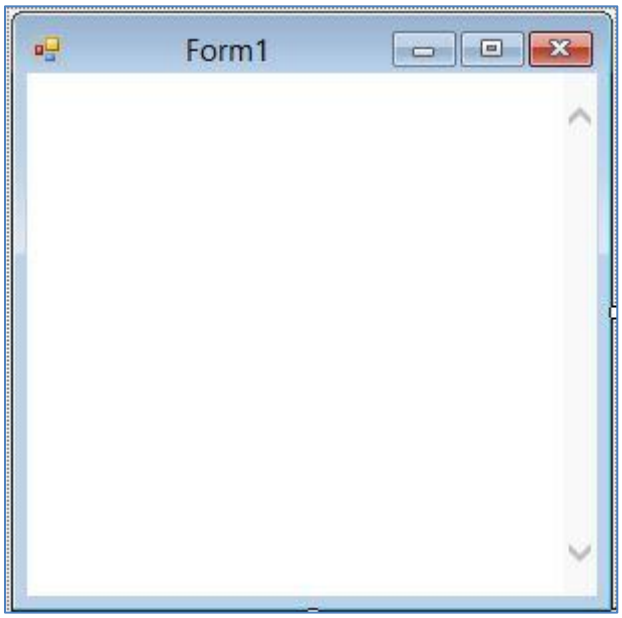
timer programlarınızda bazen belirli bir zaman içerisinde yapılması gerektiği koşullarda kronometre yani zaman ölçer görevini üstlenmektedir. Zamanlayıcının hızını siz istediğiniz şekilde artırabilir veya da azaltabilirsiniz.

timer nesnesinin properties kısmında yer alan “Interval” değeri 1000 olduğunda timer her 1 saniyede yazılan işlemi gerçekleştirecektir.



Web Browser nedir?

webBrowser ile programınızda belirli internet sitelerini açabilir veya internet üzerinden online bağlantılar gerçekleştirebilirsiniz. webBrowser ile ortalama bir web tarayıcısı programı yapmanız dahi mümkündür.

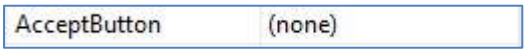


Properties nedir?

Properties ile gerek toolbox nesnelerinizi gerek form ekranınızı ve hatta gerekse programınızın genel ayarları ve çeşitliliklerini değiştirebileceğiniz ayarlama menüsüdür. Properties kelimesi zaten İngilizce "***özelikler***" anlamına gelmektedir. Her türlü nesne ve ekranın özelliklerini buradan değiştirebilirsiniz.

Properties kısmında yaptığınız ve yapmadığınız şeyler programın sistemi ve kodlama yapısı kadar önem görmektedir. Bunun asli nedeni ise kullanıcıların sistem bilgisinden öte ilk olarak görsel şekle ve kalıpsal durumlara bakması ve ona göre analiz etmesidir.

FORM PROPERTİES



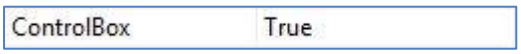
Form ekranında Enter tuşuna basıldığında otomatik olarak çalışacak seçili butonu belirler.

Form ekranın arka plan renginin ne olacağına karar verilir.

Form ekranın arka planında hangi resmin olacağına karar verilir.

Form ekranında duran arka plan resminin ekrana nasıl sığacağı ve yerleşeceği konusunda karar verilir.

Form ekranında Esc tuşunda basıldığında otomatik olarak çalışacak seçili butonu seçer.



Form ekranın sağ üstünde yer alan büyütme-küçültme ve kapatma tuşlarını kapatır(false) veya açar(true).

Form ekranında bulunan fare imlecinin şeklinin ne olacağına karar verilir.

Form ekranında yazıların fontunun, büyüklüğünün ve şeklinin ayarlandığı yerdir

Form ekranında yazıların yazınını renginin karar verildiği yerdir.

Formun çerçevesinin yani dış ekranın nasıl olacağına karar verilir.

Form ekranın en sol üstünde yer alan minik icon ögesinin değiştirilebildiği yerdir.

Form ekranında klavyeden basılacak tuşların rol alıp olmamasına karar verilir.

MaximizeBox	True

Form ekranın büyültüp büyültmediğine karar verilen yerdir.

⊞ MaximumSize	0; 0

Form ekranın en fazla kaça kaç boyutta büyütülebileceğine karar verilir.

MinimizeBox	True

Form ekranının küçültülüp küçültülmediğine karar verilen yerdir.

⊞ MinimumSize	0; 0

Form ekranın en az kaça kaç boyutta küçültülebileceğine karar verilir.

Opacity	100%

Form ekranının görünürlük derecesinin, şeffaflığının karar verildiği yerdir.

ShowIcon	True

Form ekranın sol en üstünde icon olup olmaması durumunun kararı verilir.

⊞ Size	300; 300

Form ekranın kaça kaç boyutta olacağına karar verilir.

StartPosition	WindowsDefaultLocatio

Form ekranın açılış şekli ve konumunun nasıl olacağına karar verilir.

Text	**Form1**

Form ekranın üstünde yazan başlığın ne olacağının kararının verildiği yerdir.

TopMost	False

Form ekranın en üstte durup durmamasın kararı burada verilir.

WindowState	Normal

Form ekranının Windows bildiriminin türünün kararı verilir.

Form ekranın properties kısmı olduğu gibi eklediğiniz her toolbox nesnesinin de bir properties durumu mutlaka vardır**.** Şimdi bunlara göz atalım;

Enabled	True

Bu özellik ile o nesnenin kullanıcı tarafından kullanılıp kullanılmaması özelliğini aktif veya pasif edersiniz.

FlatStyle	Standard

Nesnenin tasarımına hafif bir şıklık ve göz zevki katmaya yarayan properties ayarıdır.

⊞ Location	99; 59

İlgili nesnenin form ekranında nerede durduğuna dair bilginin verildiği yerdir.

RightToLeft	No

Yazının sağa dayalı olup olmamasının kararı burada verilir.

Text	**button1**

Nesnenin üstünde yazan yazıyı yani söz ibaresini belirler.

TextAlign	MiddleCenter

Yazının hizasının yani yerleştirilmesinin nasıl olunacağının ayarı yapılır.

Visible	True

Nesnenin görünür veya görünmez olma durumuna el atılan kısımdır.

Maximum	100

numericUpDown nesnesinde yapılabilecek en büyük sayısal değeri belirler.

Minimum	0

numericUpDown nesnesinde yapılabilecek en küçük sayısal değeri belirler.

Increment	1

numericUpDown nesnesinde sayıların kaçar kaçar artacağı burada belirlenir.

UseSystemPasswordCh	False

textBox nesnesine yazılanlar şifre görünümüyle gizli karakterler şeklinde gözüküp gözükmemesi durumunu ayarlar.

ReadOnly	False

İlgili nesneye değer girilmeden sadece okunup okunmaması durumunu belirler.

MaxLength	32767

Değer girişi yapılabilme özelliği olan bir nesneye en fazla kaç karakter girileceğinin ayarlandığı ve değiştirilebildiği yerdir.

Multiline	False

textBox gibi değer girilebilen nesnelerinin boyut sınırlandırmalarının kaldırılarak en büyük şekle sokulabilir hale getirilmesi true/false mantığıyla ayarlanır.

DropDownStyle	DropDown

comboBox nesnesinde programcı o nesnenin içine hangi seçenekleri eklediyse kullanıcı onlardan birini seçmek zorunda kalır. Kendisi yeni manüel olarak bir ekleme yapamaz.

Sorted	False

comboBox nesnesine eklenen her bir seçeneğin bütün şekliyle o nesnede alfabetik olarak sıralanıp sıralanmaması durumu buradan kontrol edilir.

Checked	False

radioButton veya checkBox gibi işaretleme özelliği olan nesnelerin başta işaretli veya işaretsiz olması buradan yönetilir.

AutoSize	False

İlgili nesnenin otomatik olan yükseklik ve genişlik ölçülerinin bir referansa bağlı kalınmadan özgürce değiştirebilme veya değiştirilememe özelliğidir.

Image	(none)

pictureBox nesnesinde resmin alındığı kaynağı adresi gösterir.

Value	0

Aldığı değerin türü sayısal yani int kalıbında olan nesnelerin değerinin göstermek miktarını belirtmek için kullanılır.

Interval	100

timer nesnesinin ne kadar süre içerisinde bir birim atlayacağını belirler.1000 interval değeri Timer nesnenin artış miktarının saniyelik olduğu anlamına gelmektedir. interval miktarı 10 olursa eğer bu artışın saliselik olduğu anlaşılacaktır.

Url	

webBrowser nesnesinin açılış esnasında hangi internet sitesini yani web adresini göstereceğini belirler.

Form ekranında ilgili nesneye tıklanınca kod eklemesi yapmak istiyorsanız formdaki o nesneye çift tıklama yapmanız gerekmektedir. Tıklama sonucu çıkacak kod sayfasında { ve } parantezleri arasındaki boşluk sizin kod yazacağınız alandır.

```
private void button1_Click(object sender, EventArgs e)
{

}
```

button1'e tıklandığı olacak olaylar bu parantez içinde yazılır.

```
private void textBox1_TextChanged(object sender, EventArgs e)
{

}
```

textBox1 nesnesine değer girişi yapıldığında olacak olaylar bu parantez içinde yazılır.

```
private void Form1_Load(object sender, EventArgs e)
{

}
```

Form1 nesnesi yüklendiğinde yani program açıldığında yapılacak olaylar bu parantez içinde yazılır.

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{

}
```

comboBox1 nesnesinden herhangi bir seçenek veya eleman seçilmesi sonrasında bu parantez içinde yazan kodlar gerçekleşir.

```
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{

}
```

checkBox1 nesnesine işaretleme yapılsa da yapılmasa da tıklanması dahilinde parantez içindeki kodlar gerçekleşir.

```
private void button1_Click(object sender, EventArgs e)
{
    this.Enabled = false;
}
```

Bu kod ile form ekranına ne var ne yoksa kilitlenir ve tıklanamaz hale gelir.

```
private void button1_Click(object sender, EventArgs e)
{
    this.Opacity = 50;
}
```

Form ekranının görünürlük yani şeffaflık derecesini 0 ve 100 arasında istediğiniz zamanlarda kod ile de değiştirebilirsiniz.

```
private void button1_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

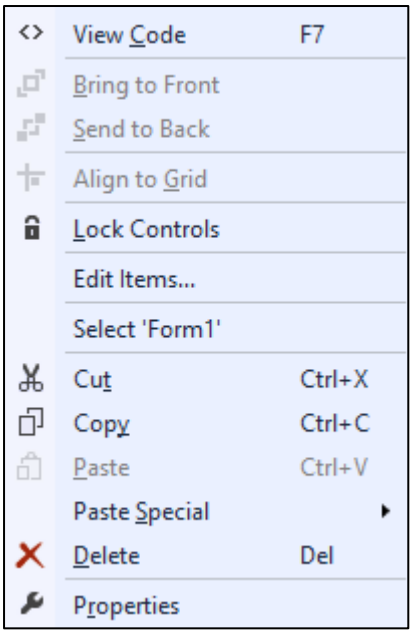
Örnek olarak bir "*Programdan Çıkış*" butonu yaparsanız kod kısmına bunu yazmanız gerekmektedir.

```
private void button1_Click(object sender, EventArgs e)
{
    System.Diagnostics.Process.Start("https://google.com.tr");
}
```

Bu kodun yazılı olduğu bir butonda kullanıcı butona tıklaması sonrasında bilgisayarındaki seçili web tarayıcısından sizin belirlediğiniz internet adresine yönlendirilecektir.

```
private void timer1_Tick(object sender, EventArgs e)
{
    label1.Text = DateTime.Now.Hour.ToString() + ":" + DateTime.Now.Minute.ToString() + "." +
      DateTime.Now.Second.ToString() + "".ToString();
}
```

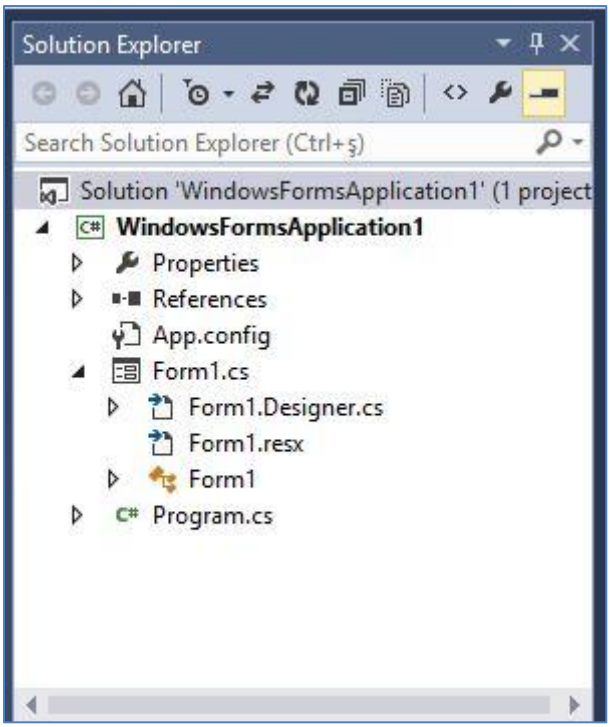
Timer1 nesnesinin Enabled özelliği "True" olduğu müddet yukarıda yazılı olan kod her saniye artarak size o anki doğru saat-dakika ve saniye bilgilerini vermekte olacaktır. Bunu programınızın herhangi bir köşesinde sisteme müdahalede bulunmayacak şekilde kullanabilirsiniz.



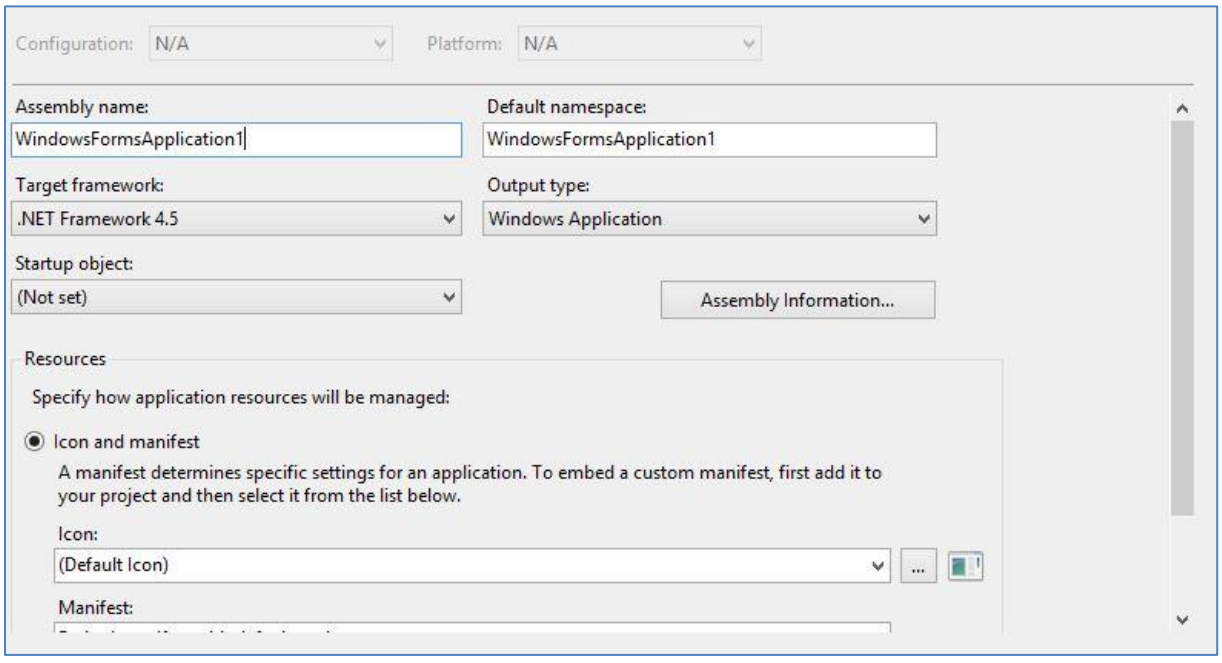
Herhangi bir eklediğiniz toolbox nesnesine sağ tıklama yaparak birkaç işlem de gerçekleştirebilirsiniz;

View Code	İlgili nesnenin kodlarını açar
Bring To Front	Nesneyi öne getirir.
Send To Back	Nesneyi en arkaya atar.
Cut	Kesme işlemi yapar
Copy	Kopyalama işlemi yapar
Paste	Yapıştırma işlemi yapar
Delete	Silme işlemi yapar
Properties	İlgili nesnenin özellikler kısmını açar

Örnek olarak program ekranınıza yani forma bir buton eklediniz ve daha sonradan bu eklemeyi yapmaktan vazgeçtiniz. Form ekranında görünüm olarak "*delete*" kısmıyla nesneyi sildiniz ama arka planda yani kod safhasında o ilgili butona ait kodlar gene mevcut. Bu gereksizliği ve fazla alan kapsama durumunu yok etmek için ilgili nesneyi hem formdan hem de kod satırından tamamen kaldırmanız size en verimli sonucu verecektir.



Solution Explorer içerisinde yer alan Properties kısmına tıkladığınız zaman aşağıdaki görüntüde bir kısım gelecektir;

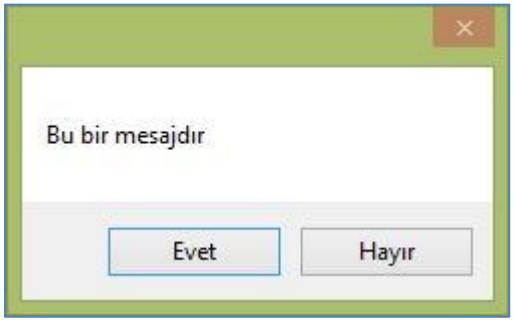


Bu çıkan yerde icon altında bulunan default icon kısmına başka bir icon yüklemesi yaparsanız kullanıcının masaüstünde bulunan. exe halinde ki uygulamasının dış görünüşü değiştirmiş olursunuz.

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Bu bir mesajdır");
}
```

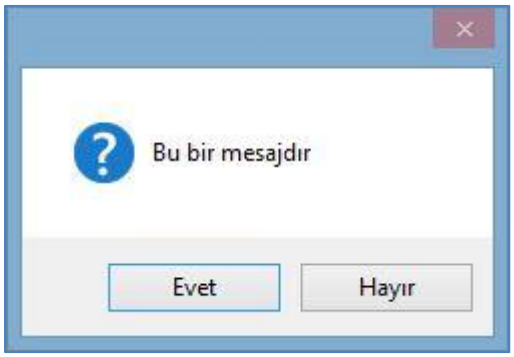
```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Bu bir mesajdır","Başlık");
}
```

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Bu bir mesajdır","",MessageBoxButtons.YesNo);
}
```

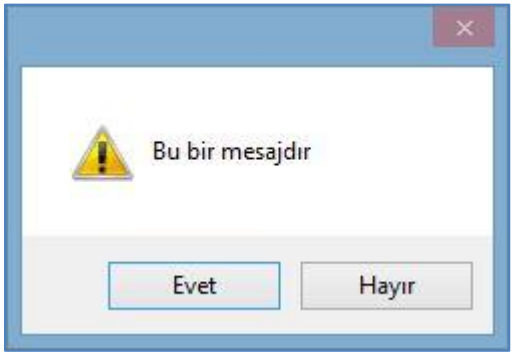


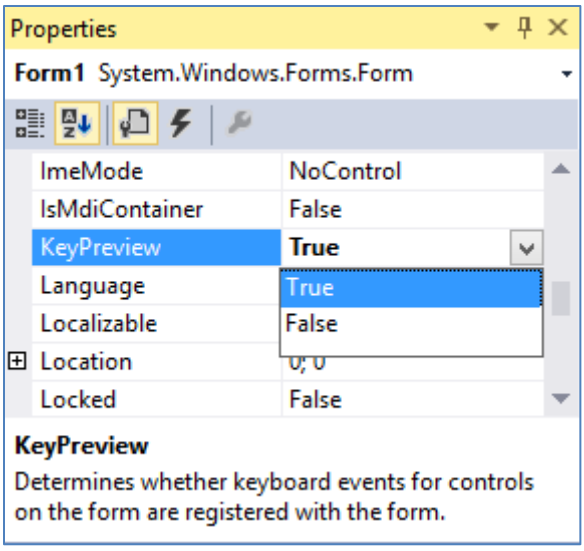
```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Bu bir mesajdır","Başlık",MessageBoxButtons.YesNo);
}
```

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Bu bir mesajdır","",MessageBoxButtons.YesNo,MessageBoxIcon.Question);
}
```

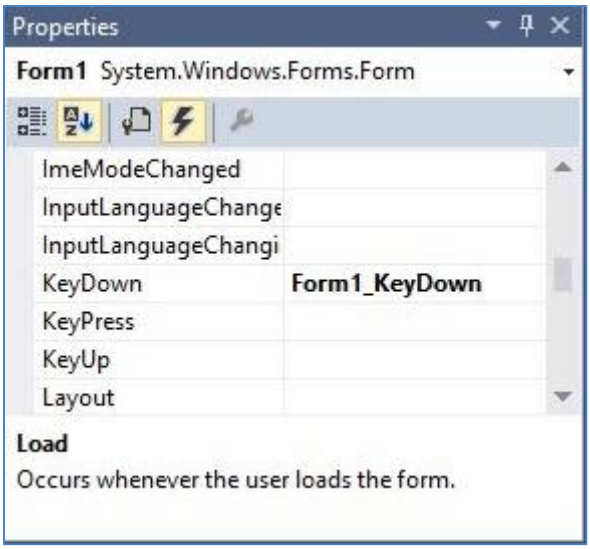


```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Bu bir mesajdır","",MessageBoxButtons.YesNo,MessageBoxIcon.Warning);
}
```





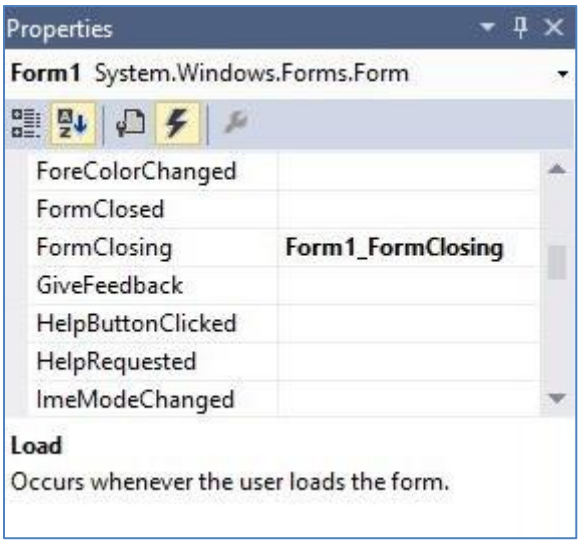
Form ekranının properties kısmından KeyPreview kısmını "*True*" olarak değiştiriniz.



Properties penceresinde yer alan şimşek resimli "*Event*" kısmına girerek KeyDown kısmını aktif hale getiriniz.

```
private void Form1_KeyDown(object sender, KeyEventArgs e)
{
    if(e.KeyCode==Keys.A)
    {
        MessageBox.Show("A tuşuna bastınız","TEBRİKLER",MessageBoxButtons.OK,MessageBoxIcon.Asterisk);
    }
}
```

Yukarıda örnek verilen kod form ekranında nerede olursa olsun A tuşuna basılması dahilinde ilgili MessageBox çıktısını verecektir.

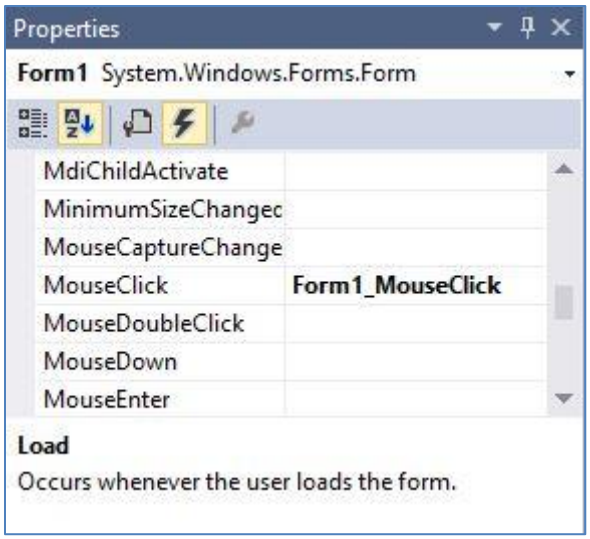


Properties kısmının "*Event*" sekmesinde yer alan FormClosing kısmını aktif hale getiriniz.

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    Application.Exit();
}
```

Bu kod sayesinde program tamamıyla kapanacaktır. Diğer türlü sağ üstteki X kısmına basarak sadece temel şekliyle sonlandırma işlemi yapılacaktır ve program bazı durumlarda arka planda açık kalmaya devam edecektir.

Bu kod ile arka planda uzantıları kalmadan her noktadan program sonlandırılmakta ve çıkış yapılmaktadır.

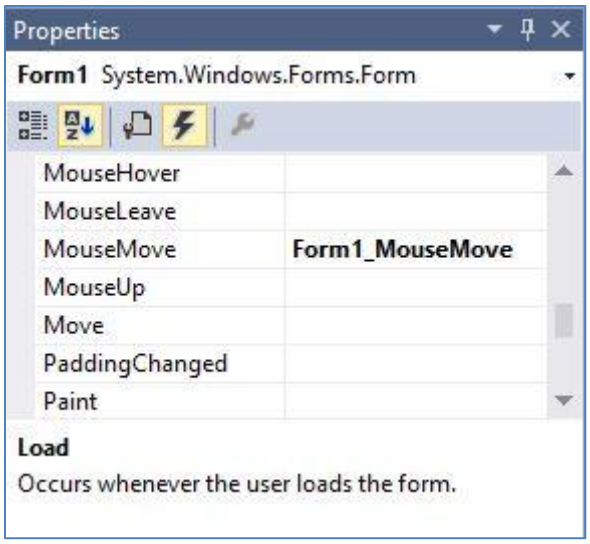


Properties kısmının "*Event*" sekmesinde yer alan MouseClick kısmını aktif hale getiriniz.

```
private void Form1_MouseClick(object sender, MouseEventArgs e)
{
    MessageBox.Show("Ekrana tıkladınız");
}
```

Bu örnekte gösterilen kod ile programda şöyle bir olay yaşanacaktır; form ekranın neresi olursa olsun kullanıcının ana ekrana yani programa fare ile temas etmesi yani tıklaması halinde ekrana "Ekrana tıkladınız" yazılı bir MessageBox çıkacaktır.

Bu özelliğe birkaç ekleme daha yaparak kullanıcının sizin programınızla ilgilenip ilgilenmediğini veya bilgisayardan ayrılıp ayrılmadığını öğrenebilirsiniz.

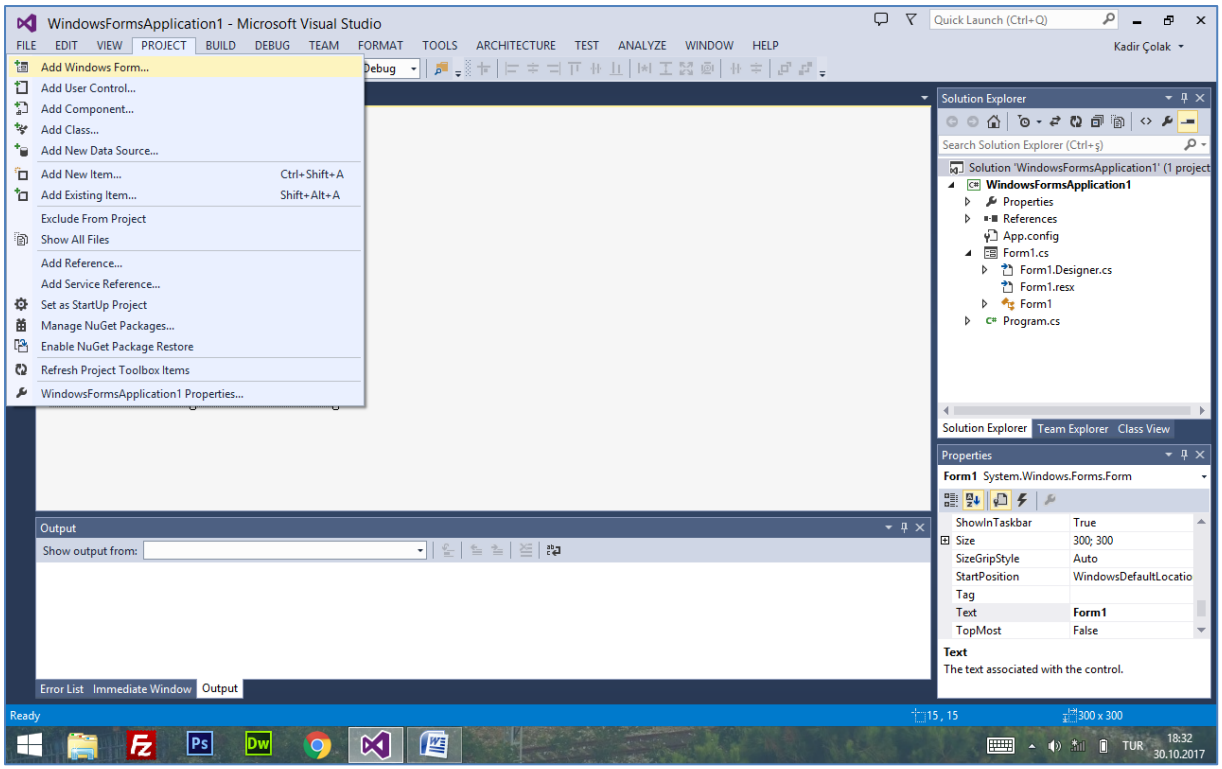


Properties kısmının "*Event*" sekmesinde yer alan MouseMove kısmını aktif hale getiriniz.

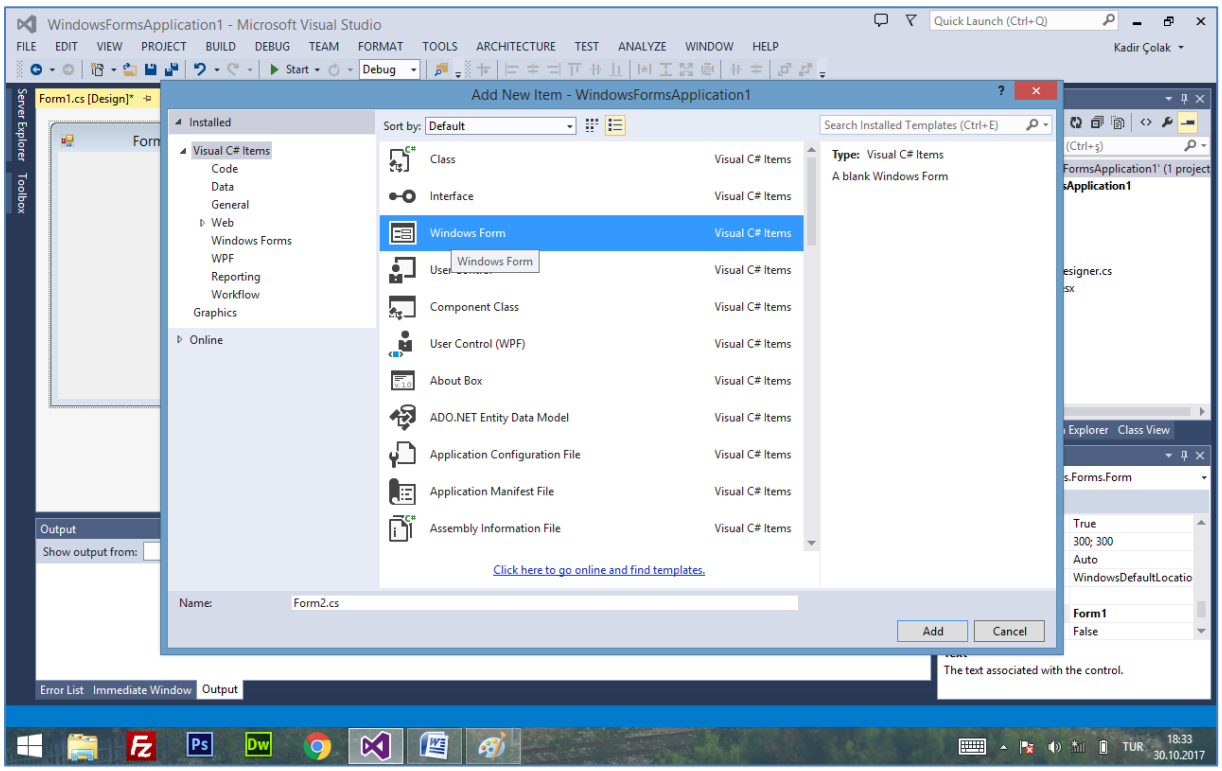
```
private void Form1_MouseMove(object sender, MouseEventArgs e)
{
    MessageBox.Show("Programda hiçbir fonskiyon gerçekleştiremezsiniz");
}
```

Bu kodu yazmanız sonucunda kullanıcı programdan hiçbir verim alamayacaktır çünkü fareyi programın yani formun herhangi bir noktasına getirip tıklama işlemi yaptırmasa bile bu ilgili kodda yer alan MessageBox devreye girecek ve adeta kendini dışarıdan gelen her şeye kapayacaktır.

Bu kullanıcıların yani bir nevi müşterilerinizin karşılaşmak istemeyeceği bir durumdur. Bu kitapta bunun yer almasının nedeni ise örnek çeşitliliğini artırmak olarak görülebilir.



Visual Studio'da sol üstte yer alan menüden sırasıyla Project > Add Windows Form adımları izlendiğinde aşağıda yer alan bir seçim ekranı çıkacaktır;



Buradan Windows Form seçildiğinde form1 ekranının yanına form2 eklenmiş olacak ve 2 farklı pencere ile program hizmet verme eğiliminde olacak demektir.

Yeni form ekleme işlemini yaptırdıktan sana bunu kodlama safhasına geçirme işlemine dair örnekler aşağıda verilmiştir;

```
private void button1_Click(object sender, EventArgs e)
{
    this.Hide();
    Form2 form2 = new Form2();
    form2.Show();
}
```

Burada açık olan form ekranı gizlenerek yeni form ekranı açılacaktır.

```
private void button1_Click(object sender, EventArgs e)
{
    Form2 form2 = new Form2();
    form2.Show();
}
```

Burada ise ilgili önceki form kapanmayacak veya gizlenmeyecek üstüne yeni form ekranı açılacaktır. Bu pek tercih edilmeyen bir seçenektir çünkü kullanıcı aynı butona basarak birden fazla kez form ekranı açtırınca üst üste biriken form ekranları hiç hoş durmayacaktır.

İsteğe bağlı olara aynı anda iki farklı form ekranı birden de açabilirsiniz;

```
private void button1_Click(object sender, EventArgs e)
{
    this.Hide();
    Form2 form2 = new Form2();
    form2.Show();
    Form3 form3 = new Form3();
    form3.Show();
}
```

Programımızda bazı birden fazla tekrar edecek aynı olayı teker teker yazmaktansa bir tane void oluşturarak aynı işlemi farklı yerlerde kullanabilirsiniz. Bunu bir örnekle açıklayalım;

```
private void ozelolay()
{
    sayi1 = 5;
    sayi2 = 7;
    toplam = sayi1 + sayi2;
    MessageBox.Show(sayi1 + "+" + sayi2 + "=" + toplam + "".ToString(), "", MessageBoxButtons.OK,
      MessageBoxIcon.Asterisk);
}
```

Bu yazdığınız private void kalıbının nerede olduğunun hiçbir önemi yok. Sadece herhangi bir toolbox nesnesinin kod parantezleri içinde olmasın yeterli.

```
private void hatamesaj()
{
    MessageBox.Show("Sistemde teknik bir arıza yaşandı. Lütfen daha sonra tekrar deneyiniz!", "",
      MessageBoxButtons.OK, MessageBoxIcon.Error);
}
```

Programınızın birçok aşamasında ve safhasında hata mesajı yaşanacaktır. Her birine teker teker MessageBox ile hata mesajı uyarısı vermektense bir tane private void oluşturmanız sonrasında o void satırını ilgili yerde çağırmanız tıpa tıp aynı işlevi gördürecektir.

```
private void button1_Click(object sender, EventArgs e)
{
    try
    {
        dogrumesaj();
    }
    catch
    {
        hatamesaj();
    }
}
```

Yukarıda butona tıklandığında try-catch ile hata olup olmadığı algılanmış ve bu algılama sonrası private void ile hazır oluşturulan metotlar çağırılmıştır.

```
Math.Pow(2, 3);
```

Parantez içinde yazan ilk sayıya x, ikinci sayıya y dersek x^y şeklinde bir tanımlama yapılabilir. Burada 2^3 mantığıyla sonucu 8 bulduracaktır.

```
Math.Round(23.564646464, 2);
```

Parantez içinde yazılan ilk sayının virgülden sonraki kaç karakter kalacağı durumu parantezin ikinci sayısında verilmiştir. Yani burada çıktı ya da sonuç 23.56 şeklinde olacaktır.

```
Math.Sqrt(25);
```

Parantez içinde yazılan sayının karekökünü hesaplayan matematiksel formüldür. Buradaki değer 5 olarak işlev görecektir.

```
Math.Min(100, 200);
```

Parantez içine yazılan iki sayıdan küçük olanı bulur.

```
Math.Max(100, 200);
```

Parantez içine yazılan iki sayıdan büyük olanı bulur.

```
Math.Floor(23.2234234);
```

Parantez içinde yazılan sayının virgülünü atar ve tam sayı (int) tipinde çıkarım yapar.

Bu sayfada verilen ve Math kalıbına dahil olan örneklerin hepsi Console Application bünyesinde uygulandığı ve kodlandığı zaman aynı değeri görecektir. Windows Form Application yapısına özel bir kod değildir, hem Console hem Form Application ekranlarında kullanıma açıktır.

```
public Form1()
{
    InitializeComponent();
}
public static string ad = "Kadir ÇOLAK";
1 reference
private void Form1_Load(object sender, EventArgs e)
{

}
```

Normal bildiğiniz int-string-char-double değişken kalıplarının başına public static eklemesi yaparak değişkenimizi her formda (ne kadar form eklemesi yaparsanız yapın) kullanılabilir hale getirirsiniz.

```
public Form2()
{
    InitializeComponent();
}
string adform1 = Form1.ad;
1 reference
private void Form2_Load(object sender, EventArgs e)
{
    MessageBox.Show(adform1 + "");
    MessageBox.Show(Form1.ad + "");
}
```

form2 ekranında form1 de oluşturulan ad değişkenini iki farklı şekilde kullanabilir, yazdırabilir ve işleme sokabilirsiniz.

Windows Form Application bünyesinde değişkenlerden yana bir hata yaşamamak istiyorsanız Console Application kodlarında ki gibi değişkenleri en üstte oluşturun. Bu üst noktanın neresi olduğunu anlatan örnek görsele aşağıda yer verilmiştir;

```
namespace WindowsFormsApplication1
{
    5 references
    public partial class Form1 : Form
    {
        1 reference
        public Form1()
        {
            InitializeComponent();
        }
        int sayi = 5;
        string ad;
        char nokta = '.';
        double pi = 3.14;
```

listBox ile her türlü ekleme-çıkarma ve güncelleme işlemleri yapabilirsiniz;

```
private void button1_Click(object sender, EventArgs e)
{
    listBox1.Items.Add("İstanbul");
}
```

Bu kodda listBox1 nesnesinin içine "İstanbul" yazılı bir eleman eklenir.

```
private void button1_Click(object sender, EventArgs e)
{
    listBox1.Items.Remove("İstanbul");
}
```

listBox1 içerisinde yer alan "İstanbul" metinli ifadeyi siler, kaldırır.

```
private void button1_Click(object sender, EventArgs e)
{
    listBox1.Items.RemoveAt(2);
}
```

listBox1 içinde yer alan 3.elemanı siler. Parantez içinde 2 yazması onun 3.elemanı sildireceği anlamına gelmektedir çünkü listBox nesnesinin ilk elemanı 0.eleman olarak değer görür.

```
private void button1_Click(object sender, EventArgs e)
{
    listBox1.Items.Clear();
}
```

listBox1 içinde yer alan bütün elemanları kayıtları siler.

```
private void button1_Click(object sender, EventArgs e)
{
    int uye = listBox1.Items.Count;
    MessageBox.Show(uye + "");
}
```

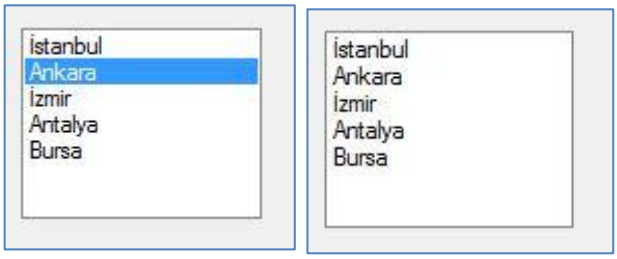
listBox içinde normal sayma mantığıyla kaç eleman olduğu Count ile bulunabilir.

```
private void button1_Click(object sender, EventArgs e)
{
   int kontrol = listBox1.Items.IndexOf("İstanbul");
   if(kontrol==0)
   {
        MessageBox.Show("İstanbul elemanı bulundu");
   }
   if(kontrol==-1)
   {
        MessageBox.Show("İstanbul elemanı bulunamadı");
   }
}
```

Yukarıda yer alan kodda IndexOf metoduyla "İstanbul" elamanın olup olmadığı int kontrol değişkenine atanmıştır. Kontrol değişkeni 0 olduğunda aranan **nesne var**, -1 olduğunda ise o ilgili **nesne yok** demektir.

```
private void button1_Click(object sender, EventArgs e)
{
    listBox1.SelectedItem = null;
}
```

Kullanıcı tarafından seçilen listBox elemanı seçilmemiş olarak dönüşüm geçirir.




```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show(listBox1.SelectedItem.ToString());
}
```

Kullanıcı o an listBox içerisinden hangisini seçerek mavi renge bürüdüyse butona tıklaması sonucunda MessageBox içerisinde o şehir yazacaktır.

Timer nesnesi içerisinde saniye görevi görüp sürekli güncellenecek artacak değerin değişken tipi **int** olmak zorundadır. Çünkü artım/azalım yani matematiksel işlem yapılacak.

```
private void timer1_Tick(object sender, EventArgs e)
{
    progressBar1.Value++;
}
```

Burada timer nesnesi her arttığında yani belirli bir süre geçtiğinde progressBar nesnesi birer birer artacaktır;

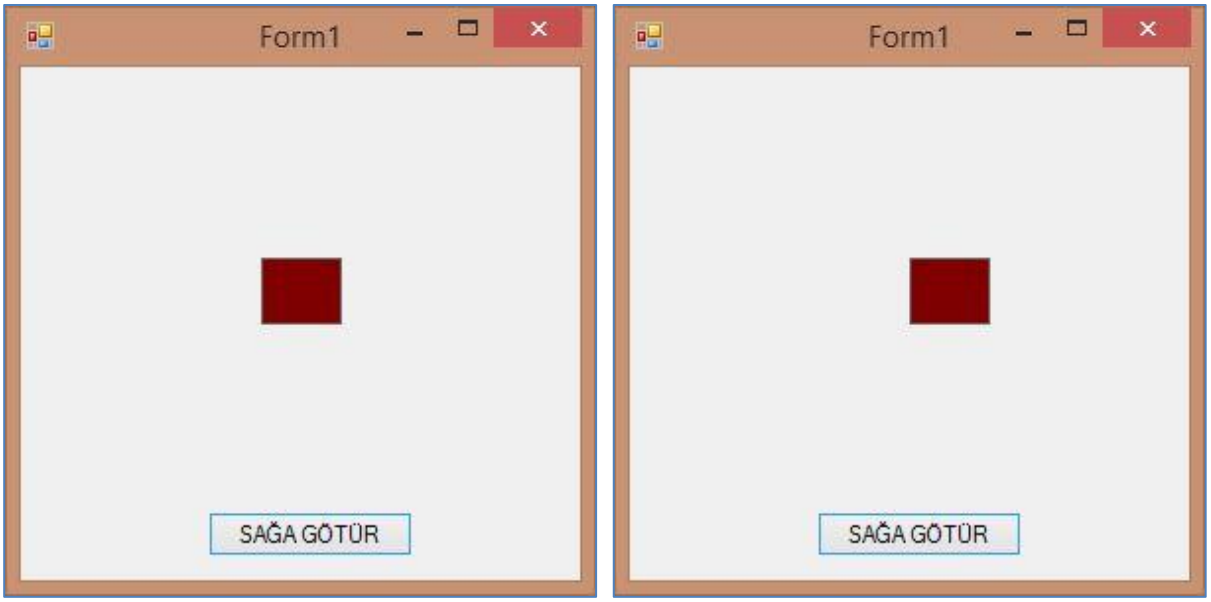
```
private void timer1_Tick(object sender, EventArgs e)
{
    saniye++;
    label1.Text = saniye + " saniye geçti";
}
```

Bu kod ile tam bir kronometre işlevi yapılmaktadır. Geçen her saniye sonrası artan saniye değişkeni her artışında label nesnesine yazdırılmaktadır.


```
private void timer1_Tick(object sender, EventArgs e)
{
    saniye++;
    if(saniye==50)
    {
        timer1.Enabled = false;
    }
}
```

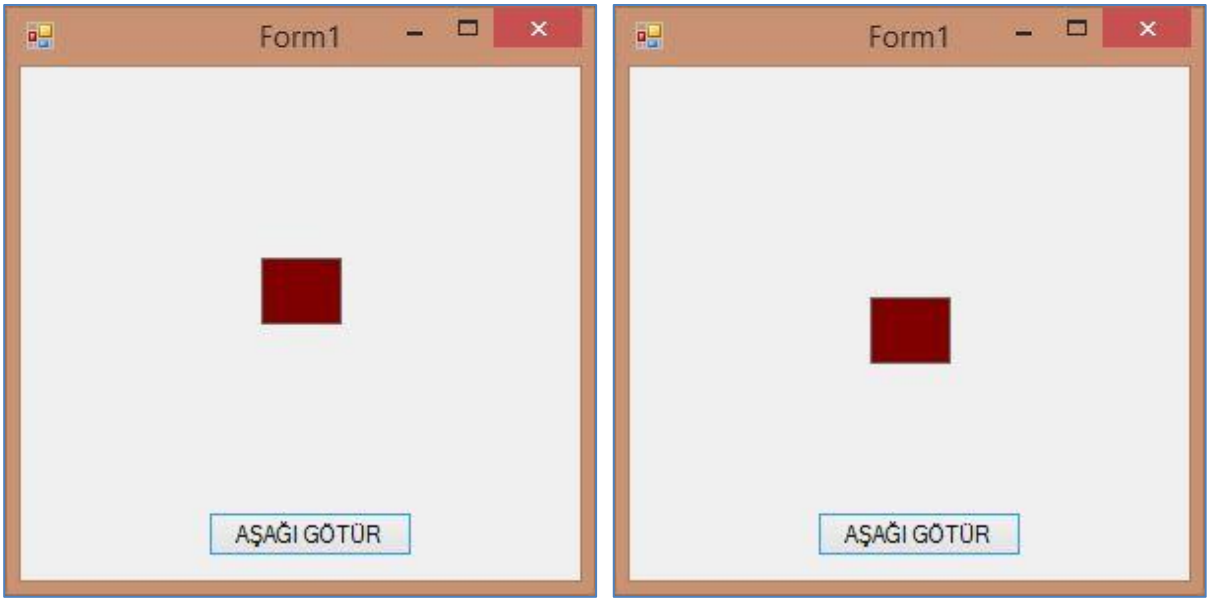
0'den itibaren her saniyede artma işlemi sonunda 50 saniye olduğu an Enabled=false; komutuyla timer artık etkisiz hale gelecektir ve duracaktır. Başka bir kod içinde timer nesnesi Enabled=true; olduğunda timer nesnesi tekrardan açılacaktır.

```
private void button1_Click(object sender, EventArgs e)
{
    pictureBox1.Left += 20;
}
```

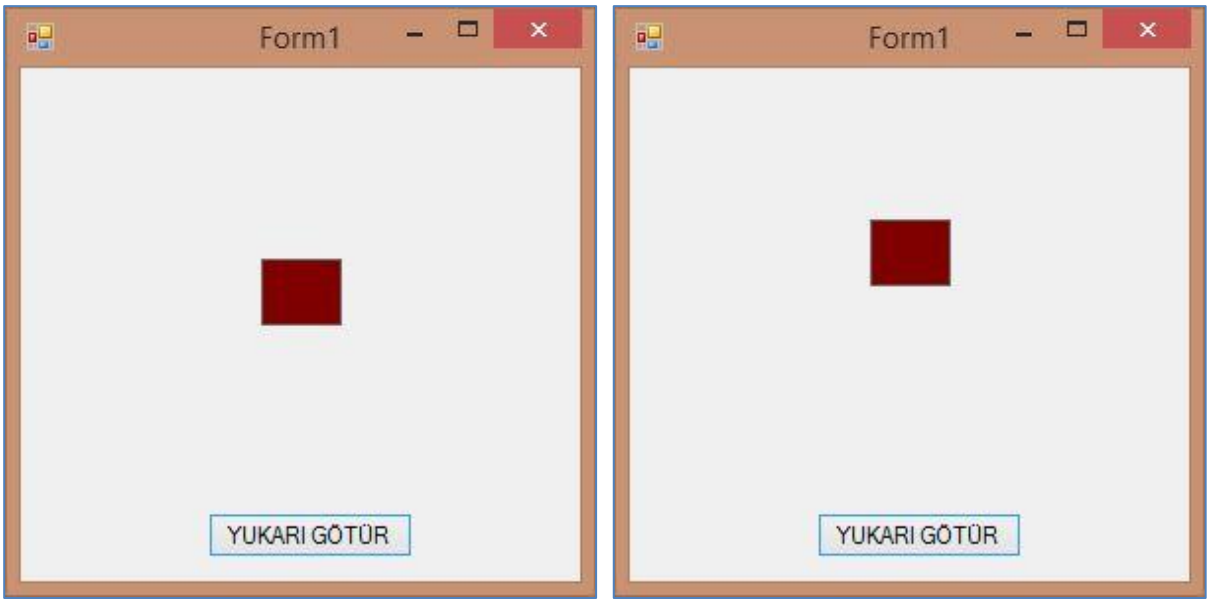



```
private void button1_Click(object sender, EventArgs e)
{
    pictureBox1.Left -= 20;
}
```

```
private void button1_Click(object sender, EventArgs e)
{
    pictureBox1.Top += 20;
}
```




```
private void button1_Click(object sender, EventArgs e)
{
    pictureBox1.Top -= 20;
}
```



```
private void button1_Click(object sender, EventArgs e)
{
    textBox1.Text = textBox1.Text.Replace(".", ",");
}
```

Yukarıda yer alan kodun yansımasında textBox nesnesine girilen . değerlerini , değeri olarak dönüştürecektir.Bu girilen değerleri filtreleme ve yapılması muhtemel yanlışları önlemek için güzel bir mekanizma sayılabilir.

Kullanıcının girdiği 3.14 değeri butona basması sonrası 3,14 olarak dönüşecektir.

```
private void button1_Click(object sender, EventArgs e)
{
    textBox1.Text = textBox1.Text.Trim();
}
```

textBox nesnesinin başında, sonunda veyahut ortasında oluşu önemsenmeden herhangi birkaç noktasına bırakılan space yani boşluk karakterlerini siler. Kullanıcı örnek olarak “a d a n a “ değerini girdiyse bu kod ile değer “adana” olarak boşlukları silinmiş şekilde değişim gösterir.

```
private void button1_Click(object sender, EventArgs e)
{
    for(int i=0;i<=1000;i++)
    {
        this.Top += 5;
        this.Left += 5;
        this.Top -= 5;
        this.Left -= 5;
    }
}
```

Bu kod ile form ekranında titreşim görünümü sağlanmaktadır. For döngüsü ile istediğiniz süre boyunca form ekranına titreşim yapabilirsiniz. Burada yazan 1000 sayısı 1 saniyeye, 3000 sayısı ise 3 saniyeye eşit olacaktır.

```
private void button1_Click(object sender, EventArgs e)
{
    textBox1.Text = textBox1.Text.Substring(0, textBox1.TextLength - 1);
}
```

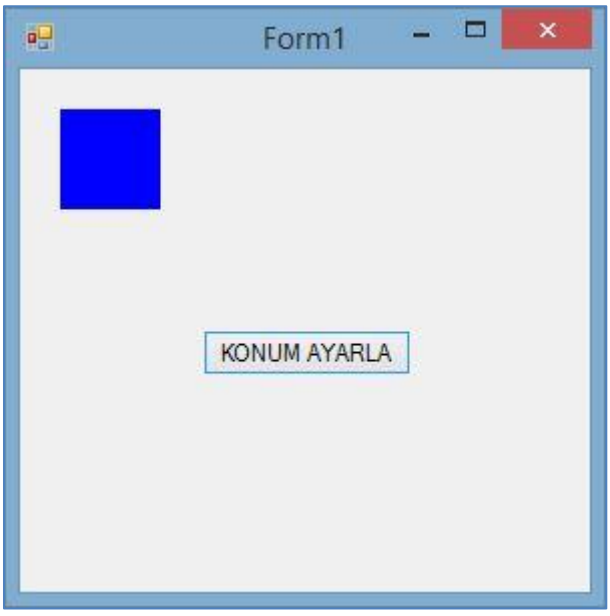
Bu kod ile klavyedeki silme tuşu görevi görülür. Yazılan değerin son harfi ya da karakteri silinir, yok edilir.

```
if (pictureBox1.Right >= label2.Left && pictureBox1.Bottom >= label2.Top)
{
    if (pictureBox1.Left <= label2.Right && pictureBox1.Top <= label2.Bottom)
    {
        MessageBox.Show("pictureBox1 ve label2 nesneleri birbirine değiyor");
    }
}
```

Yukarıdaki kod ile pictureBox1 ve label2 nesnelerinin dört yanlarından herhangi birinin ikili şekilde karşılıklı temas, dokunma durumu yapması karşılığında MessageBox devreye girecek ve form ekranında bir bildirim verecektir.

```
private void button1_Click(object sender, EventArgs e)
{
    pictureBox1.Left = 20;
    pictureBox1.Top = 20;
}
```

pictureBox nesnesinin sola ve yukarı olan uzaklığını istediğiniz ölçütlerde ayarlayabilir ve değiştirebilirsiniz. pictureBox nesnesinin sola olan uzaklığını belirleyerek otomatik olarak sağa olan uzaklığını da belirlemiş olursunuz. Aynı şekilde yukarıya verdiğiniz uzunluk ile aşağıda kalan uzunluğu ve mesafeyi belirlemiş olursunuz. Burada sola ve yukarıya olan uzaklıkları 20 olarak verilmiştir.



Mavi arka plan renkli pictureBox nesnesinin resimdeki konumu sola 20 yukarıya 20 birimdir. 300*300 ölçülerinde olan bir form ekranında genişliği 50 piksel olan pictureBox nesnesinin en sağa olan uzaklığı 300 – (50+20) ile bulunabilir. Bu işlem sonucunda bu pictureBox nesnesi sola 20, sağa ise 230 birim uzaklıkta yer almaktadır.

BİLİNMESİ GEREKEN TERİMLER:

Add: Ekle, dahil et.

Algoritma: Belirli bir sorun veya olay karşısında sırasıyla izlenecek adımlar bütünü.

BackColor: Arka plan rengi.

bool: true ve false değerleri alabilen değişken türü.

break: Kırmak, döngüden çıkmak, bitirmek.

Bug: Programda yaşanan ve sistemde sorun yaşattıran hata, eksik, giderilemeyen durum.

button: Kullanıcının basması dahilinde işe yarayan bir form nesnesi.

case: switch ihtimallerinin dışında kalan bütün senaryolarda gerçekleşecek olaylar.

catch: Hata, arıza yaşandığı durumlar.

Check: Nesne veya formun işaretlenmesi, seçilmesi.

checkBox: Kullanıcının aynı ekranda birden fazla işaretleyebildiği dikdörtgen işaretleme kutusu sayılan form nesnesi.

Clear: Komple içindekilerin hepsini birden silmek, kaldırmak.

Click: Nesne veya forma tıklanması.

comboBox: Kullanıcının içinden bir tanesini seçtiği listelenmiş şekilde seçim için sunulan menü.

Console: Tasarımı değiştirilemeyen siyah arka plana sahip standart cmd tipi ekran.

const: Sabit kalan, değişmeyen veya değiştirilemeyen.

Convert: Dönüştürme, tür ve çeşitler arasında değişim yaptırma.

Count: Toplam eleman sayısı, o anlık karakter kotası.

Cursor: Fare imlecinin şekli.

Default: Seçili, varsayılmış, hazır olarak kabul edilen.

Design: Görünüm ekranı.

double: 1'e tam bölünen veya bölünmeyen (kesirli-irrasyonel) sayıları kapsayan değişken türü.

Döngü: Belirli bir işin belirlenen kez tekrar etmesi ve gerçekleşmesi durumu.

else: if ve else if kodları dışında geriye kalan herhangi bir ihtimalin yaşanması durumu.

Enabled: Nesne veya formun kullanılabilir yani uygulamaya açık olup olmaması durumu.

exe: Programın saf şekliyle kullanıcıda görülen uygulama uzantısı.

False: Olumsuz, olmaz, gerçekleşmez.

float: double değişkeniyle eş görevi üstlenen virgüllü ve 1'e tam bölünen sayıların değişkeni.

for: İstenilen sayıda istenilen bütün işlemlerin tekrar edilmesi.

Form: Tasarımı değiştirilebilen ve yeni nesnelerin eklenmesine açık geniş uçlu bir dizayna sahip modern görsel program ekranı.

goto: Belirlenmiş bir noktaya hızlı ve aradaki kodları es geçerek ışınlanma özelliği veren kod yapısı.

Increment: numericUpDown nesnesinde sayısal değerlerin ne kadar azalıp ne kadar azalacağını belirleyen değer.

Item: Eşya, öge, ürün.

if: Programda olup olmadığı gerçekleşip gerçekleşmediği durumunun kontrol edilmesine yarayan kod yapısı.

int: 1'e tam bölünene negatif veya pozitif oluşu önemsenmeyen bütün matematiksel işleme tabi tutulabilecek sayılar.

label: Form uygulamalarında yazı yazdırma görevini üstlenen nesne.

listBox: Eleman ekleme, çıkarma ve güncelleme özelliklerini barındıran veri toplama ve depolama yeri.

Load: Yükleme, oluşturma.

Location: Nesnenin form ekranında yer alan X ve Y koordinatları, konumu.

Max / Maximum: En fazla, en çok, en büyük.

Main: Ana parça, gövde.

menuStrip: Programın en üstünde ve solda yer alan ayrı bir yönetim merkezi.

MessageBox: Form ekranında kullanıcıya mesaj verdirme görevini üstlenen özel kod.

Min / Minimum: Asgari değer, en az, en küçük.

monthCalendar: Tarihi ve zamanı göstermeye yarayan form nesnesi.

null: Boş, değeri olmayan, değersiz değişken.

Opacity: Görünürlük derecesi, şeffaflık miktarı.

numericUpDown: Sayısal girişle kısıtlanmış kullanıcının artırma ve azaltma seçeneğiyle değeri oynatabildiği form nesnesi.

pictureBox: Resim konulabilen form nesnesi.

private void: Ayrı şekilde oluşturulan ve istenildiğinde çağırılan metot kalıp.

progressBar: Yükleme çubuğu.

radioButton: Kullanıcının aynı ekranda birden fazla işaretleyemediği yuvarlak işaretleme kutusu sayılan form nesnesi.

random: Rastgele anlamına gelen ve programlamada rastgele sayı üretimi esnasında ihtiyaç duyulan özel kod yapısı.

Round: Yuvarlama.

Stop: Durmak, sonlandırmak, bitirmek.

string: Kelime, cümle veya sayısal değerleri içine alan (matematiksel işleme tabi tutmayan) en geniş olasılığa havuzuna sahip değişken türü.

switch: Kullanıcının girdiği değeri kontrol eden ve ona göre programları yönlendirme özelliğine sahip kod yapısı.

Sqrt: Math kodları içerisinde sayının karekökünü hesaplayan kod.

Text: Yazı, metinsel ifade.

textBox: Kullanıcıdan değer ve veri almaya yarayan kutu şeklindeki form nesnesi.

Timer: Süreölçer, zaman tutucu ya da diğer adıyla kronometre.

Title: Başlık, üst isim.

Trace: Oluşturulan programdaki hataları ve sorunları gözleme süreci.

try: Hata, arıza yaşanmadığı durumlar.

ToLower: Değerin bütün harflerini küçük yapan fonksiyon.

ToUpper: Değerin bütün harflerini büyük yapan fonksiyon.

ToolBox: Windows Form Application içerisine nesne eklemeye yarayan “eşya kutusu” anlamına gelen menü.

True: Olumlu, olur, gerçekleşir.

Value: Sayısal değer, sayısal ifade.

Visible: Nesne veya formun somut durumunu yani görülebilir veya görülemez olması durumu.

while: for yapısından farklı olsa da gene döngü ve tekrar içeren özel kod yapısı.

Write: Yazdırmak, yazdırma işlemi.